CRISTAL User Manual

Ruben de Groote
`ruben.degroote@fys.kuleuven.be`

August 2, 2014

Contents

1 Introduction

The CRISTAL (CRIS laser Tuning, Acquisition and Logbook) software package provides an interface to communicate with a NI-6211 DAQ card (and, presumably, any card that relies on the NIDaqmx drivers to talk to the pc). It can also communicate to the RILIS laser systems via the Labview Shared Variables engine.
This document is the starting point for a more complete manual I hope to write some day (this version was written in two hours during a night shift, so no promises regarding (literary) quality). While incomplete, I hope it can help you use CRISTAL and troubleshoot some potential problems you might encounter.

1.1 CRISTAL structure

Under the hood, CRISTAL runs three python processes (you can see these three if you open the task manager when CRISTAL is running; you will see three python processes).

1. The National Instruments Acquisition process. This process gets data from the NI card and tells it what voltages to write to its analog outputs. This process is the 'master' process, in the sense that all data is synchronized to a timer that runs in this process.
2. The network process. This process uses the OPC technology to read and write network variables (in this case, National Instruments Shared Variables). This process communicates with the RILIS systems as well as the CRIS wavemeter. All the data gathered by this process are first sent to the NI process, where they are timestamped and bundled up with the analog input and MCP count rate data.
3. The main process, responsible for the user interface and also communicates with the previous two processes, saves the data to file and plots the data.

The main process is actually the most CPU intensive, predominantly because it plots the data[1]. It is therefore a good idea to not have too many live updating plots active, since that can put quite a bit of strain on the program. All interprocess communication is buffered, which should prevent data loss, but still - better safe than sorry.

Additionally, the software also executes a labview VI on startup. This VI is like a relay messenger - reading network variables from RILIS and hosting them on the PCCRIS 6 local machine.

This information immediately tells you how you can force a full termination of CRISTAL: kill the three python processes using the task manager, and close the relay VI.

1.2 NI card interfacing

The NI 6211 USB card has analog inputs and outputs, and counter channels. Every time a trigger pulse is supplied in the clock channel, the card sends the current values of the analog input to the pc, as well as the number of counts it received since the last trigger pulse. It also sets the voltage on the analog output at this moment.
The channels into which these signals are put can be specified using the settings widget that will be discussed further down in this document.

2 Use

2.1 Running CRISTAL

Proceed as follows:

1. Make sure you are workin on PCCRIS6. Fasten your seatbelt as well.

[1]Which is why I will move the plotting to a fourth process at some point

2. Navigate to C:\CRISTALCLEAR. You should see three subfolders: Code, Data and doc, as well as a shortcut to CRISTAL.pyc. Double click on the shortcut to CRISTAL.pyc. If you do not see said shortcut, navigate into the Code directory, and double-click on CRISTAL.pyc. If for some reason even that does not work, double-click on CRISTAL.py. A code editor called Sublimetext will open. Press Ctrl-B to run the python code.
3. A command prompt should appear. If at any point the program bugs out our throws errors, these errors will be printed in this command line interface. Let me know what the errors are if this happens! You should also see a Labview program appear, which will automatically start running. You can just mimize this program, it should never be changed. After a few moments a python program will also appear. It might not jump to the foreground immediately, so check the taskbar.

So, summarized: **C:\CRISTALCLEAR, run CRISTAL.pyc.**
All data is automatically saved periodically. **NOTE:** Do not move the software anywhere but to the directory where it is located right now! Originally, the code automatically found the directory it was running in and used relative paths, but I just noticed some unexpected behaviour there, so I had to hard-code the absolute paths.

2.2 Using CRISTAL

The CRISTAL UI has three main areas:

1. Top toolbar: the **scanner** widget. This toolbar allows you to change laser frequency (by defining a target wavelength or etalon setpoints for the RILIS laser, or by defining a voltage for the cw laser).
2. Middle area: a blank canvas on which all the fun and exciting widgets will appear once you click on the buttons in the ...
3. bottom toolbar: contains buttons to open and close the other widgets of CRISTAL. Also shows the current event rate on the MCP, laser wavelength, provides a boolean status indicator of the laser system, as well as information on the proton supercycle.

The toolbars can be dragged out of the UI or repositioned in the window.
All of the widgets can be dragged and dropped by clicking and dragging the blue label at the top of the widget to rearrange the interface. All the widgets created by the bottom toolbar can be opened and closed by toggling the buttons, except for the graph widgets (see the subsection below). We will go through the scanner widget and the widgets the bottom toolbar enables one by one.

2.2.1 Scanner widget

The scanner widget is your way of telling the lasers what to do. On the right-hand side you can see a large button, initially with an orange plus-sign on it. This button handles the creation of a new scan, running of this new scan, and stopping of this scan.

- When the button is clicked for the first time, a new Capture object is created in memory. None of the actual scan parameters have been initialized though, so you can still change the scan settings at this point (more info on the scan settings in a bit).
- When the button is clicked for the second time, the scan actually starts running. Most of the interface is disabled, so no settings can be changed during a scan.
- The scan will continue until it reaches its predefined end, or until you click the button (now showing a red stop symbol) a third time. All measurements are then (gracefully) halted.

I've already mentioned the settings for the scan, but not how to actually change these. Fear no more, explanations await!
To the left of this button you can find a dropdown menu and a spinbox with a clock icon in it. These two elements combined will tell the lasers how long they have to keep fixed at every wavelength step of the scan. The dropdown menu offers the following options:

1. Time: gather data for a time (defined in ms) per frequency step
2. Triggers: gather data for number of acquisition triggers per frequency step

3. Supercycle: gather data for number of supercycles per frequency step
4. Proton Pulse: gather data for number of proton impacts per frequency step

The time/number of triggers/supercycles/proton pulses is defined by the spinbox.
The biggest part of the scanner widget is occupied by a tabbed interface.

One of the tabs can be used to freely change scan parameters like the voltage or etalon setpoints. When a scan is not running his can be done by simply changing the values in the spinboxes. When a scan is running, you actually have to press the 'Confirm' button for the variable you changed. This is to avoid accidental changes you might not want recorded.
The other tab contains a widget that helps you create a scan region CRISTAL then scans for you. On the left are two buttons. The first controls if the scan only scans upwards in the scan parameter, or if it zigzags up and down. The other (with the infinity sign) can be toggled to have the software repeat the scan across your region forever (until you untoggle this button or just stop the scan).

Underneath these two buttons is a dropdown menu you can use to change which paremeter you want to scan (e.g. voltage or an etalon setpoint).

The biggest part of this second tab is occupied by a slider which visualises the progress through the scan region. Underneath it are (initially) three spinboxes: one for the starting point of the scan, the second for the number of steps CRISTAL should scan, and the third fixes the endpoint of the scan.
By actually clicking on the slider, you can define intermediate points. You will see a marker appear which can be dragged around to change where you want this intermediate point. Two spinboxes will also appear underneath the graph, one with the value of the intermediate point. By creating two of these intermediate points, you can for example define a scanning array that scans from 0 to 3 V in 100 steps, then from 3 to 7 in 500 steps, and finally from 7 to 9 in 150 steps.
Note that you should always make sure that the values of the scan markers INCREASE as you make them - otherwise you will get some very strange zig-zagging scans. So: don't drag a marker past another one, conserve their initial ordering.

2.2.2 Settings widget

This widget is used to change the acquisition settings of CRISTAL. You can specify which channel on the NI card is the analog output, which channels are the analog inputs, and into which channel the clock signal is applied.
Note that the locations of the analog input channels should be separated by a comma, e.g. \Dev\ai1,\Dev\ai2, and in incrementing order. I might make this more user-friendly at some point.
Finally, you can also choose which laser to control - be it the cw laser system or the RILIS lasers.
The two communications processes mentioned previously are set up once and then keep running at the specified settings. So, the changes you make to the settings widget will only be applied if the 'Confirm changes and reconnect' button is clicked. As a pop-up message will inform you, rebooting the external processes will freeze the application for a few seconds. I can probably fix that at some point too, we'll see.

2.2.3 Console widget

The console widget provides a Python console with the numpy and pyqtgraph libraries already loaded in the np and pg namespaces. Go crazy! Divide 1/3 and find out that in Python 2.7 this still equals 0!

2.2.4 Logbook widget

The logbook widget provides you with a logbook interface. Every time a new capture is initialized, an timestamped entry is added to the logbook, waiting for you to fill in the blanks. Write down your name, too, since it could help us to figure out particularly obscure entries as well.
Each logbook entry can be shown or collapsed by clicking on the icon on the left of its name. You can collapse all open entries by clicking on the top-left button. Each entry contains a text box where you write your name, as well as a larger text box for comments and notes. You can always change previously made entries by clicking the 'edit'-button and then confirming your changes. The capture

entries also have a graph button on the left which you can use to show the data from earlier measurements.

Next to the collapse-all button you will find another button, which you can click to add an entry to the log that is not explicitly tied to a specific scan. Entries like this are colored blue rather than the green reserved for the capture entries.

There is one final button to explain: the square with a plus sign in it allows you to define a certain property that you would like to always see written down in the log. In all following entries a textbox will appear, reserved for just this parameter.

2.2.5 Graph widgets

Every time you click on the scatterplot icon, a graph canvas is added to the UI. By using the combo boxes at the bottom left of the graph, you can choose what you want to plot (e.g. the ion counts as function of the wavelength, or the diode signal as function of time). By clicking the gears icon on the bottom right, you get access to some options. You can choose whether you want to see the raw data or rather the binned version, if you want to see all the scans within one capture, or rather their sum or average. You can also hide or show individual scans, and change their color. These graphs can be closed by clicking the x button in the top-right.

2.2.6 Datastream widget

The line plot icon opens up a widget that shows you all of the raw signals the DAQ monitors as a function of time. The stream only goes 10000 samples back in time. By clicking the gears icon in the bottom right, you can show or hide specific data streams, as well as access a way to write the required streams to a file.

2.2.7 HELP PANIC BUTTON AAH

Clicking this opens this document. Hope that's sufficient help. . .

3 Development

Coming soon.

4 Troubleshooting / FAQ

I started the program as you explained, but no python window appears!
Check if you are sending the trigger into the NI card. Without this, the program can't start sampling from the card, which means it can't start initializing all the parameters.

Golden rule of CRISTAL: open task manager, kill python threads, close the relay VI, try again.

I was not touching anything, I swear, but the UI got unresponsive!
If the UI got unresponsive, restart the program. If you can't even close it using the normal methods, just kill the python processes using the task manager. If you did manage to just close the window, be sure to kill possible ghost python threads anyway using the task manager. They tend to linger when errors are thrown.)

The command prompt spits out the following: The specified resource is reserved. The operation could not be completed as specified.
This means a task is still running on the device. To end the task and recover, open the NIDAQmx utility from Start → National Instruments → NiMAX. Select Devices and Interfaces → NI USB 6211 then select 'Reset Device', and await confirmation.

The command line shows an error message!

Copy-paste-email-me! If it is an error relating to binning of data or plotting: I know these sometimes pop up; they're mostly harmless. Let me know anyway.

The UI is a bit sluggish.
Two options: the pc is bogging down due to god-knows-what-reason, or there are too many graphs being updated at the same time. Rebooting the computer can help with the former, just closing a few graphs with the latter (you data-junkie!).

I was playing around with the plotting options, and the graph just disappeared!
It could be that you set the bin width to be too small. Change to something more reasonable. Another possibility is that you changed what is put on the x- or y-axes, so that the graph is actually just out of view. If you hover over the graph canvas, you should see a small icon with an 'A' in it appear in the bottom-right corner. Clicking this will attempt to auto-range x- and y-axes to bring the graph in view. Also, make sure you didn't set the line color to white ;).

I attempted to plot the data from earlier measurements, but I don't see a graph/the UI gets unresponsive or slow.
If this happens, something went wrong during the data read-in from file. Not much I can do to help you if I'm not around. Just let me know (and consider not trying to look at that measurement again).
If the whole UI gets slow, just click on the graph icon again to hide the plot. If the UI got unresponsive, restart the program (be sure to kill possible ghost python threads using the task manager. They tend to linger when errors are thrown.)

CRISTAL is awesome! Can I buy you a drink?
Sure thing.