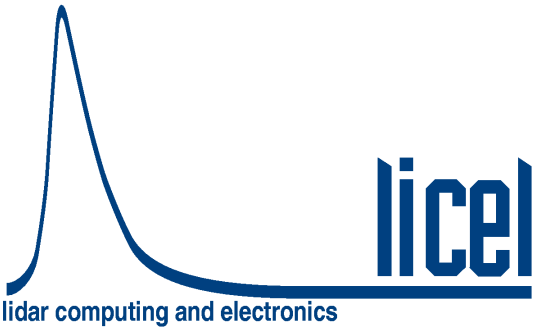


Licel Single Channel Lidar Detector [Lidarino] – Installation and Reference Manual

Licel GmbH

October 4, 2025

Contents

Chapter 1

The Licel Single Channel Lidar Detector

1.1 Overview

The Licel Lidar Detector is a single channel detection system. . It is based on a metal-channel-dynode photomultiplier. The PMT with a single photon counting systems provides one dimensional range resolved data.

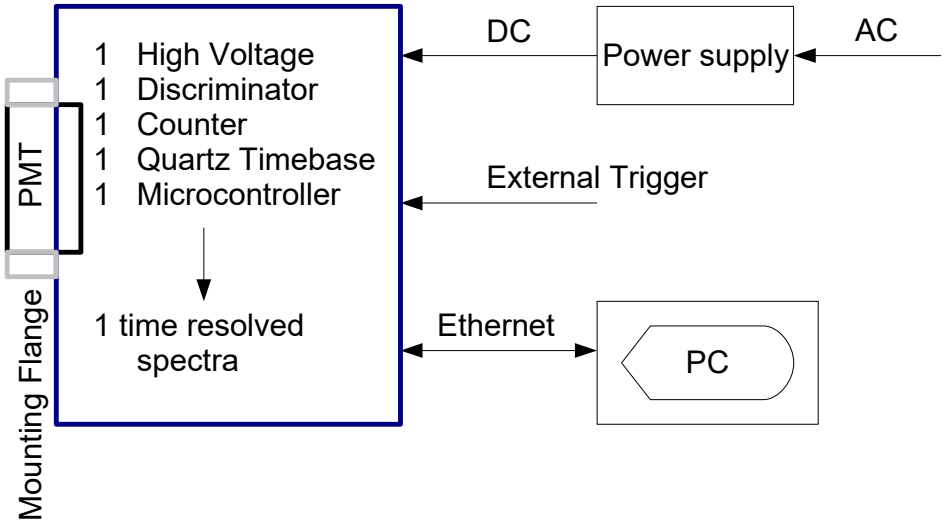


Figure 1.1:

The high voltage supply, an adjustable discriminator and the trigger logic is integrated on the module. The Ethernet interface is used together with LabVIEW software to control the measurement and readout the acquired data.

1.2 The Counter

The counter contains a microcontroller which receives its commands over the Ethernet. It will set the high voltage for the PMT and the discriminator level. It will transmit the averaged data over the Ethernet to the PC.
The counter receives the raw anode signals from the PMT. It also provides the software controlled high voltage for the PMT. The signal is then amplified and fed in to the discriminator. The threshold level is software controlled. Behind the discriminator is a counter which will count the single photons. Beginning with an external trigger the counts are recorded. A fixed quartz based time base will be used to read out the data. The microcontroller will average the data over multiple trigger events.

Once enough data has been acquired the summed up data will be transferred to the PC.

1.3 Detector

The module is based on a Hamamatsu R9880 metal can photomultplier. The mechanical outline is shown in fig. 1.2.

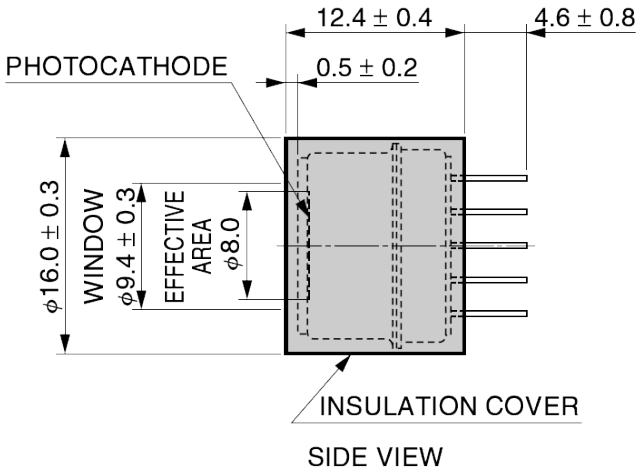


Figure 1.2: Mechanical outline of the detector

1.3.1 Gain characteristics

The gain of the cathode depends on the voltage setting. The pulse height distributions at 900 V and 950 V are shown in the plots below.

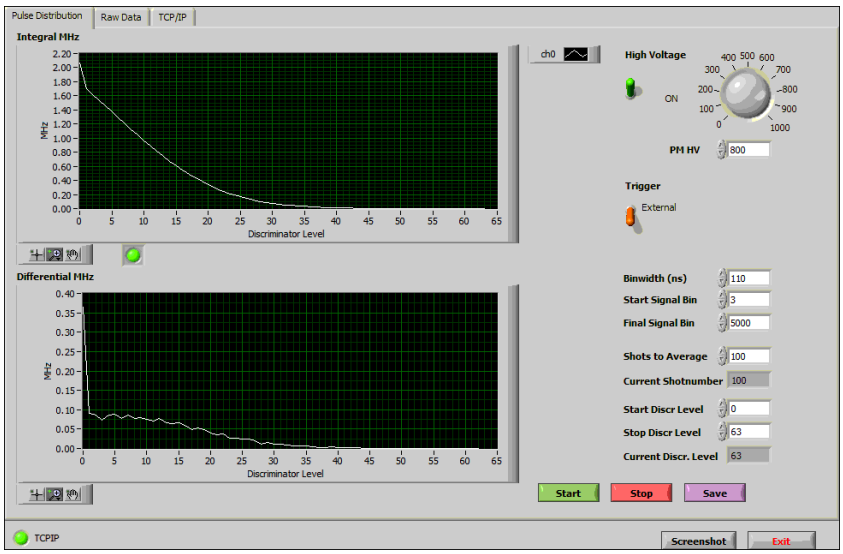


Figure 1.3: Pulse height distribution at 800 V

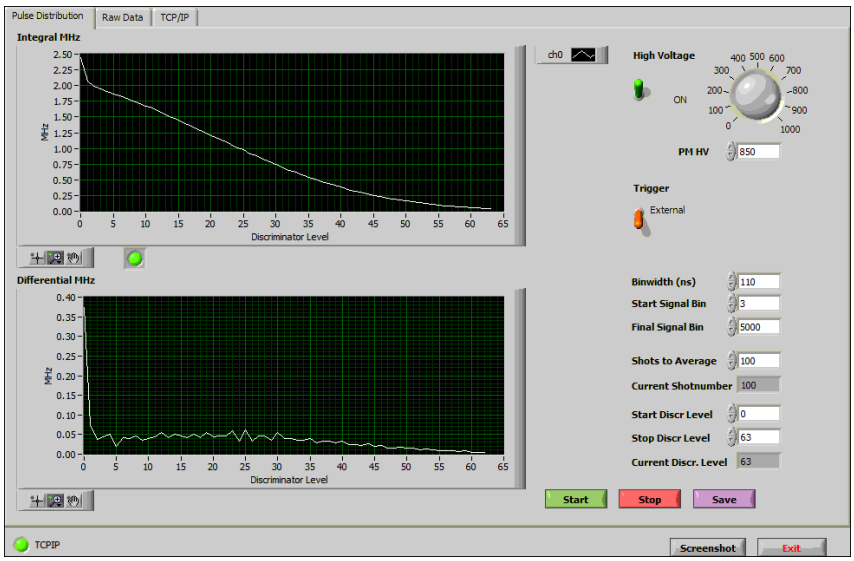


Figure 1.4: Pulse height distribution at 850 V

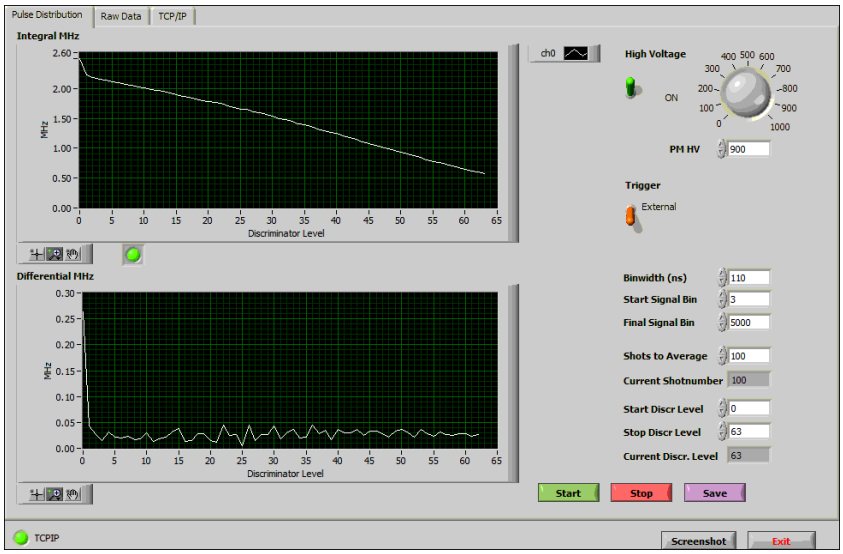


Figure 1.5: Pulse height distribution at 900 V

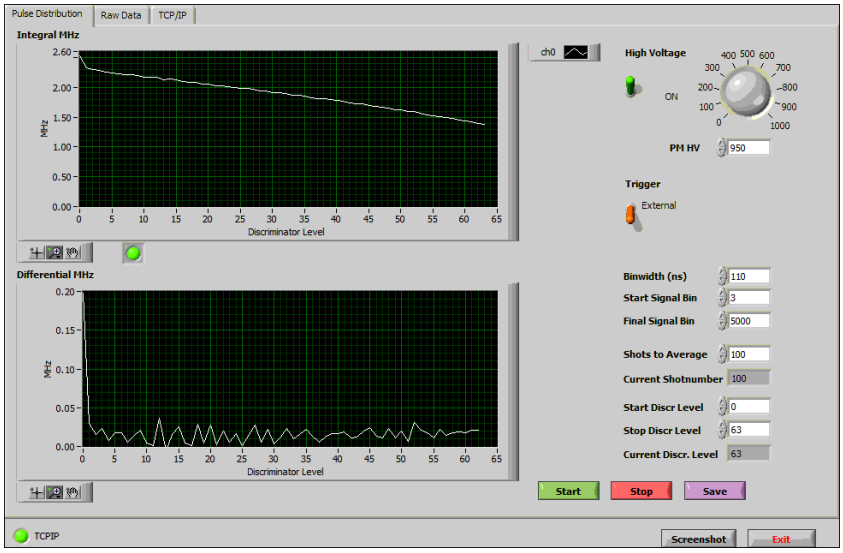


Figure 1.6: Pulse height distribution at 950 V

Chapter 2

Hardware Installation

The Licel Lidar Detector consists of the following equipment:

1. power supply unit
2. detector & electronics housing
3. mounting flange (premounted at the detector & electronics housing)
4. DC power cable (Lemo 7 pole)
5. TEC cable (Lemo 4 pole)
6. line cable
7. Ethernet cross link cable
8. trigger cable (Lemo CAMAC to BNC coaxial)

2.1 Power Supply Unit

The power supply cassette can be integrated to standard 19” form factor systems. The line connector is located at the rear side. The power switch, the control LED, and the DC output connector are on the front side.

Figure 2.1: Front (left) and rear sides of the power supply cassette

2.2 Detector & Electronics Housing

The detector & electronics housing contains the detector, the counter and the network interface. On top the DC input connector (7 pole Lemo), the network reset switch, the trigger input (Lemo CAMAC), and the Ethernet connector (RJ45) are located as seen in the figure. The 4 pin Lemo connector is for future shutter extensions.

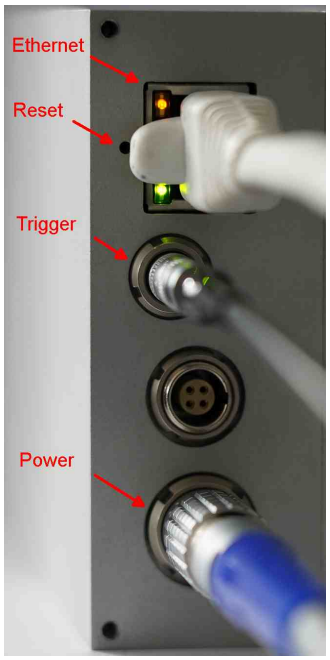


Figure 2.2: Detector & electronics housing

2.2.1 Mechanical dimensions

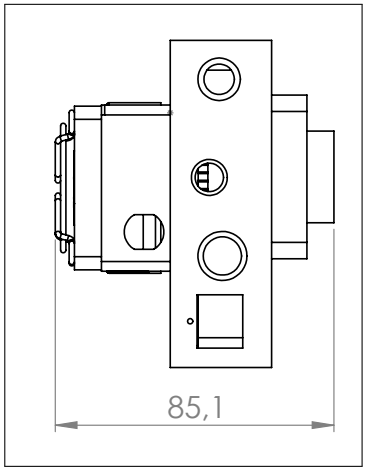


Figure 2.3: Side View

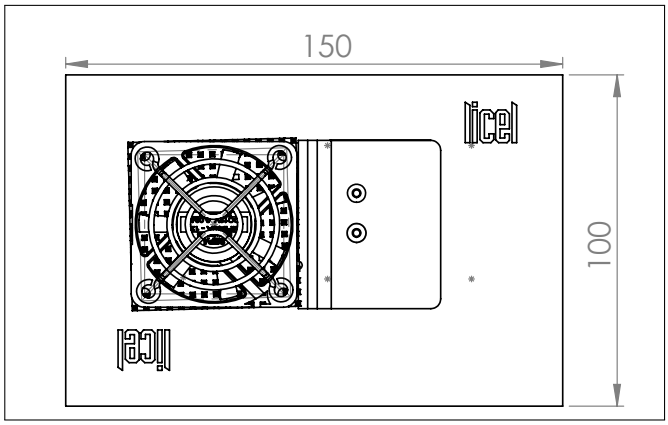


Figure 2.4: Top View

2.3 Mounting the Detector & Electronics Housing

To connect the detector & electronics housing to the detection box the mounting flange is used. The mounting interface is 05 Thorlabs thread.
Please make sure that daylight illumination of the detector is carefully avoided during mounting.

Figure 2.5: Mount SM05

2.4 Cable Connections

The following cable connections are required:

1. line connection to the rear side of the power supply unit (line cable included)
2. DC supply connection from the front of the power supply to the detector & electronics housing (Lemo 7 pole cable included)
3. trigger input from a trigger source to the trigger input at the detector & electronics housing (Lemo CAMAC to BNC cable included)
4. Ethernet connection to the detector & electronics housing. A cross link cable for a direct connection to the computer is provided. A straight patch cable is required when connecting to an Ethernet switch or hub.

Chapter 3

Software Installation

Licel provides a package of software modules for setting up the Licel Lidar Detector for network operation, and for operating the Licel Lidar Detector. These software modules are written in LabVIEW's G language. The software is provided as LabVIEW source for users who have LabVIEW (beginning with version LabVIEW 2010) 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 (2000|XP|Vista) computer. Licel provides the software on a CD ROM.

3.1 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 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...) or onto a CD ROM.

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...) or burning them onto a CD ROM.
2. Scan your disks to find all versions of the following directories (or files with similar names if you migrate from an older version of Licel's Acquisition Software) and delete them once you have made backups of them. Delete all other `LLB` files.

    1. `project\PM32Channel_src.lvproj`
    2. `source\AdvancedViewer.llb`
    3. `source\Datafile.llb`
    4. `source\LicelAcquis.llb`
    5. `source\LicelFile.llb`
    6. `source\LicelGraph.llb`
    7. `source\LicelModule.llb`
    8. `source\LicelTCPIP.llb`
    9. `source\LicelTCPIP_API.llb`
    10. `source\LicelUtil.llb`

11. source\PMT32Channel.llb
12. source\PMT32ChannelPulse.llb
13. source\PMT32Viewer.llb
14. source\Search Controllers.llb

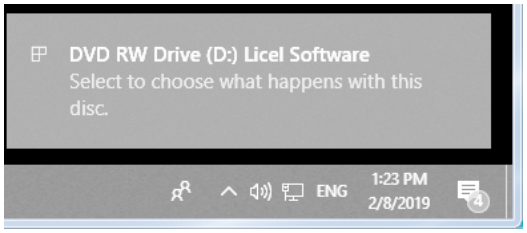
Please note: Licel may have provided individual software solutions with additional or less directories and/or LabVIEW library files than noted in the list above.

3. Search the directory your older version of Licel LabVIEW libraries reside and backup all initialization files (`*.ini`).

3.2 The Licel CD ROM

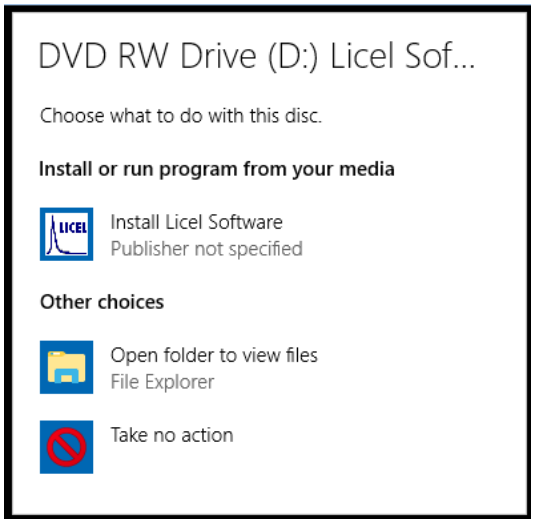
The standard CD ROM provided by Licel contains both, the LabVIEW sources and the Windows Installer for installing the Windows applications, and furthermore a documentation folder. Licel may add customer specific components on the CD ROM.

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.



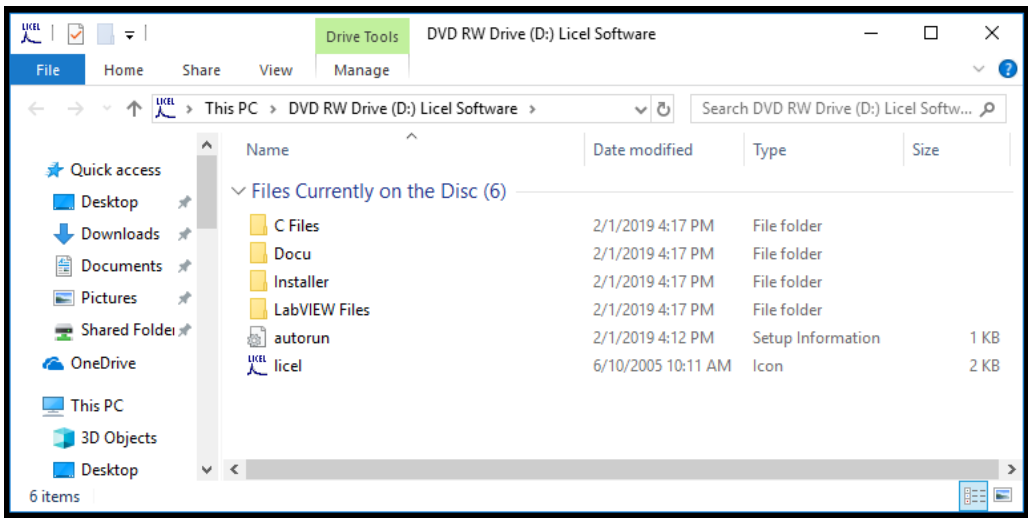
Please click on the pop-up message.

3. The following selection dialog should appear:

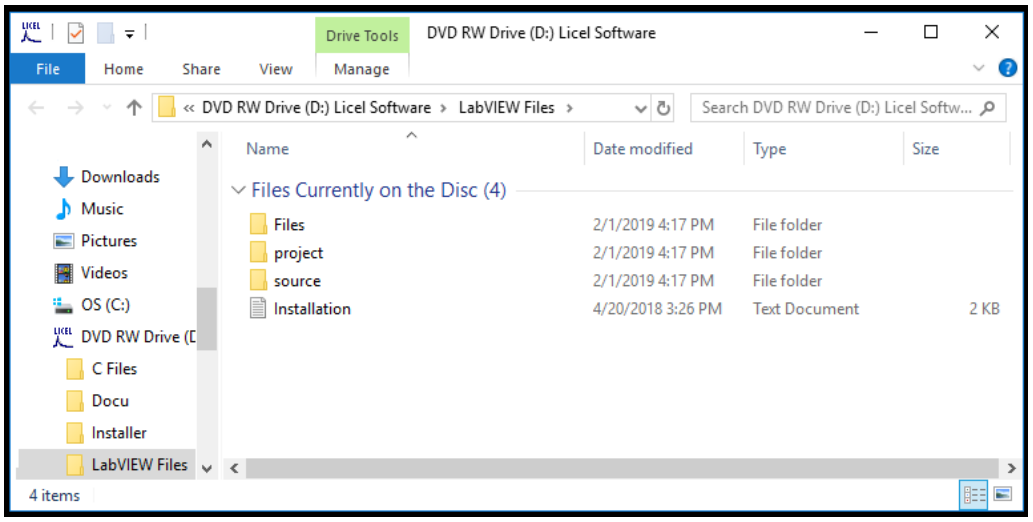


In older Windows operation systems a similar dialog will automatically come up.

- 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 3.3.
- 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.

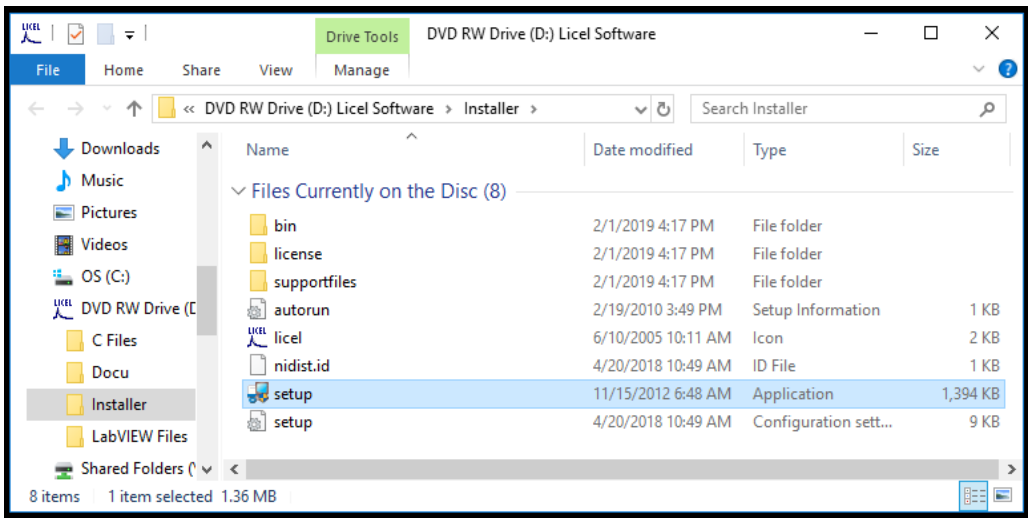


Please note the remarks according to existing LabVIEW library files. Please refer to the section 3.4 for further details.

– 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.



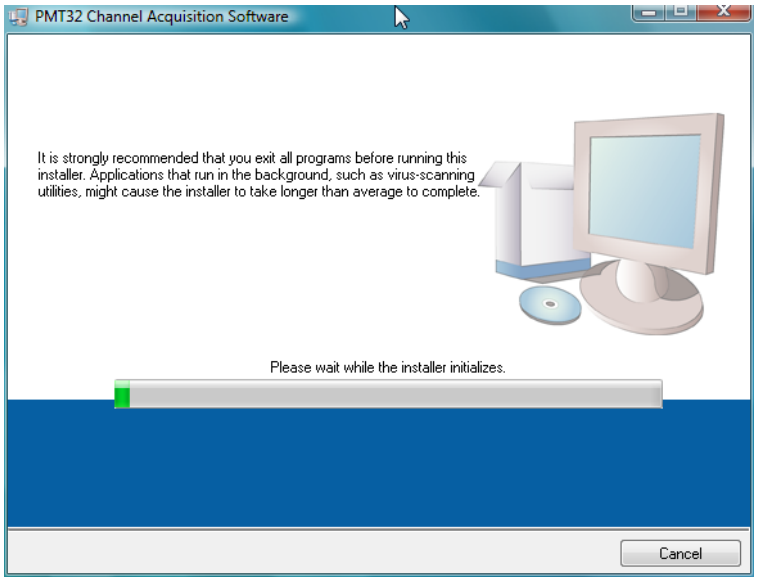
Please proceed to the section 3.3 afterwards.

3.3 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. 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 Licel Software** 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.

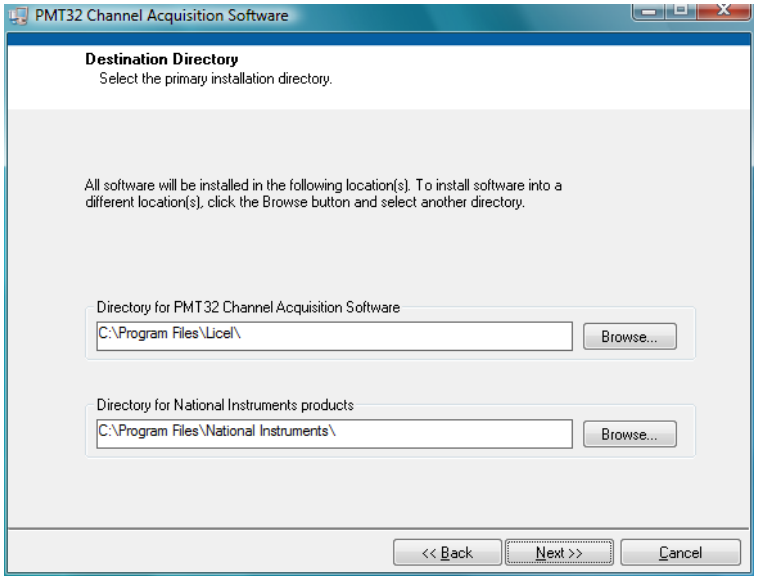
The Windows Installer dialogs will guide you through the installation process.

1. The installation will start with the following screen. Please wait until the installer has initialized.



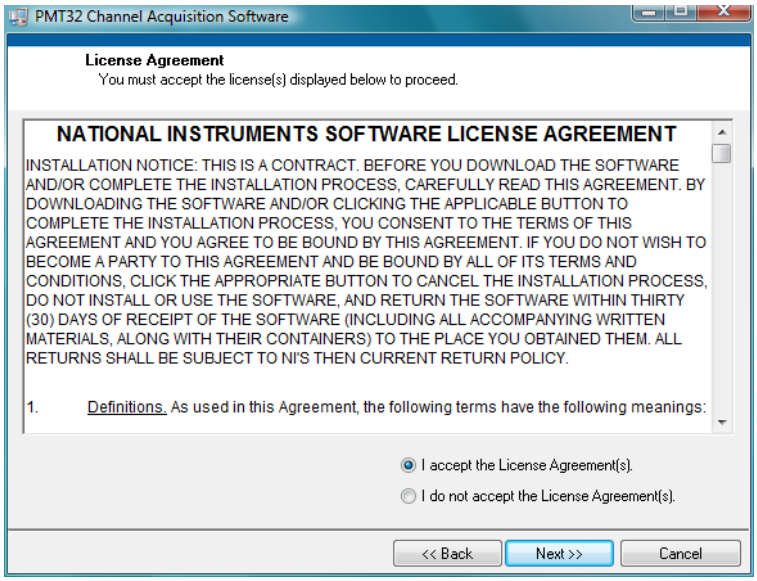
2. The next screen will allow you to choose the installation directory (standard: `<Program Files`

`Directory>\Licel`) and specify the directory where former National Instruments products are installed. The latter directory will be the root directory for the LabVIEW runtime engine.

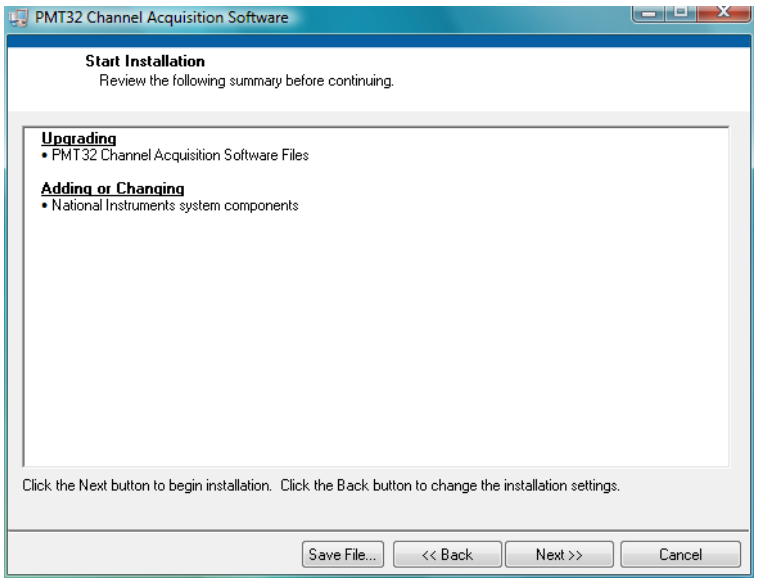


If you would like to change one of these directories click **Browse** and choose or create a directory of your choice. Click **Next** to proceed.

3. As National Instrument's LabVIEW runtime engine has to be installed you are asked to accept a software license agreement for National Instruments software. Plese check *I accept the license agreenment(s)* and proceed by clicking **Next**

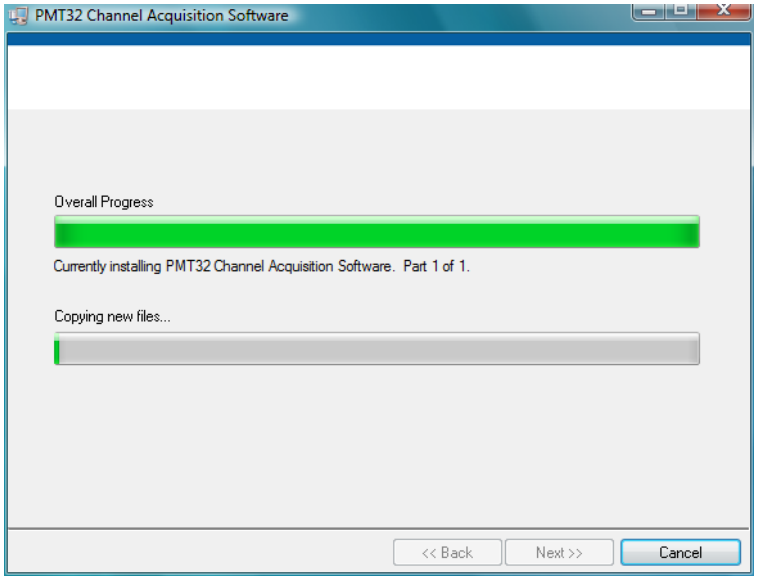


4. The next screen will inform about the software components that will need to be installed on your system.

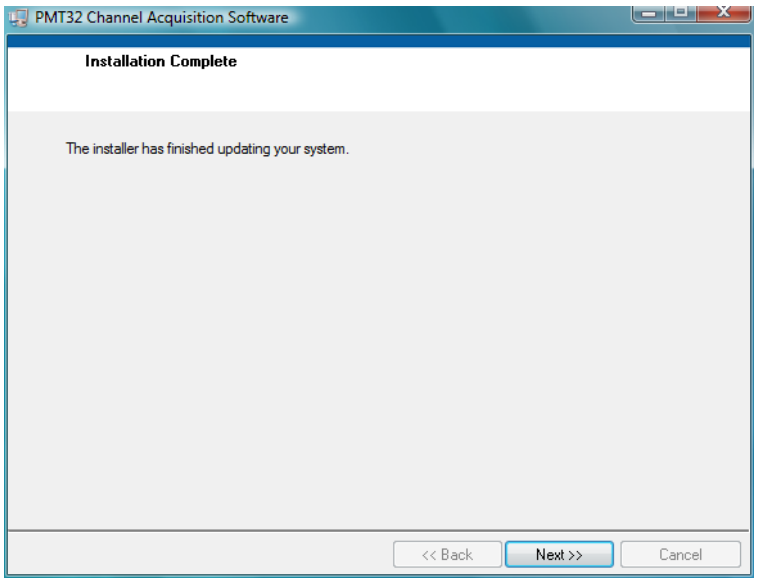


Please click **Next** to proceed.

5. The next screen contains the progress indicators during the installation.

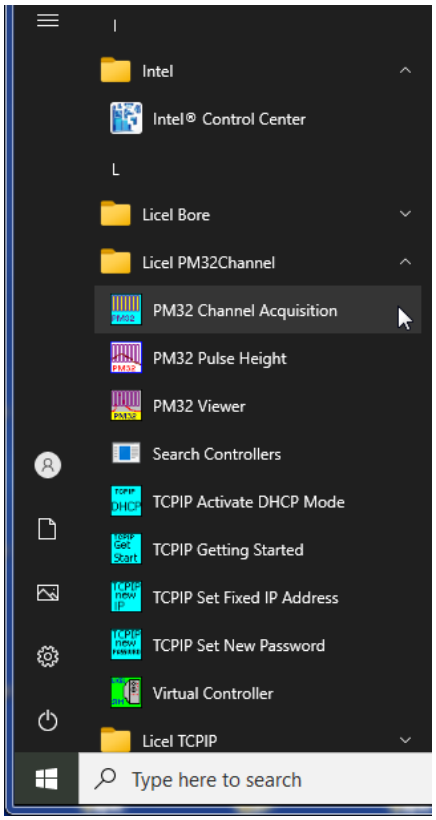


6. When the installation has finished please click **Next** to proceed.



7. If it is is necessary to restart the computer after the applications a dialog will pop up with an appropriate button for the restart procedure.

8. After the installation has successfully been completed you are able to start the Windows applications through the corresponding entry in the program group **Licel PM32Channel** in the Windows start menu:



The links to start the applications can be found after opening the program group as seen above.

9. If you have backuped 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).

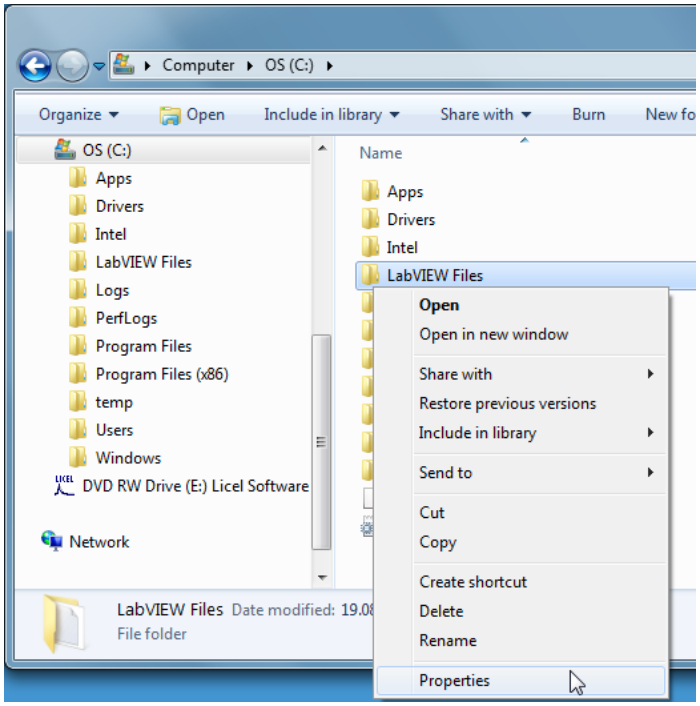
3.4 Installing the Licel LabVIEW Sources

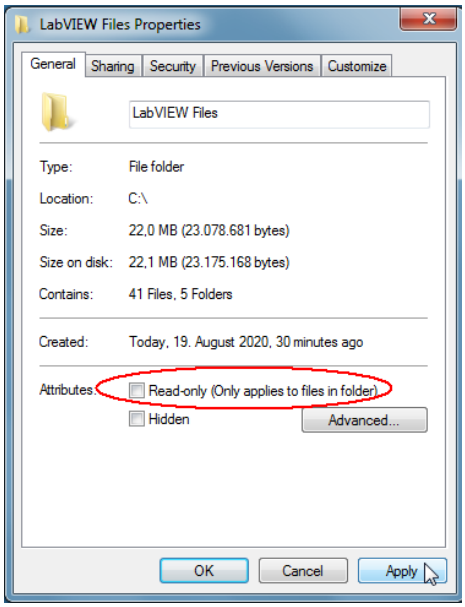
To install the Licel LabVIEW sources you may choose between the following options:

- Copy all files contained in the directory `LabVIEW Files` from the CD ROM to a directory of your choice.
- If you downloaded the Licel software from http://www.licel.com/software.htm please unpack the content from the downloaded zip file and copy it to a directory of your choice (keep all directory hierarchies!).

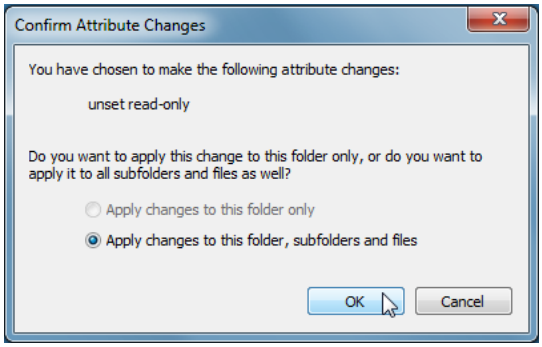
Please note that in the case the software is copied from a CD you may have to unselect the *"Read-only"* attribute for the destination folder.

1. This is done by selecting the directory and right-clicking on it. Select **Properties** from the context menu.



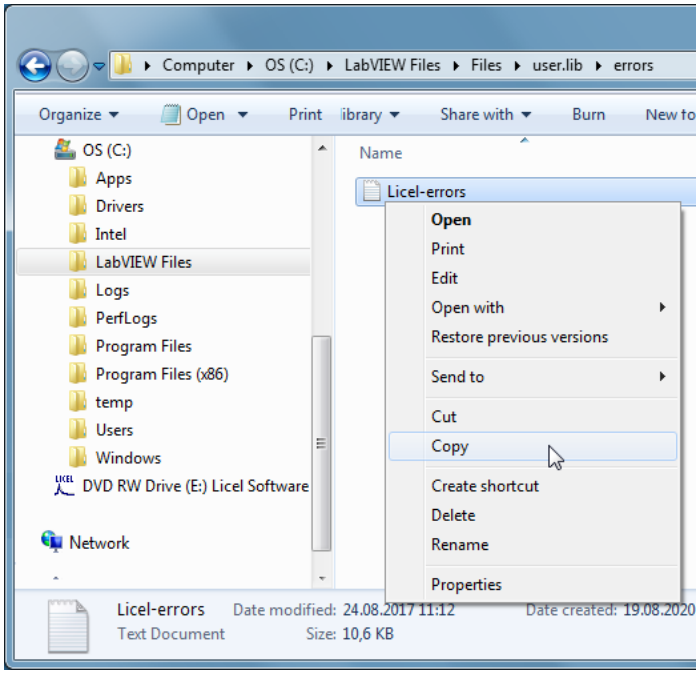


2. Verify that the `"Read-only"` attribute is not checked, uncheck it if necessary. Click *OK* and check in the next dialog *Apply changes to this folder, subfolders and files*. Leave the dialog by clicking *OK*

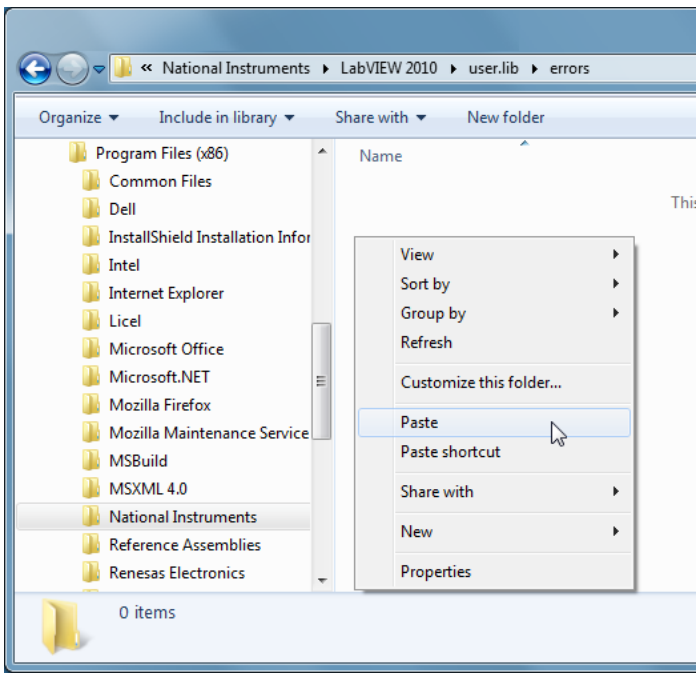


Licel provides one or more user-defined LabVIEW error code files. LabVIEW will use these files to generate hints in error messages. Before you will have to copy these error code files to an appropriate location where LabVIEW will find them. For this

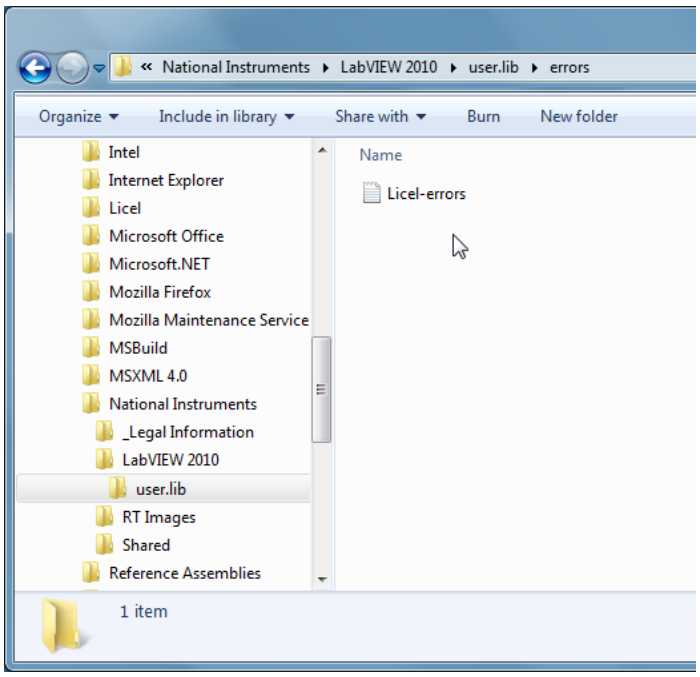
1. Locate the error code files in Licel's LabVIEW sources: they are located in the sub folder `<LabVIEW Files Folder>\Files\user.lib\errors`
2. Select all files `*-errors.txt` and copy them



3. Navigate to the folder
   `<LabVIEW installation directory>\user.lib\errors`,
   create the sub folder `errors` if necessary. Paste the copied file(s) to that directory

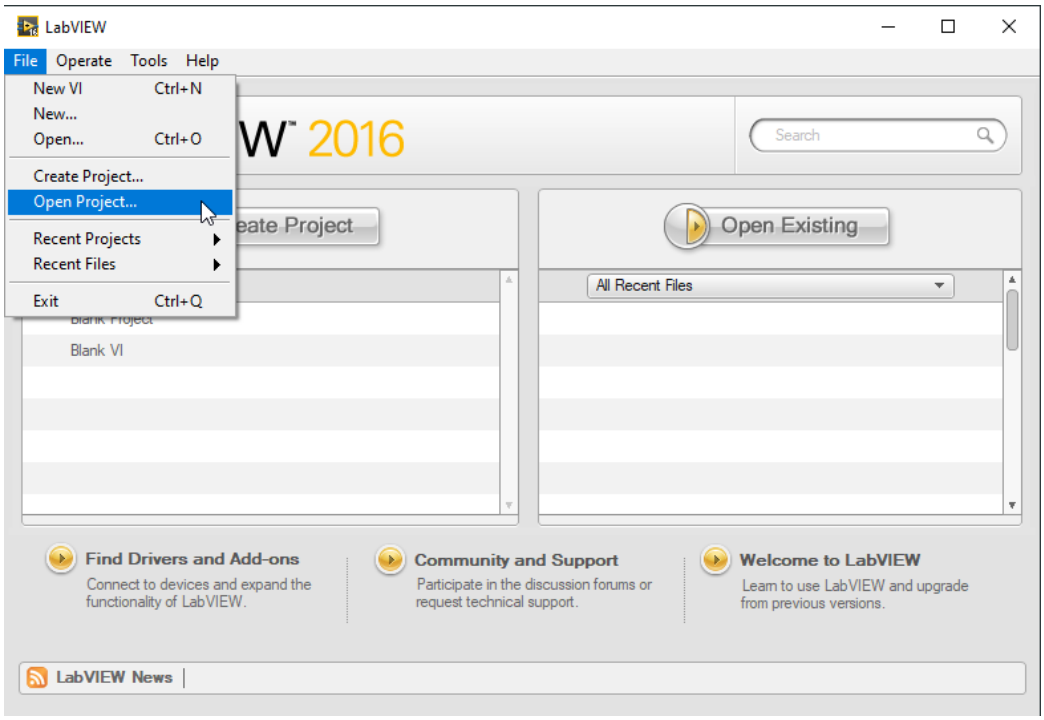


4. All copied LabVIEW error code files should be seen now:

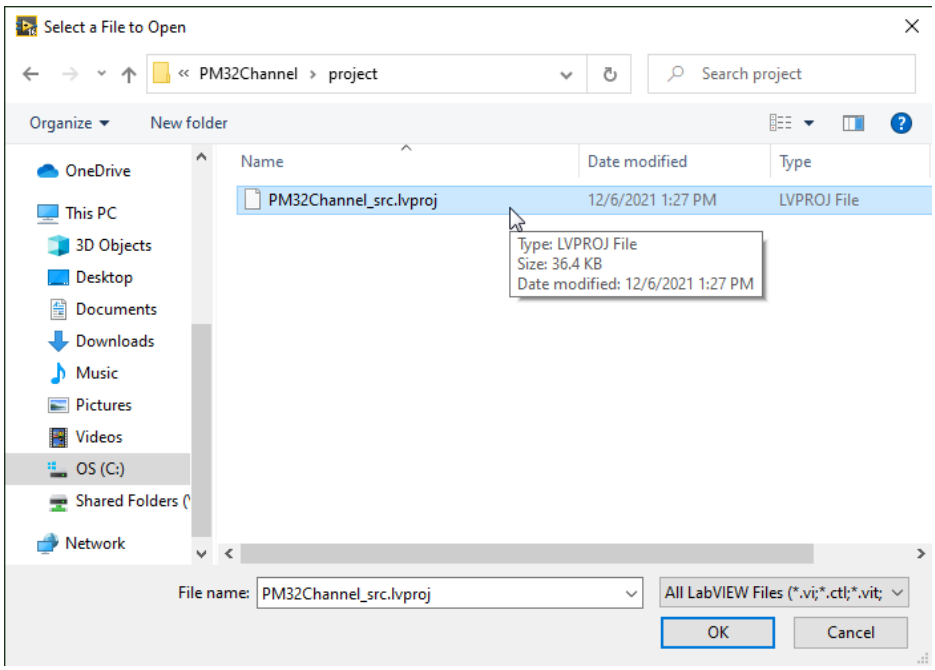


Now you should be able to run all the files. If you are still having problems, apply a mass compile to the directory where the software was extracted to:

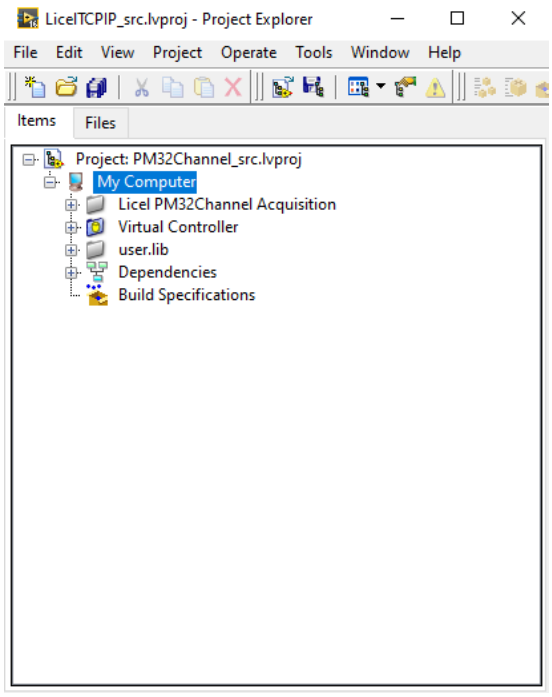
1. Start LabVIEW.
2. Select the menu entry *Open Project...* in the *File* menu



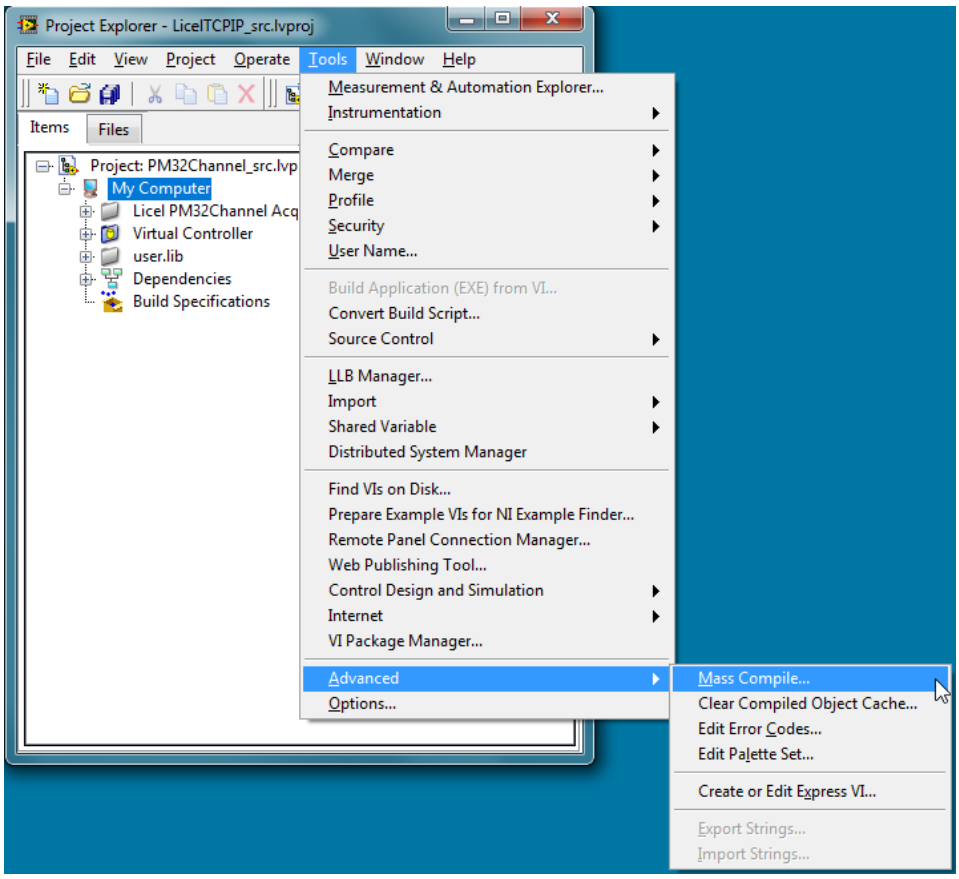
3. Navigate to the folder where you copied the Licel LabVIEW sources to



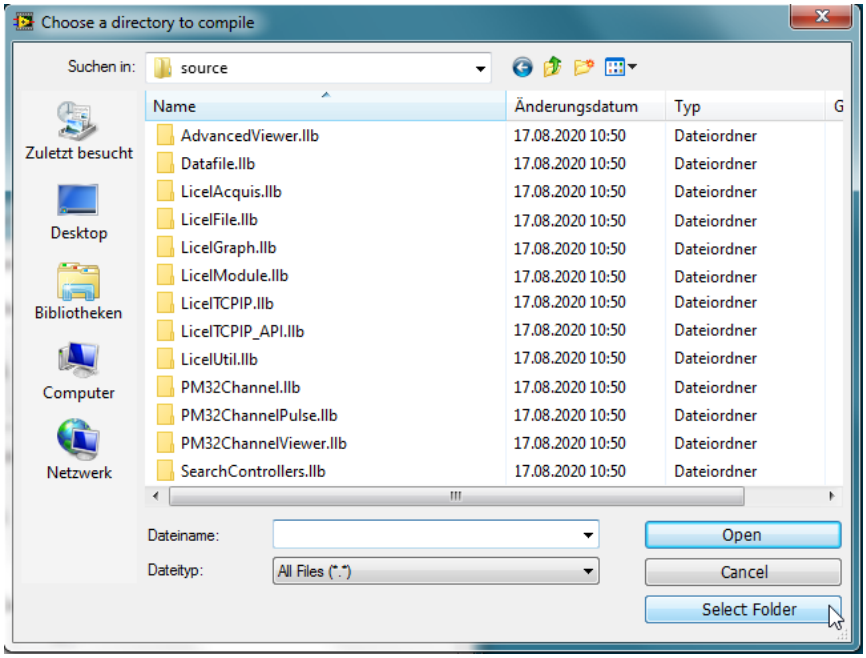
4. Open the project `LicelTCPIP_src.lvproj` in the subfolder `project`



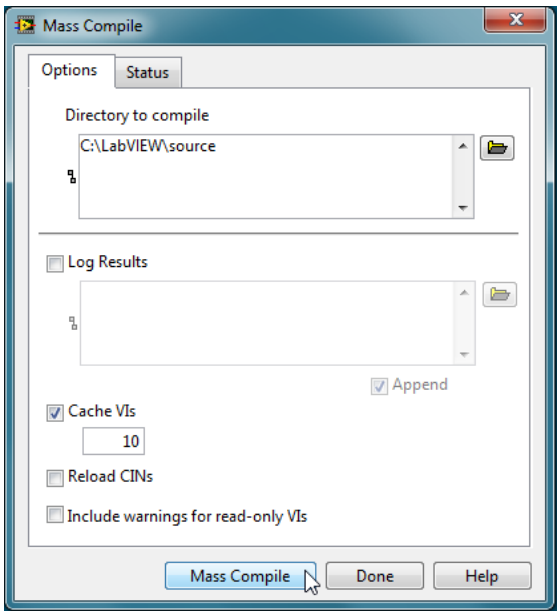
5. Select the menu *Tools*, then *Advanced*, and finally *Mass Compile...*.

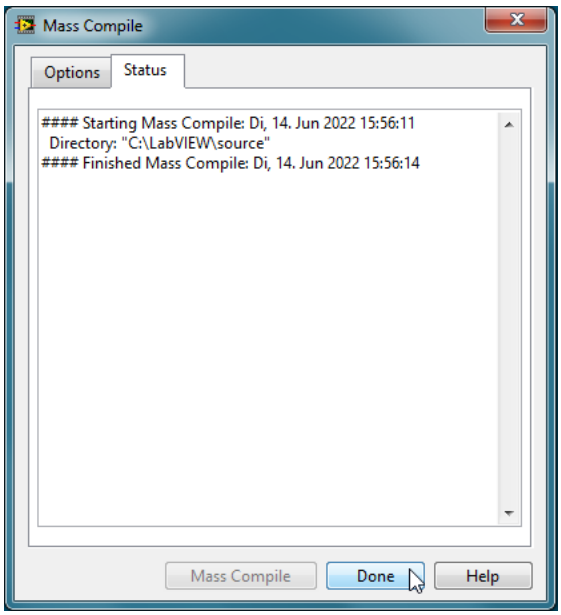


6. You will be asked to select a folder, select the **source** folder under the target directory of your LabVIEW files.



7. Press *Mass Compile* in the next dialog.





8. Later the mass compile status will be shown. There should't be any problems.

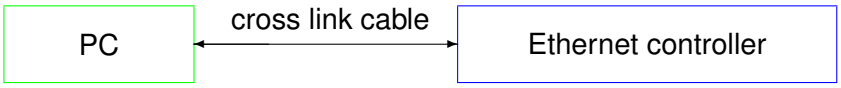
Please note that the removal of older libraries is a necessity, since LabVIEW often links to various libraries with the same name. As a result, if a library is installed twice, one can not be certain which library is actually being used.
If you still have any problems, please contact Licel for further assistance.

Chapter 4

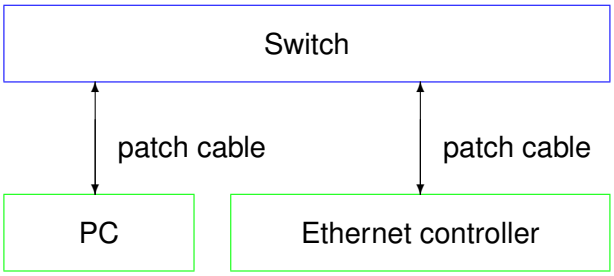
Setting up the Network

4.1 Network Introduction

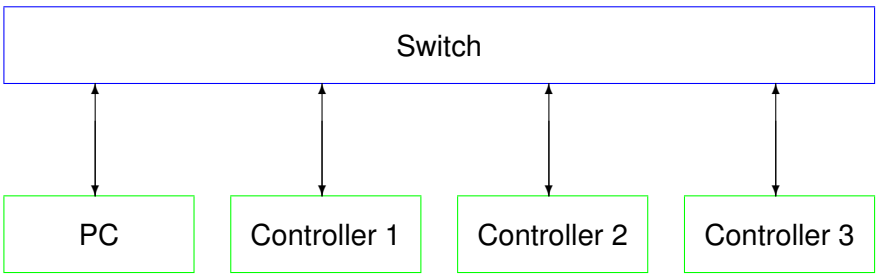
To control a Licel Ethernet controller a working TCPIP connection is required. This can be reached by two ways, using a cross link cable, which creates a one to one connection between the PC and the Ethernet Controller or with patch cables and a switch



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. Never use IP addresses beginning with `169.254.` because the corresponding IP address range `169.254.x.y` is reserved as the DHCP fallback range for network clients that try to connect to a DHCP server but do not find any in the local network.

4.2 Preparations

To operate the Licel Ethernet 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.

4.3 Network Information

The Licel Ethernet 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 (4.5.2) to enable the Licel Ethernet 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 gateway

should be set by yourself. **Please make sure that the IP address is unique in your network.** If you have more than Licel Ethernet controller make sure that they use different addresses. Refer to the subsection Fixed IP Address (4.5.1). The system ships with all Ethernet controllers set to the default address of `10.49.234.234`. In order to setup a system with multiple controllers one needs to do this procedure with each controller in sequence where only one controller is connected to the network at a time. Otherwise one would end up with multiple controllers sharing the same default address which would prevent a successful setup procedure. Never use a fixed IP address beginning with `169.254.` because the corresponding IP address range `169.254.x.y` is reserved as the DHCP fallback range for network clients that try to connect to a DHCP server but do not find any in the local network.

2. The default ports used by the Ethernet controller are 2055 and 2056. Can these ports be used? If you have more than one Licel Ethernet controller the addresses should be different but the ports can be identical for them.

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?

4. Is the default network mask `255.255.255.0` suitable for the communication bewtween the PC and the Licel Ethernet controller? A "'`255`"' at the first n positions of the network mask mean that the first n numbers of the IP addresses of both, the PC and the Licel Ethernet Controller, at the corresponding positions must be the same.

4.4 Network Preparation

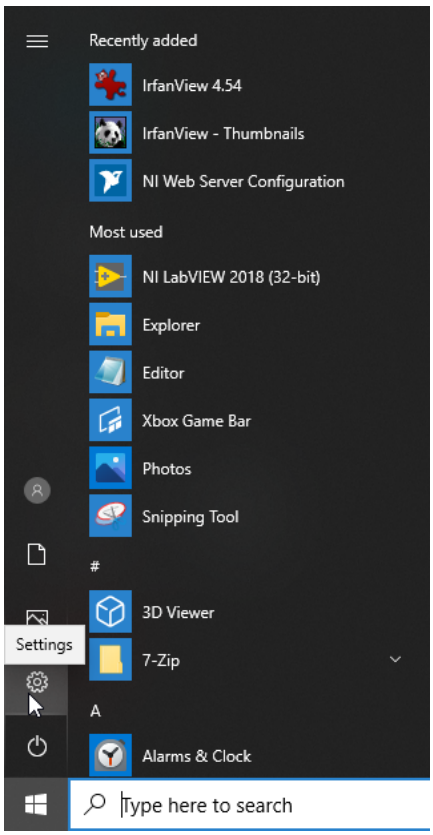
After having installed the Licel Windows applications or the Licel LabVIEW modules on your PC you are ready to change the network configuration parameters of the Licel Ethernet Controller according to the local network settings described in the previous section.

4.4.1 Establish the Connection

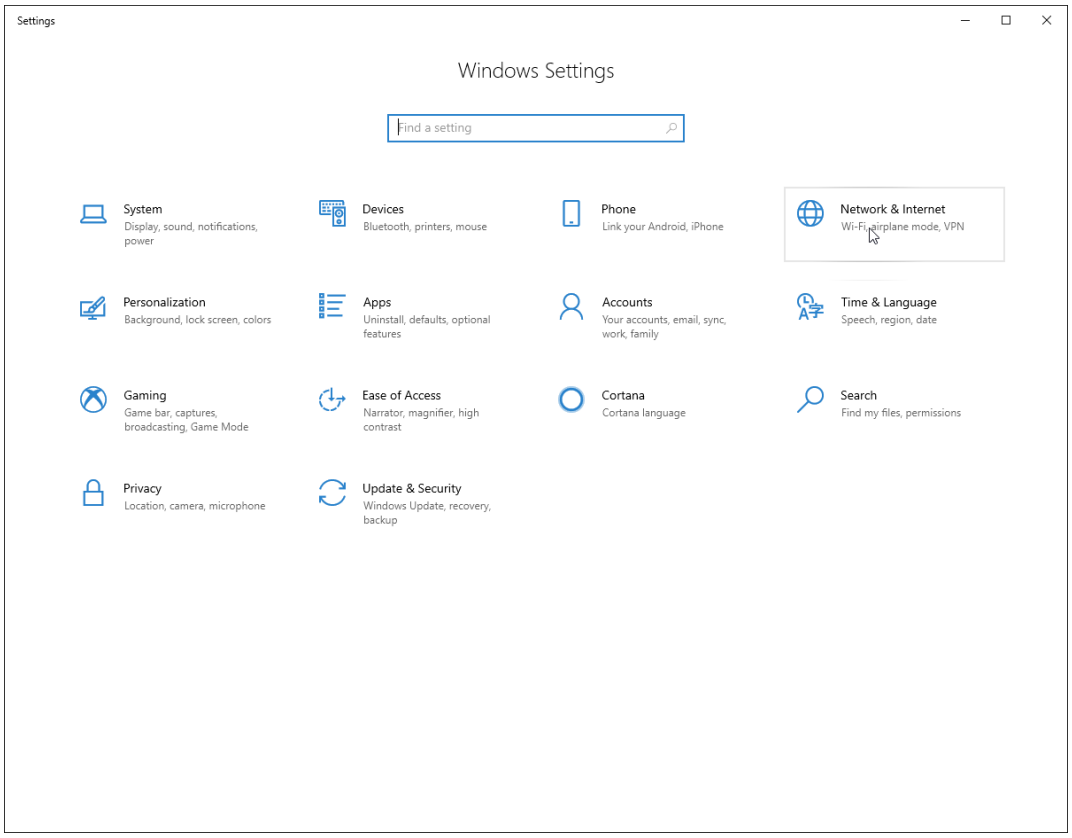
A straight-forward way to do this is the following procedure. You will need local administrator rights on your PC for the following steps:

1. 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:

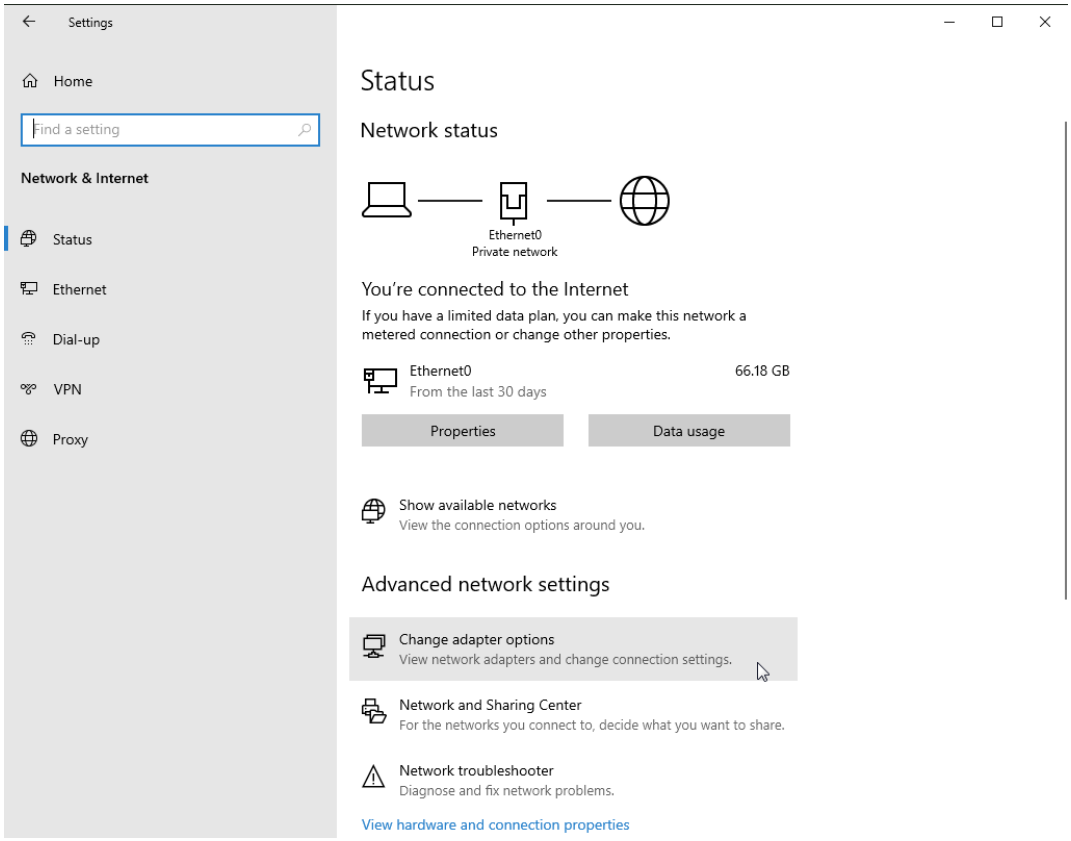
   (a) Click on the  button, and then on *Control Panel*.



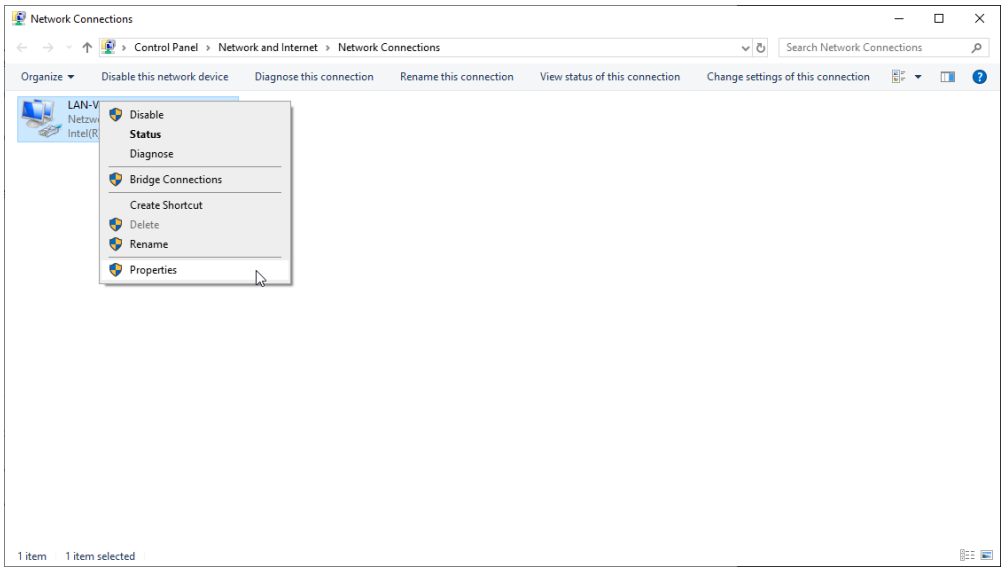
(b) Once the control panel has come up click on *Network and Internet*.



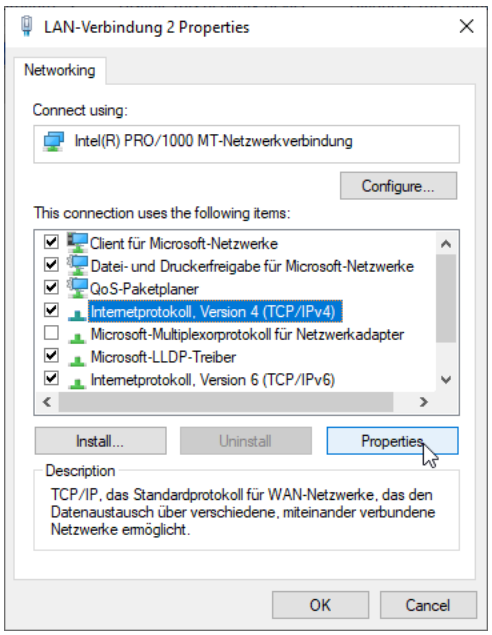
(c) In the next window click on *Change adapter options* in the *Advanced network settings* section.



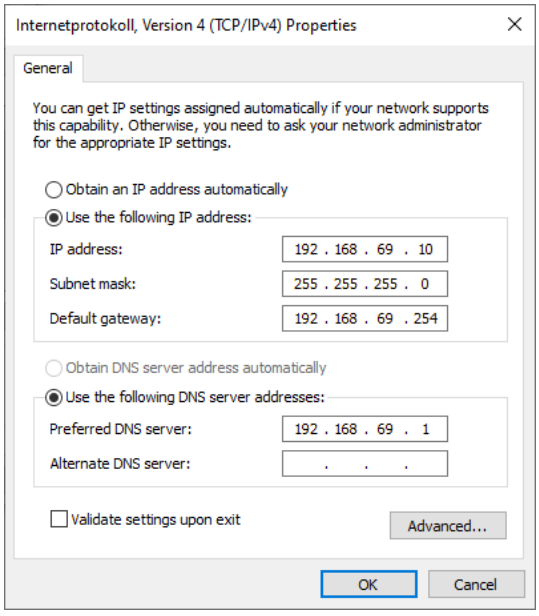
(d) The installed network connections will be shown, right-click on the local Ethernet connection to be used with the Licel Ethernet Controller and choose **Properties** from the context menu.



2. 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.



3. 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. Please note that the addresses and settings given here are examples only!



   (a) If *Obtain an IP address automatically* is active on the Ethernet adapter you will use for the communication with your Licel Ethernet controller you will work in a configuration where both, your PC and your Licel Ethernet controller will finally connect to your local network via a switch. In that case you should find out information about the PC's current IP address (range) to be able to assign a fixed IP address to the Licel Ethernet controller later:

      i. Open a command prompt window (DOS box).

      ii. 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. . . . . . . . . : 192.168.69.10
        Subnet Mask . . . . . . . . : 255.255.255.0
        Standard-Gateway. . . . . . :
```

iii. If the shown IP address begins with `169.254.` there is a network problem: your PC attemps to connect to a DHCP server but does not find any. Therefore, an IP address in the reserved DHCP fallback range `169.254.x.y` is assigned. Please fix your netwirk problem before you continue.

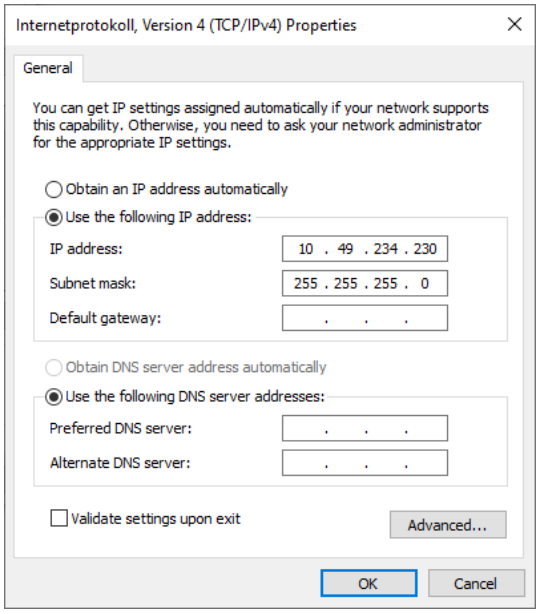
iv. If not remember the shown IP address (in the example: `192.168.69.10`).

(b) if *Obtain an IP address automatically* is not checked, note the *IP address*, the *Subnet mask*, and the *Default gateway*.

(c) Remember the checkbox *Obtain DNS server address automatically*, and

(d) if *Obtain DNS server address automatically* is not checked, note the *DNC server* addresses if available.

4. If activated disable DHCP (checkbox *Obtain an IP address automatically*) and manually assign an IP address within the default address range of the Licel Ethernet Controller. A good choice would be `10.49.234.230`. **Never use the default address (`10.49.234.234`) of the Licel Ethernet Controller as IP address for your PC.**



5. Quit the dialog by pressing *OK*.

6. Power up the rack with the Licel Ethernet Controller and connect the PC with the controller using the red **cross-link cable** shipped together with your hardware.

Now you should be able to access the Licel Ethernet Controller via your Ethernet adapter. Please test this first connection with the methods given in the next section.

4.4.2 Diagnostics

Please carry out the following steps to verify that the connection of the Licel Ethernet 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).

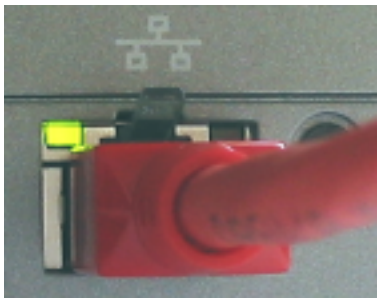
(b) Type `ipconfig` and press enter. At least one of the Ethernet adapters should show the address that you previously set (`10.49.234.230`). The response should be similar to the following:

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

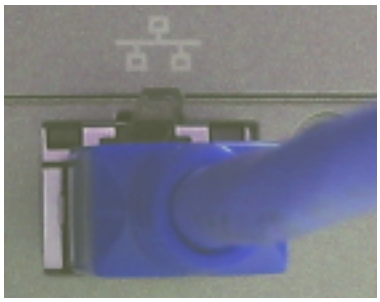
4. Verify that the Licel Ethernet Controller is accessible via the network now:

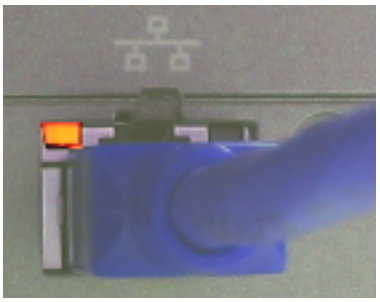
   (a) Open a command prompt window (DOS box) or use the one from above.

   (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.

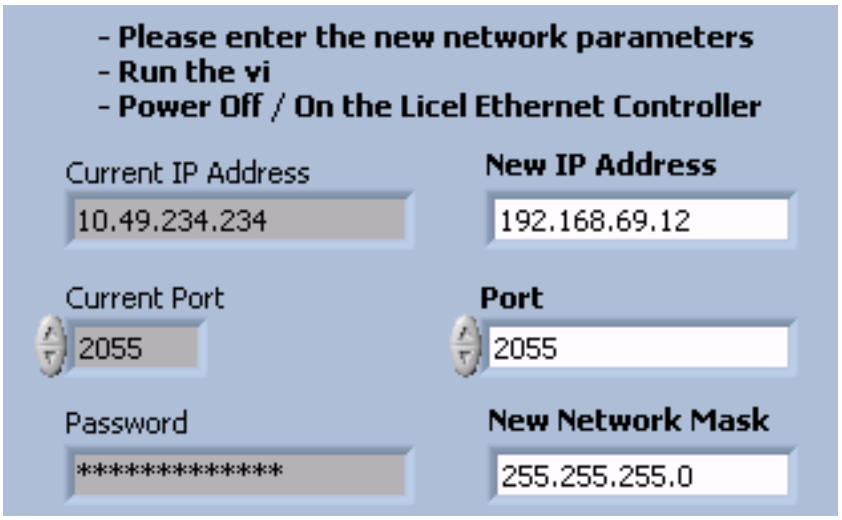
4.5 Network Setup

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.

4.5.1 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.



2. Set the desired network parameters in the fields *New IP Address* and *Port*. In this eaxample we set the Licel Ethernet controller's IP address to `192.168.69.12` because this IP address lies in the same IP address range where we found our PC (in our example `192.168.69.10`). Never use a fixed IP address beginning with `169.254.` because the corresponding IP address range `169.254.x.y` is reserved as the DHCP fallback range for network clients that try to connect to a DHCP server but do not find any in the local network.

3. Check the *New Network Mask*: is the default network mask `255.255.255.0` suitable for the communication bewtween the PC and the Licel Ethernet controller? A "`255`" at the first n positions of the network mask mean that the first n numbers of the IP addresses of both, the PC and the Licel Ethernet Controller, at the corresponding positions must be the same.

4. Do not forget to enter the correct administrator *Password*.

5. Run the vi by pressing the start  button. It should finish without opening an error message dialog.

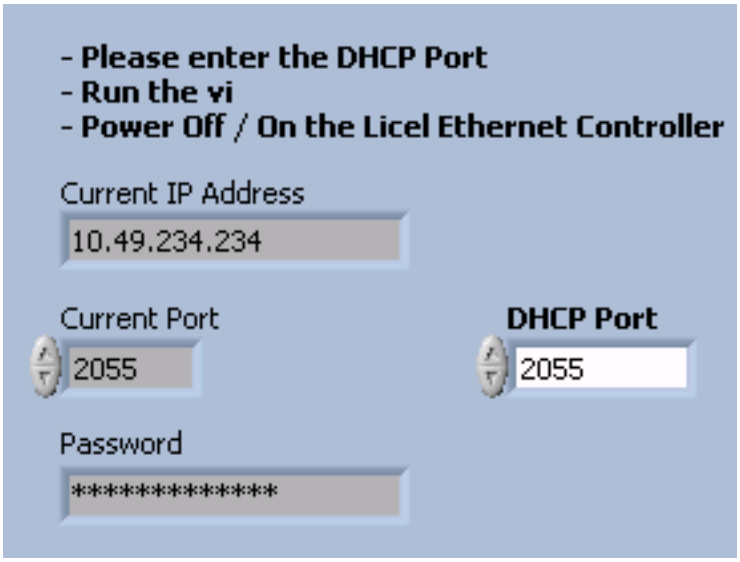
6. Turn the Licel Ethernet Controller off and switch it on again. Wait **approximately 20 – 30 seconds**.

7. A `ping 10.49.234.234` executed from a command prompt (DOS box) should now time-out.

4.5.2 DHCP Mode

In order to configure the Licel Ethernet 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.

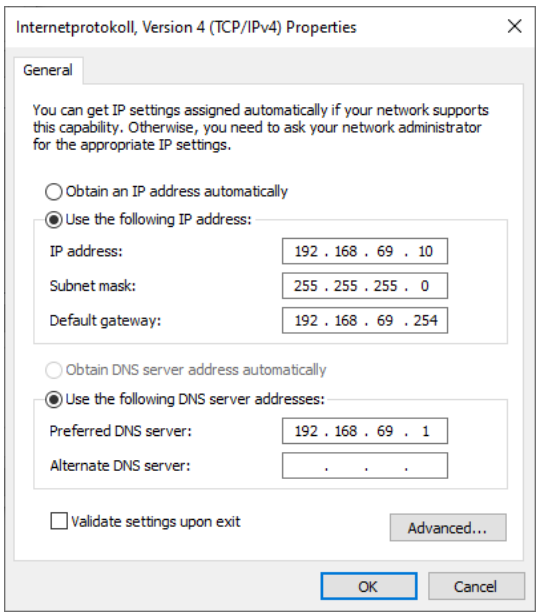


2. Set the desired *DHCP Port* number.
3. Do not forget to enter the administrator administrator *Password*.
4. Run the vi by pressing the start button. It should finish without opening an error message dialog.
5. Turn the Licel Ethernet Controller off and switch it on again. Wait **approximately 20 – 30 seconds**.
6. A `ping 10.49.234.234` executed from a command prompt (DOS box) should now time-out.

4.6 Reconfigure the PC

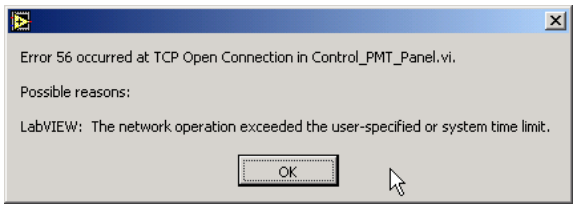
After you successfully configured the Licel Ethernet Controller the following last steps have to be carried out to reconfigure your PC for the local network and to test the connection to the Licel Ethernet Controller:

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 Licel Ethernet Controller.

5. Quit the dialog by pressing *OK*.
6. Reboot your PC.
7. Connect the Licel Ethernet 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 Licel Ethernet Controller. If the Ethernet controller is in DHCP mode, 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 TCPIP Getting Started.vi` or the corresponding Windows application to be started from the Windows start menu.
10. A TCP/IP timeout error with LabVIEW's error code 56 may be caused by a wrong IP address.

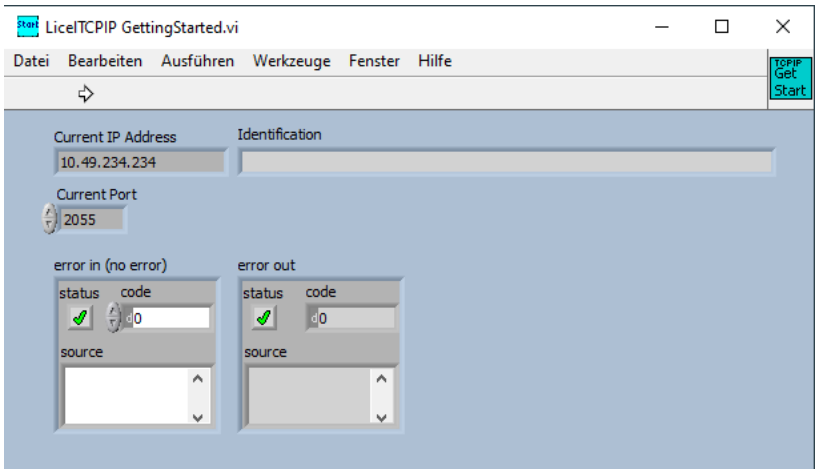


Please check carefully that the values for *IP Address* and *Port* match with the parameters set at the Licel Ethernet Controller. Set the correct values as defaults for future operation. Other reasons for errors with code 56 are non-existing connections (check if the cable in use is correct) or unstable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

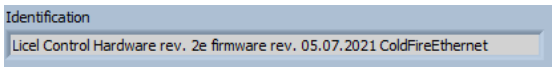
4.7 Test the TCP/IP Setup – Getting Started

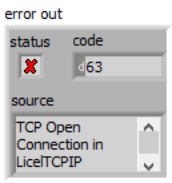
Once you assigned an IP address to your Licel Ethernet Controller, turned it off and switched it on again, and reconfigured the PC's IP address you may like to test whether or not a TCP/IP connection is possible from your PC to the Licel Ethernet Controller using the new TCP/IP settings.
For this you could start the Windows Application *TCPIP Getting Started* from the Windows start menu. If you are using the LabVIEW sources open the corresponding VI

`LicelTCPIP GettingStarted.vi` from the LabVIEW project in the subfolder `Licel PM32 Acquisition\TCPIP`.
The front panel of *Getting Started* will open.



Please continue as follows:

1. Enter the *Current IP address* and *Port* as you have set them for the Licel Ethernet Controller in one of the previous sections.
2. Run the program using the run button 
3. The program will attempt to open a TCP/IP connection to the Licel TCPIP controller and request the identification string using a low level TCPIP command.
4. The *Identification* field should hold the controller's identification string now.
   

5. In the case of an error *error out* will show up with an error mark, an error *code* and an error *source*, here an example is shown:
   

   .

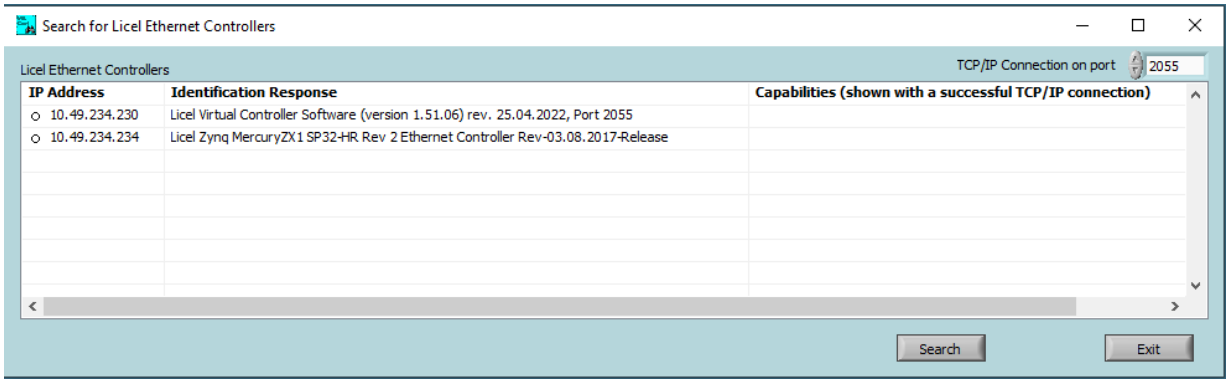
4.8 Search Controllers

Licel provides the utility *Search Controllers* to search for Licel Ethernet Controllers in your Local Network.

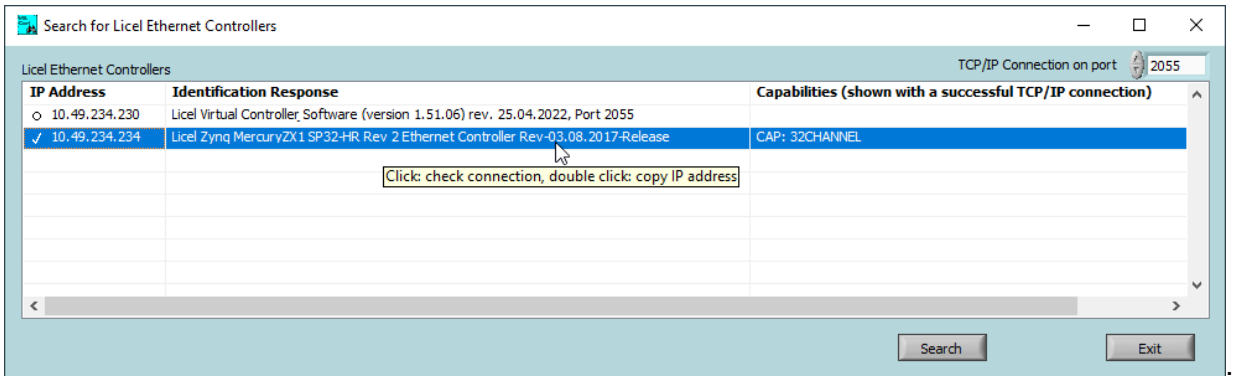
- You can start the Windows Application *Search Controllers* from the Windows start menu. Then the program opens and immediately starts to run.
- If you are using the LabVIEW sources open the corresponding VI `Search Controllers.vi` from the LabVIEW project in the subfolder `Licel PM32 Acquisition\TCPIP`. Run the VI by using LabVIEW's run button  .

Search Controllers uses UDP polls to find Licel Ethernet Controllers in your network. Your firewall might ask for your allowance to do that. Use *Search Controllers* to make sure that all your controllers

have been set to the correct IP addresses or to identify a specific controller e.g. after you have set it to the DHCP mode. After starting *Search Controllers* will display the found controllers in a table. Please click on the Search button if the table remains empty or if you would like to repeat a search because you have switched some controllers on or off.



In the example we see one controller at the IP address 10.49.234.234, and additionally we see a *Virtual Controller* running on the PC's IP address (10.49.234.230 in this example). On your system you will see the IP addresses you assigned to your Licel Ethernet Controllers.
At the left of each line you see a circle symbol which indicates that no deatils are known about the corresponding controller at the moment ○.
Now click on one line representing one of the controllers. *Search Controllers* will attemps to open a TCP/IP using the port top right TCP/IP Connection on port 2055. If the TCP/IP connection could be successfully opened a checkmark ☑ will appear on the left. Furthermore *Search Controllers* will requst the controller's capabilities using a low level TCP/IP command. In the following example you will recognize that the controller at the IP address 10.49.234.234 has the capability *32CHANNEL* i.e. it is a Multispectral Lidar Detector



If the capability field is empty please check that no other software is communicating with the selected controller. If a connection is not successful (e.g. because the controller has been switched off in the meantime or the port is wrong) an appropriate sign is displayed on the left ⊘.
A double click on a line representing a controller will copy it's IP address to the clipboard.

4.9 TCP/IP Connection Parameters (Software)

To work properly with the Licel Ethernet 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 Licel Ethernet Controller following the network setup section.

If a connection to the controller using the *IP Address* and *Port* in the corresponding control fields is not successful the applications will continiue to try to connect to the controller. The user may change the *IP Address* and *Port* during these reconnection attempts.
Defining the *IP Address* and *Port* is different for the Windows applications and the LabVIEW sources.

Windows Applications: Initialization Files

The Windows applications communicating with the Licel Ethernet Controller read their TCP/IP parameters *IP Address* and *Port* from initialization files.
An example for an initialization file holding the TCP/IP information is given below:

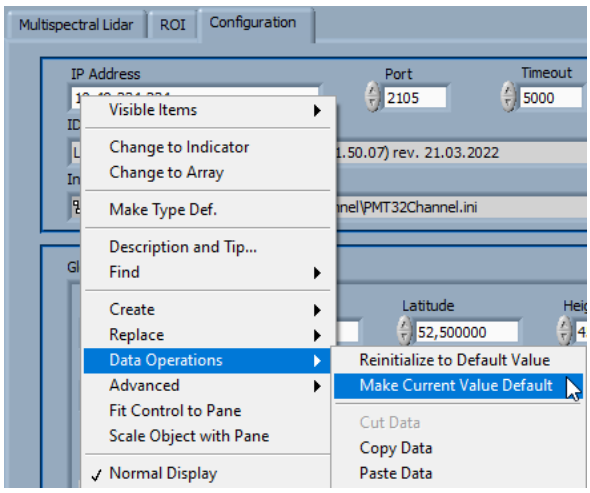
```
[TCPIP]
UseValues=TRUE
IPAddress=10.49.234.234
Port=2055
```

You may directly edit the corresponding initialization file using a text editor like `Notepad`. You must change the values for the *IP Address* and *Port* to the values you will set following the Instructions in the network setup section. Or change the *IP Address* and *Port* while the application is running and not yet connected, the application will try to build up a connection until it has success.
When the applications is able to open a TCP/IP connection with the given *IP Address* and *Port* these values will be written to the initialization file. The Windows apllication will use these values at the next start of the program.
The applications *PM2 Channel Acquisition* and PM32 Pulde Height Distribution use the same initialization file `PMT32Channel.ini`.

LabVIEW: Setting Default TCP/IP Parameters

The LabVIEW VIs will not read the `IPAddress` and `Port` from the initialization file as the Windows applications do. However, when opening a LabVIEW VI within a LabVIEW development environment, default values can be defined for controls on the panel of a LabVIEW VI. This is especially convenient and recommended for the TCP/IP parameters **IP Address** and *Port*. Change the values to the values you set following the Instructions in the network setup section.

1. Open the vi using LabVIEW, do not run the vi.
2. Enter the value for the IP address into the control named *IP Address*.
3. Right-click on the control *IP Address* $\longrightarrow$ the context menu opens.
4. Select **Data Operations** $\longrightarrow$ a sub menu opens.
5. Select **Make Current Value Default**.

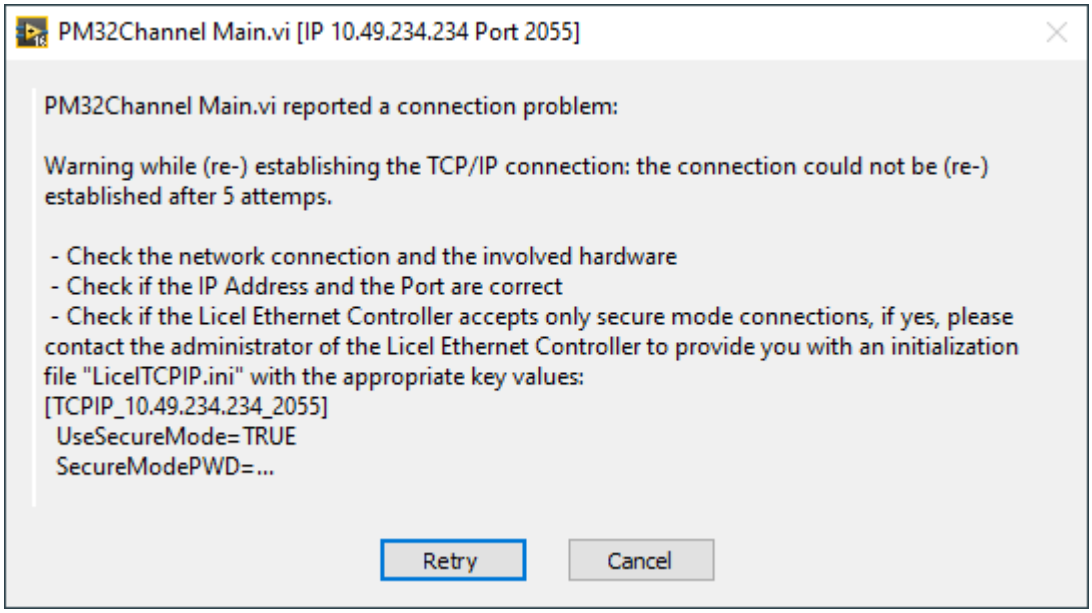


6. Repeat this procedure for *Port*.
7. Save the vi.

Although the LabVIEW VIs do not read the *IP Address* and *Port* from the initialization file they will save these values to the file for documentation whenever a TCP/IP connection could successfully be opened.

4.9.1 TCP/IP Connection Problems (Software)

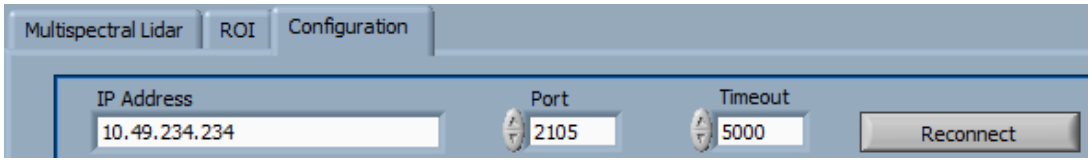
Both, *PM2 Channel Acquisition* and PM32 Pulde Height Distribution have a built-in mechanism to re-establish the TCP/IP connection to the Licel Ethernet Controller when the connection is lost or when the connection is not successful after the program start.
If the reconnection mechanism is not successful after 5 attempts the software assumes that some basic TCP/IP settings may be incorrect. Therefore the following error message is displayed:



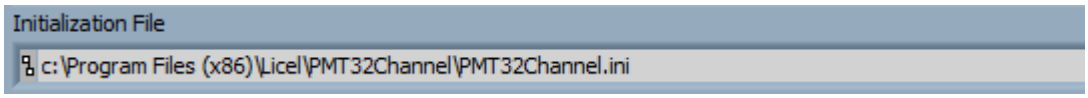
In the case that this dialog comes up please

- check the network connection and the involved hardware. Check whether the Licel Ethernet Controller and all other Licel hardware 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 Licel Ethernet 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 Licel Ethernet Controller
   - Using the LabVIEW vi, just enter the required values on the *TCP/IP* page and save them as defaults.



     If the warning dialog is not closed by a user (Retry or Cancel) it will close automatically and the program will continue to attempt to connect to the Licel Ethernet Controller using the given *IP Address* and *Port*.
   - If you are running a Windows application you should check the IP address and port in the corresponding initialization file. You will see the full path of the file in a file path indicator on the *TCP/IP* page.



     While a Licel Windows application **is running** (and has not yet a TCP/IP connection) you may enter the *IP Address* and the *Port* directly. If a connection can be established (i.e. the values are correct) the parameters will be written to the appropriate initialization file directly after successfully establishing a TCP/IP connection.

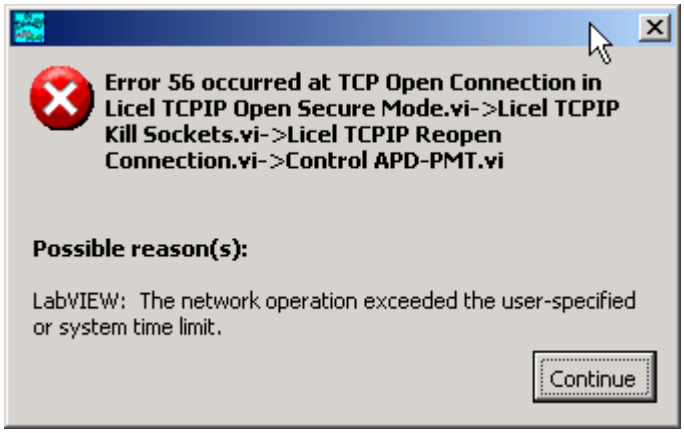
- check whether the Licel Ethernet Controller operates in secure mode. If secure mode is used please check the following section in the file `LicelTCPIP.ini`:

```
[TCPIP_<Controller-IP-Address>_<Controller-Port>]
UseSecureMode=TRUE
SecureModePWD=<SecureModePassword>
```

  where `Controller-IP-Address` and `Controller-Port` are the IP address and port of the Licel Ethernet Controller, respectively. If necessary, ask your administrator for the correct password for usage in secure mode.

You have the following choices to continue when the warning dialog appears:

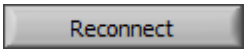
1. Click *Retry* to continue to reconnect to the Licel Ethernet Controller.
2. Click *Cancel* to exit. The program will display an error message (here an example for *Control APD-PMT*, the dialog's appearance may depend on the LabVIEW version):



3. Do nothing – the application will automatically close the warning dialog and try again to connect to the TCP/IP controller with the current *IP Address* and *Port*.

4.9.2 Change the Ethernet Controller (in the Application)

If you have more than one Licel Ethernet Controllers (e.g. 2 Multispectral Lidar Detectors) and recognize that you accidentially connected to the wrong controller you may easily change the controller:

1. Enter the *IP Address* and *Port* of that controller you really would like to connect to.
2. Press the 

button.

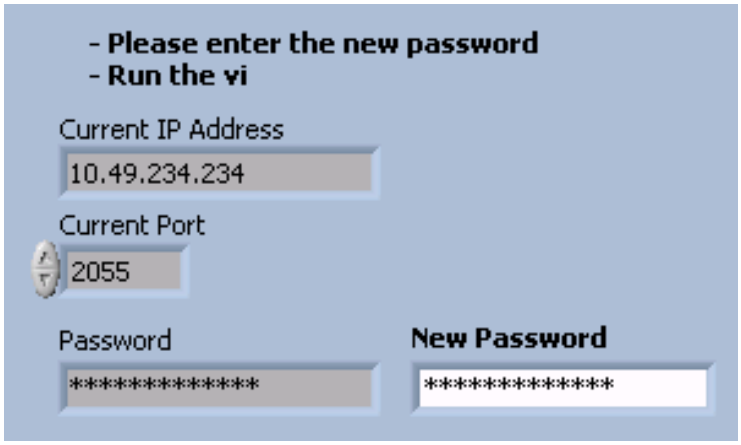
The application will then close the open TCP/IP connection and reconnect with the new *IP Address* and *Port*.

4.10 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.

4.10.1 Changing the Administrator Password

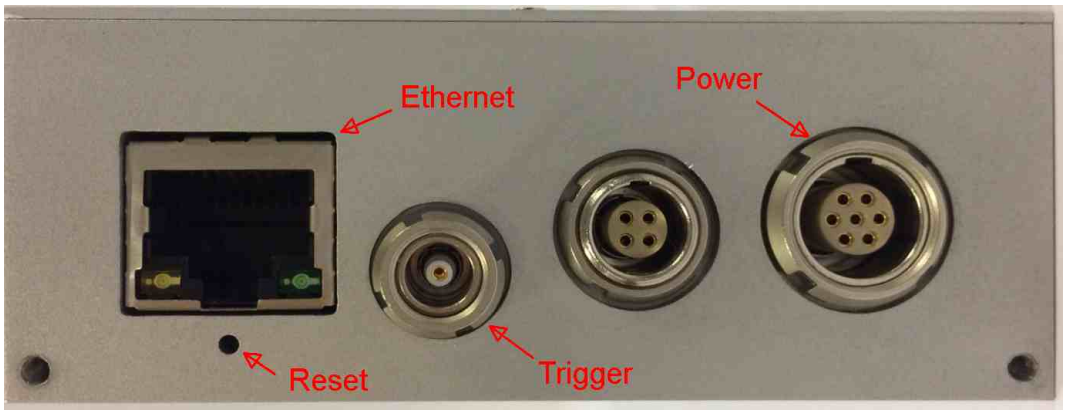
The Licel Ethernet Controller is shipped with the default administrator password "'*Administrator*"'. In order to change this password which grants administrative access to the controller, please carry out the following steps:

1. Open `Licel TCPIP 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.

4.11 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 5 seconds after switching the unit on, wait for 45 seconds.

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 5

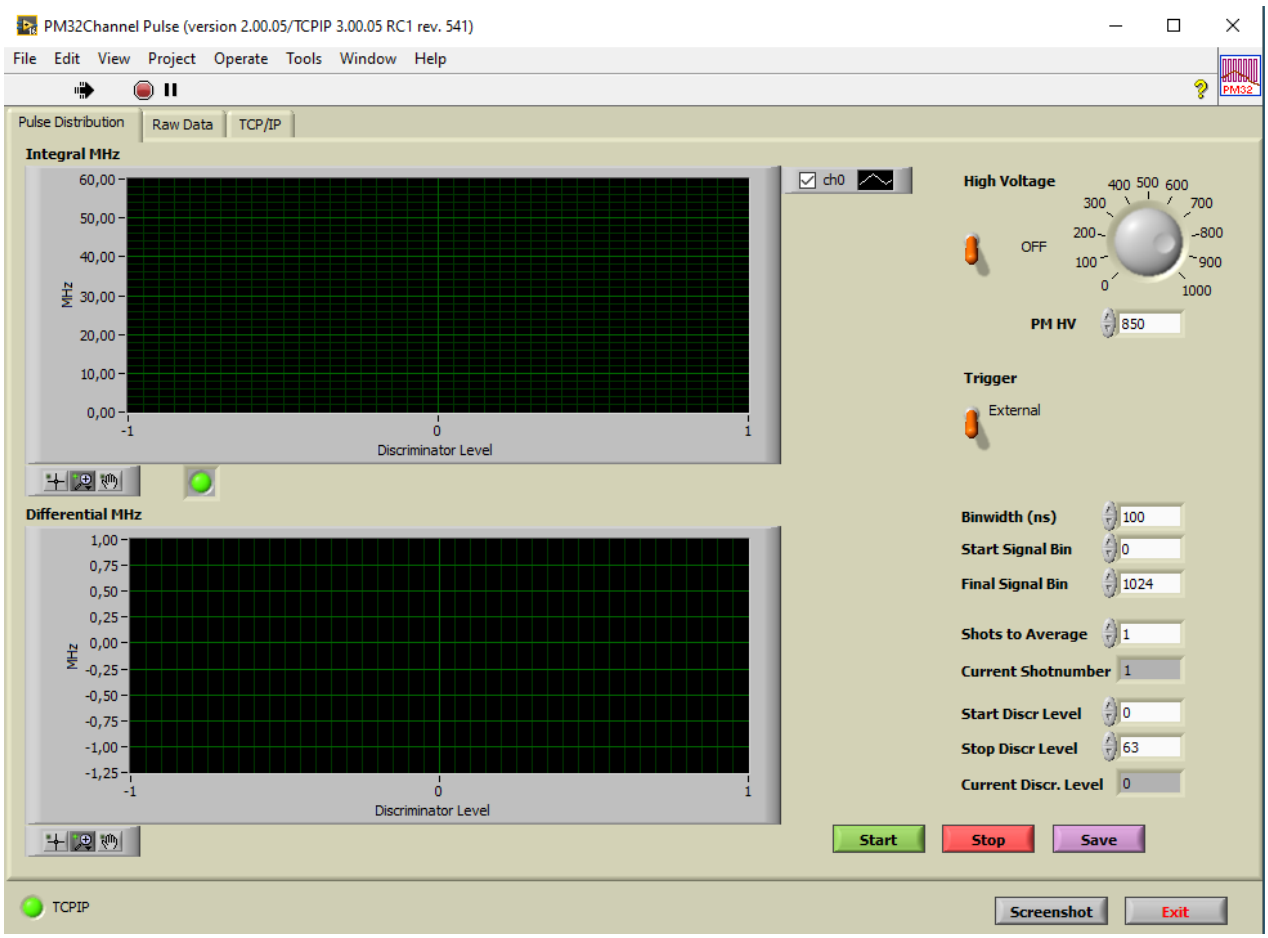
Software Modules

5.1 Overview

This chapter describes how to use the data acquisition software as well as the functions of the individual controls and indicators. In order to work through the information in this tutorial, the hardware and network setup must be completed. The next sections give brief introductions to the software module *PM32 Pulse Height Distribution*, *PM32 Channel Acquisition*, and *PM32 Viewer*.

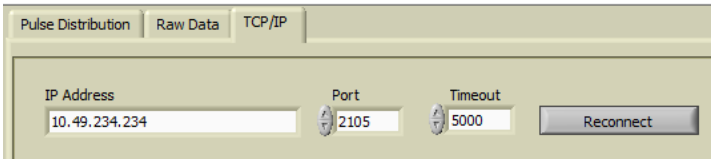
5.2 PM32 Pulse Height Distribution

With the *PM32 Pulse Height Distribution* utility you are able to acquire pulse height distributions for either all or selected channels of the Licel Multispectral Lidar Detector.
If you are working with the LabVIEW sources you can load the *PM32 Height Distribution* utility by doubly clicking on `PMT32Channel Pulse.vi` in the LabVIEW project. If you installed the Windows applications please start the program by selecting the entry *PM32 Pulse Height* in the *Licel PM32Channel* section of the Windows Start menu. After doing so, you should see a screen similar to the one below.

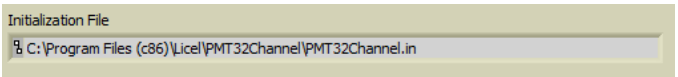


1. Before you start please enter the correct values for the *IP Address* and *Port*. You should already have set these values for the Licel Multispectral Lidar Detector.

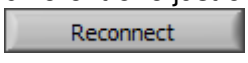
   - Using the LabVIEW vi, just enter the required values on the *TCP/IP* page and save them as defaults.



   - If you run the Windows application you may enter the values while the program is running. They will be written to the initialization file `PMT32Channel.ini` as soon as the program has successfully established a TCP/IP connection to the controller. You will see the full path of the file in a file path indicator on the *TCP/IP* page. Of course you are free to edit the initialization file before starting the program and enter the values for the *IP Address* and *Port* there.



   - If you would like to disconnect from the current controller and to connect the software to a different one just change the values of the *IP Address* and *Port* and click the



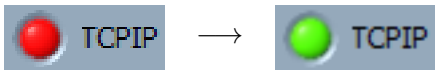
   button.

2. Please switch the Licel Multispectral Detector on and connect a trigger to the trigger input.

3. To start the program press the **Run** button at the top left of the screen.

The Windows application will start automatically when called for the first time.

4. After a short time the *TCPIP* indicator should change its color from red to green indicating a successful connection with the Licel Multispectral Lidar Detector. If the indicator remains red and/or an error is indicated, please check the values for *IP Address* and *Port*, change them on the program's panel if necessary. Check if the Licel Multispectral Lidar Detector is running and that all network connections are correct.



To record pulse height distributions please follow the steps below:

1. Make sure that your TCP/IP connection is alive,
2. Enter the desired voltage and switch the *High Voltage* on. .
3. Select the *Start Signal Bin*, *Final Signal Bin*, and the *Shots to Average*.
4. Select the *Start Discr*iminator *Level* and the *Stop Discr*iminator *Level*.
5. Click on the 

button. The Licel Multispectral Lidar Detector will acquire the desired number of shots for each discriminator level between the *Start Discr*iminator *Level* and the *Stop Discr*iminator *Level*. The acquired data of the active channels is averaged between *Start Signal Bin* and *Final Signal Bin*. The result is displayed in the graphic plots *Integral MHz* and *Differential MHz*. The most bottom display is intended to show the *Acquired Raw Data* for the active channels.
6. Click on the 

button to stop a running pulse height distribution.
7. By clicking the 

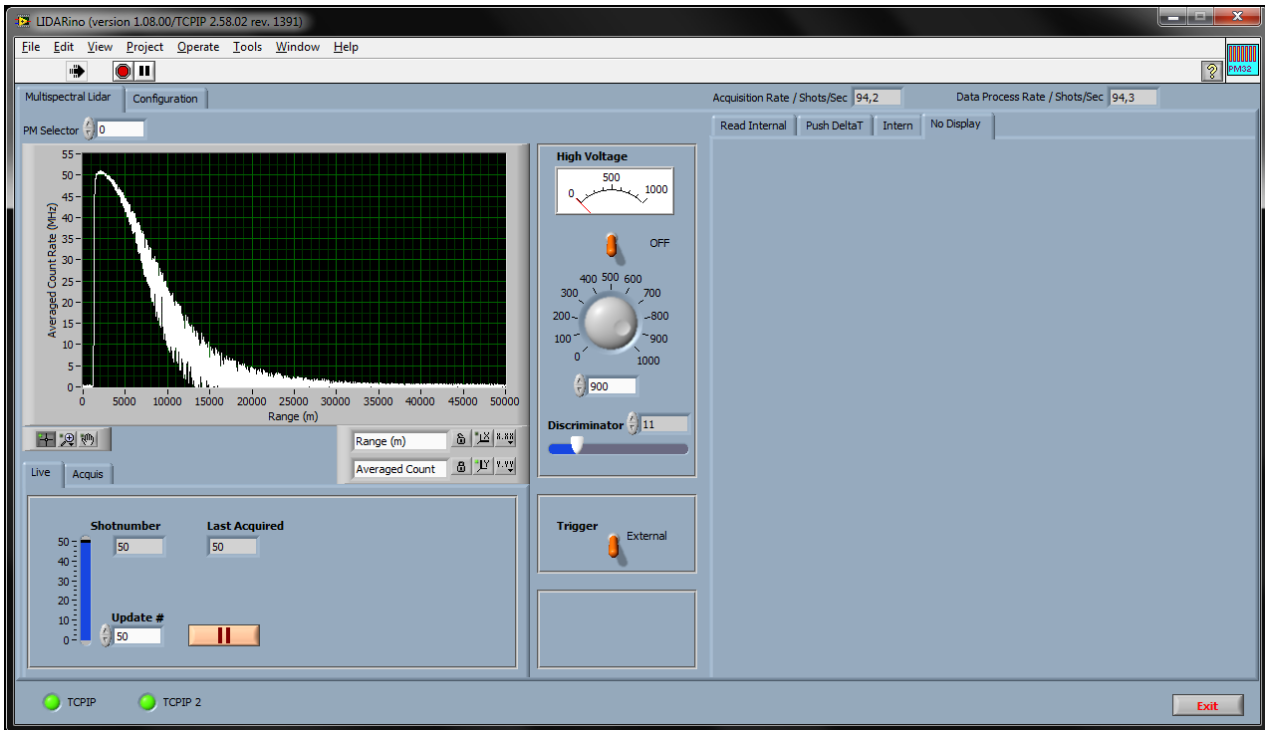
button and choosing a file name in a subsequent dialog the *Inegral MHz* data is saved to a text (ascii) file.
8. Clicking the 

button and choosing a file name in a subsequent dialog will save the current panel to a portable network graphics (png) file.
9. Clicking on 

or ✕ at the top right of the window will stop the application's execution. When run as a Windows application the front panel window will close. In the LabVIEW development system the window will remain open and you may restart the program using LabVIEW's run button  .

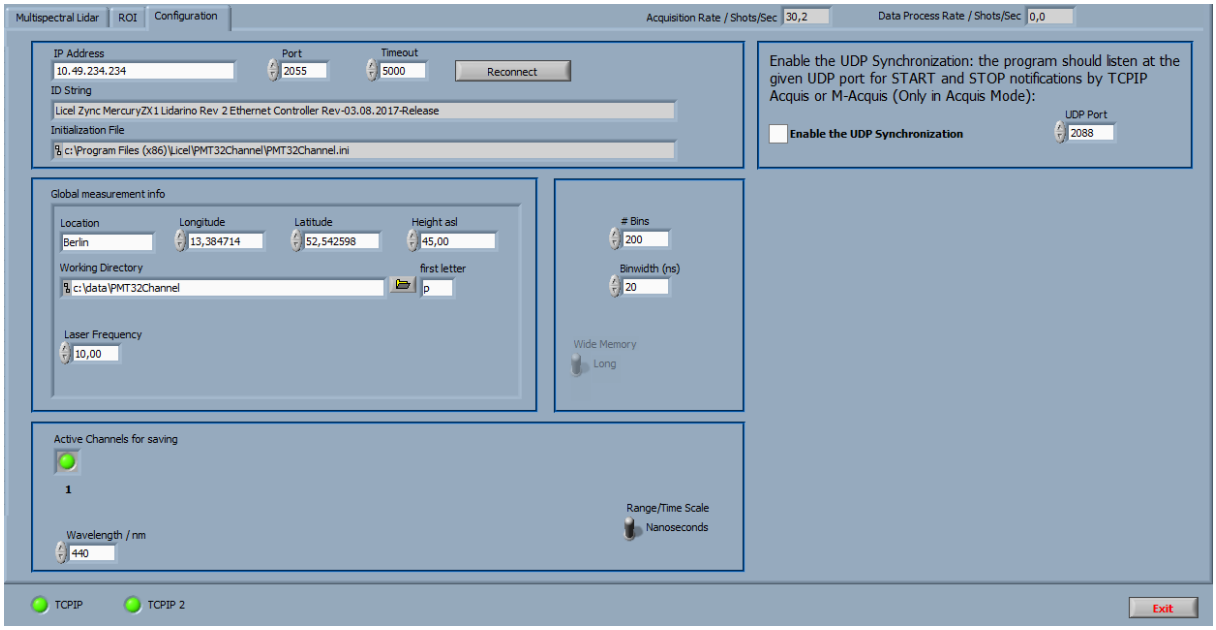
5.3 PM32 Channel Acquisition

You can load *PM32 Channel Acquisition* by either double clicking on the `PMT32Channel.vi` in the LabVIEW project. If you installed the Windows applications please start the program by selecting the entry *PM32 Channel Acquisition* in the *Licel PM32Channel* section of the Windows Start menu. After doing so, you should see a screen similar to the one below.



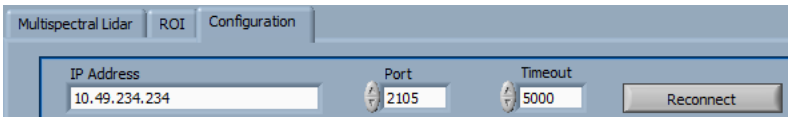
1. The program's front panel is structured using three pages, *Multispectral Lidar*, *ROI*, and *Configuration*. General settings are defined on the page *Configuration*. The acquisition settings and the main data display are located on *Multispectral Lidar*. The page *ROI* enables to inspect the 32 channel spectrum in a defineable range.

2. TCP/IP connection parameters, global parameters, and basic acquisition settings are defined on the on the *Configuration* page.

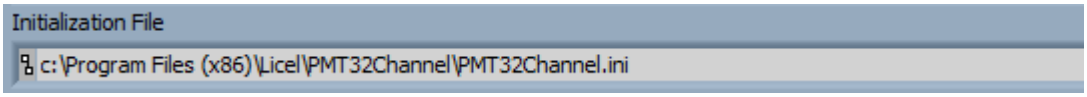


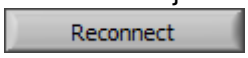
3. Before you start please enter the correct values for the *IP Address* and *Port*. You should already have set these values for the Licel Multispectral Lidar Detector.

   - Using the LabVIEW vi, just enter the required values on the *TCP/IP* page and save them as defaults.



   - If you run the Windows application you may enter the values while the program is running. They will be written to the initialization file `PMT32Channel.ini` as soon as the program has successfully established a TCP/IP connection to the controller. You will see the full path of the file in a file path indicator on the *TCP/IP* page. Of course you are free to edit the initialization file before starting the program and enter the values for the *IP Address* and *Port* there.

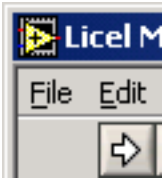


   - If you would like to disconnect from the current controller and to connect the software to a different one just change the values of the *IP Address* and *Port* and click the 

button.

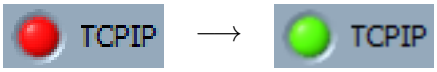
4. Please switch the Licel Multispectral Detector on and connect a trigger to the trigger input.

5. To start the program press the **Run** button at the top left of the screen.



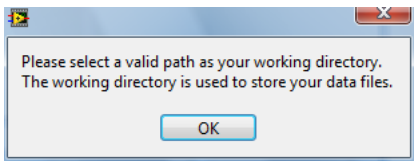
   The Windows application will start automatically when called for the first time.

6. After a short time the *TCPIP* indicator should change its color from red to green indicating a successful connection with the Licel Multispectral Lidar Detector. If the indicator remains red and/or an error is indicated, please check the values for *IP Address* and *Port*, change them (on the program's panel or in the initialization file) if necessary. Check if the Licel Multispectral Lidar Detector is running and that all network connections are correct.



7. If the controller sends the data via a second TCP/IP connection a second *TCPIP* indicator will be visible.

If you run the program for the first time it is recommended to verify and set some parameters. If the storage path for the data files found in the initialization file is invalid a message will inform you:



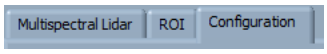
Please refer to the next subsection to set the working directory and the other parameters.

Clicking on 

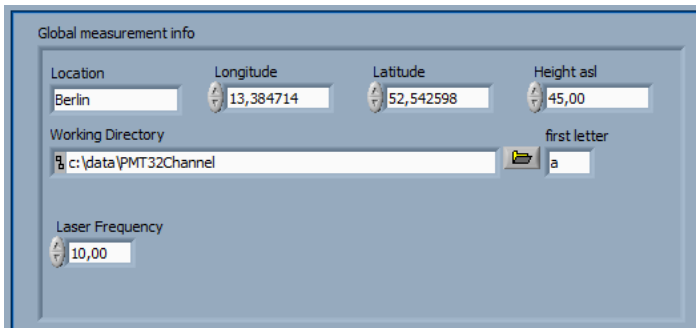
or ✕ at the top right of the window will stop the application's execution. When run as a Windows application the front panel window will close. In the LabVIEW development system the window will remain open and you may restart the program using LabVIEW's run button  .

Global Configuration Parameters

The global configuration parameters can be set in the middle of the tab page *Configuration*



:



The *Working Directory* is the location where you want data files to be stored and the *First Letter* is a letter that will be used as a prefix for the file names. Directly enter the path of the *Working Directory* into the control field or browse your file system using the browse  button. The format of the file names is

`?YYMDDhh.mmssxx`

where `?` is the *First Letter*, `YY` is the year of the century, where `M` is the month (hexadecimal, 0 – C), `DD` is the day of the month, `hh` is the current hour of the day, `mm` are the minutes, `ss` the seconds, and xx the first 2 decimal places of the seconds.

For example the filename

`a0552011.281650`

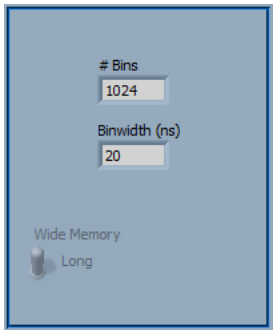
is a file that would have been taken on May 20, 2005 (or 2105...). The operator set a to be the first letter (as in the screenshot) and the time was 11:28:16.50.

The other information above has no effect on the program execution, it is only stored in the data file headers for later reference. The fields available are your current *Location* (e.g. Berlin), the *Longitude* and *Latitude* of your location, the *Height asl* (above sea level) of the location of your acqusition system, and the repetition rate of your laser (*Laser Frequency*).

All these parameters (as well as the *Working Directory* and the *first letter*) are written to the initialization file `global info pm32.ini` once you change any of it. Upon starting, the program preloads the information from the file that is located in the same directory as the libraries or the Windows applications, respectively.

The data file format is described in an appendix.

Bin Width



The number of bins (*# Bins*) and the *Binwidth (ns)* can be set on the right side at the *Configuration* page. A new entered value is coerced to the allowed range of the controller. Please note the dependency on the number of acquired shots. Whether the *Binwidth* is in *nanoseconds* or in *meters* depends on the setting of the *Range/Time Scale* switch below.

Channel Configuration

Also on the *Configuration* tab page the Lidarino channel can be configured.

The following settings are available:

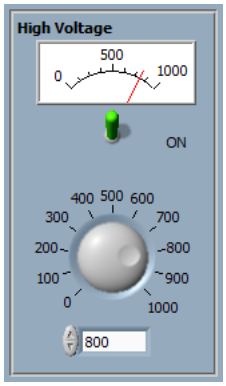
Active Channels for Saving There is only one channel available with a Lidarino.

Wavelength / nm this wavelength is the wavelength of the Lidarino channel. This setting is stored in the initialization file.

Range/Time Scale this switch defines how the scales of the transient plots on the *Multispectral Lidar* and *ROI* are shown, either in *nanoseconds* or in *meters*. This settings affects the appearance of the Binwidth, too.

Setting The High Voltage

The *High Voltage* is set in the middle of the page *Multispectral Lidar*.

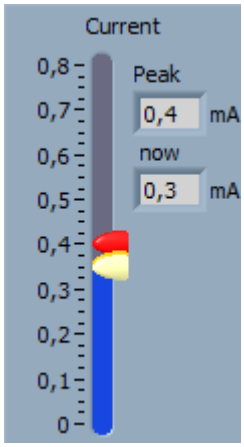


The *High Voltage* value is saved into the initialization file. The voltage set at the Licel Multispectral Lidar Detector is displayed at the top.

Current Warning Level

To find an appropriate warning level for the current value after setting the high voltage proceed as follows:

1. Measure power consumption in darkness:
   Set the HV to your preferred setting. Keep the detector in darkness. Read the current value in the *Intern* tab.



2. Measure power consumption with lidar signal:
   Now take a lidar measurement and read the current again. Usually this value will be the same as your dark value.

3. Set the warning level slightly above the measured value(s) before. Now you have a warning level which is adapted to your HV setting and to your operating conditions.

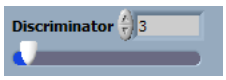
4. If you found an appropriate value for the warning level for future usage you could change that maximum value in the initialization file :

```
[HV_Current]
Max_mA = 0.3
```

If an operator sees *no signal* but the *Photomultiplier Overload* warning is shown, then he will know that there is too much light on the detector and he should decrease the overall light level.
You can play with pulsed signals (LED source) with variable duty cycles and use pretty strong pulses. You will then see *blooming effects* when secondary electrons flow into neighboring channels. In this condition you might see a weak increase of the current meter.

Discriminator Level

The *Discriminator* Level is set in the middle of the page *Multispectral Lidar*, as well, the corresponding control is located directly below the *High Voltage* switch.



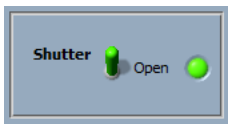
The *Discriminator* level and the *High Voltage* value are saved to the initialization file. The voltage set at the Licel Multispectral Lidar Detector is displayed at the top.

External / Internal Trigger

In normal operation it is assumed that an external trigger source is available. For testing purposes or calibration with a continuous spectral lamp the controller is capable to generate an internal trigger signal. The function is enabled by the *Trigger* switch.

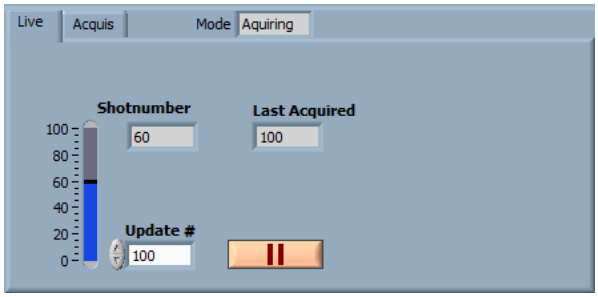
Mechanical Shutter

A *Shutter* switch is operable if an external mechanical shutter is supported by the controller.



The Live Mode

PM32 Channel Acquisiton will always start in the *Live* mode. The *Mode* can be switched with the mode selector at the top in the middle of the *Multispectral Lidar* tab page. Please refer to the screenshot above.

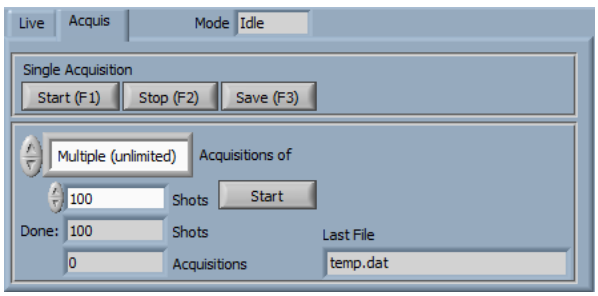


In the oscilloscope-like *Live* mode an acquisition is started and the shot number is monitored and displayed in the *Shotnumber* field. When the detector has acquired the desired number of shots *Update #* and is ready to be read out, the data is read and displayed at the left side. The only necessary action in the software to obtain a signal is to set a value for the *High Voltage* and switch on the high voltage on the right. The *Mode* above is the current acquisition mode of the Licel Multispectral Lidar Detector.
If the *Update #* is set to 1 the acquisition is started with the `TRANSMIT` option. Data is immediately read from the Licel Multichannel Lidar Detector. If no trigger is received in this mode this is indicated at the top.

An acquisition during the live mode can be stopped using the *Freeze*  button e.g. for a deeper inspection of the displayed trace or for making a screenshot. The live mode restarts after clicking on 

The Acquisition Mode

The *Acquisition* mode is intended for single acquisitions or a number of fixed or unlimited multiple acquisitions including file storage.



The following list explains the functionality of the buttons and input fields related to acquiring data and saving it to files.

Start (F1) starts a single acquisition with the number of *Shots* specified below. The acquisition stops when the number of *Shots* has been reached or when the user presses *Stop*. The number of acquired shots is displayed below. All acquired data are displayed and the data corresponding to the *Selected Channels for Saving* are saved to a temporary *Last File* `temp.dat` in the *Working Directory* specified before.

Stop (F2) Stops a running acquisition.

Save (F2) Saves the current temporary file to a file named according to the file name convention.

Multiple acquisitions are started with the

Start button in the bottom section from the figure above. Once a multiple acquisition has been started the label of the button changes to

Stop and could be used to stop a running acquisition. The data corresponding to a completed acquisition is written to a file in the *Working Directory*. The file name is displayed in the *Last File* field. The number of acquired shots and the number of completed acquisitions are displayed. The program acquires data and writes files until *Stop* is pressed. Optionally a fixed number of acquisitions to acquire can be set by switching the *Multiple (unlimited)* field to 

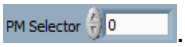
. If a fixed number of acquisitions is set the acquisition will stop automatically after the specified number of data files has been written.

Data Display

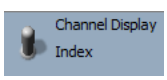
The data is displayed on the two pages *Multispectral Lidar* and *ROI*.

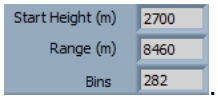
ROI

Two display indicators are available on the page *ROI*:

1. **Individual channel display (left)**: here the averaged count rate for an individual channel is displayed. The channel is selected by the *PM Selector* 

. The cursors are used to specify the integration region for the spectrum (right). The channel selector as well as the scale settings are used for the channel display on the page *Multispectral Lidar*, as well.
2. **Spectrum between Cursors (rigth)**: the spectrum is obtained by integrating all acquired channels in the range specified by the cursors in the individual channel display. The specified range is shown at the bottom below the graphic display. The *Channel Display* x-axis may be switched either to the channel *Index* display or to a *Wavelength* scale according to the *Center Wavelength* and *nm per Channel* from the configuration.





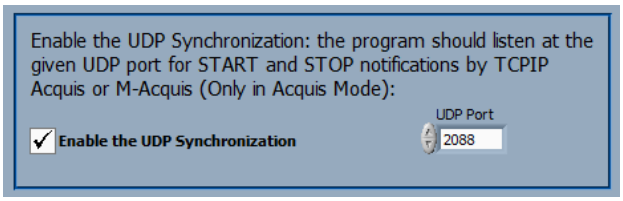
.

UDP Synchronization

It is often important to acquire data that is in synchronization with data from other applications. For that, the PM32 software is able to receive `START` and `STOP` commands via UDP. If the other data acquisition software the PM32 acquisitions should be synchronized with is capable to send those `START` and `STOP` commands we will get nearly synchronized data.

Licel's TCPIP Acquisition software *TCPIP Acquis* for acquiring data from Licel's transient recorders is capable to send `START` and `STOP` during multiple acquisitions. To achieve a synchronization with *Licel's TCPIP Acquis* follow the steps below:

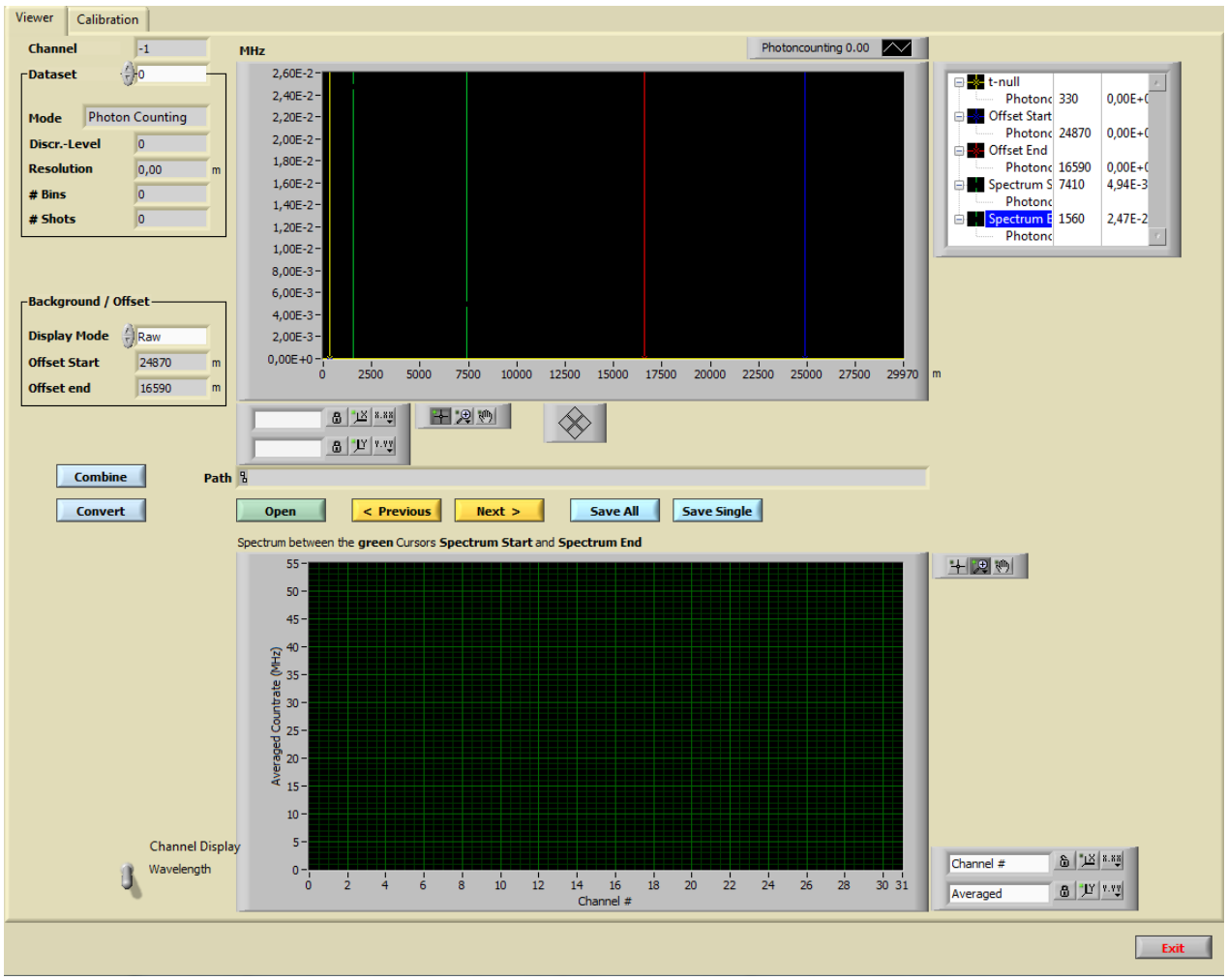
1. Configure *Acquis* to publish `START` and `STOP` commands via UDP. Choose either *Broadcast* or set an IP address equal to the IP address of the PC where the *PM32 Channel* software is running. Make sure that the selected port must not be blocked by firewalls in the network.
2. In the PM32 channel software go to the tab page *Configuration* (top right).



3. Enable the UDP synchronization by checking the corresponding checkbox.
4. Enter the UDP Port. This should be the same port as configured in the UDP configuration of the acquisition software.
5. Start a multiple acquisition at *Acquis*. From then the *PM32 Channel* software will automatically Start and STOP nearly together with the acquisition software.

5.4 PM32 Viewer

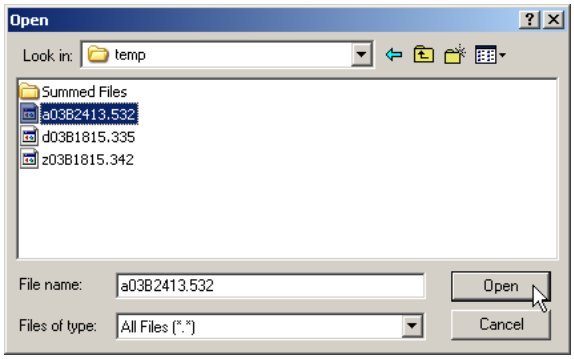
You can load the *PM32 Viewer* by either double clicking on the `PMT32 Viewer.vi` in the LabVIEW project. If you installed the Windows applications please start the program by selecting the entry *PM32 Viewer* in the *Licel PM32Channel* section of the Windows Start menu. After doing so, you should see a screen similar to the one below.



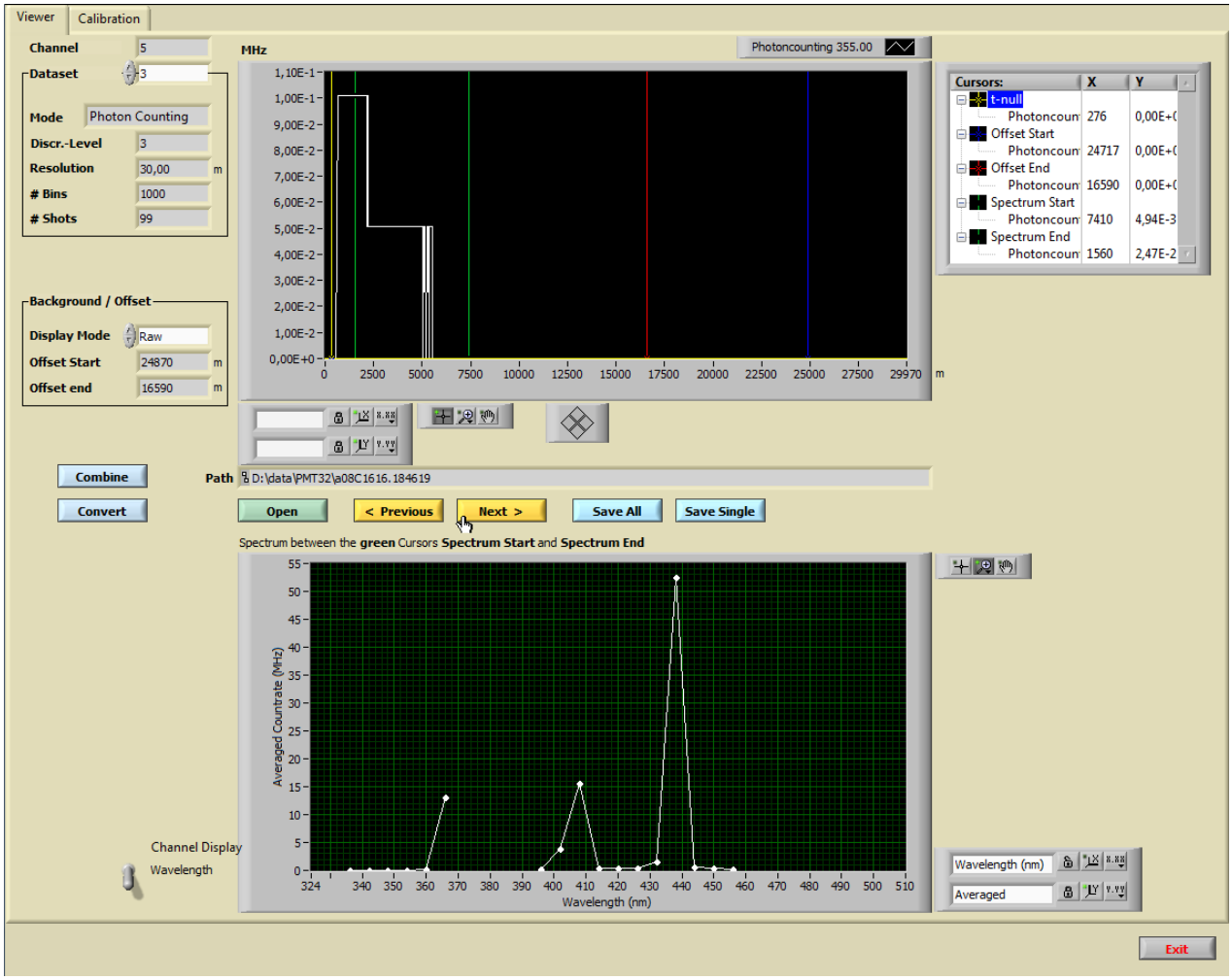
To start the program press the **Run** button at the top left of the screen.

The Windows application will start automatically when called for the first time.
A desired data file may be loaded by pressing the *Open* button.

A file selection dialog will appear. At the program start this dialog will come up without pressing the button.

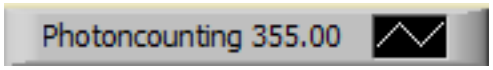


Select a valid data file, press open and the dataset number specified by the *dataset* control will appear in the graph indicator.



The channel index corresponding to the dataset read from the file is displayed top left

The wavelength is displayed in the graph legend

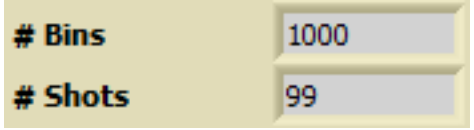


and the units used for the y-axis (MHz)is seen in the upper left hand corner of the graph

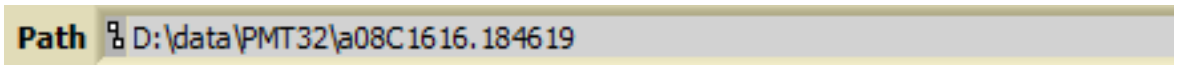
The discriminator level used during the acquisition is displayed

Furthermore the bin resolution is given in meters

and the number of range bins and the number of acquired shots are displayed:

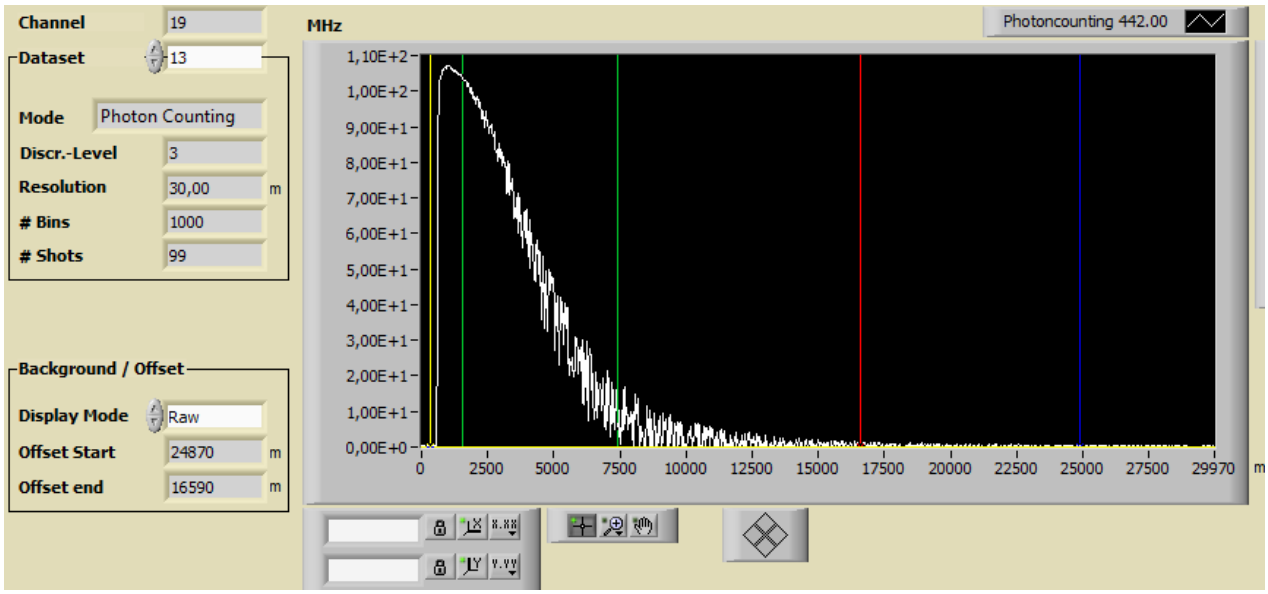


The full path to the current data file is shown in the *path* indicator.

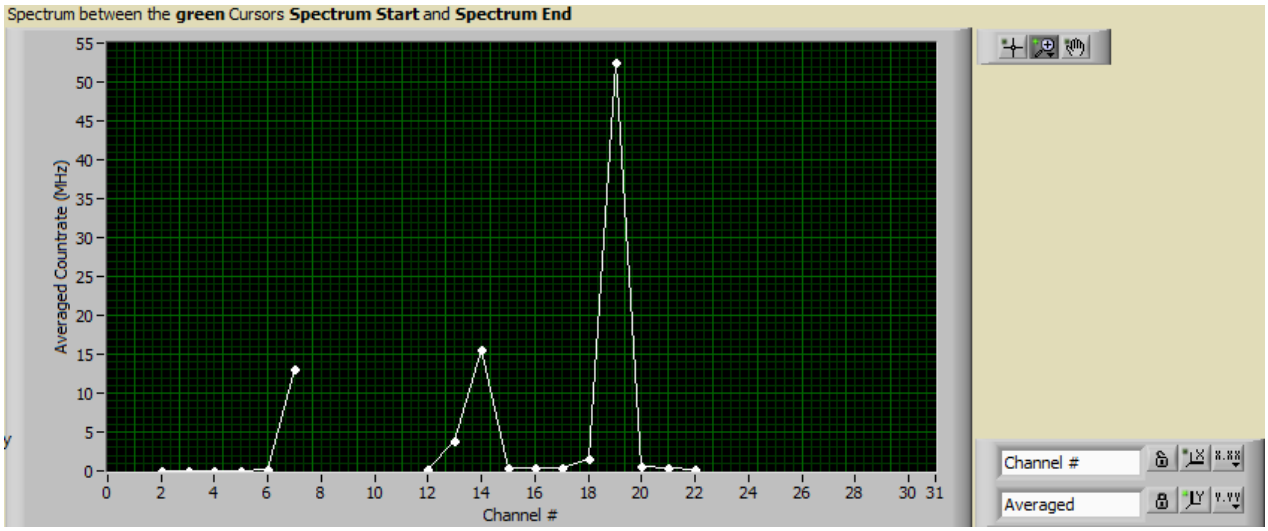


If you would like to see a different dataset (spectral channel) from the loaded file, use the *Dataset* control to choose it.

Below, dataset 13 (Channel 19 in our example) is shown.

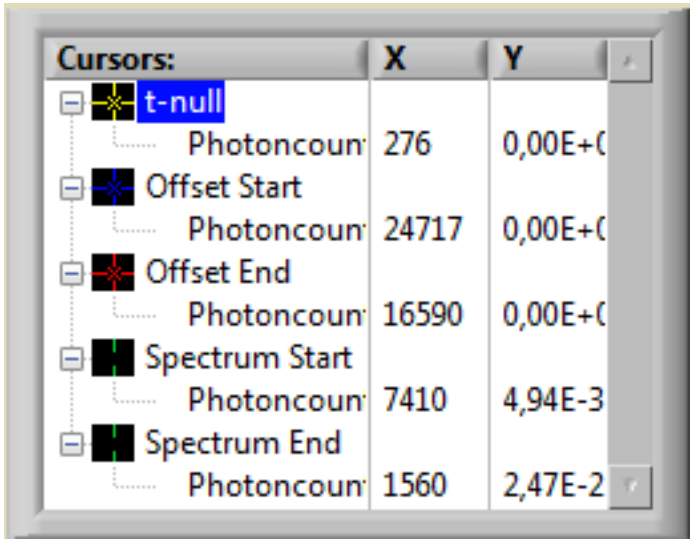


The two green cursors define the range for the integration of the spectral display at the bottom of the program's panel. Please note that not all channels have been saved in our example so the drawn line representing the spectrum is broken at the "missing" channels.

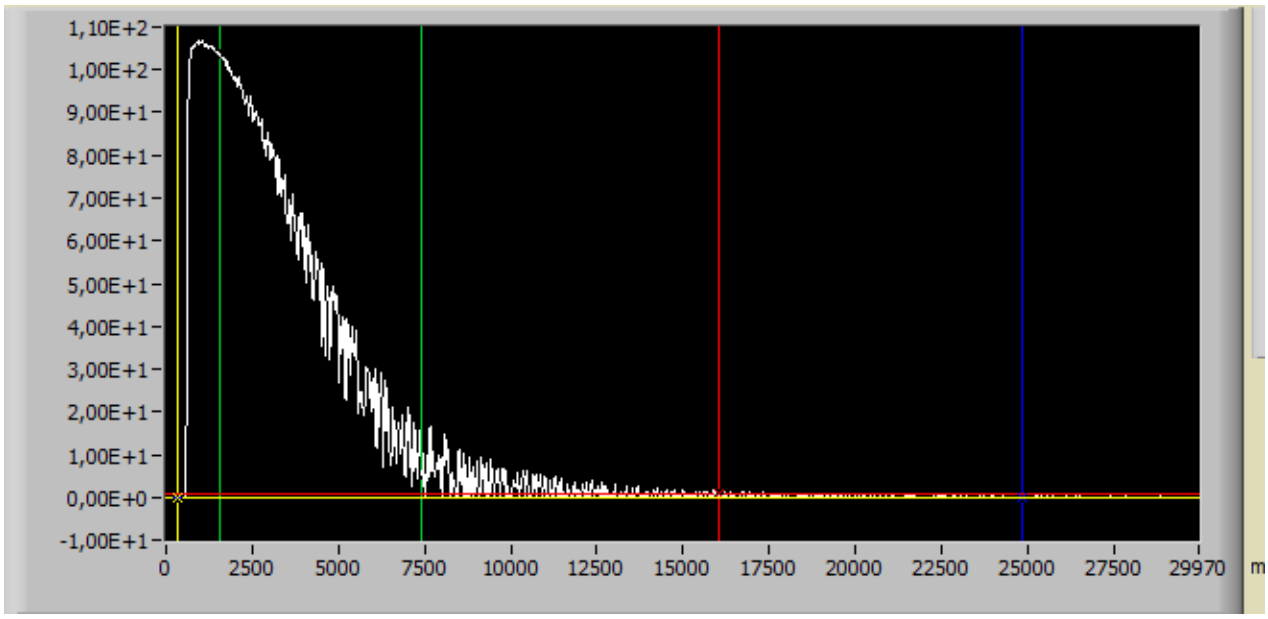


The x-axis may be set with the *Channel Display* switch either to *Index* or *Wavelength*.

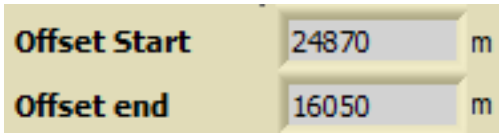
All cursor coordinates are displayed in the cursor list:



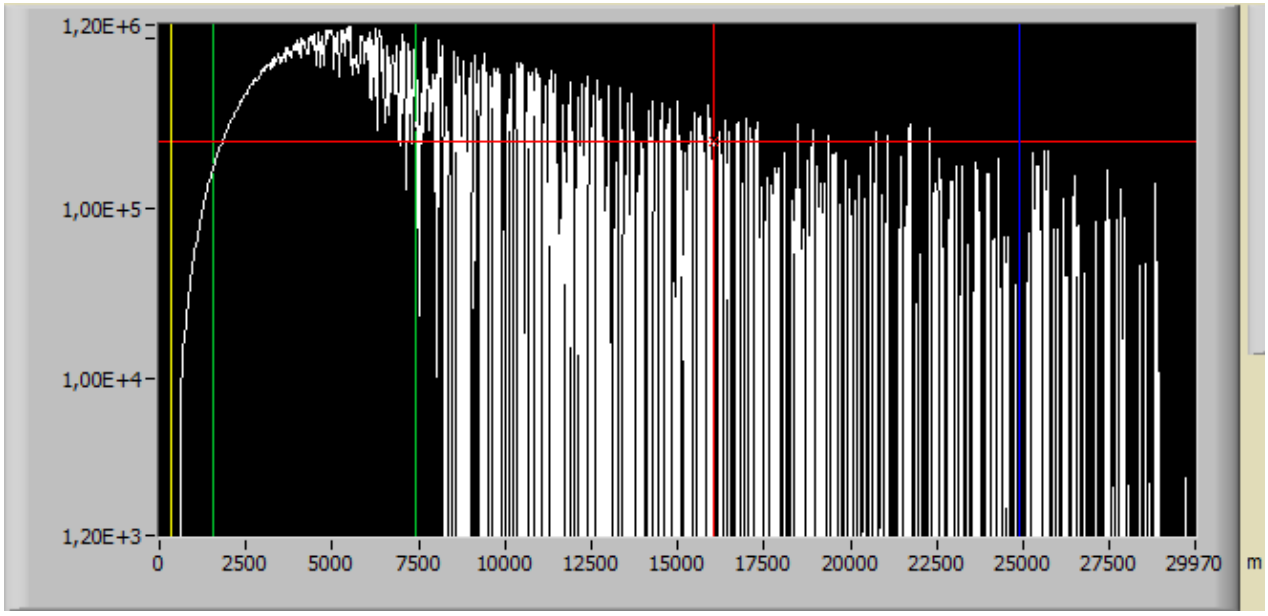
Currently the data is being shown in raw mode. The *Offset Start* (blue) and *Offset End* (red) cursors can be used to correct for the baseline offset. If the *Display Mode Offset Corr.* is used, then the mean value of the signal between these two cursors will be subtracted from the signal in order to create a baseline corrected signal. Use the cursor controls to move the blue and red cursors (Offset Start and Offset End) to a region which will be evaluated to generate the new baseline. Change the *Display Mode* to *Offset Corr.* and observe that the baseline of the signal changes.



In the image above, you can see that the baseline is now closer to zero. The end values of the region to be used to evaluate the baseline are shown in the *Offset Start* and *Offset End* indicators.

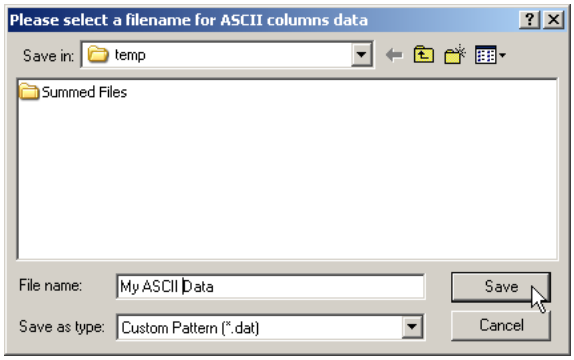


The data can also be displayed in the *Pr2* Pr^2 mode which corrects for the power loss due to the length of travel of the signal. The key parameter for the *Pr2* mode is *t-null* (yellow cursor) which defines the starting point of the signal. When switching to the *Pr2* mode, the data will look similar to the following.



Note that the signal is only corrected for power loss after the yellow cursor, which is the t-null point. All values before *t-null* are left unchanged and those after *t-null* are corrected for the power loss due to distance. The difference in the display is due to the fact that the scaling has been changed to a logarithmic scale for easier viewing.
Please remember that the *Pr2* signal is always offset corrected, as well, using the blue and red cursors.
Once you have adjusted your signal and would like to save it to a file in ASCII format as it is displayed, press the *Save Single* button.

A file dialog appears asking you to name the ASCII file.



Enter the desired file name press *Save*. The file extension `.dat` will automatically be added to the file name unless you choose a different extension. Afterwards the data is saved in ASCII format as a column and can be imported into other programs for further evaluation.

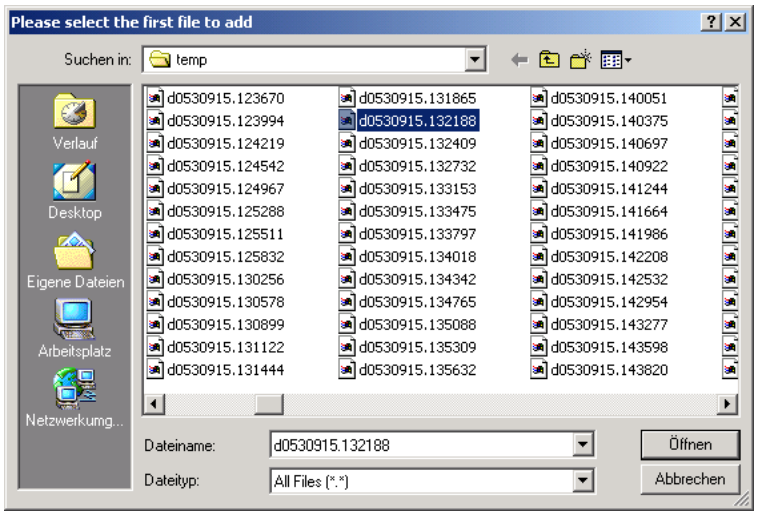
converts all datasets contained in the actual data file to an ASCII format file and appends the extension `.dat` to the end of the actual file's name. The whole file can then be imported into other programs.
If you would like to load the next file or previous file in a time series, this can be done by pressing the < *Previous* or *Next* > buttons.

By pressing one of these buttons, either the file acquired before or after the current file will be dis-

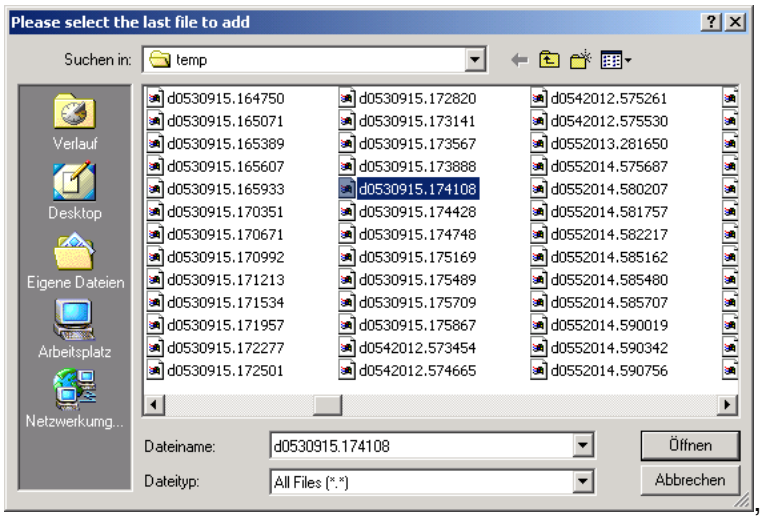
played if it exists in the same directory. You can then manipulate the data using the aforementioned capabilities of *PM32Channel Viewer* and save the data from the new datasets to ASCII if desired.
Two data file utilities may be called from the *PM32Channel Viewer*, one to sum the data values of several files to one single file (`Datafile Addfiles Interface.vi`), and another to convert the data from several files to corresponding ASCII files (`Datafile Batch Converter.vi`).
By pressing *Combine* `Datafile Addfiles Interface.vi` is interactively called to sum the data contained in a set of subsequently recorded data files.

You will have to specify:

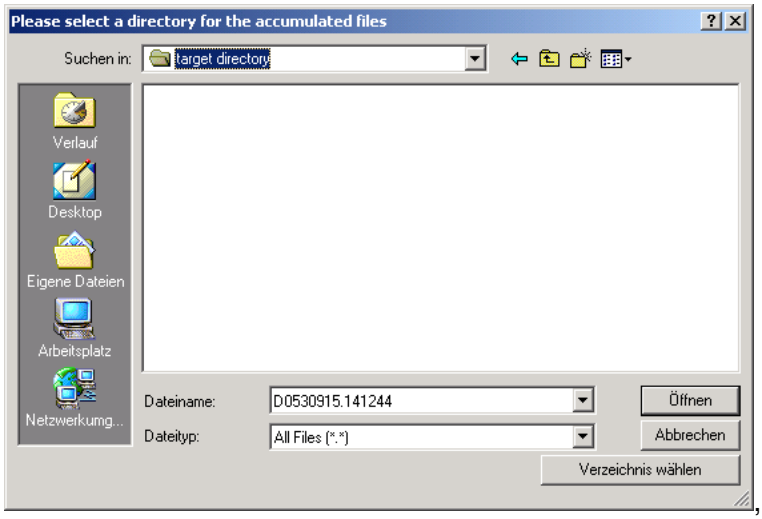
1. the name of the first file to add



2. the name of the last file to add

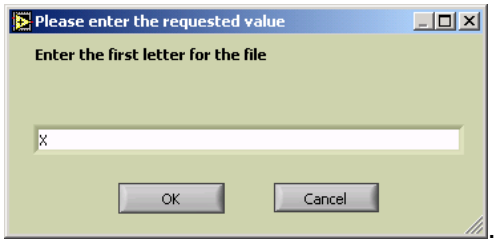


3. the name of the target directory for the file containing the summed data.



, and

4. the first letter of the file name



.

Both the first and the last file must reside in the same directory. The data from the files with acquisition dates/times lying between the first and the last files (including them) are summed and written to a target file into the target directory. The target file's name begins with the first letter, and the rest of the name is taken from the first selected file.
By pressing *Convert* `Datafile Batch Converter.vi` is interactively called to convert the data contained in a set of subsequently recorded data files to ASCII files.

The selection mechanism to select the first file, the last file, and the target directory is the same as for the sum operation.
Both the first and the last file must reside in the same directory. The data from the files with acquisition dates/times lying between the first and the last files (including them) are converted as described above for the *Save All* operation. Each data file will result in an ASCII file named by the original file name with the additional extension `.txt`.

Clicking on 

or ✕ at the top right of the window will stop the application's execution. When run as a Windows application the front panel window will close. In the LabVIEW development system the window will remain open and you may restart the program using LabVIEW's run button  .

Chapter 6

Data Acquisition using the Licel Lidarino Detector

6.1 Data Acquisition Overview

This section describes how to acquire data from the Licel Lidarino Detector. It shows in detail the two modes of data acquisition supported by the detector - The PUSH mode and the SLAVE mode and lists the steps/procedures for operating the detector in the respective modes.

Please Note : Usage of the term ”Detector” refers to the Licel Lidar Detector which is abbreviated as ” Lidarino ”, ”Controller” refers to the Licel TCPIP Ethernet Controller present inside the Lidarino and ”Client” refers to the Data Acquisition Software running on the PC.

6.1.1 SLAVE Mode

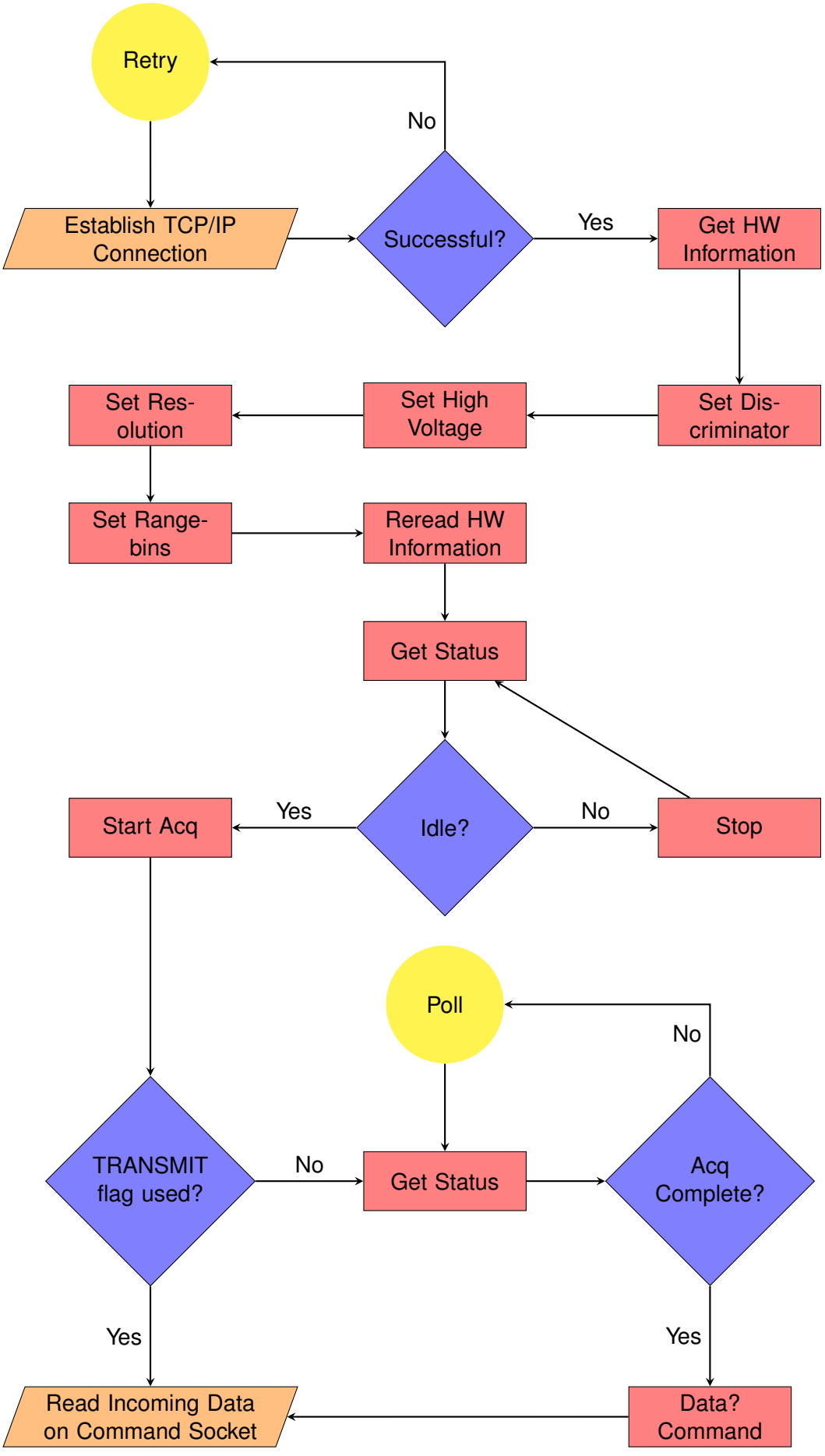
In the SLAVE mode the detector always expects a command from the client in order to perform any kind of action and send a reply. Hence in this mode detector is the SLAVE and the client is the MASTER. The different commands accepted by the detector and its corresponding replies are listed in the the TCP/IP Command List and Syntax in the appendices chapter of this manual. Please refer to the appendices for more information on the corresponding commands. The typical steps involved for acquiring data from the detector in the SLAVE mode are :

- Step 1 → TCP/IP Connection : Open up a connection to the COMMAND Socket. All the communication to the detector in the SLAVE mode takes place on the bi-directional COMMAND Socket. The COMMAND Socket is by default on port 2055.
- Step 2 → Hardware Information : Once the connection to the detector is established, the hardware information of the detector must be read out in order to know the different capabilities of the detector. This can be obtained by issuing the HW? command on the COMMAND Socket.
- Step 3 → Discriminator Level : A suitable discriminator level must be set before start of data acquisition. This can be done by issuing the DISC command on the COMMAND Socket.
- Step 4 → Highvoltage : The detector must have its Photo Multiplier Tube (PMT) high voltage turned on in order to detect photons. Set a suitable level of high voltage by issuing the PMTG command on the COMMAND Socket.
- Step 5 → Resolution : When the detector is powered on the default resolution of the detector is 10 ns. A different resolution can be set by issuing the RES command on the COMMAND Socket.
- Step 6 → Rangebins : When the detector is powered on the default rangebins of the detector is 8000. A different value can be set by issuing the RANGE command on the COMMAND Socket.

- Step 7 → Status : The current status of the detector must be determined before the start of the data acquisition. Issue a STAT? command on the COMMAND Socket to find out the current status of the detector. If the detector is not in the Idle state issue a STOP command on the COMMAND Socket to stop the detector and bring it to the idle state.
- Step 8 → Reread HW Information : Settings the bin width has an impact how many shots can be aquired and the information is also needed to see what bin width the controller internally uses. The bin width also influences the compression scheme in push mode. This can be obtained by issuing the HW? command on the COMMAND Socket.
- Step 9 → Start : After all the above procedures are complete the detector can be instructed to begin its data acquisition. Issue a START command on the COMMAND Socket to start the data acquisition. The START command makes use of additional flags as explained in command syntax. If the detector is intended to be used in the SLAVE mode, care must be taken that the START command is either sent without any optional flags or with the TRANSMIT flag under special circumstances. Under no condition should the PUSH flag be used if the detector is intended to be used in the SLAVE mode.

  Please Note : If the TRANSMIT flag is used, care must be taken that no other command is sent on the COMMAND Socket until all the data is received from the controller. In other words, the COMMAND Socket must be blocked until the complete reception of data. It is for this reason that the use of the TRANSMIT flag is recommended only when a small target shot number has to be acquired.
- Step 10 → Status : If the detector was started without the TRANSMIT flag, the client should monitor the data acquisition status of the detector by polling with the STAT? command on the COMMAND Socket. The shot number should keep increasing till the target shots is reached. Once the target shots is reached , the data is ready to be read out. If the detector was started with the TRANSMIT flag , it automatically transmits the data once the acquisition is completed and hence there is no need to poll with the STAT? command.
- Step 11 → Data : Once the detector has acquired all the shots it is ready to transmit the data to the client. Issue the DATA? command on the COMMAND Socket to instruct the detector to transmit the data. If the TRANSMIT flag was used with the START command then the detector automatically transmits the data to the client without the need for a DATA? command.

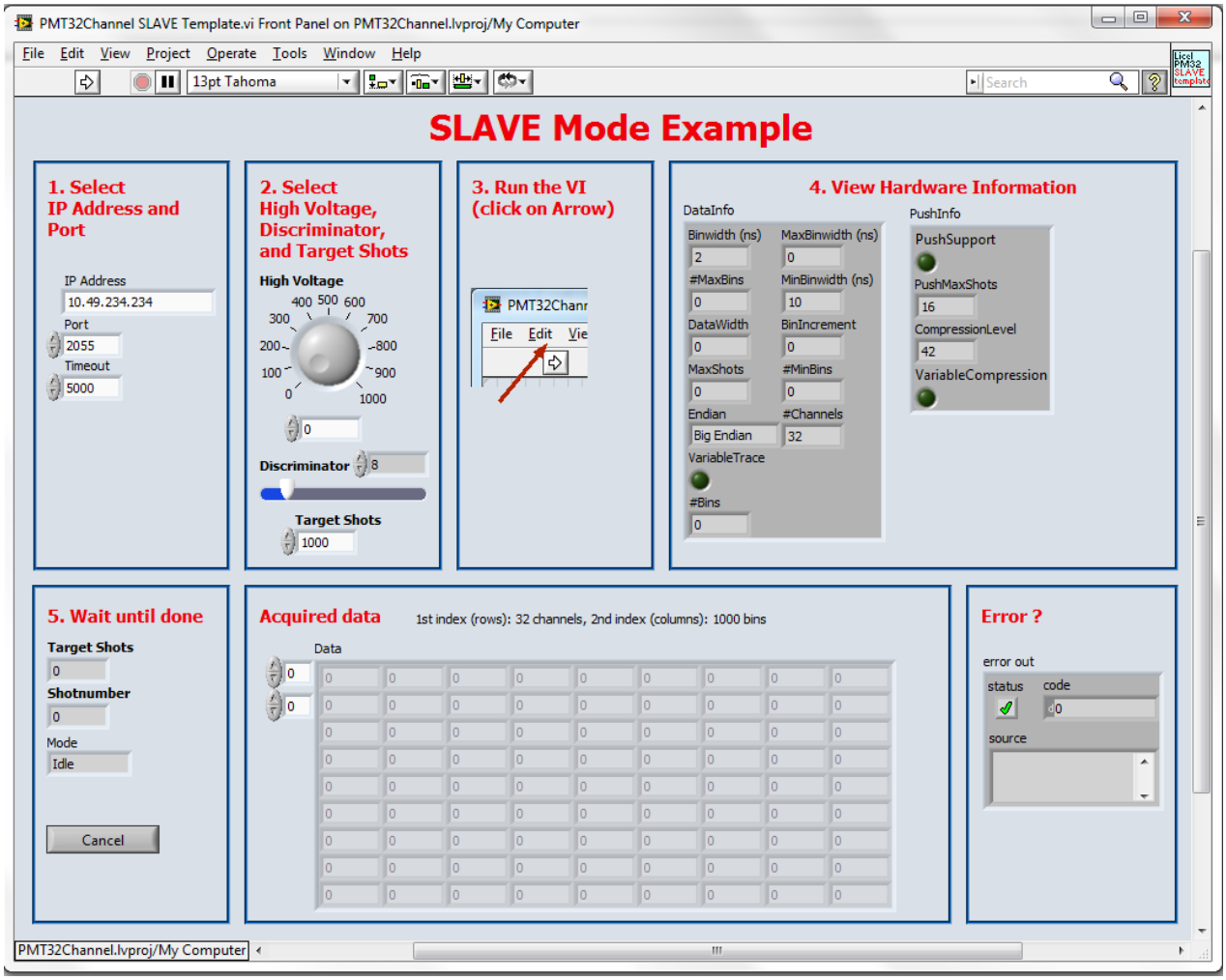
All of the above steps are illustrated in the form of a flowchart on the next page for easy reference.


Retry
No
Establish TCP/IP Connection
Successful?
Yes
Get HW Information
Set Res-olution
Set High Voltage
Set Dis-criminator
Set Range-bins
Reread HW Information
Get Status
Start Acq
Yes
Idle?
No
Stop
Poll
No
TRANSMIT flag used?
No
Get Status
Acq Complete?
Yes
Yes
Read Incoming Data on Command Socket
Data? Command

Flowchart : Licel SP32 Detector SLAVE Mode Operation

LabVIEW SLAVE Mode Template

As part of the LabVIEW sources the template VI `PMT32Channel SLAVE Template.vi` is included. It is located in the LabVIEW LLB `PMT32Channel.llb` and may easily be opened from the delivered LabVIEW project.



Flowchart : Front Panel of the PMT32Channel SLAVE Template.vi

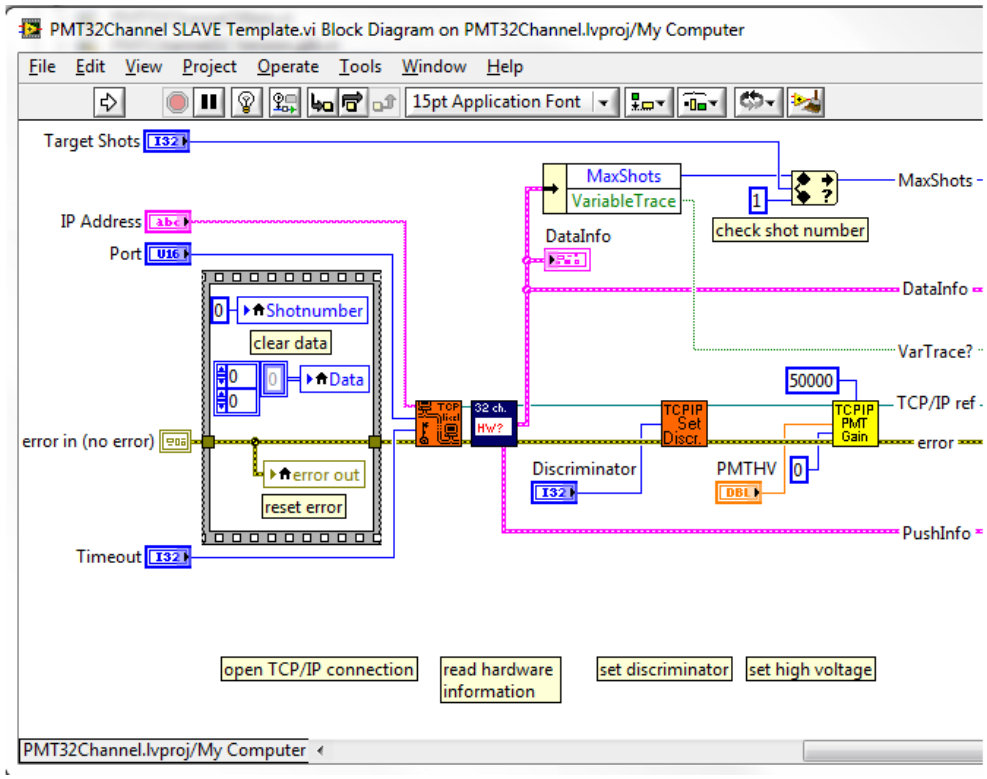
To run the template VI

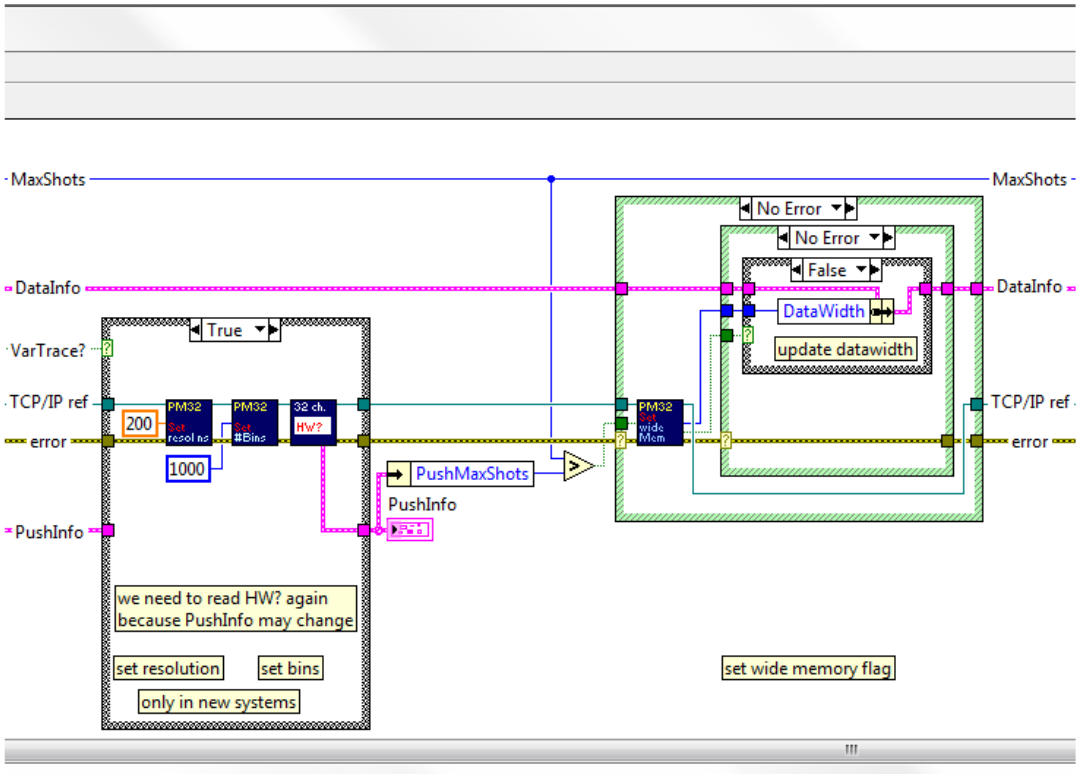
1. Select the IP Address and Port
2. Select the desired High Voltage and Discriminator level
3. Rrun the VI by clicking on the arrow icon.

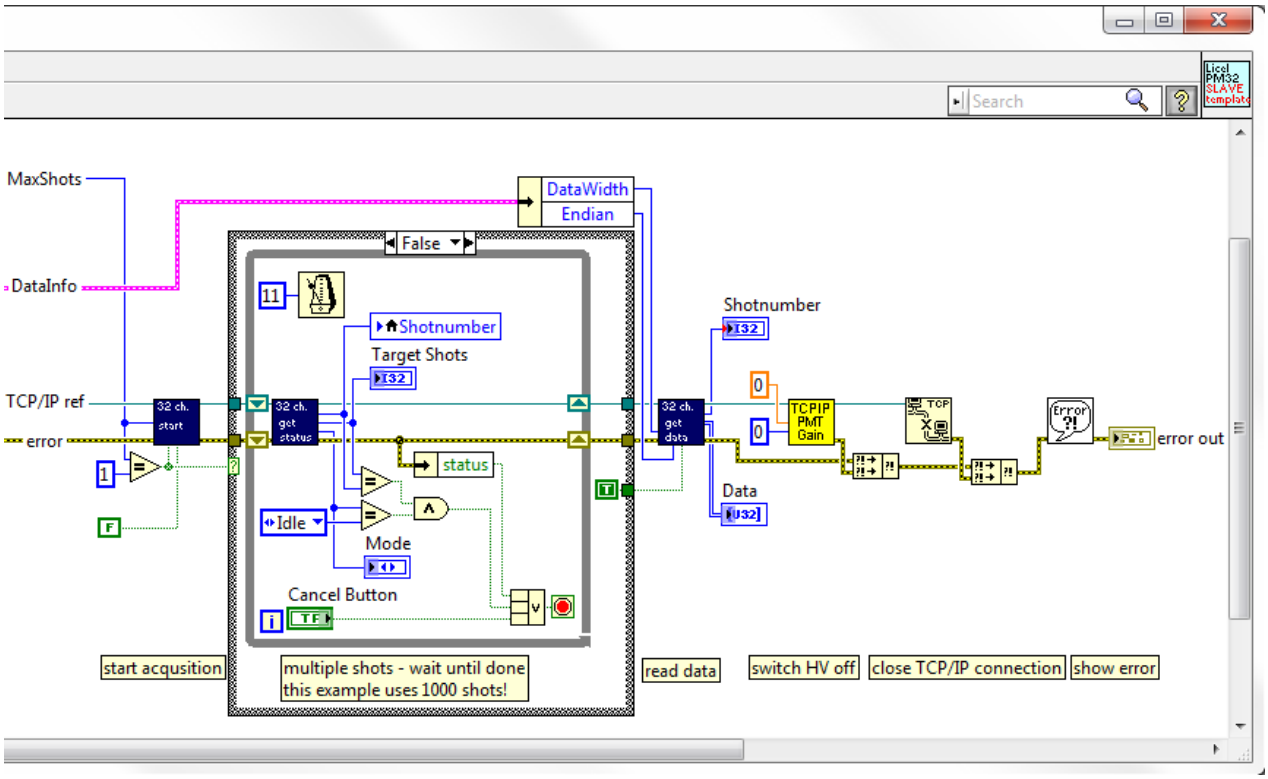
The VI will then

1. Open a TCP/IP connection with the device at the given IP Address and Port (only 1 attempt in the template VI),
2. Read the hardware information and display it in the corresponding indicators,
3. Set the discriminator level to the selected value,
4. Set the high voltage to the selected value,
5. Set the resolution to 200 ns if supported by the hardware,
6. Set the range bins to 1000 if supported by the hardware,
7. Start an acquisition with 1000 shots to acquire.
8. During the acquisition the number of shots and the acquisition status is displayed,
9. When the acquisition is finished the data is read and displayed.
10. At the end the VI will set the high voltage back to 0 V
11. Close the TCP/IP connection.
12. If an error occurs it will be shown in a standard dialog and on the front panel indicator.

These execution steps can be seen on the block diagram of the VI in the next figure:


PMT32Channel SLAVE Template.vi Block Diagram on PMT32Channel.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Target Shots
IP Address
Port
Timeout
error in (no error)
Shotnumber
clear data
Data
error out
reset error
MaxShots
VariableTrace
DataInfo
check shot number
MaxShots
DataInfo
VarTrace?
50000
TCP/IP ref
error
Discriminator
PMTHV
PushInfo
open TCP/IP connection
read hardware information
set discriminator
set high voltage
PMT32Channel.lvproj/My Computer


MaxShots
DataInfo
VarTrace?
TCP/IP ref
error
PushInfo
True
200
1000
PM32
32 ch.
HW?
PushMaxShots
PushInfo
we need to read HW? again because PushInfo may change
set resolution
set bins
only in new systems
No Error
No Error
False
DataWidth
update datawidth
set wide memory flag
MaxShots
DataInfo
TCP/IP ref
error

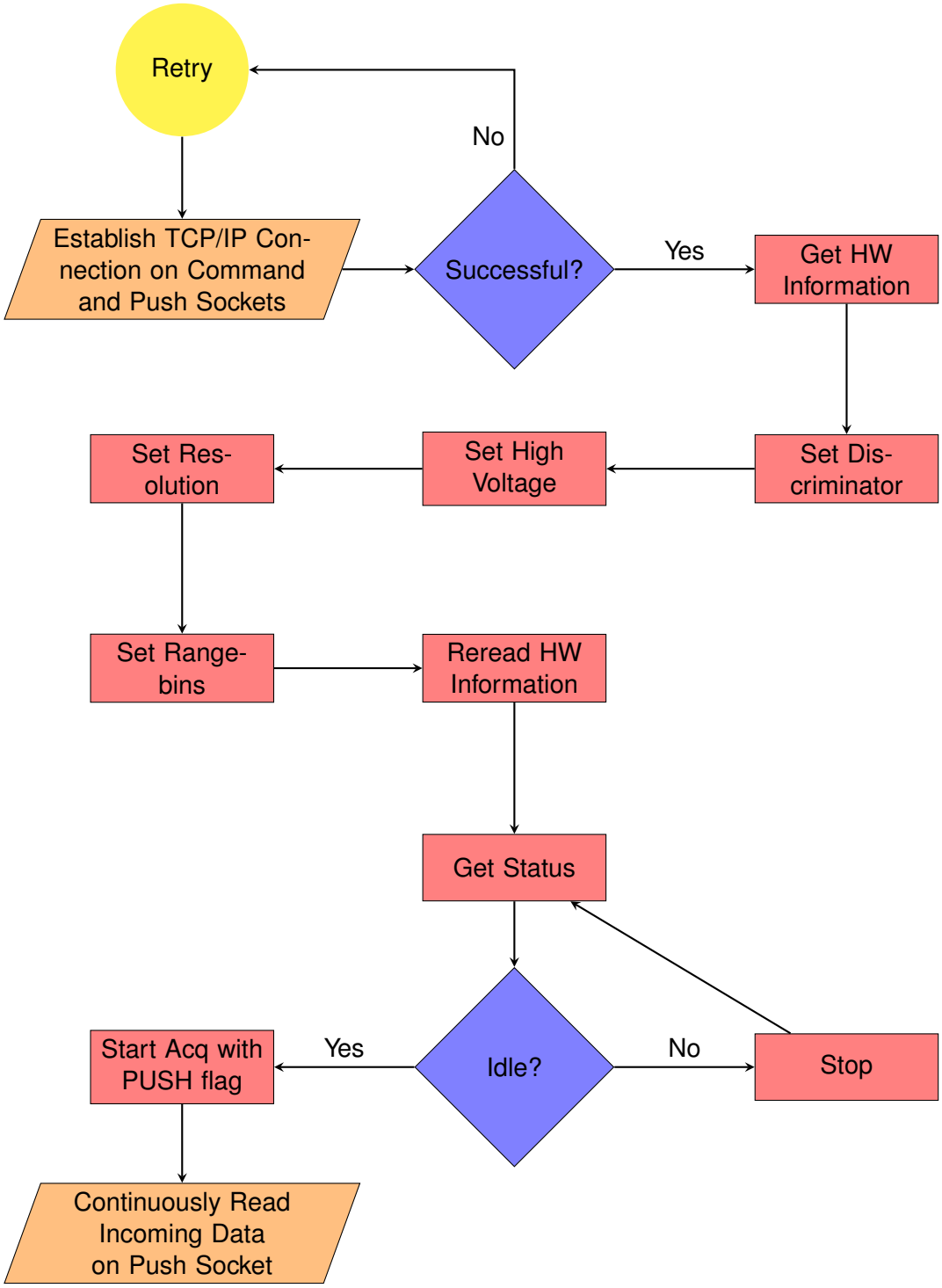

MaxShots
DataInfo
TCP/IP ref
error
DataWidth
Endian
False
11
Shotnumber
Target Shots
I32
32 ch. start
32 ch. get status
status
Idle
Mode
Cancel Button
TF
start acquisition
multiple shots - wait until done
this example uses 1000 shots!
32 ch. get data
read data
Shotnumber
I32
Data
U32
0
TCPIP PMT Gain
TCP
Error
error out
switch HV off
close TCP/IP connection
show error

6.1.2 PUSH Mode

The SLAVE mode can sometimes cause a lot of delays due to Nagle Algorithm when the number of shots to acquire is very low (like in the case of single shot acquisition mode) due to the large number of small sized commands being issued. In order to overcome this the PUSH mode was introduced. In this mode, the detector takes on the roll of the MASTER and the client acts as the SLAVE. The detector is by default initialized to operate in the SLAVE mode after powering on. The PUSH mode is activated only when the PUSH flag is used along with the START command. Once the PUSH mode is activated, the detector does not need any additional commands for data acquisition. It transmits the data once the predetermined number of shots are acquired and automatically restarts the detector for the next acquisition cycle. The PUSH mode makes use of a separate PUSH Socket in order to send data to the client. The PUSH Socket is on the port COMMAND Socket + 1 and hence the default port for the PUSH Socket is 2056. The PUSH Socket is a unidirectional communication socket. It is only used to send the acquisition data from the detector to the client. All commands sent to the detector should be on the COMMAND Socket and the client can not use the PUSH Socket for any purpose other than to receive the data.

Once the detector is put into the PUSH mode it starts the acquisition of the requested shots, transmits it on the PUSH Socket once the acquisition is completed and restarts itself for the acquisition of the next cycle. This process loops until a STOP command is sent by the client on the COMMAND Socket. The STOP command disables the PUSH mode and sets the detector back into the SLAVE mode. Please refer to the section on Efficient use of SP32 for Data Acquisition for more information on when to make use of the PUSH mode. The typical steps involved in acquiring data from the detector in the PUSH mode are :

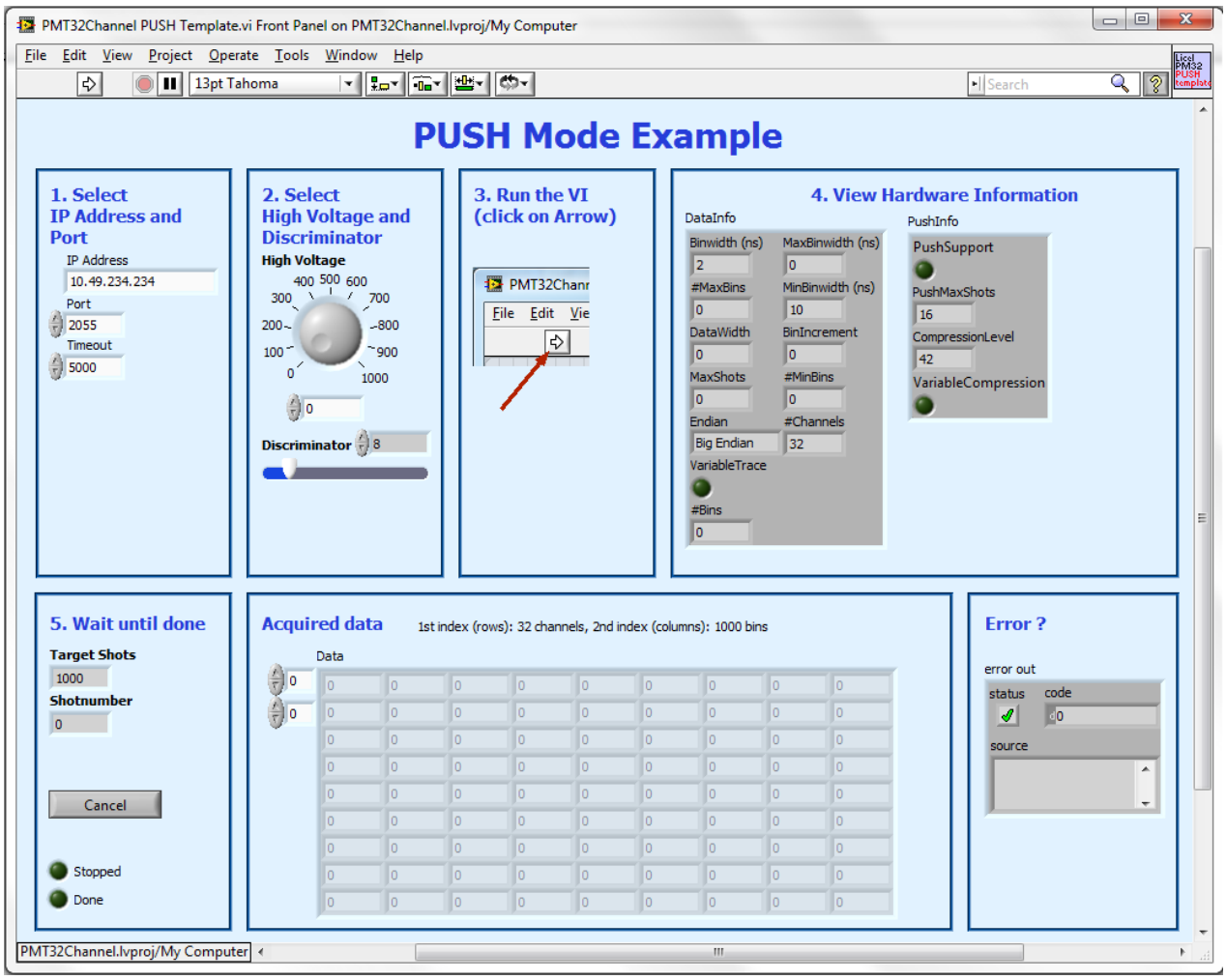
- Step 1 → TCP/IP Connection : Open up a connection to the COMMAND Socket and PUSH Socket on the corresponding ports. The detector makes use of the COMMAND Socket to receive commands and send replies. The PUSH Socket is used to send acquisition status and data to the client.
- Step 2 → Step 8 : These steps are exactly the same as in the SLAVE mode Operation.
- Step 9 → Start : Issue a START command for an acquisition of the required number of shots capable of being averaged internally along with the PUSH flag enabled. The maximum shots capable of being averaged internally is determined by the HW? command reply by the MAXPUSHSHOTS parameter. The PUSH mode is only enabled after the START command is sent with the PUSH flag. Once the PUSH mode is enabled, all the client has to do is read and process the incoming data on the PUSH Socket.



Flowchart : Licel SP32 Detector PUSH Mode Operation

LabVIEW PUSH Mode Template

As part of the LabVIEW sources the template VI `PMT32Channel PUSH Template.vi` is included. It is located in the LabVIEW LLB `PMT32Channel.llb` and may easily be opened from the delivered LabVIEW project.



Flowchart : Front Panel of the PMT32Channel Push Template.vi

To run the template VI

1. Select the IP Address and Port
2. Select the desired High Voltage and Discriminator level
3. Run the VI by clicking on the arrow icon.

The VI will then

1. Open a TCP/IP connection (command socket) with the device at the given IP Address and Port (only 1 attempt in the template VI),
2. Read the hardware information and display it in the corresponding indicators.
3. If the PUSH mode is not supported by the PM32 detector controller the program will terminate.

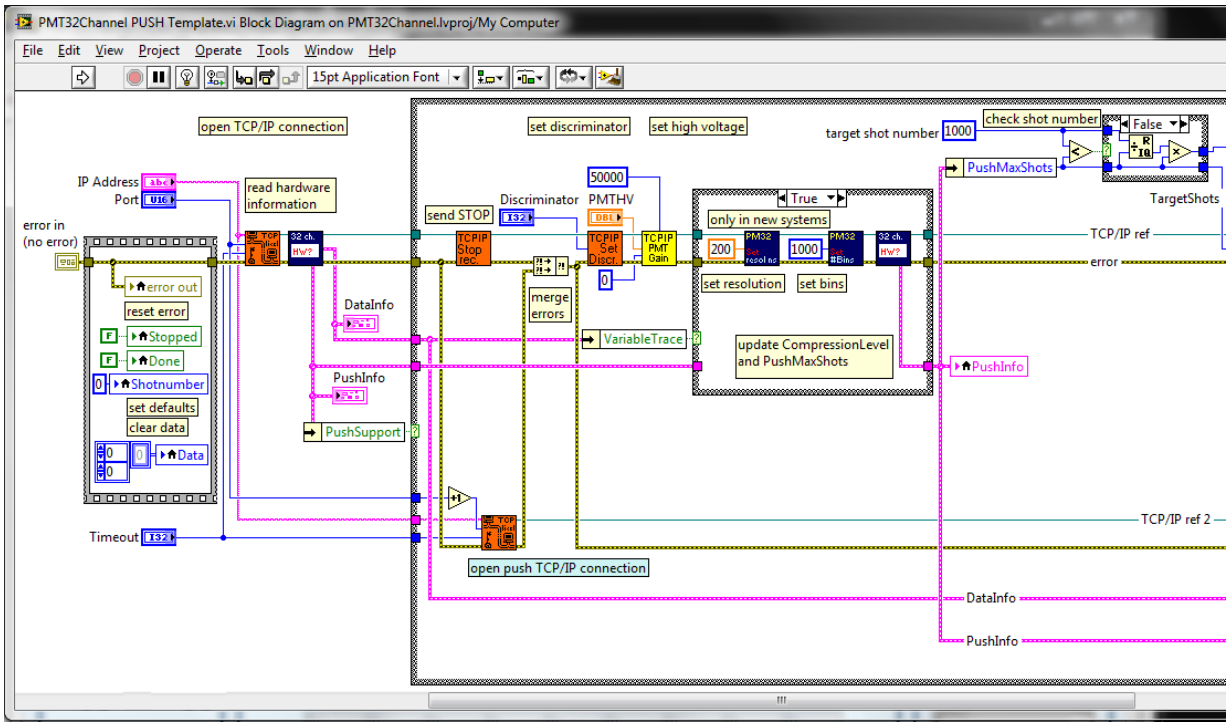
If the PUSH mode is supported the VI proceeds by executing the following operations in parallel on the `COMMAND` and `PUSH` Socket.

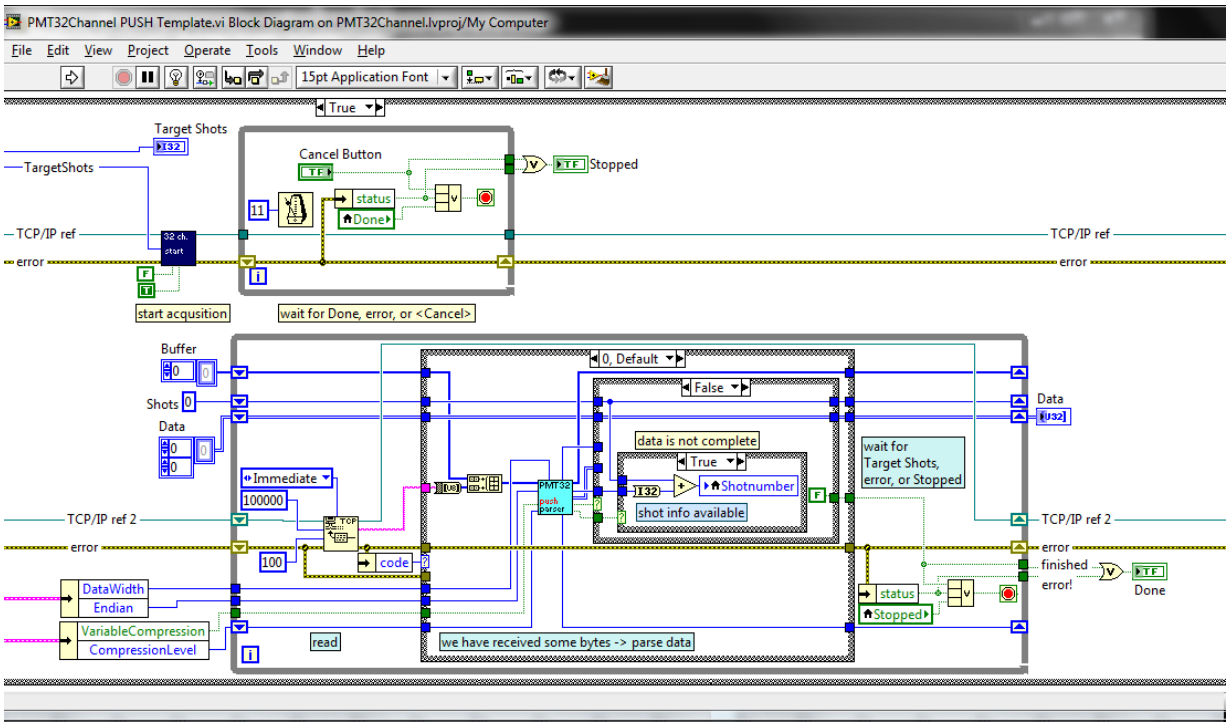
	COMMAND Socket	**PUSH Socket**
4.	Sending a STOP command to stop any data transfer (e.g. if the software has not been correctly stopped in a previous run)	Open a TCP/IP connection (PUSH socket) with the device at the given IP Address and (Port + 1).
5.	Set the discriminator level to the selected value,	
6.	Set the high voltage to the selected value,	
7.	Set the resolution to `200` if supported by the hardware,	
8.	Set the range bins to`1000` if supported by the hardware,	
9.	Reed back the actual hardware information,	
10.	Start a PUSH mode acquisition.	
11.	Stop if the acquisition at the PUSH socket has been **Done** or <**Cancel**> is pressed ((then set **Stopped** = TRUE)) or a previous error has been detected.	Read bytes from the PUSH socket. Extract data from complete data telegrams. Update the shot number. Stop if the **Target Shots** have been reached (then set **Done** = TRUE) or the acquisition has been **Stopped** or a previous error has been detected.
12.		The acquired data is displayed.
13.	A STOP command is sent via the command socket	Data at the PUSH socket is trashed.
14.	The VI sets the high voltage to 0 V.	The TCP/IP PUSH socket is closed

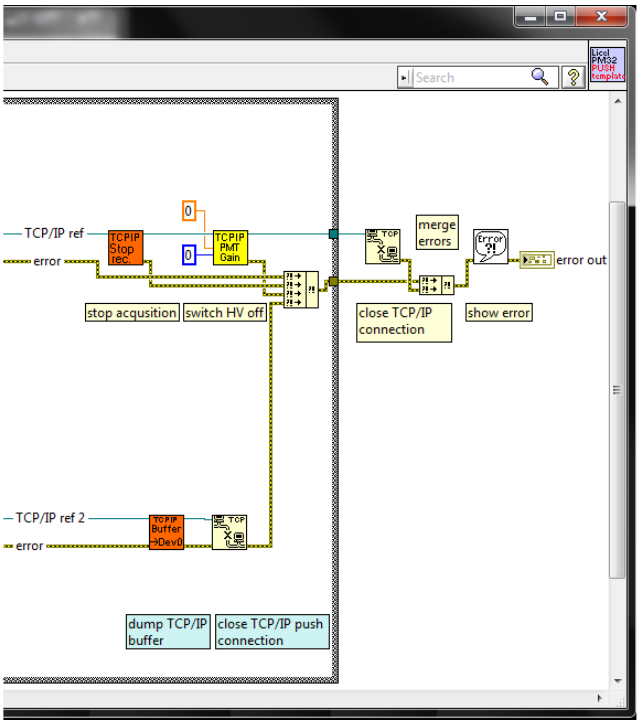
Program end:

15. The will close the TCP/IP connection.
16. If an error occurs it will be shown in a standard dialog and on the front panel indicator.

These execution steps can be seen on the block diagram of the VI in the next figures:






Search
TCP/IP ref
error
TCPIP Stop rec.
0
0
TCPIP PMT Gain
merge errors
error out
stop acqusition
switch HV off
close TCP/IP connection
show error
TCP/IP ref 2
error
TCPIP Buffer ->Dev0
dump TCP/IP buffer
close TCP/IP push connection

6.2 Transmitted Data Format

The transmitted data from the detector consists of two parts namely `Preamble/Header` followed by the `Channel Trace Data`. The `Preamble/Header` contains information about the current data set necessary for the client to read the incoming `Channel Trace Data`. Both these parts have different formats based on the mode of operation of the detector.

6.2.1 SLAVE Mode

Preamble/Header

The `Preamble/Header` of the data set transmitted in the `SLAVE` mode has a fixed 16 bytes length. **Note: This header has less parameters than the psuh mode header as in slave mode there is no compression and no need to synchronize.** This might change in future revisions, so that both haeders are identical.

Dataset Marker = 0xFFFFFFFF [4 bytes]
Acquired Shots [4 bytes]
Number of Traces [4 bytes]
Rangebins [4 bytes]

Figure : Slave Mode Transmission Data Preamble/Header

Dataset Marker [4 bytes unsigned int] :
The `Data Marker` is a fixed `4` byte value used to denote the start of a data set. The client must use the `Data Marker` of the `Preamble/Header` as a reference to the start of the data set for further processing.

Acquired Shots [4 bytes unsigned int] :
The `Acquired Shots` is a `4` byte value used to denote the number of internally averaged shots present in the current data set. This has to be equal to the `target shot number` of the `START` command.

Number of Traces [4 bytes unsigned int] :
The `Number of Traces` is a `4` byte value used to denote the total number of traces present in the acquired data. In the `SLAVE` mode each trace corresponds to a single channel of the detector. Hence this is a fixed value of `32`.

Rangebins [4 bytes unsigned int] :
The `Rangebins` is a `4` byte value used to denote the total number of rangebins present in each trace of the data set. This is equal to the `CURRENTRANGEBINS` parameter of the `HW?` command reply.

Channel Trace Data

The `Channel Trace Data` follows the `Preamble/Header`. In the `SLAVE` mode operation the detector transmits the `Channel Trace Data` without any form of compression. The uncompressed data consists of a single trace . Since the detector has `1` channels, the data consists of singel trace and a trace is as wide as the number of rangebins. This can be represented as the following 2D array:

```
UNCOMPRESSED_DATA[32][no of rangebins]
```

Where the size (in bytes) of each element of the 2D array is as determined in the BINSIZE parameter of the HW? command reply. Shown below is the diagram illustrating how the acquired Channel Trace Data is stored in the internal memory of the detector.

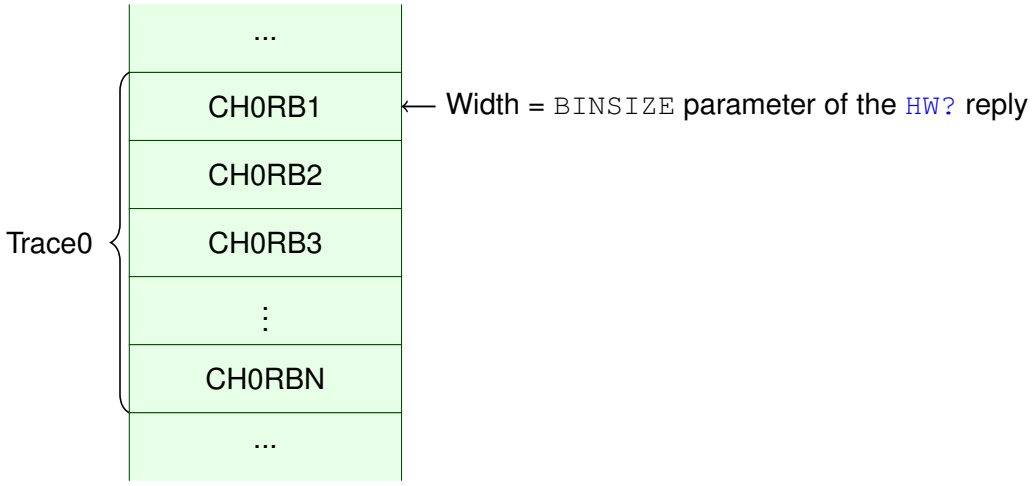


Figure : SLAVE Mode Channel Trace Data Memory Diagram

Where CH X RB Y represents Channel X Rangebin Y and CH X RB N represents Channel X Rangebin N where "N" is the number of rangebins.

6.2.2 PUSH Mode

Preamble/Header

The Preamble/Header of the data set transmitted in the PUSH mode has a fixed 32 bytes length. **Note: This header has more parameters than the slave mode header as in slave mode there is no compression and no need to synchronize.**

Dataset Marker = 0xFFFFFFFF [4 bytes]
Acquired Shots [4 bytes]
Compressed Traces [4 bytes]
Rangebins [4 bytes]
Data Time Stamp [8 bytes]
Current Sensor Value [4 bytes]
Data Compression Factor [4 bytes]

Figure : PUSH Mode Transmission Data Preamble/Header

Dataset Marker [4 bytes unsigned int] :
The Dataset Marker of the PUSH mode is the same as that of the SLAVE mode. The client must use the Data Marker as a boundary to separate the continuously incoming data sets on the PUSH Socket when the Detector is operated in the PUSH mode. It also serves as a reference to the start of current dataset.

Acquired Shots [4 bytes unsigned int] :
The `Acquired Shots` of the `PUSH` mode is the same as that of the `SLAVE` mode.

Compressed Traces [4 bytes unsigned int] :
The `Compressed Traces` is a `4` byte value used to denote the total number of compressed traces present in the acquired data after the suitable compression technique is applied.

Rangebins [4 bytes unsigned int] :
The `Rangebins` of the `PUSH` mode is the same as that of the `SLAVE` mode.

Data Time Stamp [8 bytes double] :
The `Data Time Stamp` is a time stamp in milliseconds (accurate to micro seconds) associated with each data set transmitted.

Current Sensor Value [4 bytes unsigned int] :
The `Current Sensor Value` indicates the Analog to Digital converted value of the current sensor reading which measures the current drawn by the high voltage module that powers the Photo Multiplier Tube (PMT) on board the `Lidarino` detector.

Data Compression Factor [4 bytes unsigned int] :
The `Data Compression Factor` indicates the compression factor with which the detector is going to transmit the acquired data. This value is 0 for a Lidarino.

When a `PUSH` mode acquisition is running, the detector also transmits the Data Preamble/Header with only the `Dataset Marker` and `Acquired Shots` information. All other fields of the Preamble/Header are set to `0`. This is sent to indicate to the user the current status of the internal shot acquisition during an active `PUSH` mode. Please refer to the section on `PUSH Mode Internal Shot Acquisition Status` for more information.

Channel Trace Data

The `PUSH` mode `Channel Trace Data` format is as the Slave mode for a Lidarino.

6.3 PUSH Data Time Stamps

Every dataset transmitted by the detector in the `PUSH` mode on the `PUSH Socket` has a time stamp associated with it. This is the time in milliseconds since powering on the detector. The `PUSH Data Time Stamp` can be used to monitor the interval in which the push datasets arrive at the client. For a certain laser trigger frequency and target shot number - the detector acquires data, timestamps the data, transmits the data and finally restarts itself to acquire the next acquisition cycle at fixed intervals. Hence the datasets arriving at the client must have a fixed timestamp difference between two consecutive datasets. If there is an irregularity in the difference between two consecutive time stamps even though the target shots, trace length parameters and the laser trigger frequency is kept constant , there could be a possible loss of shots.

The detector has an internal buffering mechanism which stores the data acquired in temporary buffers if the detector is not able to transmit the data fast enough and restart itself before the arrival of the next laser trigger. However if there is heavy load on the ethernet, the untransmitted data gets continuously buffered and can cause the temporary buffers to reach its limit very quickly. In such situations the subsequent datasets cannot be buffered and will over write the previous datasets in

the internal memory of the detector. When this data is finally transmitted the timestamp difference between two consecutive datasets will be twice the expected value because of the missing dataset that was overwritten.

For applications where the emphasis is on least possible shot loss, it is strongly recommended to make full use of the internal shot averaging capability. Internal data averaging ensures loss less acquisition of the required number of shots. In applications where internal averaging capability of the detector is not sufficient and external averaging is necessary, it is recommended to maximize the internal averaging and minimize the external averaging. For more information on efficient use of the detector for acquisition please refer to the section on Efficient use of SP32 for Data Acquisition in this chapter. For more information on internal and external averaging please refer to the section on Internal and External Data Averaging in this chapter.

6.4 PUSH Mode Internal Shot Acquisition Status

When the detector is operated in the PUSH mode the client effectively hands over the control to the detector. The detector sends the data sets to the client on the PUSH socket once the acquisition is completed. If the acquisition time is small (low number of shots to acquire and/or a high laser repetition rate) then the rate at which the detector sends data to the client is relatively much faster when compared to the case where the acquisition time is large (large number of shots to acquire and/or a low laser repetition rate). Hence in cases where the acquisition time is large, it is important to know the internal shot acquisition status. The detector sends the current shot acquisition status using a full sized PUSH mode Data Preamble/Header. When the PUSH mode Data Preamble/Header is used to send the internal shot acquisition status, all its fields are set to 0 except for the Dataset Marker, the Acquired Shots and the Time Stamp fields. When such a Data Preamble/Header is received it should be used only to update the shot number and time stamp information.

6.5 Standard Resolution and High Resolution Mode

If the HW? command of the controller returns the HIGHRES parameter, then the Lidarino detectors supports two modes of operation - High Resolution Mode and Standard Resolution Mode.

In the Standard Resolution Mode the detector supports a binlength varying from the lowest value of 10 ns to a maximum value in steps of 10. The maximum supported binlength is determined by the MAXBINLEN parameter of the HW? command. In the Standard Resolution Mode, the data is acquired by the controller according to the resolution and rangebin settings. Hence even if the detector is configured to acquire the maximum rangebins, the maximum trace length acquirable can be varied based on the resolution setting alone.

If a resolution less than 10 ns is required, the detector automatically switches to the High Resolution Mode. In the High Resolution Mode, the data is always acquired in the lowest binlength (highest resolution) setting which can be determined by the MINBINLEN parameter of the HW? command. Thus the maximum trace length that can be acquired in the High Resolution Mode is fixed. However, it is possible for the controller to output data with higher binlengths by combining adjacent rangebins. Hence in the High Resolution Mode the controller can output data with binlengths (MINBINLEN x 2), (MINBINLEN x 4) and (MINBINLEN x 8) by combining adjacent 2,4 and 8 bins respectively. Since the data with higher binlengths are produced by combining adjacent bins, the maximum number of rangebins acquirable decreases proportionately based on the scaling of the binlengths.

6.6 Internal and External Data Averaging

The Lidarino detector is capable of averaging up to a certain number of shots in its internal memory. This is known as Internal Averaging. The maximum number of shots it can average internally is made known to the client in its reply to the HW? command. Please refer to the HW? command in the appendices for more information. If the desired number of shots to acquire is larger than the maximum number of shots the detector is capable of averaging internally, the client has to additionally average the data externally in order to reach the desired shot number. This is known as External Averaging. This is performed by operating the detector in the PUSH mode.

The PUSH mode operation makes use of data compression mechanisms to optimize transmission. Hence when performing External Averaging using the PUSH mode, care must be taken to correctly decompress the incoming data before averaging them. A clear understanding of the data compression techniques used by the Detector to transmit data on the PUSH Socket is necessary before performing the decompression at the client side. Please refer to the PUSH Mode Transmission Data Compression for more information on these techniques.

Consider the following example: Suppose we have to acquire 4000 shots of data from the Lidarino detector. The following steps lists the brief procedure to go about it:

- Step 1 : Determine the hardware information of the detector using the HW? command. Let the reply from the detector be :

  ```
  HW: 2 500.0 8000 2 100 LE PUSH: 100 2 VARCOMP VARTRACE 2000 1000.0
  ```

  From the reply it can be concluded that the detector is a little-endian system, supports PUSH mode, supports variable compression data transmission, can average a maximum of 100 shots internally and transmits the data with a compression factor of 2. Hence in order to acquire 4000 shots a mechanism of 100 shots internal averaging and 40 datasets external averaging is required.

- Step 2 : Start the Detector to acquire 100 shots data in the push mode. The corresponding command will be :

  ```
  START 100 PUSH.
  ```

- Step 3: Read the first incoming dataset on the push socket, decompress the data section of the incoming dataset and store it in the summation array. The summation array now has just 1 dataset of 100 internally averaged shots.

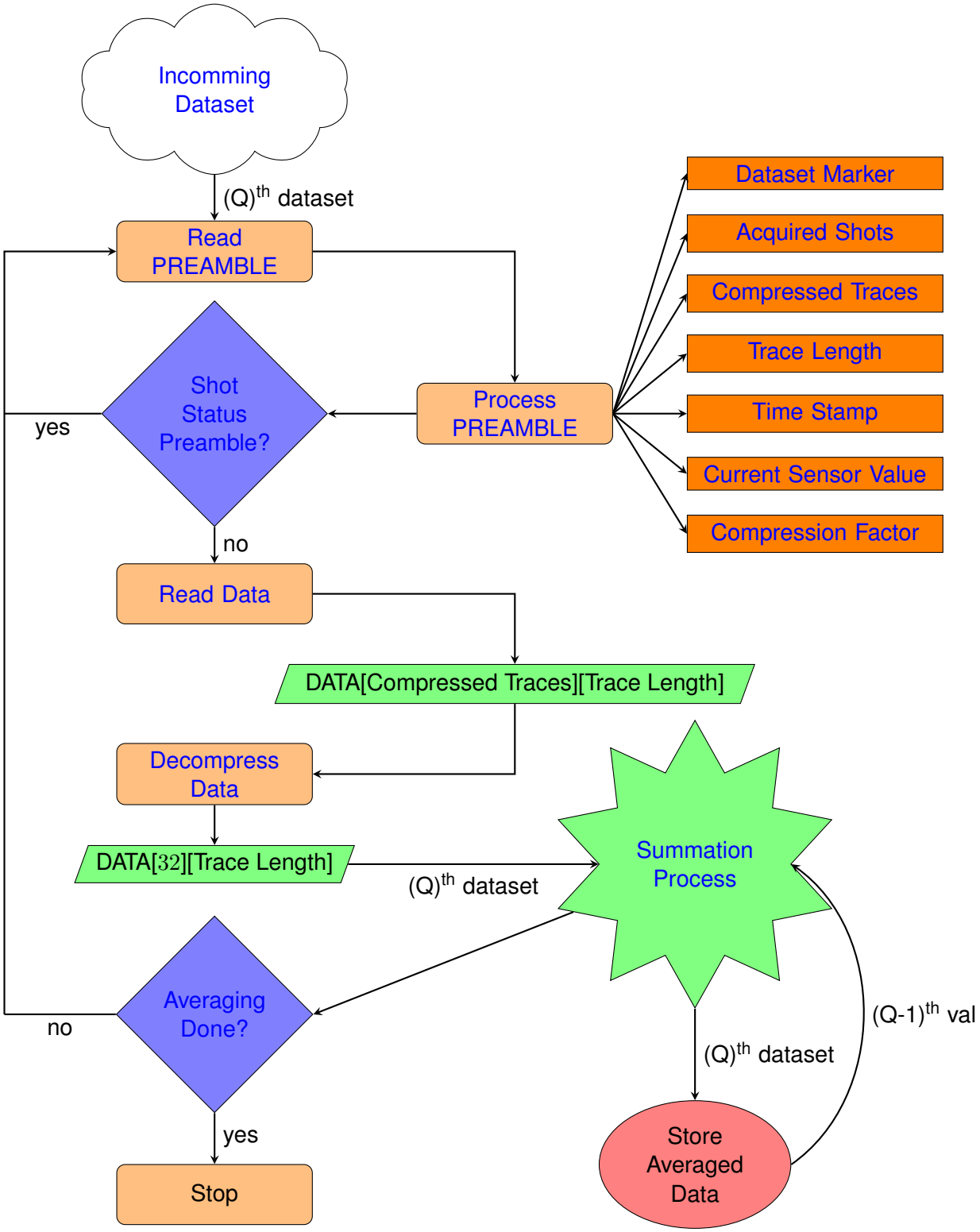
- Step 4 : Read the second incoming dataset on the push socket, decompress the data section of the incoming dataset and add it to the summation array. The summation array now has 2 datasets of 100 internally averaged shots. Hence the summation array has 200 shots of averaged data.

- Step 5 : Continue the above process for 38 more incoming datasets. Once this is finished the summation array will have 40 datasets of 100 internally averaged shots. Hence the summation array has 4000 shots of averaged data.

It is recommended to internally average with the the maximum possible shots as this is the fastest and in some cases most accurate way to acquire a given number of shots. Using a mechanism with internal averaging of shots far below its maximum possible capability and consequently increasing the number of external averaged datasets, puts heavy load on the Ethernet transmission. This could lead to possible loss of shots as the detector is not able to transfer the data fast enough to the client before restarting itself for the next acquisition cycle. This possible loss of shots can be detected by observing irregularities in the time stamps of the transmitted data sets. Please refer to the PUSH Data Time Stamps section of this chapter for more information.

6.7 Client Side External Averaging of PUSH Data

The following flowchart outlines the procedure for performing `External Averaging` by the client application.



Flowchart : External Averaging of PUSH Mode Data at the Client Application

Incomming Dataset : The continuous incoming data on the Push Socket.

Read PREAMBLE: The size of the PUSH Data Preamble is 32 bytes. The PREAMBLE must be read first and processed in order to determine the parameters needed to read the rest of the Channel Trace Data. If the received PREAMBLE has all its fields set to 0 except the Dataset Marker and the Acquired Shots, then the received PREAMBLE is just shot status indicator and not the PREAMBLE for the Channel Trace Data. This much be used only to update the shots acquired parameter at the client software. Care must be taken to determine if a byteswap is necessary based on the Endianess.

Read Data: The BINSIZE parameter from the HW? command reply along with the Compressed Traces and Trace Length parameters extracted from the PREAMBLE provides for all the parameters necessary to determine the size of the Channel Trace Data to be read.

Decompress Data: A decompression routine is needed to extract the compressed traces back into its original format where each channel corresponded to a separate trace.

Summation Process: The decompressed data is accumulated in the summation process. The $(Q)^{th}$ dataset is summed with the averaged data upto the $(Q\text{-}1)^{th}$ dataset and stored in the averaged data memory.

Averaging Done?: After the summation process , check if the required external averaging datasets are completed. If yes , stop the process by issuing a STOP command. If not read the PREAMBLE and repeat the above steps again.

6.8 Wide Memory Mode for Extended Internal Averaging

If the HW? command of the controller returns the WIDEMEM parameter, then the Lidarino detectors supports the Wide Memory Mode. The Lidarino detectors that support Wide Memory Mode can acquire large number of shots in its internal memory thereby removing the need for client side external averaging of data. In the Wide Memory Mode, the controller acquires the maximum possible shots in its internal fast SRAM and quickly transfers the data to its DDR3 memory before re-enabling itself to acquire the next cycle of shots.The trigger is blocked only during the data transfer between the SRAM and DDR3 memory. Thus this mode offers a better performance when compared to External Averaging where the trigger is blocked during the data transfer and ethernet transmission phase. Thus the Wide Memory Mode can be used to completely replace External Averaging. This also greatly reduces the complexity in data acquisition of large shots for the end user.

The wide memory mode should be activated if the target shots to acquire is larger than the maximum possible shots that can be acquired internally in the fast SRAM memory of the controller. The maximum possible shots that can be acquired in the fast SRAM memory is equal to the MAXPUSHSHOTS parameter of the HW? command. When the Wide Memory Mode is not activated and the user issues a START command with target shots greater than the MAXPUSHSHOTS, the controller responds with a failed message. When the Wide Memory Mode is activated, it changes the width of the internal controller memory from 2 bytes to 4 bytes to accommodate the accumulation of large shot numbers. Please refer to the WIDEMEM command for more information on how to activate and deactivate the Wide Memory Mode.

When the Wide Memory Mode is active, the data should be acquired in the SLAVE Mode as the

PUSH Mode is disabled by the controller. The controller responds with a failed message if the START command is sent with the PUSH flag as long as the Wide Memory Mode is active.

6.9 Efficient use of Lidarino Detector for Data Acquisition

This section explains the recommended methods for the efficient acquisition of data using the Licel Lidarino Detector.

The data acquisition process of the Lidarino detector can be divided into two steps - the internal acquisition and the data readout. During the internal acquisition the detector accepts a trigger and acquires the full trace length of the data and stores it in its internal fast SRAM. After receiving a trigger the detector blocks all other triggers until the full trace length of the data is acquired. Once the full trace length is acquired there is a dead time of 6 rangebins after which the detector will be ready to accept the next trigger. The internal acquisition process continues until the desired number of shots has been acquired.

For example if the number of rangebins is 8000 bins and the resolution is 10 ns (trace length = 12 km), the acquisition time for a single shot would be (8000 x 10 ns = 80000 ns). Factoring in the dead time of 6 rangebins between two triggers, the system will be ready to accept the next trigger after 80060 ns. This corresponds to an approximate trigger rate of 12.49 KHz.

After the internal acquisition process is finished, the data has to be readout to the slower DDR3 DRAM so that it can be prepared for ethernet transmission. The data readout is a slow process and will block the trigger for a time period that depends on the trace length and the number of shots being acquired. The internal acquisition process ensures data acquisition without loss of shots provided the acquisition time for a single shot is less than the time between two trigger pulses

The mode in which the Lidarino detector should be operated is entirely dependent on the application requirements. If the application needs a single data set of a fixed number of shots then the Lidarino should be used in the SLAVE mode. If the application needs a continuous stream of data sets at regular intervals, each containing a fixed number of shots, then the PUSH mode must be used. For example if laser repetition rate is 100 Hz (10 ms) and the user needs to acquire a continuous stream of 100 shots of data where each data set arrives at fixed intervals corresponding to the laser repetition rate (10 ms x 100 shots = 1000 ms), then it is recommended to operate the Lidarino in the PUSH mode. It is also recommended to use the PUSH mode in single shot applications or for applications that make use of the External Averaging mechanism. The PUSH mode implements an optimized data transmission mechanism which works for both the single shot acquisition and for large shot numbers, making the Lidarino detector very flexible. However due to these optimizations, the processing of the data at the client application becomes much more complicated when compared to the SLAVE mode.

For controllers that support the Wide Memory Mode, External Averaging should be avoided. Care must be taken to use the Wide Memory Mode only in applications which require the acquisition of large shot numbers. If it is used to acquire a small number of shots, the controller would transmit a lot of redundant zeroes which would be very inefficient.

The number of range bins to acquire has to be tuned for applications where a continuous stream of data sets at regular intervals, containing a fixed number of shots, is expected. If the number of range bins to acquire is very high then the data readout and transmission times are proportionally long. Since the trigger is blocked during the data readout phase the detector will miss the next trigger if the readout time is too long. In single shot applications where a continuous stream of single shot data sets are expected, the trigger is blocked during both the read out and transmission phase. The missing of triggers is more prominent in such applications as both the readout and transmission time contribute to blocking the trigger. The number of rangebins to acquire has to be tuned until the time

stamp difference between two incoming data sets reach a stable expected value. Please refer to the section on PUSH Data Time Stamps for more information on using the data time stamps to detect missed triggers (loss of shots).

The ethernet controller inside the Lidarino detector supports ethernet transfer speeds of 10Mbps, 100Mbps and 1000Mbps [1Gbps]. The link speed is auto-negotiated with the client hardware upon powering on the Lidarino during the boot process. It is highly recommended to make use of the 1Gbps transfer speed capability of the controller by using a client PC that supports 1Gbps ethernet speed. If the Lidarino detector is connected to the client PC via a switch box/hub, make sure the switch box/hub supports 1Gbps speed. This ensures the best performance and response time of the system, especially in applications where data transfer rates are crucial.

Chapter 7

Simulation of a Multichannel or Single LIDAR Detector

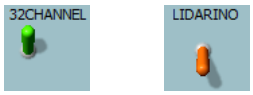
The simulation program *Virtual Controller* is — as for other Licel Ethernet Controllers — available as part of the Windows Applications. Start the *Virtual Controller* using the corresponding entry in the Windows start menu. The *Virtual Controller* is not included in the LabVIEW sources.
The *Virtual Controller* is capable to simulate most of the functions of the Licel Multispectral Lidar Detector and of the Licel Single Channel Lidar Detector [Lidarino]. This will help a user to develop and test his own software as debugging is possible without having the hardware installed.
The *Virtual Controller* is accessed via the IP address of the PC where the program is running and the port specified in the initialiazation file. If your client software is running on the same PC as the *Virtual Controller* you may use `127.0.0.1` (localhost) as IP address.

7.1 Virtual Controller – General

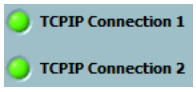
After starting the *Virtual Controller* the main page *General* is shown.

Here you may

- Switch between the Licel Multispectral Lidar Detector and the Licel Single Channel Lidar Detector. If none of these switches are active you will receive error messages when communicating with the *Virtual Controller*.



- Inspect the TCPIP connection(s). The second connection should be active if the *Virtual Controller* simulates a push mode controller and a client is connected.



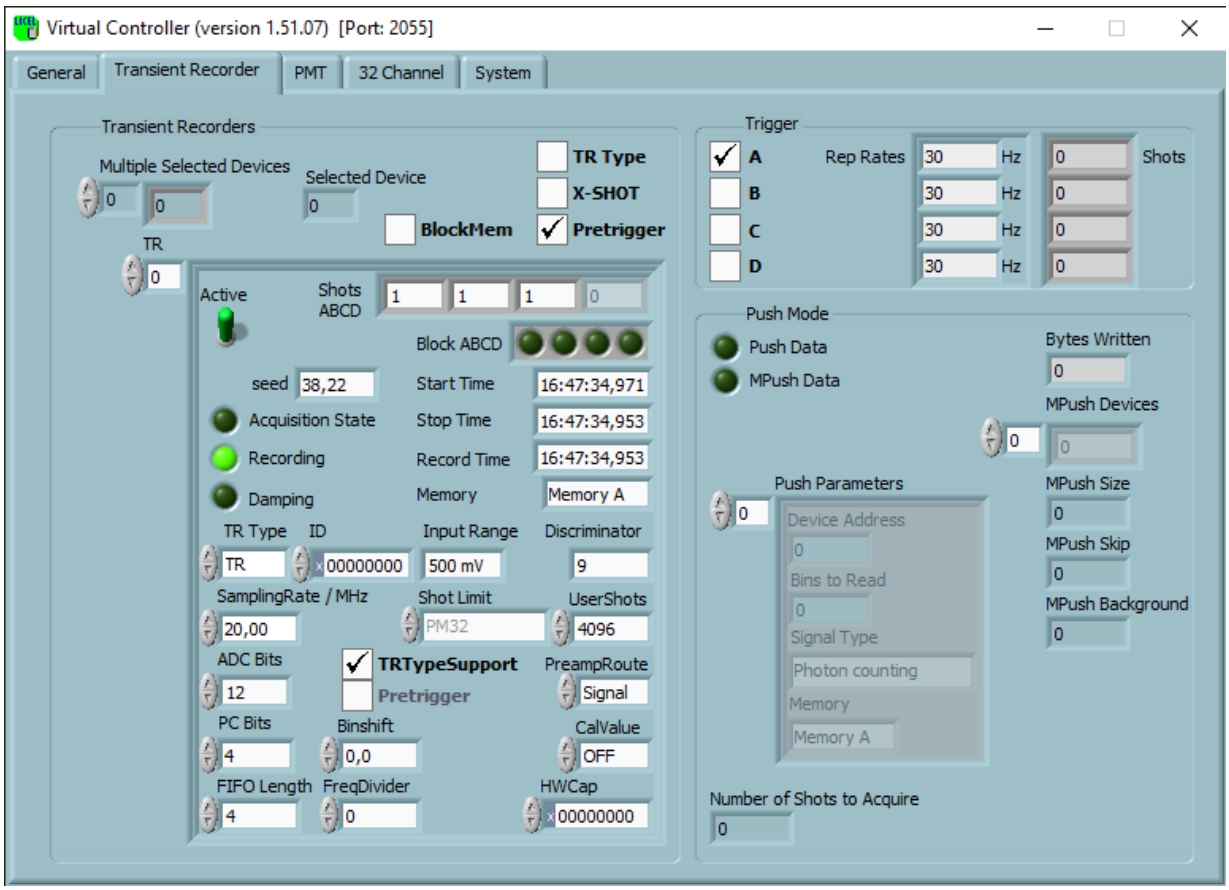
- Switch off the *Virtual Controller*.

  You may restart it using LabVIEW's run button 

- Close the *Virtual Controller*'s window as usual with the Windows close button ✕ at the top right of the window.

7.2 Virtual Controller – Transient Recorder

Internally, the *Virtual Controller* uses the same mechanism to simulate Lidar signal as it does for transient recorders. Because of this the *Transient Recorder* page is accessable.

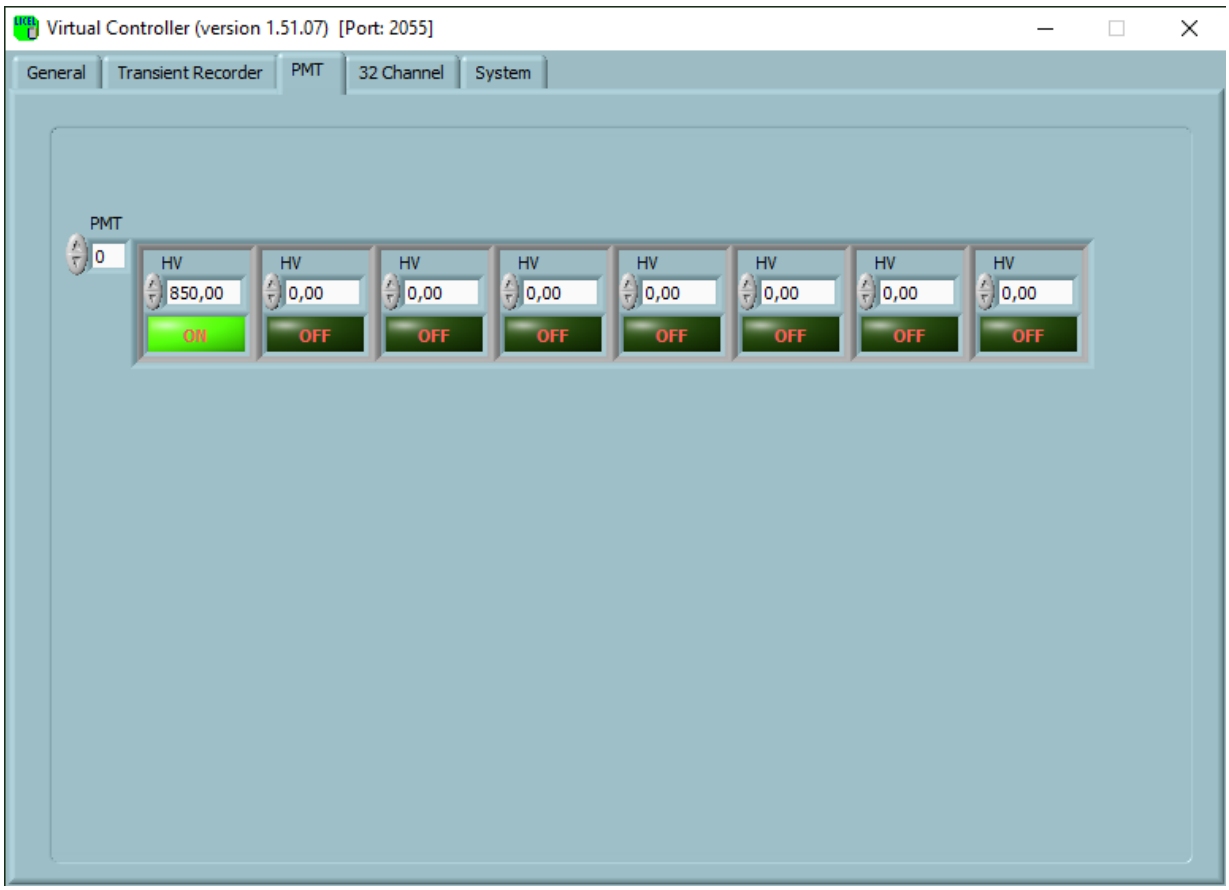


Please do not modify anything on this page except for the change of the *Rep*etition *Rate* of *Trigger A*:

Note the current shot number during an acqusiition is displayed on the page *32 Channel*.

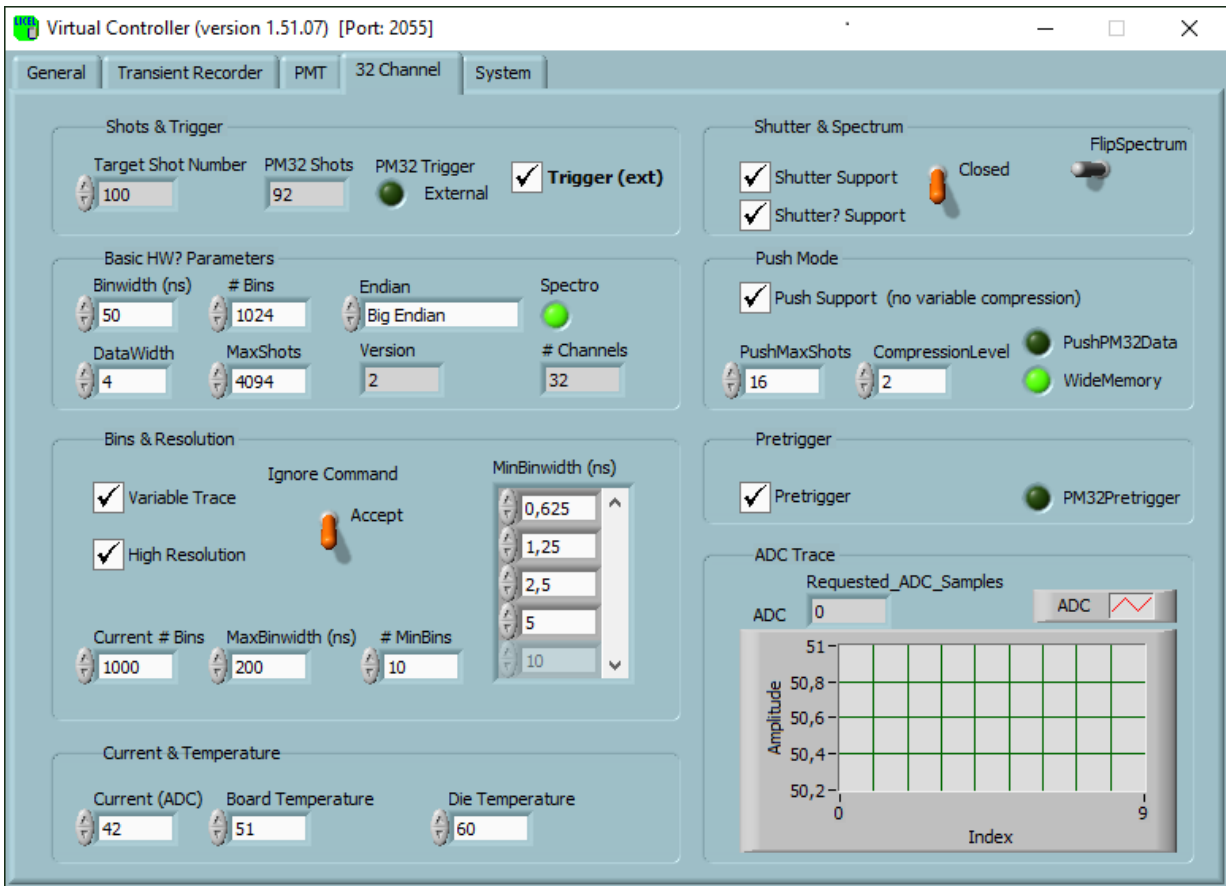
7.3 Virtual Controller – PMT

On the *PMT* page you may control that the high voltage setting is working. Please not that only one *PMT* indicator should be active.



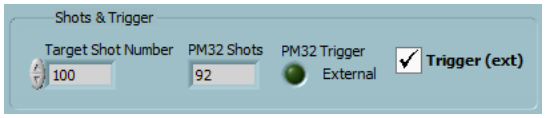
7.4 Virtual Controller – 32 Channel

The page *32 Channel* is the most important page for the simulation of the Multichannel or Single LIDAR Detector.



Shots & Trigger

In the *Shots & Trigger* section



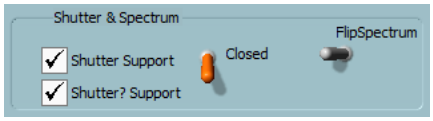
you will

- see the *Target Shot Number* received with the last `START` command.
- see the current shot number *PM32 Shots*. It updates whenever the client software requests the status (STAT?) or when data is send via the push socket.
- inspect wether or not the client software has set the trigger mode to *Internal* (green LED) or *External* (dark LED)using the `SIM` command.
- manually set the external trigger using the *Trigger (ext)* checkbox. Checking or unchecking the bos is a simulation for plugging and unplugging a trigger cable to the controller.

The current shot number will change only if the *Trigger (ext)* is checked or the *PM32 Trigger* is set to *Internal*.

Shutter & Spectrum

Here, the simulation of the mechanical shutter and the spectrum orientation are set.



- *Shutter Support* determines whether or not the control of a mechanical shutter is supported i.e. the switching using the `SHUTTER` command. The switch *Closed*|*Open* will show the shutter position according to a received `SHUTTER` command.
- *Shutter? Support* determines whether or not the current shutter position can be requsted using the `SHUTTER?` command.
- *Flip Spectrum* determines whether or not the simulated spectrum is flipped i.e. the assignment to the 32 channels is reverted. This option can be used to match the simulated output to the exit of your spectrometer. The option has no effect on the simulation of a *LIDARINO*.

Basic HW? Parameters

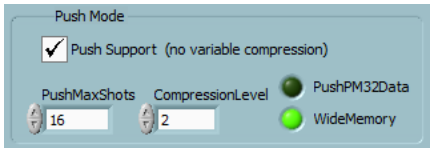
In the *Basic HW? Parameters* section

you will see basic parameters most of them included in responses to the `HW?` command:

- The LED *Spectro* should be light green when either the *32CHANNEL* or the *LIDARINO* on the first page are active. If not many commnds will return `unkown command`.
- *Binwidth (ns)* is the current binwith in nanoseconds. Please note that this value can be changed when a *High Resolution* conroller is simulated and the `RANGEBINS` command is used.
- *# Bins* is the maximum number of bins the controller is able to return.
- The *Endian* defines the byte order (endianness) of the data bytes when sending data (*Big Endian* or *Little Endian*).
- The *Data Width* defines the width in bytes of the data. Note that the *Data Width* can be changed using the `WIDEMEM` command.
- *MaxShots* contains the maximum number of shots.
- *Version* contains the `HW?` version and depends on the *High Resolution* setting: unchecked: version = 1, checked: version = 2.
- the number of channels *# Channels* depends on whether *32CHANNEL* (=32) or *LIDARINO* (=1) are active.

Push Mode

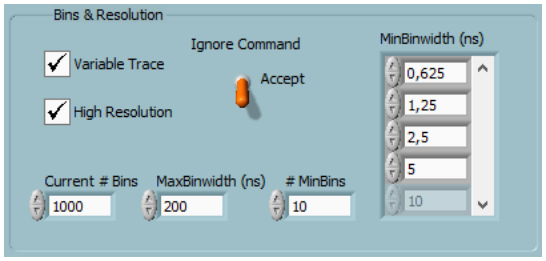
In this section some settings related to a push mode simulation may be set.



- The checkbox *Push Support* enables the simulation of the push mode. Please note that variable compression is not supported.
- The *PushMaxShots* describe the maximum number of shots for using the push mode. Once the (global) maximum allowed shot number *MaxShots* is larger than the *PushMaxShots* the *DataWidth* can be modified using the `WIDEMEM` command.
- The *CompressionLevel* can be set for the simulation in this section, as well.

Bins & Resolution

To simulate more options especially for HR spectrometer controllers the following control fields are available in the *Bins & Resolution* section.



Here you are able to

- enable the *Variable Trace* option. If enabled, the current number of bins *Current # Bins* and the maximum binwidth *MaxBinwidth (ns)* can be changed using the `RANGEBINS` and `RESOLUTION` commands, respectively. With the *Ignore Command* switch in up position you may simulate a situation where the controller ignores a command.
- enable the *High resolution* option for the activation of the minimum bin number *# MinBins* as the lower limit in the `RANGEBINS` command and the values in the list of *MinBinwidth (ns)* as the lowest allowed values in the `RESOLUTION` command.

Pretrigger

Here, the simulation of setting the pretrigger is available.

- The activation of the `PRETRIG` command simulation is driven by the checkbox *Pretrigger*.
- The LED *PM32Pretrigger* reflects the received pretrigger setting. Please note that the pretrigger setting has no influence on the simulated data.

Current & Temperature

In the section *Current & Temperature*

the following input fields are available:

- the *Current (ADC)* is the simulated ADC value of the on board high voltage supply current sensor which is returned with the `CURRENT?` command as well as part of the `STATUS?` and of the push data header described in the context of the `START PUSH` command.

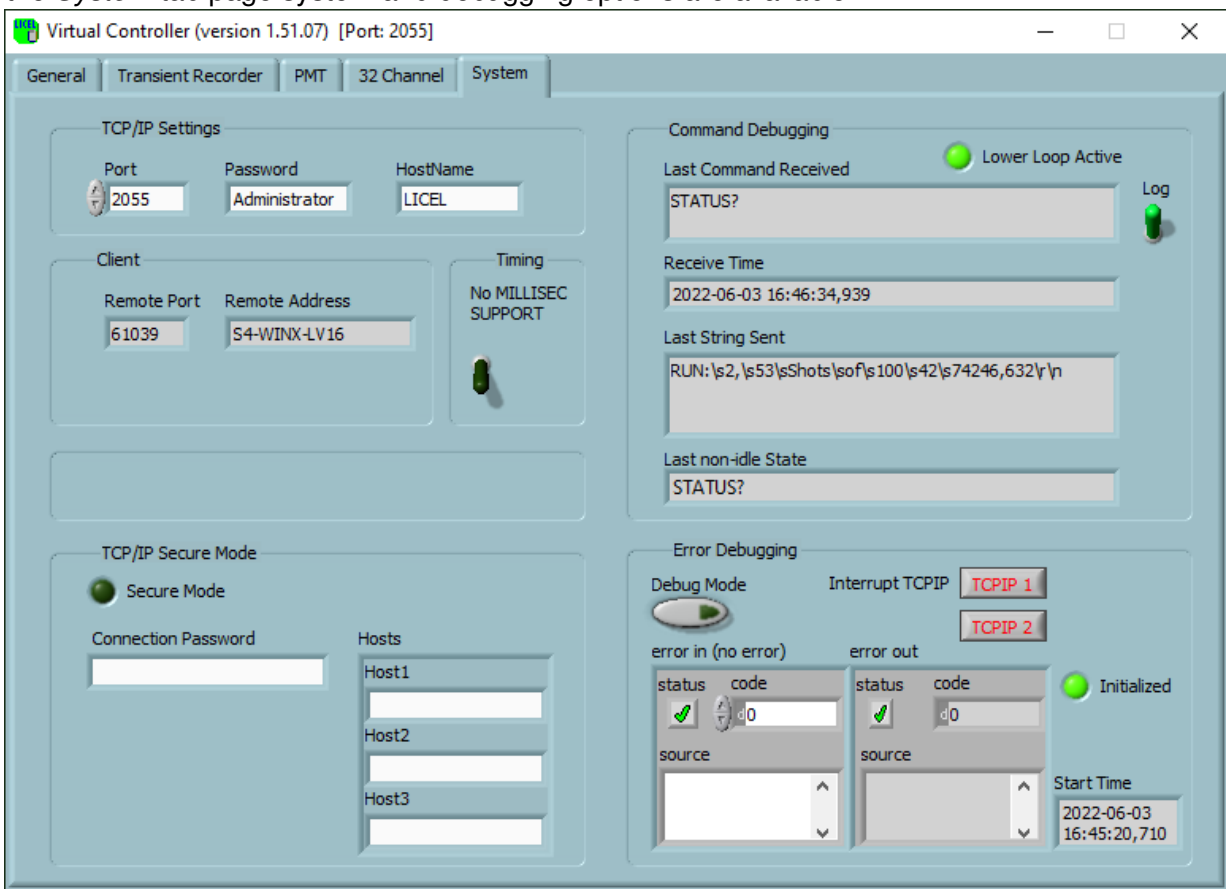
- the *Board Temperature* is the simulated board temprature returned with the `TEMP?` command.
- the *Die Temperature* is the simulated die temprature returned with the `DIETEMP?` command.

ADC Trace

This is an internal section for later use.

7.5 Virtual Controller – System

On the *System* tab page system and debugging options are available.



TCP/IP Settings In this section the **Port** and the **Password** for the Virtual Controller may be changed. Note that **Port** is used directly for the command socket (TCPIP Connection 1), and **Port**+1 is the port for the push socket (TCPIP Connection 2). The **Password** is the administrator password of the Virtual Controller which is needed to be sent with certain commands like the command to change the IP address. Note that a change of the IP address of the Virtual Controller is not possible, as the IP address is set in the computer's system setup. The **Hostname** is used to simulate Ethernet Controllers with the corresponding capability.

Client When a client is connected to the Virtual Controller it's **Remote Port** and **Remote Address** are displayed here.

Timing Here, the millisecond timer support (MILLISEC? command) of the Virtual Controller can be switched off to simulate very old Ethernet controllers.

TCP/IP Secure Mode This section is not used by the *Virtual Controller* in the simulation of *32Channel* or *LIDARINO* controllers.

Command Debugging This part of the front panel contains information about the TCP/IP communication flow (**Last Command Received**, the corresponding **Receive Time** and **Last String Sent**), and about the loop activities of the Virtual Controller (**Last non-idle State** and **Lower Loop Active**). **Last non-idle State** describes the internal state of the command socket loop of the program while **Lower Loop Active** indicates that the push socket loop to handle requests on **Port**$+1$ is alive. This indicator should always light in a green color. Switching the **Log** switch to the up position will make the *Virtual Controller* write all received commands to a log file named `Virtual Controller.exe.log`.

Error Debugging This section contains the standard LabVIEW error clusters. If **Debug Mode** is set errors in the Virtual Controller are shown. Timeout errors (code=56) are normal during operation. The buttons **Interrupt TCPIP** (**TCPIP 1** and **TCPIP 2**) may be used to interrupt the TCP/IP connections to inspect the behavior of an application to such an event. This is especially useful if a mechanism to reconnect has been implemented to the application when TCP/IP relevant errors occur. The LED indicator **Initialized** shows whether or not the program has been initialized.

Chapter 8

SP32 Controller TCPIP Command List and Syntax

8.1 TCP/IP Command List and Syntax

This section lists and describes the TCPIP command syntax for the Licel TCPIP Ethernet Controller present inside the Licel `Lidarino` Detector. Most commands can be sent either in its short or long format. All commands sent to the controller should end with `<CRLF>`(carriage return line feed), and all replies from the controller will end with `<CRLF>`. This will not be shown explicitly in this document.

If an unknown command is sent to the controller, then the controller sends

`<COMMAND>unknown command`

back to the client where `<COMMAND>` is the command sent by the client to the controller.

Please Note: Through out this document `HOST` refers to the the Ethernet Controller of the Licel `Lidarino` Detector and `CLIENT` refers to the PC Acquisition System

Short	Long
HW?	HARDWARE?
CAP?	CAPABILITY?
CURRENT?	CURRENT?
DATA?	DATA?
DIETEMP?	DIETEMP?
DISC	DISCRIMINATOR
IDN?	IDENTIFICATION?
MSEC?	MILLISEC?
PASS	PASSWORD
PMT?	PMTSTATUS?
PMTG	PMTGAIN
PRETRIG	PRETRIG
RES	RESOLUTION
RANGE	RANGEBINS
SHUTTER	SHUTTER
SHUTTER?	SHUTTER?
SIM	SIM
START	START
STOP	STOP
STAT?	STATUS?

TEMP?	TEMP?
WIDEMEM	WIDEMEMORY
TCPIP	TCPIP

HARDWARE?

HW?

Requests the hardware capabilities of the Lidarino detector. The format of the reply from the controller is as follows :

```
HW: HWREV BINLEN MAXRANGEBINS BINSIZE MAXSHOTS ENDIANNESS
PUSH: MAXPUSHSHOTS CMPFACTOR VARCOMP VARTRACE CURRENTRANGEBINS
MAXBINLEN HIGHRES: MINBINLEN MINRANGEBINS WIDEMEM
```

HWREV is the hardware revision number of the detector.
BINLEN is the current resolution/binlength (in nano seconds) setting of the detector.
MAXRANGEBINS is the maximum acquisition range bins.
BINSIZE is the size (in bytes) of each range bin.
MAXSHOTS is the maximum number of shots the detector is capable of internally averaging.
ENDIANNESS is the endianness of the system (big-endian/little-endian)(BE/LE).
PUSH indicates that the detector supports MASTER/PUSH mode.
MAXPUSHSHOTS is the maximum number of shots the detector is capable of averaging internally in the PUSH mode. This is also equal to the maximum number of shots the detector can acquire in its fast SRAM.
CMPFACTOR is the compression factor of the PUSH mode data.
VARCOMP indicates that the detector supports variable compression.
VARTRACE indicates that the detector supports variable trace length.
CURRENTRANGEBINS is the current rangebins setting of the detector.
MAXBINLEN is the maximum binlength (minimum resolution) of the detector.
HIGHRES indicates that the detector supports HIGH RESOLUTION mode.
MINBINLEN is the minimum binlength (highest resolution) of the detector.
MINRANGEBINS is the minimum number of rangebins of the detector.
WIDEMEM indicates that the detector supports wide memory summation (extended internal averaging).

An example would be :

```
HW: 2 50.0 8000 2 10000 LE PUSH: 100 2 VARCOMP VARTRACE 2000 1000.0
HIGHRES: 0.625 100 WIDEMEM
```

Please Note:

- The HW? command is appended with new features as they are added to the Lidarino detector. Hence older detectors may not support all of the above mentioned features.
- Data compression techniques are applied only when the detector is operated in the PUSH mode. For more information about data compression please read the section on PUSH Mode Transmission Data Compression.

CAPABILITY?

CAP?

Requests the capabilities of the ethernet controller.
The controller's response is:

```
CAP: Capability
```

Where `Capability` is the fixed expression `32CHANNEL` in the case of the Licel `Lidarino` Detector. The response is :

```
CAP: Lidarino
```

CURRENT?

Explicitely return the ADC value of the on board high voltage supply current sensor (no decimal places).
The controller's response is for example:

```
Current:  42.
```

DATA?

Requests data from the controller.
Sending a `DATA?` command will instruct the controller to transmit the acquired data in the following *binary format*

Data Preamble/Header :

`Fixed Marker = 0xFFFFFFFF`	4 byte unsigned int
`Number of Shots`	4 byte unsigned int
`Number of Traces`	4 byte unsigned int
`Number of Bins`	4 byte unsigned int

Acquired Data :

`DATA [number of traces] × [number of bins]`

Different to the response of the other commands, no <CRLF> is added here. The acquired data transmitted by the controller in response to the `DATA?` command is always in the uncompressed format. This is different from the transmitted data in the `PUSH` mode which makes use of data compression techniques to optimize data transfer.

DIETEMP?

Explicitely return the die temperature in ℃.
An example reply would be:

```
DIETEMP: 55.000000.
```

DISCRIMINATOR <Integer>

DISC <Integer>

Sets the discriminator level of the detector. Valid values for the discriminator are 0–63.

An Example : To set the discriminator level to 16, send the command :

```
DISCRIMINATOR 16
```

The corresponding reply from the controller is:

```
DISCRIMINATOR set to 16
```

If the `Integer` value is out of range the reply is

```
DISCRIMINATOR Failed.  Value out of range
```

IDENTIFICATION?

IDN?

Requests the controller to send its identity and firmware revision. An example reply is as follows :

```
Licel Zynq MercuryZX1 SP32-HR Rev 2 Ethernet Controller Rev-03.08.2017-Release
```

MILLISEC?

MSEC?

Requests the millisecond timer value of the ethernet controller. The corresponding reply is :

```
MILLISEC: time
```

Where `time` is a number (in milliseconds) since the powering on of the ethernet controller expressed in a double precision format.

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

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

Changes the password of the ethernet controller. The actual password is required to change the IP configuration of the ethernet controller. 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 controller.
An example would be :

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

This will change the password to MyPassword. The corresponding reply from the controller is :

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

If an error occurs (wrong `Old Password`, non-matching `New Password` entries, or empty `New Passwords`) the reply is :

```
PASSWORD not set
```

PMT? <Device Number>

PMTSTATUS? <Device Number>

Returns the status of the PMT with the specified device number. The reply parameters are <HV value in Volts> <HV on/off> <local/remote>. The PMT device number for the `Lidarino` detector is fixed to `0` and does not support any other device number.
An example to request the status of the PMT would be :

```
PMT? 0
```

An example corresponding reply from the controller could be :

```
PMT 970 on remote
```

Which indicates that the PMT is in remote mode, the HV power supply is on and is set to 970 Volts.
Another example reply could be:

```
PMT 0 off remote
```

Here, the HV power supply is off and set to `0` , which is the default return value when the PMT is off.
If a request for the status of the PMT with a non-zero device number is made, the reply is :

```
PMT 5 is not available
```

Where the number 5 is the device number of the non-existent PMT.

PMTG <Device Number> <Voltage>

PMTGAIN <Device Number> <Voltage>

This command sets the gain voltage applied to the dynodes of the PMT with the specified device number. The PMT device number for the `Lidarino` detector is fixed to `0` and does not support any other device number.
An example would be :

```
PMTG 0 980
```

This will set the gain voltage of the PMT with device number `0` to `980` volts. The corresponding reply from the controller is :

```
PMTG executed
```

If a non-zero device number is specified, the reply is :

```
PMT 3 is not available
```

Where the number 3 is the device number of the non-existent PMT.

PRETRIG <ON|OFF>

Enables (`PRTRIG ON`) or disables (`PRETRIG OFF`) the prtrigger. The command is available only if setting the pretrigger is allowed. The reply by the controller is

```
PRETRIG executed.
```

RES <value>

RESOLUTION <value>

This command sets the resolution at which the detector must acquire the data. Please refer to the section on `Standard Resolution and High Resolution Mode` for more information about the resolution settings supported by the controller.

An example would be :

```
RESOLUTION 50
```

This instructs the controller to change the resolution of the detector to `50` ns. If the controller changes the detector resolution successfully, then the reply from the controller is :

```
RESOLUTION executed
```

If an invalid resolution setting is received by the controller, it ignores the command and outputs the corresponding error message :

```
RESOLUTION ignored.<Error Message>
```

Please Note: Only controllers that display the `VARTRACE` parameter in the `HW?` command reply support variable resolution settings. Older controllers without this feature have a predetermined fixed resolution setting.

RANGE <value>

RANGEBINS <value>

This command sets the number of rangebins the detector must acquire. The highest and the lowest number of rangebins acquirable is determined by the `MAXRANGEBINS` and `MINRANGEBINS` parameters of the `HW?` command reply respectively.

An example would be :

```
RANGEBINS 4000
```

This instructs the controller to change the total rangebins to acquire to `4000`. If the controller changes the rangebins successfully, then the reply from the controller is :

```
RANGEBINS executed
```

If an invalid rangebin setting is received by the controller, it ignores the command and outputs the corresponding error message :

```
RANGEBINS ignored.<Error Message>
```

Please Note: Only controllers that display the `VARTRACE` parameter in the `HW?` command reply support variable rangebin settings. Older controllers without this feature have a predetermined fixed rangebin setting.

SHUTTER <OPEN|CLOSED>

Opens or closes the mechanical shutter of the spectrometer. This command is available as an option. The reply by the controller is

```
SHUTTER executed.
```

SHUTTER?

Requst the current position of the mechanical shutter. This command is available as an option. The reply by the controller is

`SHUTTER 1` if the shutter is `OPEN` or `SHUTTER 0` if the shutter is `CLOSED`.

SIM <ON|OFF>

Sets the trigger mode to *internal* (simulate a trigger, `SIM ON`) or *external* (`SIM OFF`). The reply by the controller is

```
SIM executed.
```

START <target shot number> [PUSH] / [TRANSMIT]

Starts an acquisition of the detector with `target shot number` shots to acquire. The `START` command can be used in 3 different ways. It can be sent without any optional `PUSH` / `TRANSMIT` flags in which case the command just starts the acquisition of the detector for the `target shot number`. An example of this command would be :

```
START 16
```

This will instruct the controller to start the acquisition of 16 shots.

The `START` command can be sent with either of the optional flags in which case the controller will show modified behavior.

When the `START` command is sent with the `PUSH` flag, it activates the `PUSH` mode mechanism of the controller. In this mode the data is sent automatically on the `PUSH Socket` once the `target shot number` has been acquired and restart itself to acquire the next set of `target shot number`.

Different to the data response sent for the `DATA?` command, the data preamble/header sent on the `PUSH Socket` has :

- An additional time stamp information in milliseconds (accurate to micro seconds) for each of its data sets.
- ADC value of the on board high voltage supply current sensor.
- Compression factor of the data.

The acquired data format is transmitted using compression techniques to optimize data transmission.
The data sent on the PUSH Socket is in the following binary format :

Data Preamble/Header :

Fixed Marker = 0xFFFFFFFF	4 byte unsigned int
Number of Shots	4 byte unsigned int
Number of Traces	4 byte unsigned int
Number of Bins	4 byte unsigned int
Data Time Stamp	8 byte double
Current Sensor Value	4 byte integer
Data Compression Factor	4 byte integer

Acquired Data :

DATA [no of compressed traces] × [no of bins]

An example of this command would be :

```
START 16 PUSH
```

This will instruct the controller to start acquisition of 16 shots with the PUSH mode enabled.

When the START command is sent with the TRANSMIT flag, the data is sent automatically on the COMMAND Socket once the target shot number has been acquired. Different to the Push mode (that is activated by the PUSH flag) the TRANSMIT flag does not restart the detector automatically to acquire the next set of target shot number. The data response for this command is exactly the same as the response for the DATA? command.
An example of this command would be :

```
START 1 TRANSMIT
```

This will instruct the controller to start acquisition of 1 shot and transmit the data automatically on the COMMAND Socket.
The reply from the controller is the same for all the variants of the START command and is :

```
START executed
```

Please Note: Once the controller is started in the PUSH Mode, only the STOP command can bring the controller back into the SLAVE Mode.

STOP

Stops the data acquisition of the detector. The corresponding reply from the controller is :

```
STOP executed
```

Please Note: The STOP command puts the controller back into the SLAVE Mode.

STAT?

STATUS?

Returns the current status of the controller. The format of the reply is :

RUN: AcqStatus, ShotNum Shots of TargetShotNum Current Timestamp

AcqStatus is the acquisition status of the detector, which can be either 0 (Idle),
1 (Armed), or 2 (Acquiring).
ShotNum is the current shots acquired.
TargetShotNum is the target shots to acquire set by the START command.
Current is the ADC value of the on board high voltage supply current sensor.
Timestamp in milliseconds (accurate to micro seconds) since powering on the controller.

An example reply would be:

```
Run:  2, 10 Shots of 100 65535 9070.000000
```

TEMP?

Explicitely return the board temperature in ℃.
An example reply would be:

```
Temperature:  51.000000.
```

WIDEMEMORY < 0 | 1>

WIDEMEM < 0 | 1>

Sending the command `WIDEMEM` along with the parameter `1` enables the Wide Memory Summation Mode of the controller (if it is supported). The parameter `0` disables it.

The format of the reply is :

```
WIDEMEM DataWidth
```

Where `DataWidth` is the size in bytes of the incoming data.

An example command would be

```
WIDEMEM 1
```

The corresponding reply from the controller would be :

```
WIDEMEM 4
```

This indicates that the `WIDEMEM` command was accepted and the data size is changed to `4` bytes.

TCPIP <"ip#"> <"subnet mask"> <"Gateway"> <"Port"> <"Password">

TCP <"ip#"> <"subnet mask"> <"Gateway"> <"Port"> <"Password">

Sets the IP address, subnet mask, gateway and ports that are used for TCP connections. Please note that the port number, port number+1 and port number+2 are used by the controller. This command will only be executed if the password corresponds with the controller's internally stored password. The defaults settings are :

IP Address :	10.49.234.234
Subnet Mask :	255.255.255.0
Gateway :	empty
Port :	2055 .

In this case the ports 2055, 2056 and 2057 are used by default. The Port 2055 is used for the bidirectional communication with the controller and is called the `COMMAND Socket`. The port 2056 is used for the unidirectional PUSH mode data transfer from the controller and is called the `PUSH Socket`. 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 would be :

```
TCPIP "197.13.17.23" "250.250.250.29" " " "2013" "Administrator"
```

This 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 ports 2013, 2014 and 2015 would be used.
The corresponding reply from the controller is :

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

If the password is incorrect, the reply is :

```
TCPIP failed due to invalid password
```

TCPIP ”DHCP” <”Port”> <”Password”>

TCP ”DHCP” <”Port”> <”Password”>

Enable DHCP mode on the network controller. The controller will listen at the specified Port, Port+1 and Port+2. This command will only be executed if the password corresponds with the controller's internal password. If not the controller replies with the following response :

```
TCPIP failed due to invalid password
```

If the command is successfully executed the controller replies as follows :

```
DHCP activated
```

The controller comes up with the defaults described in the `TCPIP IP` command.

Please Note: A hardware reset will disable the DHCP mode.

8.2 TCP/IP Command Logging

In Licel's Windows Applications and by default in Licel's LabVIEW sources the logging of the TCP/IP comands and replies is enabled. To achieve that in the LabVIEW sources the *Conditional Disable Symbol* `CMDLOG` is set `TRUE` in the LabVIEW project.
There are two ways to activate TCP/IP command logging:

1. Activate the command logging by entering or changing the following lines in the initialization file `LicelTCPIP.ini`:
   ```
   [LogCmd]
   LogCmd = TRUE
   ```
   The command logging will then be active from the start of a program.
2. Activate or deactivate the command logging by using the TCP/IP API command `CMDLOG`.

Chapter 9

SP32 C Sources/Drivers

9.1 `Lidarino` Detector C Drivers

This section describes the C Drivers/Sources that are provided for the `Lidarino` detector. Shown below is the directory structure of the `Lidarino` C Drivers.

```
SP32-C-Drivers
├── licel_sp - Core Algorithm routines
│   ├── Common Routines
│   │   ├── Licel_SP_Startup_Seq
│   │   ├── Licel_SP_Shutdown_Seq
│   │   ├── Licel_SP_Read_Preamble
│   │   └── Licel_SP_Write_Data2File_Seq
│   ├── SLAVE Mode Routines
│   │   ├── Licel_SP_SlaveMode_Main
│   │   ├── Licel_SP_SlaveMode_Acquisition_Seq
│   │   ├── Licel_SP_SlaveMode_DataRead_Seq
│   │   └── Licel_SP_SlaveMode_Read_Dataset
│   └── PUSH Mode Routines
│       ├── Licel_SP_PushMode_Main
│       ├── Licel_SP_PushMode_Acquisition_Start
│       └── Licel_SP_PushMode_Read_Dataset
├── licel_sp_util - Data processing utility routines
│   ├── Licel_SP_UTIL_Write_AdvancedViewerData
│   ├── Licel_SP_UTIL_Reconstruct_SlaveMode_2Byte_TraceData
│   ├── Licel_SP_UTIL_Reconstruct_SlaveMode_4Byte_TraceData
│   └── Licel_SP_UTIL_Decompress_PushData
└── licel_sp_tcpip - TCP/IP command routines
    ├── Licel_SP_TCPIP_Get_HWConfig
    ├── Licel_SP_TCPIP_Start_Acq
    ├── Licel_SP_TCPIP_SlaveMode_Get_AcqStatus
    ├── Licel_SP_TCPIP_SlaveMode_Get_Data
    ├── Licel_SP_TCPIP_Stop_Acq
    ├── Licel_SP_TCPIP_Set_Resolution
    ├── Licel_SP_TCPIP_Set_Rangebins
    ├── Licel_SP_TCPIP_Enable_Wide_Accumulation_Memory
    └── Licel_SP_TCPIP_Dissable_Wide_Accumulation_Memory
```

In order for the driver functions to access the numerous parameters, three structures are used to

store and keep track of the corresponding parameters. These are :

- Spectro_Application - Used to store the application acquisition parameters.
- Spectro_Measurement - Used to store the application measurement parameters.
- Spectro_Hardware - Used to store the controller's hardware configuration information.

9.1.1 licel_sp

This category contains block level algorithm functions that implement the flow charts illustrated in the Data Acquisition Overview section. They are categorized into Common Routines, SLAVE Mode Routines and PUSH Mode Routines. All the functions in this category return 0 if successful or a negative number in case of an error.

Common Routines :

This category contains routines that are common to both the PUSH Mode and the SLAVE Mode of the controller.

- **Licel_SP_Startup_Seq**

  Syntax :

  ```
  int Licel_SP_Startup_Seq( SOCKET cmd_soc,
  Spectro_Application *sp_app, Spectro_Hardware *sp_hw )
  ```

  Description :

  The startup sequence function performs the following operations :

  - Stops the controller using the STOP command.
  - Gets the software id of the controller using the IDN? command.
  - Gets the hardware information of the controller using the HW? command and assigns the parameters to the Hardware Parameters Structure.
  - Assess the validity of the rangebins and resolution settings in the Application Parameters Structure with that of the Hardware Parameters Structure. If the controller supports variable trace then the required rangebins and resolution are set else the default hardware parameters are used.
  - Sets the discriminator level of the controller using the DISC command.
  - Sets the PMT high voltage level of the controller using the PMTG command.

- **Licel_SP_Shutdown_Seq**

  Syntax :

  ```
  int Licel_SP_Shutdown_Seq( SOCKET cmd_soc )
  ```

  Description :

  The shutdown sequence performs the following operations :

  - Stops the controller using the STOP command.
  - Turns off the PMT high voltage by setting it to 0.
  - Disables the wide accumulation memory.

- **Licel_SP_Read_Preamble**

  Syntax :

  ```
  void Licel_SP_Read_Preamble( SOCKET cmd_soc, SOCKET push_soc
  Spectro_Hardware *sp_hw, Spectro_Application *sp_app,int mode )
  ```

  Description :

  This function is used to read the Preamble/Header data of the controller in both the Slave Mode and Push Mode. The parameters extracted from the preamble are used to dynamically set up the memory traces required to read the rest of the incoming data.

- **Licel_SP_Write_Data2File_Seq**

  Syntax :

  ```
  int Licel_SP_Write_Data2File_Seq( Spectro_Measurement *sp_mes,
  Spectro_Application *sp_app, unsigned int *advanced_viewer_data )
  ```

  Description :

  The write data to file sequence performs the following operations :

  – Prepares the name of the data file to write the data.
  – Writes the acquired data into the file in the Licel Advanced Viewer Data File format. The file name and the data format is described in the section Data File format

  Slave Mode Routines :

  This category contains routines that are specific to the SLAVE Mode operation of the controller.

- **Licel_SP_SlaveMode_Main**

  Syntax :

  ```
  int Licel_SP_SlaveMode_Main( SOCKET cmd_soc,
  Spectro_Hardware *sp_hw, Spectro_Application *sp_app,
  Spectro_Measurement *sp_mes, unsigned int *advanced_viewer_data )
  ```

  Description :

  This is the main function for all the SLAVE Mode routines. The slave mode routines consists of two main sequences namely Acquisition Sequence and Data Read Sequence.

- **Licel_SP_SlaveMode_Acquisition_Seq**

  Syntax :

  ```
  int Licel_SP_SlaveMode_Acquisition_Seq( SOCKET cmd_soc,
  Spectro_Hardware *sp_hw, Spectro_Application *sp_app )
  ```

  Description :

  The slave mode acquisition sequence function performs the following operations :

  – Checks the number of shots to acquire. If the required shots is greater than the maximum shots acquirable the application turns on the wide summation memory of the controller. If wide summation memory is not supported the application adjusts the target shots to the maximum possible value.
  – Starts the Lidarino in SLAVE Mode to acquire the target number of shots.

- Polls the status of the acquisition using the STAT? command until the acquired shots is equal to the target shot number.

- **Licel_SP_SlaveMode_DataRead_Seq**

  Syntax :

  ```
  int Licel_SP_SlaveMode_DataRead_Seq( SOCKET cmd_soc,
  Spectro_Hardware *sp_hw, Spectro_Application *sp_app,
  unsigned int *advanced_viewer_data )
  ```

  Description :

  The slave mode data read sequence performs the following operations :

  - Issues the DATA? command which instructs the controller to transmit the acquired data.
  - Reads the incoming SLAVE Mode Preamble/Header.
  - Extracts the parameters from the preamble and uses it to read the Channel Trace Data.
  - Reconstructs multiple byte data.

- **Licel_SP_SlaveMode_Read_Dataset**

  Syntax :

  ```
  int Licel_SP_SlaveMode_Read_Dataset( SOCKET cmd_soc,
  Spectro_Application *sp_app, Spectro_Hardware *sp_hw,
  unsigned int *advanced_viewer_data )
  ```

  Description :

  The slave mode read data set function is used to read the Slave Mode Channel Trace Data. Since the controller transmits uncompressed data in the slave mode, a decompression routine is not required on the client side. However, the raw data is transmitted as an unsigned char data type and hence has to be reconstructed at the client side if the data width is greater than 1 byte. After the acquisition of the RAW Data, data reconstruction functions are used to reconstruct the data according to its corresponding data widths. Please note that the data is reconstructed taking into consideration the endianness of the incoming data.

Push Mode Routines :

This category contains routines that are specific to the PUSH Mode operation of the controller.

- **Licel_SP_PushMode_Main**

  Syntax :

  ```
  int Licel_SP_PushMode_Main( SOCKET cmd_soc, SOCKET push_soc
  Spectro_Hardware *sp_hw, Spectro_Application *sp_app
  Spectro_Measurement *sp_mes, unsigned int *advanced_viewer_data )
  ```

  Description :

  This is the main function for all the PUSH Mode routines. The push mode routines consists of two main routines namely Acquisition Start and Data Read Loop.

- **Licel_SP_PushMode_Acquisition_Start**

  Syntax :

  ```
  int Licel_SP_PushMode_Acquisition_Start( SOCKET cmd_soc,
  Spectro_Hardware *sp_hw, Spectro_Application *sp_app )
  ```

  Description :

  This function performs the following :

  - Analyzes the maximum number of shots to acquire in the PUSH Mode. If the target shots to acquire is greater than the maximum push mode shots then the application turns on external averaging mode.
  - Starts the controller in the PUSH Mode to acquire the target number of shots.

- **Licel_SP_PushMode_Read_Dataset**

  Syntax :

  ```
  int Licel_SP_PushMode_Read_Dataset( SOCKET push_soc,
  Spectro_Hardware *sp_hw, Spectro_Application *sp_app,
  unsigned char *incomingtrace, unsigned char *rearrangeBuff,
  unsigned short *decompBuff, unsigned int *advanced_viewer_data )
  ```

  Description :

  This function performs the following operations :

  - Reads the incoming PUSH Mode data.
  - Decompresses the received PUSH Mode data.
  - If the external averaging mode is turned on by the application, this routine keeps reading the incoming data on the push_soc till the required number of shots have been acquired.

9.1.2 licel_sp_util

This category implements utility helper functions that are called by the functions in the licel_sp category.

- **Licel_SP_UTIL_Write_AdvancedViewerData**

  Syntax :

  ```
  int Licel_SP_UTIL_Write_AdvancedViewerData( Spectro_Application *sp_app
  Spectro_Measurement *sp_mes, char *filename,
  time_t tStartTime, unsigned int *advanced_viewer_data )
  ```

  Description :

  This function is used to write the acquired data into a file in the Licel Advanced Viewer Data format.

- **Licel_SP_UTIL_Reconstruct_SlaveMode_2Byte_TraceData**

  Syntax :

  ```
  void Licel_SP_UTIL_Reconstruct_SlaveMode_2Byte_TraceData( unsigned char
  *incomingdata, unsigned short *reconstructedtrace,
  Spectro_Application *sp_app, Spectro_Hardware *sp_hw )
  ```

  Description :

  This function is used to reconstruct a 2 byte wide data trace from the received raw data.

- **Licel_SP_UTIL_Reconstruct_SlaveMode_4Byte_TraceData**

  Syntax :

  ```
  void Licel_SP_UTIL_Reconstruct_SlaveMode_4Byte_TraceData( unsigned char
  *incomingdata, long *reconstructedtrace,
  Spectro_Application *sp_app, Spectro_Hardware *sp_hw )
  ```

  Description :

  This function is used to reconstruct a 4 byte wide data trace from the received raw data.

- **Licel_SP_UTIL_Decompress_PushData**

  Syntax :

  ```
  void Licel_SP_UTIL_Decompress_PushData( Spectro_Application *sp_app
  Spectro_Hardware *sp_hw, unsigned char*incomingtrace,
  unsigned char *rearrangeBuff, unsigned short *decompBuff )
  ```

  Description :

  This function is used to decompress the received PUSH Mode data.

9.1.3 licel_sp_tcpip

This category implements low level TCPIP functions used to communicate with the Lidarino controller. All functions in this subsection have a common return type and return a positive number if successful. On failure it either returns zero or a negative number.

- **Licel_SP_TCPIP_Get_HWConfig**

  Syntax :

  ```
  int Licel_TCPIP_GetSpectroHWConfig( SOCKET s, int *sphwRev,
  double *spBinlen, int *spBins, int *spNumBytes,
  int *spMaxShots, int *spMaxPushShots, bool *spBE,
  bool *spPushMode, int *spCompFact, bool *spVarTrace,
  bool *spVarComp, int *spCurBins, double *spMaxBinlen,
  bool *spHighRes, double *spMinBinLen, int *spMinBins,
  bool *spWideMem )
  ```

  Description :

  Gets the controller Hardware Configuration parameters.

- **Licel_SP_TCPIP_Start_Acq**

  Syntax :

  ```
  int Licel_SP_TCPIP_SlaveMode_Start_Acq( SOCKET s, int targetshots,
  int startmode )
  ```

  Description :

  Starts the acquisition of the Lidarino controller in the mode specified by startmode.

- **Licel_SP_TCPIP_SlaveMode_Get_AcqStatus**

  Syntax :

  ```
  int Licel_SP_TCPIP_SlaveMode_Get_AcqStatus( SOCKET s, int *AcqStatus,
  int *currentshots, int *current )
  ```

Description :

Get the current acquisition status of the controller.

- **Licel_SP_TCPIP_SlaveMode_Get_Data**

  Syntax :

  ```
  int Licel_SP_TCPIP_SlaveMode_Get_Data( SOCKET s )
  ```

  Description :

  Get the acquired data from the controller.

- **Licel_SP_TCPIP_Stop_Acq**

  Syntax :

  ```
  int Licel_SP_TCPIP_Stop_Acq( SOCKET s )
  ```

  Description :

  Stop the data acquisition of the controller.

- **Licel_SP_TCPIP_Set_Resolution**

  Syntax :

  ```
  int Licel_SP_TCPIP_Set_Resolution( SOCKET s, int resolution )
  ```

  Description :

  Set the resolution setting of the controller (only for controllers that support variable trace acquisition).

- **Licel_SP_TCPIP_Set_Rangebins**

  Syntax :

  ```
  int Licel_SP_TCPIP_Set_Rangebins( SOCKET s, int rangebins )
  ```

  Description :

  Set the rangebins setting of the controller (only for controllers that support variable trace acquisition)

- **Licel_SP_TCPIP_Enable_Wide_Accumulation_Memory**

  Syntax :

  ```
  int Licel_SP_TCPIP_Enable_Wide_Accumulation_Memory( SOCKET s )
  ```

  Description :

  Enables the wide summation memory of the controller for Internal Shot Averaging (only for controllers that support wide summation memory).

- **Licel_SP_TCPIP_Dissable_Wide_Accumulation_Memory**

  Syntax :

  ```
  int Licel_SP_TCPIP_Enable_Wide_Accumulation_Memory( SOCKET s )
  ```

  Description :

  Dissables the wide summation memory of the controller (only for controllers that support wide summation memory).

Function arguments

cmd_soc	*The client TCPIP socket to communicate with the command socket of the Lidarino controller.*
push_soc	*The client TCPIP socket to communicate with the push socket of the Lidarino controller.*
sp_app	*The Spectrometer Application Structure object used to store application acquisition parameters from the configuration file.*
sp_hw	*The Spectrometer Hardware Structure object used to store the controller's hardware parameters from the HW? command.*
sp_mes	*The Spectrometer Measurement Structure object used to store application measurement parameters from the configuration file.*
sp_preamble	*The Spectrometer Preamble Structure object used to store parameters from the data preamble.*
advanced_viewer_data	*The acquired data in unsigned int format.*
incomingdata	*The raw incoming data from the controller in unsigned char format.*
reconstructedtrace	*The reconstructed trace from the raw data (dependent on data width).*
filename	*The name of the file to write the acquired data.*
tStartTime	*The data acquisition start time stamp.*
s	*The client TCPIP communication socket.*
sphwRev	*The controller hardware revision number.*
spBinlen	*The length of each rangebin (in nano seconds).*
spMaxBinlen	*The maximum length (minimum resolution) of each rangebin (in nano seconds).*
spBins	*The maximum number of rangebins.*
spCurBins	*The current rangebins setting.*
spNumBytes	*The size (in bytes) of each element in the incoming data trace.*
spVarComp	*Variable compression factor flag.*
spVarTrace	*Variable acquisition trace flag.*
spHighRes	*High Resolution Mode support flag.*
spMinBinLen	*Minimum binlength / Highest Resolution possible.*
spMinBins	*Minimum number of bins acquirable.*
spWideMem	*Wide Memory Summation support flag.*
resolution	*The resolution setting of the controller.*
rangebins	*The rangebins setting of the controller.*
spMaxShots	*The maximum shots acquirable by the controller.*

spMaxPushShots	*The maximum shots acquirable in the Push mode.*
spBE	*Big Endian data format flag. If this flag is set then the incoming data is in big endiann format.*
spPushMode	*Push Mode supported flag.*
spCompFact	*The compression factor of the incoming data traces.*
current	*The ADC value from the Current Sensor. This is displayed as an integer.*
targetshots	*The target number of shots to acquire.*
startmode	*The controller start acquisition mode.3 (Transmit), 2 (Master/PUSH Mode), 1 (Slave Mode(default)).*
AcqStatus	*The acquisition status of the system. 0 (Idle), 1 (Armed), or 2 (Acquiring).*
currentshots	*The current number of shots acquired by the system.*
incomingtrace	*Memory to receive the incoming data in unsigned char format.*
rearrangeBuff	*Scratchpad Memory to perform decompression of push mode data.*
decompBuff	*Memory to store the decompressed push mode data.*
mode	*Mode of operation. 2 (Master/PUSH Mode), 1 (Slave Mode(default)).*

9.2 SP32 Detector Sample Application

The file SP32_Master_Sample_Application.cpp contains the application that demonstrates the use of the SP32 C drivers for data acquisition.

9.2.1 SP32_Master_Sample_Application

The Master Sample application implements both the SLAVE Mode and PUSH Mode of the SP32. This application makes use of an Application Configuration File to configure the application acquisition parameters. The user can enter the desired values in the corresponding fields of the configuration file to work with different parameters.

The application opens up a connection to the command socket of the Lidarino controller using the host IP address information in the application configuration file. The default port number of 2055 is used to connect to the command socket. After the connection is successfully established the application executes the *Licel_SP_Startup_Seq*. Once the start up sequence is completed, based on the mode of operation selected in the configuration file, the application executes one of the following functions :

- *Licel_SP_SlaveMode_Main*.
- *Licel_SP_PushMode_Main*.

The application closes the connection to the command socket once the acquisition is completed. The data file is written to the ”data” folder which must be created by the user.

Please Note : For this application to work the application configuration file must be stored in the same directory as the application executable and the ”data” folder must be created in the output directory as specified in the application configuration file.

9.2.2 Application Configuration File

The sample applications provided make use of an ASCII based application configuration file called `standard.cfg`. This configuration file contains parameters the user can enter to interact with the application. The parameters can be classified into two groups :

Application Measurement Parameters :

char [8]	*Location of the experiment*
integer	*Altitude*
integer	*Zenith Angle*
double	*Longitude*
double	*Latitude*
char	*Leading letter of the file name*
char [80]	*Output directory for data.*
char [20]	*IP address of the Host.*

Application Acquisition Parameters :

integer	*Target Shots to acquire*
string	*Activate High Voltage. (1 = turn on high voltage, 0 = turn off high voltage)*
integer	*High Voltage value (in Volts)*
integer	*Laser Rep Rate*
integer	*Discriminator level (range 0-63)*
integer	*Total number of acquisition cycles*
float	*Central Wavelength in nm.*
float	*Dispersion per Channel in nm.*
int	*Resolution (in nano seconds).*
int	*Rangebins.*

Please note : This chapter only includes `Lidarino` Detector specific function documentation. Please refer to the Licel Transient Recorder and Ethernet-Controller Programming Manual for additional functions. The manual call be downloaded at http://www.licel.com/programmingManual.pdf

9.3 Data File format

This describes the file format written by `PMT32 Channel Acquisition` and other Licel acquisition software. The files are interoperable between the different platforms. The file format is a mixed ascii-binary format where the first lines describe the measurement situation, below follow the dataset description and then raw data as 32-bit integer values itself.

9.3.1 Sample file header

```
a08C1114.3122161
Berlin   11/12/2008 14:31:22 11/12/2008 14:31:22 0035 0013.4 0052.5 00
0001000 0010 0000000 0000 32
1 1 1 01000 1 0800 0030 00323.9 0 0 00 000 00 001003 1.1905 BC0
1 1 1 01000 1 0800 0030 00330.1 0 0 00 000 00 001003 1.1905 BC1
...
1 1 1 01000 1 0800 0030 00516.1 0 0 00 000 00 001003 1.1905 BC1F
```

Line 1

Filename — *string.*
Format: `?YYMDDhh.mmssxxx`

- `?` - *The first letter can be chosen freely.*
- `yy` - *two numbers showing the years in the century*
- `M` - *one number containing the month as a hexadecimal number (December ≡ C)*
- `DD` - *two numbers containing the day of month*
- `hh` - *two numbers containing the hours since midnight*
- `mm` - *two numbers containing the minutes*
- `ss` - *two numbers containing the seconds*
- `xxx` - *three numbers containing the milliseconds(3 decimal places of the seconds)*

Line 2

Location	*String with 8 Letters*
Start Time	`dd/mm/yyyy hh:mm:ss`
Stop Time	`dd/mm/yyyy hh:mm:ss`
Height asl.	*four digits (in meter)*
Longitude	*four digits (including - sign) dot one digit*
Latitude	*four digits (including - sign) dot one digit*
zenith angle	*two digits (in degrees)*

Line 3

Laser 1 Number of shots	*7 digits (integer)*
Pulse repetition frequency for Laser 1	*5 digits (integer)*
Laser 2 Number of shots	*7 digits (integer)*
Pulse repetition frequency for Laser 2	*5 digits (integer) (Not used for the Licel Multispectral Lidar Detector)*
number of datasets in the file	*2 digits (integer)*

Dataset description

Active	*1 if dataset is present, 0 otherwise*
Analog/Photon Counting	*Always: Photon Counting ≡ 1 in case of the Licel Multispectral Lidar Detector*
Laser source	*one digit Always Laser 1 ≡ 1 in case of the one digit*
Number of bins	*5 digits*
1	*backward compatibility*
PMT high voltage	*four digits (in Volt)*
bin width	*two digits dot two digits (in meter)*
Channel wavelength	*five digits dot one digit (in nm)*
0 0 00 000	*backward compatibility*
number of ADC bits	*two digits. (00 in case of the Licel Multispectral Lidar Detector)*
number of shots	*6 digits (integer)*
analog input range/discriminator level	
	one digit dot 4 digits (discriminator level in case of the Licel Multispectral Lidar Detector)
Dataset descriptor	*BC ≡ photon counting, the number is the channel number as a hexadecimal (0 ... 1F).*

The data set description is followed by an extra `CRLF`. The datasets are 32bit integer values. Datasets are separated by `CRLF`. The last dataset is followed by a `CRLF`. These `CRLF` are used as markers and can be used as check points for file integrity.

Chapter 10

LabVIEW Driver VIs

In this chapter an overview about the provided LabVIEW VIs is given. The Licel TCPIP driver VIs are located in the LabVIEW library `Licel TCPIP.llb`. The `PM32` detector specific VIs are located in LabVIEW library `PM32 Channel.llb`.

Please Note : The abbreviation `Lidarino` is referred to as `PM32` in this chapter for naming compatibility purpose.

10.1 Licel TCPIP Driver VIs

10.1.1 Top Level VIs

Licel TCPIP Activate DHCP Mode.vi

This VI is used to activate DHCP for the transient recorder controller.
This VI uses the default password **Administrator** and the default port **2055**. If the port has been changed, you must change the **current port** to the proper value. The **DHCP port** is the port that will be used for DHCP communication. After DHCP mode has been set, communication will be lost until the acquisition computer is configured for DHCP communication as well.

Licel TCPIP Disable Secure Mode.vi

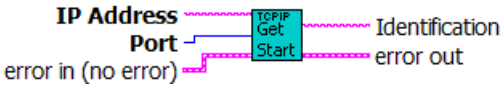
This VI is used to disable the Secure Mode of the Licel Ethernet Controller. The initialization file LicelTCPIP.ini is modified to allow future access without using the Secure Mode login.

Licel TCPIP Enable Secure Mode.vi

This VI is used to enable the Secure Mode of the Licel Ethernet Controller. The initialization file LicelTCPIP.ini is modified to allow future access using the Secure Mode login. This file should be copied to the same directory where Licel TCPIP.llb resides on all PCs from where access is allowed.

Licel TCPIP Getting Started.vi

This VI gets the identification information from the transient recorder controller.



Licel TCPIP Set Fixed IP Address.vi

This VI is used for setting the new IP configuration for the transient recorder controller.

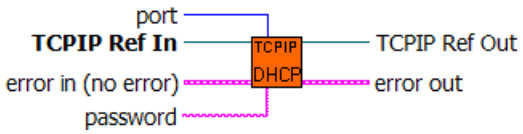
Licel TCPIP Set New Password.vi

This VI is used for setting the new password for the Licel Ethernet Controller.

10.1.2 Controller related VI's

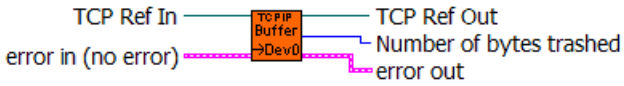
Licel TCPIP Activate DHCP.vi

This VI is used to activate the DHCP mode of the transient recorder controller.
In order to do so, the user must enter the proper password and port number for the controller. After DHCP mode has been set, communication will be lost until the acquisition computer is configured for DHCP communication as well.



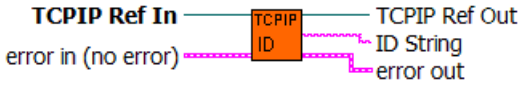
Licel TCPIP Dump TCPIP Buffer.vi

This VI empties the TCPIP buffer by reading all the data that is available in the buffer.
The **Number of bytes trashed** shows how many bytes were read from the buffer and disposed of.



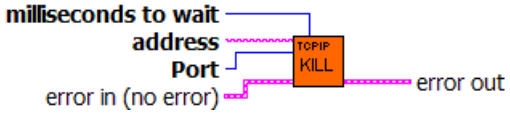
Licel TCPIP Get ID.vi

Gets the identification string from the transient recorder controller.



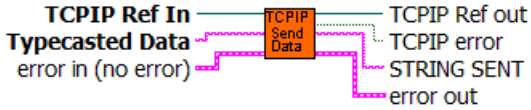
Licel TCPIP Kill Sockets.vi

This VI opens a new connection to the TR and sends the command to close down and reset all TCPIP connections. After doing this, the VI shuts down its TCPIP connection and waits the specified number of milliseconds, **milliseconds to wait**, before returning.



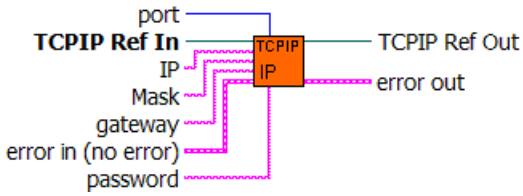
Licel TCPIP Send Data.vi

Adds a `CRLF` to the end of the string and sends it via TCPIP using the TCPIP reference input



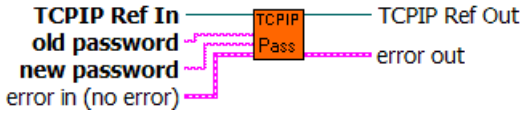
Licel TCPIP Set IP Parameter.vi

This VI is used to configure the transient recorder controller for static IP communication. With it, the values of the **IP** address, **port** number, subnet **mask**, and **gateway** can be set.



Licel TCPIP Set Password.vi

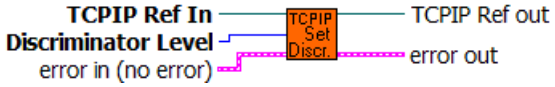
This VI is used for setting the password of the transient recorder controller.
This password must be given in order to change the IP configuration of the controller.



10.1.3 Transient Recorder Commands applicable to PM32 Detector

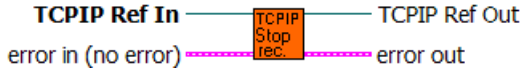
Licel TCPIP Set Discriminator Level.vi

Sets the discriminator level between 0 and 63 for the selected transient recorders.



Licel TCPIP Stop Acqusition.vi

This VI stops the acquisition process after the next received trigger.



10.1.4 PMT Commands applicable to PM32 Detector

Licel TCPIP PMT Get Status.vi

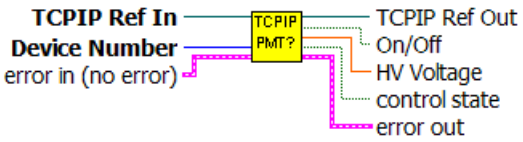
This VI gets the status of the PMT with the corresponding device number.
The values that are returned are the

HV Voltage : this is the actual gain voltage

On : this boolean is true if the gain voltage power supply is on, otherwise it is false

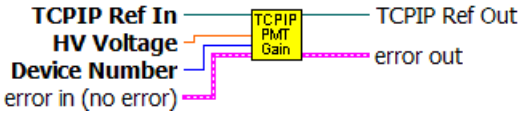
control state : if true, the PMT is being controlled remotely, if false, then the PMT is being controlled locally

For usage with the PM32 detector controller set **Device Number = 0**.



Licel TCPIP PMT Set Gain.vi

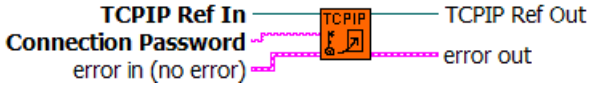
Sets the Gain Voltage for the PMT specified by the Device Number to the value specified by HV Voltage
For usage with the PM32 detector controller set **Device Number = 0**.



10.1.5 Network Security

Licel TCPIP Login Secure Mode.vi

Send the LOGON command to work in secure mode. Reads a string from TCPIP, attempts to convert the string to 2 U32 numbers used to encrypt the password to 2 output U32 numbers using the Blowfish encryption algorithm. These output numbers are converted to a hexadecimal string to be used in the LOGON command. If the LOGON command fails the controller will close the connection without any notification.

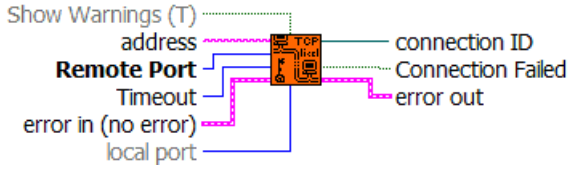


Licel TCPIP Open Secure Mode.vi

Open a TCP/IP connection to the Licel controller in secure mode. The vi tries to open the initialization file LicelTCPIP.ini to read the values for the keys UseSecureMode and SecureModePWD from the SecureMode section:

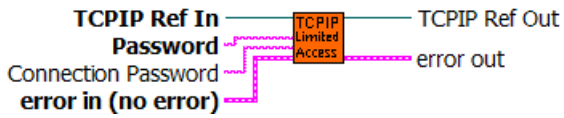
```
[SecureMode]
UseSecureMode=TRUE
SecureModePWD=ConnectMe
```

If the initialization file is found and UseSecureMode is true and SecureModePWD is found the vi will send the password using the LOGON command (Licel TCPIP Login Secure Mode.vi). Otherwise just the TCP/IP connection will be opened.



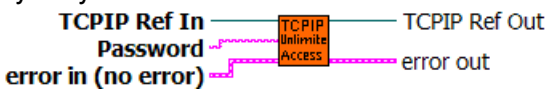
Licel TCPIP Set Access Limited.vi

Enables the limited access to the controller, i.e. activates the secure mode. Access is granted only for IP addresses as specified with the WHITELIST command. Moreover the connection password is specified.



Licel TCPIP Set Access Unlimited.vi

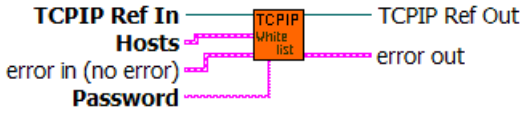
Disables the limited access to the controller, i.e. deactivates the secure mode. Access is granted for everybody.



Licel TCPIP Set Whitelist.vi

This VI is used to set the allowed hosts at the controller. In order to do so, the user must enter the appropriate password and 3 host strings to allowed IP addresses or IP address ranges. Such a string must be specified in the following format:

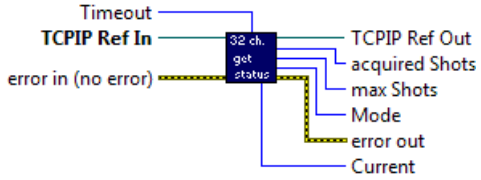
xx.xx.xx.xx	a single IP address,
xx.xx.xx.255	an IP address range (0:255),
	or may be empty.



10.2 PM32 Detector Specific VIs

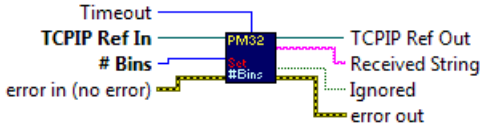
PMT32Channel Get Status.vi

Returns the acquisition mode of the PM32 detector, the acquired shots, and the target shots. Optionally a value Inum representing the current is returned. The current in mA is obtained by: $I[\text{mA}] = I_{\text{num}} * \text{Scale} + \text{Offset}$, Scale defaults to 0.0028172248 mA, Offset defaults to -0.0143994941 mA.



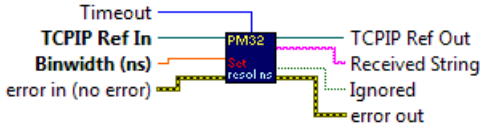
PMT32Channel Set NumBins.vi

Set the number of bins to read. This VI is supported only by controllers returning VariableTrace = TRUE in the hardware information request.



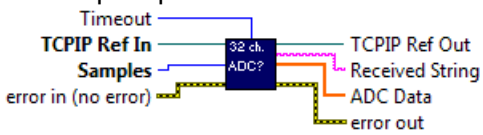
PMT32Channel Set Resolution.vi

Set the resolution in nanoseconds. This VI is supported only by controllers returning VariableTrace = TRUE in the hardware information request.



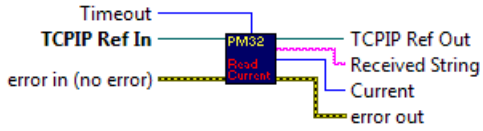
PMT32Channel Read ADC.vi

Read Samples points of the ADC trace.



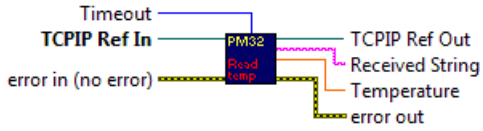
PMT32Channel Read Current.vi

Read the current value I_{num} of the PM32 detector.
The current in mA is obtained by: $I[\text{mA}] = I_{\text{num}} * \text{Scale} + \text{Offset}$, Scale defaults to 0.0028172248 mA, Offset defaults to -0.0143994941 mA.



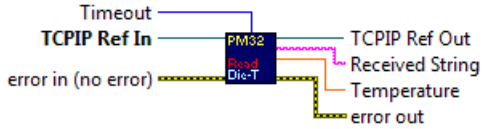
PMT32Channel ReadTemp.vi

Read the temperature in deg C from the PM32 detector controller



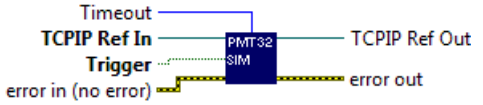
PMT32Channel ReadDieTemp.vi

Read the die temperature in deg C.



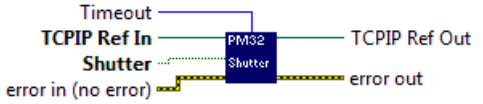
PMT32Channel Simulation.vi

Switch the trigger simulation on (generate an internal trigger) or off (expect an external trigger)



PMT32Channel Shutter.vi

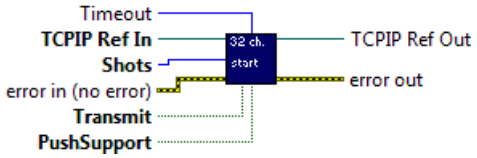
Switch the the mechanical shutter (if installed and supported).



PMT32Channel Start.vi

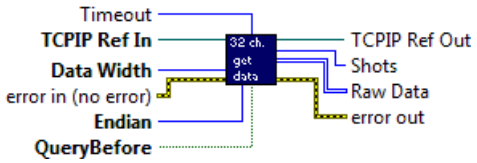
Starts an acquisition of Shots shots at the PM32 detector.
If **TRANSMIT** is true the acquired data will be returned without any further request for data.

If **PushSupport** is true the PM32 detector will continuously transfer sequences of acquired data of **Shots** shots via the TCP/IP data socket (push socket).



PMT32Channel Get Data.vi

Request data (if **QueryBefore** = TRUE).
Read data from the multispectral detector.
If **QueryBefore** = FALSE a subsequent acquisition must have been started with the **TRANSMIT** option.



PMT32Channel HW Info.vi

Returns the hardware information about the PM32 detector controller. Default values are used in the case that parameters are not available from the current PM32 detector controller.
The returned values are bundled into two clusters:

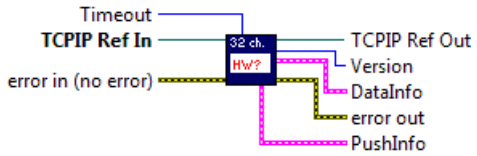
- **Data Info**

  1. **Version**
  2. (current) **Binwidth (ns)**
  3. **#MaxBins** (maximum number of bins to return)

  1. **DataWidth** (default: 2 bytes)
  2. **MaxShots** (default: 4096)
  3. **Endian** (default: big endian)
  4. **VariableTrace** (= FALSE if not supported)
  5. (current)
  6. **#Bins** (changeable if VariableTryce = TRUE, otherwise **#Bins** = **#MaxBins**)
  7. **MaxBinwidth (ns)** (= **Binwidth (ns)** if **VariableTrace** = FALSE, otherwise **Binwidth (ns)** is changeable)

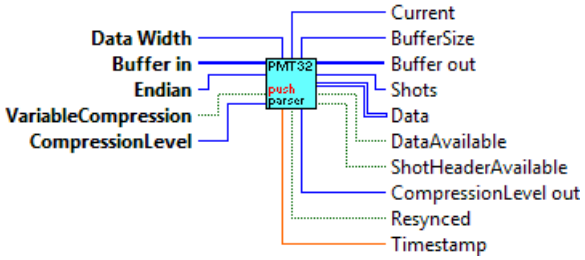
- **PushInfo**

  1. **PushSupport** TRUE: the controller is capable to transfer acquired data via the TCP/IP data socket (push socket) (= FALSE if not supported)
  2. **PushDataSize** (use if **PushSupport** = TRUE)
  3. **CompressionLevel** (use if **PushSupport** = TRUE)
  4. **VariableCompression** (use if **PushSupport** = TRUE)



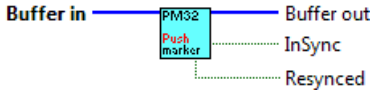
PMT32Channel Push Parser.vi

Parse the pushed data stream for the data heder and the acquired data bytes.



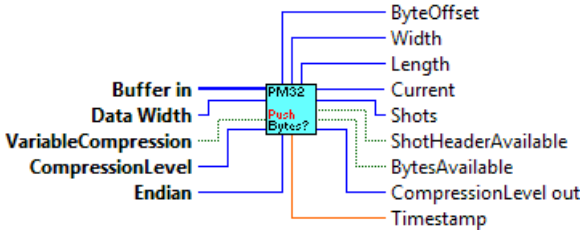
PMT32Channel Push CheckMarker.vi

Sub-VI of `PMT32Channel Push Parser.vi`. Test the data stream for the availablity of the marker DWord (0xFFFFFFFF) at the beginning. Search for it if necessary.



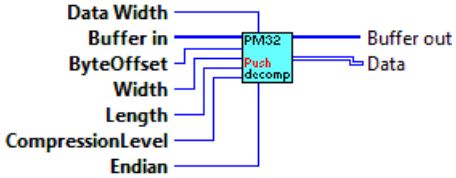
PMT32Channel Push EnoughDataBytes.vi

Sub-VI of `PMT32Channel Push Parser.vi`. Check whether or not the data header is complete and if yes whether or not enough data according to the extracted values for Width, Length, and Data Width is available.



PMT32Channel PushDecompress.vi

Sub-VI of `PMT32Channel Push Parser.vi`. Extract the data from the data stream according to the parameters ByteOffset, Width, Length, Data Width, Compression Level, and Endian.



Chapter 11

Initialization Files

The software *PM32 Channel Acquisition* and *PM32 Pulse Height Distribution* read certain value from the initialization file `PMT32Channel.ini`. Some of these values are written to the file when they are changed. The initialization file `global info pm32.ini` is used by *PM32 Channel Acquisition*, only. In the next section we refer to *PM32 Channel Acquisition* as *Acquis*, and to *PM32 Channel Acquisition* as *Pulse*.

11.1 The initialization File `PMT32Channel.ini`

In the initialization File `PMT32Channel.ini` several values are stored on change while the software is running. These values will be read when the software is started the next time.
The values in the `TCPIP` section are read only if the software is started as a Windows Apllication. The values are used by *Acquis* and *Pulse*:

```
[TCPIP]
UseValues = TRUE
IPAddress = "10.49.234.234"
Port = 2055
```

The keys `IPAddress` and `Port` are automatically written to the file after successfully establishing a TCP/IP connection.
The following list contains the other initialization file keys.

- `[PMT]`
  `HV = 800.000000`

  The key's value is written when the *High Voltage* is changed and will be used when the software is started (*Acquis* and *Pulse*).

- `[TR]`
  `Discriminator = 3`
  `SamplingRate = 5.000000`

  The key `Discriminator` is written when the *Discriminator* is changed and it's value will be used when the software is started (*Acquis*).
  The key `SamplingRate` is written for documentation (*Acquis*) when the internally used sampling rate is derived from the binwidth (in nanoseconds, obtained with the `HW?` command when the software initializes).

- `[Live]`
  `Shots = 100`

  The key's value is written when the *Update #* (update number) in the *Live* mode is changed and will be used when the software is started (*Acquis*).

- [Acquis]
  Shots = 1000
  Records = 1

  The key Shots and the key Records are written when the number of *Shots* and the number of records (*Acquisitions*), respectively, are changed in the *Acquis* mode and will be used when the software is started (*Acquis*).

- [ROI]
  CursorStartX = 4000,000000
  CursorEndX = 6000,000000
  Scale = 1

  The keys CursorStartX and *CursorEndX* are written when the cursor positions change on the *ROI* page or when the *Range/Time Scales* is changed (for documentation). The key Scale is changed when the *Range/Time Scales* is changed. The Scale value is used for the next start of the program (*Acquis*).

- [Resolution]
  Binwidth_ns = -1.000000
  NoBins = -1

  The key NoBins is written to the initialization file when the number of bins *# Bins* on the *Configuration* page changes (if the controller supports variable traces). The key Binwidth_ns is written when the *Binwidth (ns)* or the *Ninlength (m)* are changed. The values are used for the next start of the program (*Acquis*) but they may be overwritten when obtaining the *HW?* values. Values of -1 mean that the keys have never been updated.

- [Channels]
  CenterWavelength = 420,000000
  IndexDisplay = 1

  The key CenterWavelength and IndexDisplay are written to the initialization file when the control fields *Wavelength / nm*, (located on the *Configuration* page in *Acquis*), and the switch *Channel Display* (at the page *ROI* in *Acquis*) are changed. The values are read when *Acquis* initializes. When operating a Single channel Lidar Detector (Lidarino) the keys nmWidt and Active may appear but are not used.

- [Pulse]
  HV = 800.000000
  Shots = 10
  Active = 4294967295

  The values in this section are used to initialize the corresponding controls in *Pulse*. They will be updated when changing the *High Voltage* value, the number of *Shots* and the control to activate the channels (Active is a bit field as described above).

- [HV_Current]
  Max_mA = 0.4
  CurrentScale = 0.0008
  CurrentOffset = 0

  The values in this section are not changed by the software but read when starting *Acquis*. Max_mA is used to set the warning level (red slider in the dynode current indicator).
  CurrentScale und CurrentOffset are used to calculate the dynode current in mA from the *ADC value* of the on board high voltage supply current sensor: *Current* $=$ (*ADC value* $\times$ CurrentScale $+$ CurrentOffset.

- `[TCPIP_API]`
  `Active = FALSE`
  `Port = 2088`

  The settings here are used to enable the TCP/IP API of *Acquis*.

11.2 The initialization File `global info pm32.ini`

The initialization file contains global settings for the PM32 acquisition used by *PM32 Channel Acquisition*. The `working_directory` is used as start path by the *PM32 Viewer*.

```
[global_info]
Location = "Berlin"
Longitude = 13.400000
Latitude = 52.500000
Height_asl = 45.000000
working_directory = "/D/data/PM32channel/20220608"
first_letter = "p"
frequency1 = 10.000000
```

These initialization file keys correspond to the inut fields *Location*, *Longitude*, *Latitude*, *Height asl*, *Working Directory*, *first letter*, and *Laser Frequency* on the Configuration page of the PM32 channel acquisition software. All values are written to the initialization file when the corresponding parameters are changed, therfore, there is no need to change them manually. The initialization file keys `frequency2`, `Zenith`, `frequency3`, `Azimuth`, `Info`, `frequency4`, `NoSafeIncompleteFiles`, `SaveOverflow`, and `SyncViewer` might appear in the file for compatibility with other applications. These parameters will not be used by the PM32 software.

Chapter 12

TCP/IP API

The basic functions of the *Licel Main* software can be accessed from third party applications via TCP/IP. For this *Licel Main* implements a TCP/IP server listening on a defineable port.

12.1 Enable the TCP/IP API

To activate the TCP/IP server the following initialization file keys in `PMT32Channel.ini` have to be aligned:

```
[TCPIP_API]
Active = TRUE
Port = 2088
```

If `Active` is set `TRUE` a listener will be started using the specified TCP/IP port (`Port = 2088`).

12.2 Command List

The following list contains the supported commands. The commands must be sent with an additional `<CRLF>` (`0x0D0A`) and the responses will end with a `<CRLF>`, as well.

- `VER?`
  - *Parameters*
  - *Description* Return the program version number as displayed in the Windows title bar
  - *Reply* `VER <version>`
- `DIRECTORY`
  - *Parameters* `<Directory>`
  - *Description* Set the `directory` for the data files. The directory path must be given in Windows syntax.
  - *Reply* `DIRECTORY executed`
- `WAVELENGTH`
  - *Parameters* `<CentralWavelength> <nmPerChannel>`
  - *Description* Sets the `CentralWavelength` in nanometers and the `nmPerChannel`. Note that the `nmPerChannel` may be negative dependent on the geometry of the used spectrograph/spectrometer.
  - *Reply* `WAVELENGTH executed`

- `PMT`
  - *Parameters* `<HV> <VoltageSwitch>`
  - *Description* Set the PM voltage `HV` and the HV switch `VoltageSwitch`. `PMT 950 1` sets the voltage to 950 V and switches it on, `PMT 950 0` switches the voltage off while leaving the set voltage at 950 V
  - *Reply* `PMT executed`
- `DISCRIMINATOR`
  - *Parameters* `<DiscriminatorLevel>`
  - *Description* Set the Discrminator Level, the allowed range is 0 . . . 63
  - *Reply* `DISCRIMINATOR executed`
- `SHUTTER`
  - *Parameters* `<0|1>`
  - *Description* Set the shutter (0: Closed, 1: Open)
  - *Reply* `SHUTTER executed`
- `TRIGGER`
  - *Parameters* `<0|1>`
  - *Description* Set the trigger (0: External, 1: Internal)
  - *Reply* `TRIGGER executed`
- `RUNMODE`
  - *Parameters* `<0|1>`
  - *Description* Set either the Live mode (0) or the Set either the Acquis mode (1).
  - *Reply* `RUNMODE executed`
- `LSHOTS`
  - *Parameters* `<NumberOfShots>`
  - *Description* Sets the `NumberOfShots` to acquire in *Live Mode*.
  - *Reply* `LSHOTS executed`
- `SHOTS`
  - *Parameters* `<NumberOfShots>`
  - *Description* Sets the `NumberOfShots` to acquire per record when starting a multiple acquisition using the command `AUTO 1`.
  - *Reply* `SHOTS executed`
- `RECORDS`
  - *Parameters* `<NumberOfRecords>`
  - *Description* Sets the `NumberOfRecords` to acquire when starting a multiple acquisition using the command `AUTO 1`. A value of `0` will set the number of records to *unlimited* i.e. acquire until `AUTO 0` is sent.
  - *Reply* `RECORDS executed`
- `AUTO`
  - *Parameters* `<0|1>`
  - *Description* Start or stop `NumberOfRecords` (or unlimited) automatic acquisitions of `NumberOfShots` shots.
  - *Reply* `AUTO executed`
- `STATUS?`
  - *Parameters*
  - *Description* Request the current software status, i.e. the run mode (*Live* or *Acquis*), automatic acquisition status (0 or 1), number of acquired records (completed files), number of acquired shots in the current record, PMT voltage, Shutter (0 = closed or not supported, 1 = open).
  - *Reply* `STATUS RUNMODE=1 AUTO=1 RECORDS=2 SHOTS=77 PMTHV=950 SHUTTER=1` (example)

- `FILE?`

  Parameters

  Description Request the path of the last written file.

  Reply `FILE z:\data\2260912.395192` (example)

- `CMDLOG?`

  Parameters

  Description Request whether or not the TCP/IP commands are currently logged. Command logging is supported in Licel's Windows and Linux applications and by default in the LabVIEW sources.

  Reply `CMDLOG 0|1`

- `CMDLOG`

  Parameters `0|1`

  Description Switch the TCP/IP command logging on (`1`) or off (`0`). Command logging is supported in Licel's Windows and Linux applications and by default in the LabVIEW sources.

  Reply `CMDLOG executed`