OPTICAL SOFTWARE USER MANUAL (DRAFT)

REFERENCE	QU 5659
REVISION AND ISSUE:	Rev A
DATE:	9th April 2018
CLIENT:	Imperial College
AUTHOR:	Mike Parkin
BUSINESS DEVELOPMENT MANAGER:	Chris Garratt

Austin Consultants Ltd
Carpenter House Innovation Centre, Broad Quay, Bath, BA1 1UD   Company number: 06352633,
UK VAT Number – 932283234
AustinConsultants.com |0800 772 0795 |info@austincontultants.com

Document Summary

This document aims to give an overview on how to use the Optical Software Application and a description of the main screens contained within in it. It also provides a high level description of the LabVIEW architecture that has been used to write it and how modifications can be made to it as additional features and requirements arise in the future.

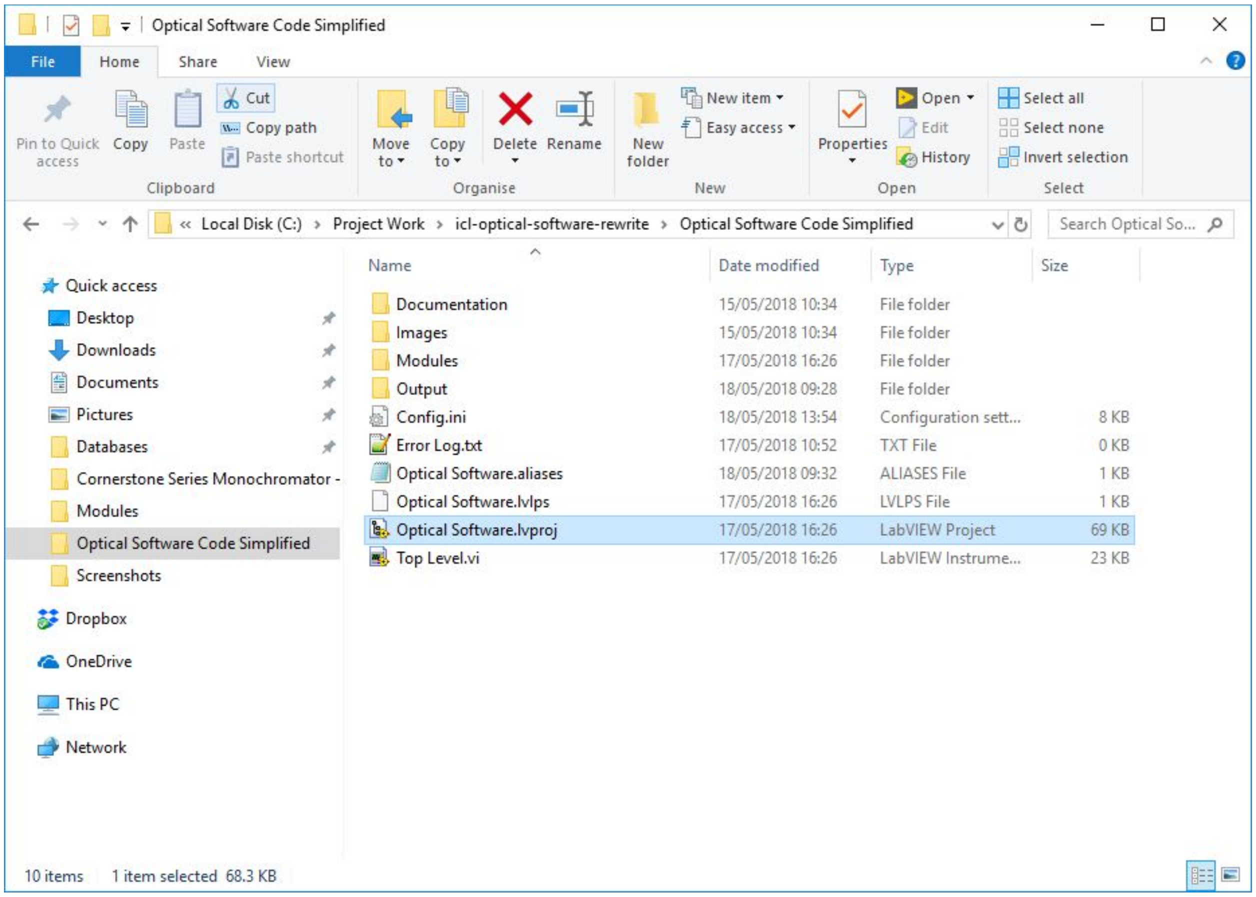
Software Overview

Starting the Application

In order to start the application, the LabVIEW project file must first be opened. This is what links the application to all of its dependencies.

To do this, navigate to where the LabVIEW project file (Optical Software.lvproj) is located and double click to open it.

NOTE: This application has been developed in LabVIEW 2017 so must be opened with this version of LabVIEW. It is not possible to open it with a previous version of LabVIEW.

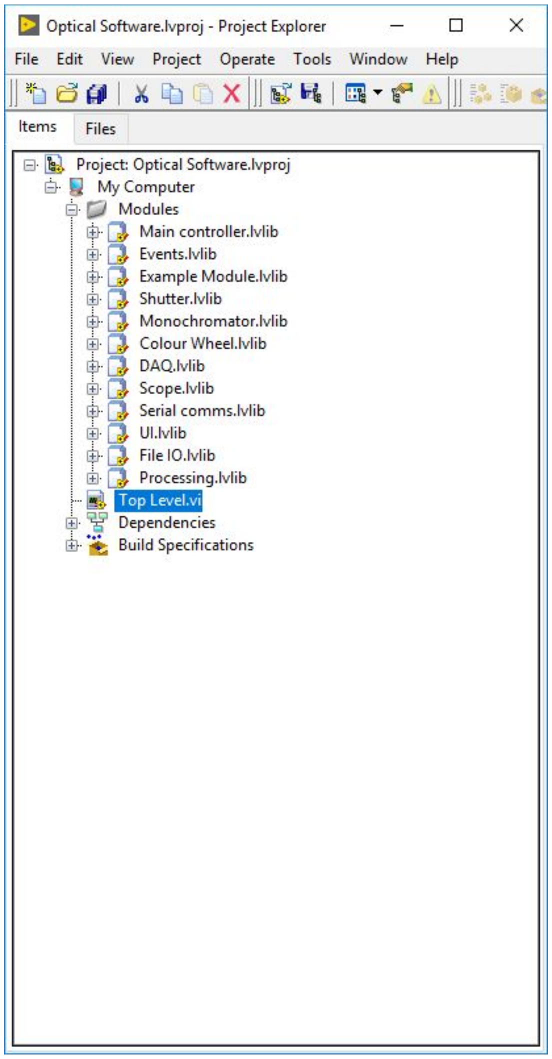


« Local Disk (C:) › Project Work › icl-optical-software-rewrite › Optical Software Code Simplified

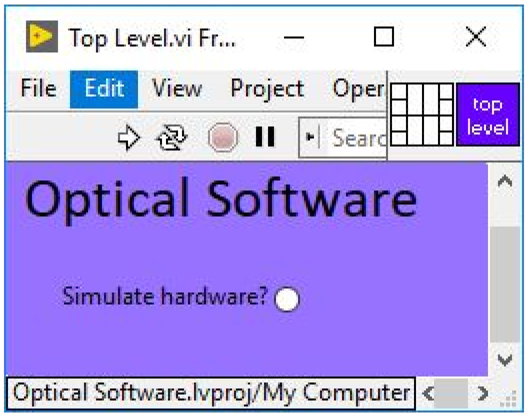
Name	Date modified	Type	Size
Documentation	15/05/2018 10:34	File folder	
Images	15/05/2018 10:34	File folder	
Modules	17/05/2018 16:26	File folder	
Output	18/05/2018 09:28	File folder	
Config.ini	18/05/2018 13:54	Configuration sett...	8 KB
Error Log.txt	17/05/2018 10:52	TXT File	0 KB
Optical Software.aliases	18/05/2018 09:32	ALIASES File	1 KB
Optical Software.lvlps	17/05/2018 16:26	LVLPS File	1 KB
Optical Software.lvproj	17/05/2018 16:26	LabVIEW Project	69 KB
Top Level.vi	17/05/2018 16:26	LabVIEW Instrume...	23 KB

10 items 1 item selected 68.3 KB

This will open the LabVIEW Project Explorer window as seen below. Double click 'Top Level.vi' to open the window from where the application can be started.

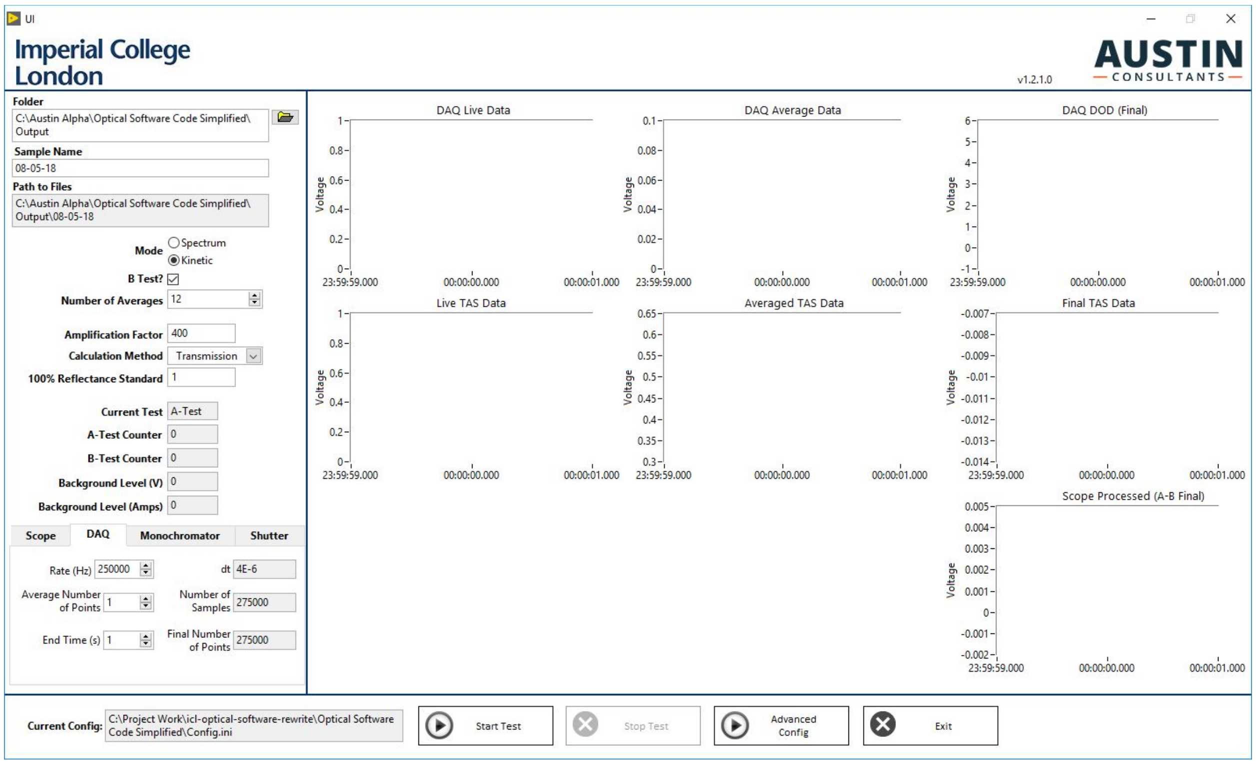


This will open the window seen below. From here the user can set whether to run with real or simulated hardware. Press the Grey arrow to start the application and see the UI.



User Interface - Overview

When the application is started the user will be presented with a UI as seen below. This is where various test parameters can be entered and where tests can be started and stopped. During a test, data will be visible on the graphs and the indicators on the left hand side will be updated.



Folder: C:\Austin Alpha\Optical Software Code Simplified\Output

Sample Name: 08-05-18

Path to Files: C:\Austin Alpha\Optical Software Code Simplified\Output\08-05-18

Mode: Spectrum [✓ Kinetic]

B Test? ✓

Number of Averages: 12

Amplification Factor: 400

Calculation Method: Transmission

100% Reflectance Standard: 1

Current Test: A-Test

A-Test Counter: 0

B-Test Counter: 0

Background Level (V): 0

Background Level (Amps): 0

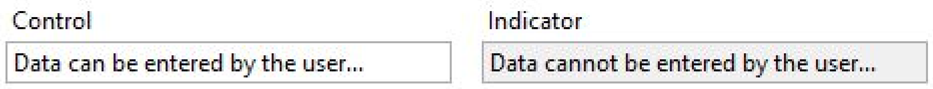
Rate (Hz): 250000 dt: 4E-6

Average Number of Points: 1 Number of Samples: 275000

End Time (s): 1 Final Number of Points: 275000

Current Config: C:\Project Work\icl-optical-software-rewrite\Optical Software Code Simplified\Config.ini

The key thing to notice is that anything with a **white** background is a control where information *can* be entered by the user. Anything with a **grey** background is an indicator and *cannot* be updated by the user.

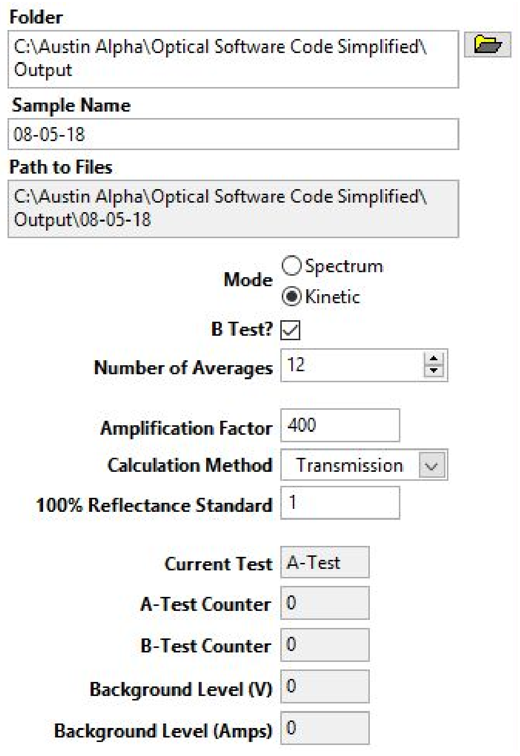


Control: Data can be entered by the user...

Indicator: Data cannot be entered by the user...

User Interface - Left Pane

The pane on the left hand side of the UI contains a mixture of both controls and indicators where the user can set certain test parameters and also observe the progress of ongoing tests.



Folder	C:\Austin Alpha\Optical Software Code Simplified\Output
Sample Name	08-05-18
Path to Files	C:\Austin Alpha\Optical Software Code Simplified\Output\08-05-18
Mode	○ Spectrum ● Kinetic
B Test?	✓
Number of Averages	12
Amplification Factor	400
Calculation Method	Transmission
100% Reflectance Standard	1
Current Test	A-Test
A-Test Counter	0
B-Test Counter	0
Background Level (V)	0
Background Level (Amps)	0

Folder - The directory specified by the user where any data files will be saved.

Sample Name - The sub-folder within the Folder location where any data files will be saved.

Path to Files - The location of the data files for that test (Folder + Sample Name).

Mode - Allows the operator to select whether to run tests in Spectrum or Kinetic mode.

B Test? - Choose whether to run a B Test in addition to an A Test.

Number of Averages - The number of averages/acquisitions per test.

Amplification Factor - The value defined by the user that links the Background Level (V) to Background Level (Amps).

Calculation Method - The type of calculation that is applied at the end of each test.

100% Reflectance Standard - Value used in the Kubelka-Munk calculation.

Current Test - Informs the operator what test is currently running.

A -Test Counter - Informs the operator how many averages have currently been acquired for an A-Test.

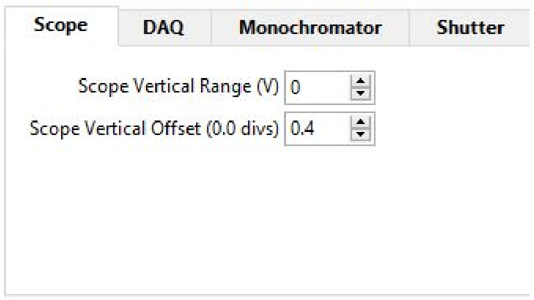
B-Test Counter - Informs the operator how many averages have currently been acquired for a B-Test.

Background Level (V) - The background voltage calculated from DAQ data at the end of an A-Test.

Background Level (Amps) - The background level in Amps (Background Level (V) x Amplification Factor).

The User Interface displays a selection of tabs showing high level configuration parameters and controls for the devices connected to the system. The ability to modify advanced configurations is shown later in the document.

User Interface - Scope

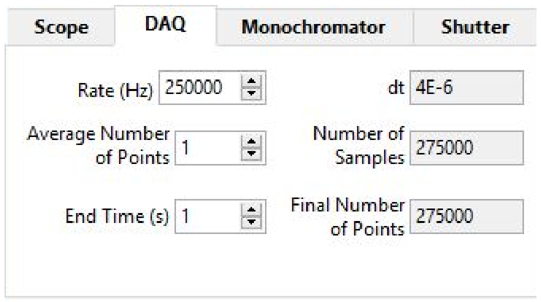


Scope Vertical Range (V) 0
Scope Vertical Offset (0.0 divs) 0.4

Scope Vertical Range (V) - The vertical range setting sent down to the scope.

Scope Vertical Offset (0.0 divs) - The vertical offset setting sent down to the scope.

User Interface - DAQ



Rate (Hz) 250000
dt 4E-6
Average Number of Points 1
Number of Samples 275000
End Time (s) 1
Final Number of Points 275000

Rate (Hz) - The rate at which the DAQ will sample.

Average Number of Points - The number of points that the acquired DAQ data will be averaged by.

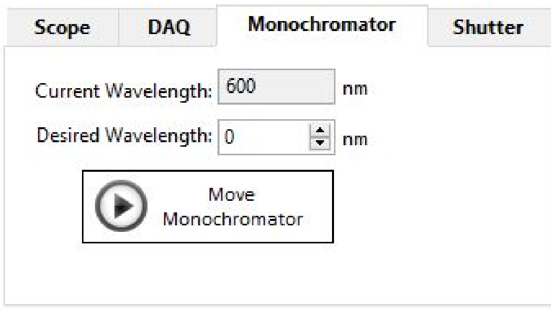
End Time - The length of time the DAQ will sample for.

Dt - The time between each data point acquired.

Number of Samples - Total number of samples ((Rate x End Time) x 1.1).

Final Number of Points - Total number of points after averaging.

User Interface - Monochromator



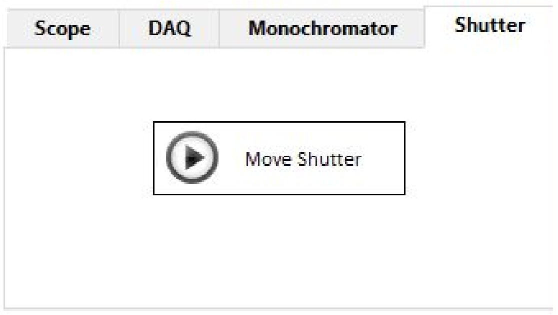
Current Wavelength: 600 nm
Desired Wavelength: 0 nm

Current Wavelength (nm) - The current wavelength that the monochromator is/will be showing.

Desired Wavelength (nm) - The wavelength, entered by the operator, that the monochromator will be moved to when the ***Move Monochromator*** button is pressed.

Move Monochromator - Moves the monochromator to the wavelength specified

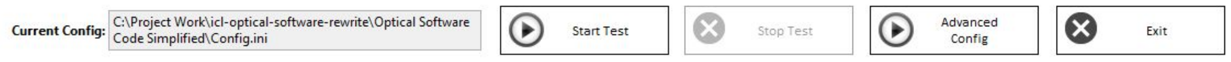
User Interface - Shutter



Move Shutter - Moves the shutter based on settings in the advanced config.

User Interface - Bottom Pane

At the bottom of the UI there is a series of buttons that the operator uses to Start and Stop tests as well as Exiting the application safely. When a test is started certain buttons on the UI are disabled to prevent the operator mistakenly interfering with the current test.


Current Config:
C:\Project Work\icl-optical-software-rewrite\Optical Software Code Simplified\Config.ini
Start Test
Stop Test
Advanced Config
Exit

Current Config - Displays the configuration currently being used by the application.

Start Test - Starts and runs a test based on the configuration set by the operator

Stop Test - Stops the current test that is running.

Advanced Config - Takes the operator through the Advanced Configuration screen which specified low level settings of the devices connected to the system - See *Advanced Configuration* section.

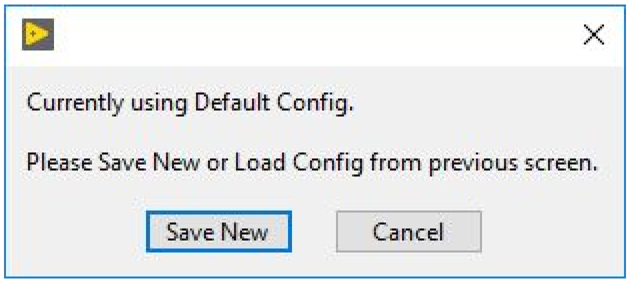
Exit - Clears the buffers and Exits the application safely.

Advanced Configuration

When the application is started it will *always* load the default configuration file. All settings and configurations are saved to a config file. The default configuration file is saved to the same location as the project file and cannot be saved or overwritten from the application. However, it can be used as a basis for operators to create and save their own configuration file.

Selecting the ***Advanced Config*** button enables the user to make changes to the configuration settings currently being displayed, save it as a new file or load an existing configuration file.

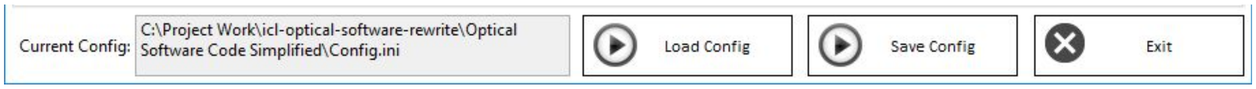
The operator can set a configuration that suits and save it using the ***Save Config*** button. If the operator tries to overwrite the default configuration file, a pop up will warn them to save it as a new one with a different name.


Currently using Default Config.

Please Save New or Load Config from previous screen.

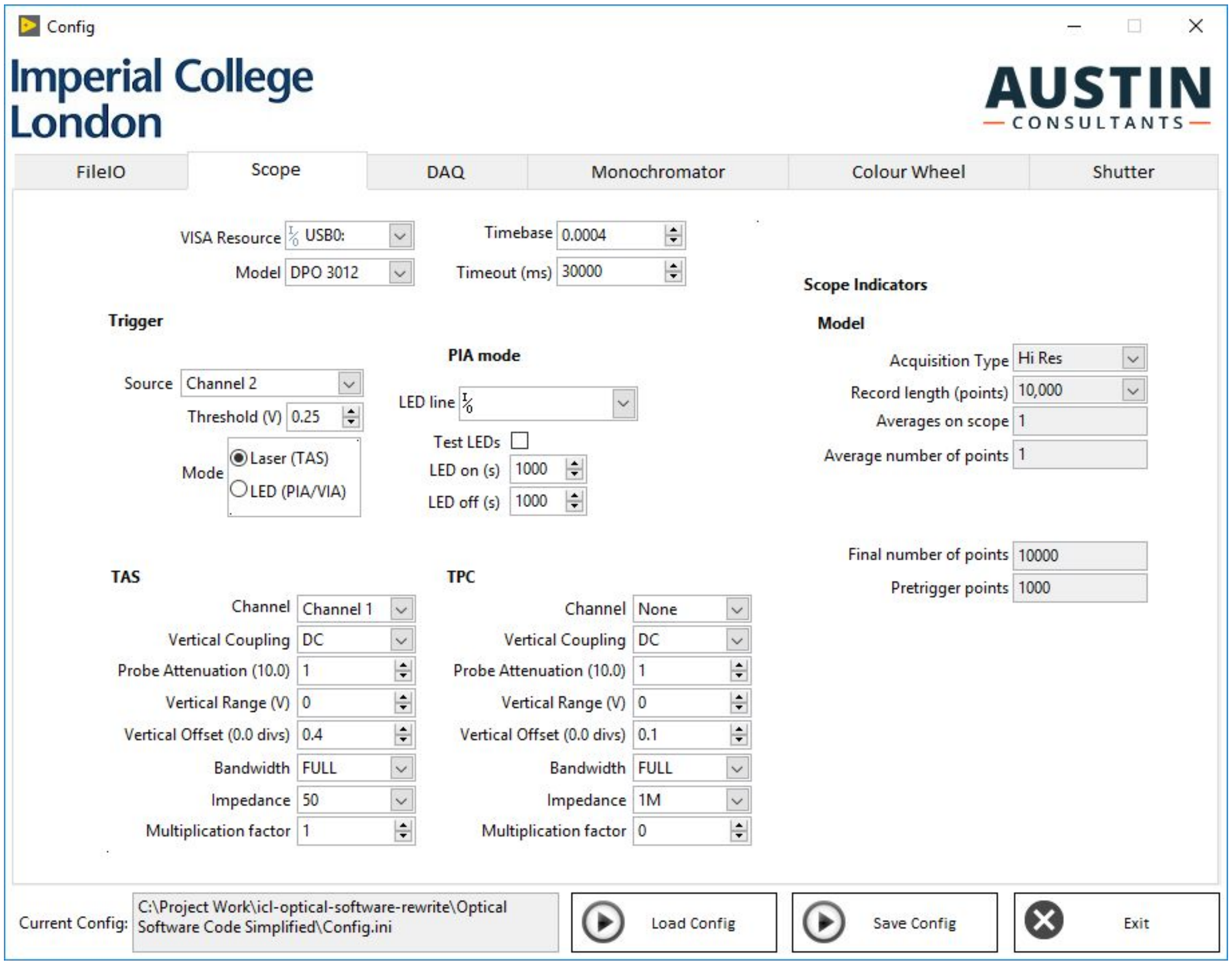
The application will then guide the operator through the process of saving the new file - which will then be used by the application and can then be used in the future.

To load an existing configuration, select the ***Load Config*** button. The operator will select the configuration file and populate the application with the settings contained in that file. This must be done every time the application is started.


Current Config: C:\Project Work\icl-optical-software-rewrite\Optical Software Code Simplified\Config.ini

The Advanced Configuration screen allows the operator to modify all settings that aren't shown on the main UI. These settings include thing such as the COM Port associated with a device or the type of Scope and Monochromator being used in the system - things that are unlikely to change frequently.

Whenever a setting is changed the operator must ***Save Config*** in order for the new settings to take effect.



VISA Resource: USB0:
Model: DPO 3012
Timebase: 0.0004
Timeout (ms): 30000

Trigger

Source: Channel 2
Threshold (V): 0.25
Mode: [✓ Laser (TAS)] LED (PIA/VIA)

PIA mode

LED line:
Test LEDs: ☐
LED on (s): 1000
LED off (s): 1000

Scope Indicators

Model

Acquisition Type: Hi Res
Record length (points): 10,000
Averages on scope: 1
Average number of points: 1

Final number of points: 10000
Pretrigger points: 1000

	TAS	TPC
Channel	Channel 1	None
Vertical Coupling	DC	DC
Probe Attenuation (10.0)	1	1
Vertical Range (V)	0	0
Vertical Offset (0.0 divs)	0.4	0.1
Bandwidth	FULL	FULL
Impedance	50	1M
Multiplication factor	1	0

Current Config: C:\Project Work\icl-optical-software-rewrite\Optical Software Code Simplified\Config.ini

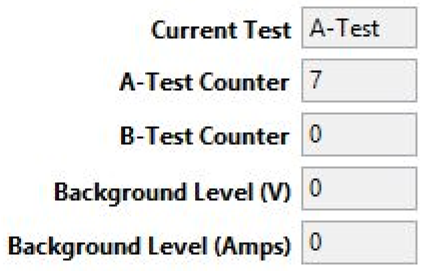
Selecting the ***Exit*** button will take the operator back to the main User Interface.

Executing a Test - Kinetic Mode

If the test has been started in Kinetic mode, the operator will be asked to confirm the current wavelength displayed by the monochromator and the desired wavelength of the monochromator. Ie, the wavelength will this test be run at.

When confirmed, the monochromator will move to the requested wavelength and the colour wheel will automatically move to the appropriate setting. The DAQ and Scope will then start acquiring based on the trigger mechanism specified in the configuration.

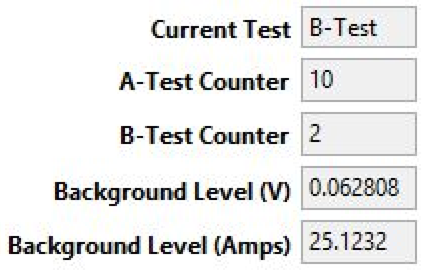
During the test, the UI will display the number of tests completed (for either A-Test and B-Test).



Current Test	A-Test
A-Test Counter	7
B-Test Counter	0
Background Level (V)	0
Background Level (Amps)	0

At the end of an A-Test, the data acquired will be processed and data files generated at the location specified by the operator. The filename of these data files will be prefixed with the wavelength that they were testing at. By this point the Background Level (V) and Background Level (Amps) will have been calculated and updated on the UI.

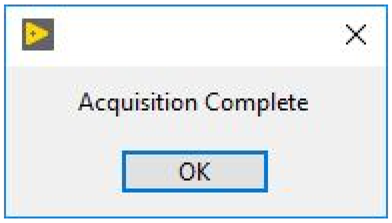
The application will then automatically move the shutter and start a B-Test. The B-Test, if uninterrupted, will always run the same number of averages as the A-Test; meaning the counters should always match at the end (the screenshot below is during a B-Test).



Current Test	B-Test
A-Test Counter	10
B-Test Counter	2
Background Level (V)	0.062808
Background Level (Amps)	25.1232

At the end of a B-Test, the data acquired will be processed in the same way as the A-Test. However, this time there will also be A-B calculation stage and an AB Stitching stage. These will be written to a file and saved to the same location.

The shutter will then return to its original position and a pop up will appear informing the operator that the acquisition has completed.

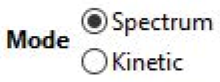


Acquisition Complete

OK

Executing a Test - Spectrum Mode

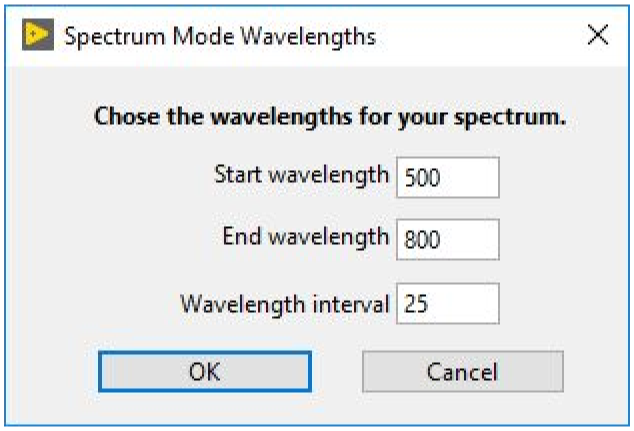
Spectrum Mode is similar in the fact that it runs A-Tests and B-Tests across a number of averages as specified by the operator.



Mode [✓ Spectrum] Kinetic

However, the main difference is that it runs multiple series of A and B Tests consecutively, each time at a different wavelengths. Eg, A-Test at 600nm, B-Test at 600nm, Data processed and files created, A Test at 700nm, B-Test at 700nm, Data files processed etc...

When the Mode on the UI is specified as Spectrum and a test is started, the operator will be asked to enter the current wavelength of the monochromator. This is followed by a request to define the desired wavelengths. The operator must set the ***Start Wavelength***, ***End Wavelength*** and ***Wavelength Interval*** and press OK. A series of wavelengths is created and shuffled into a random order that will be used to determine what wavelength the monochromator and colour wheel moves to at the start of each test.



Spectrum Mode Wavelengths

Chose the wavelengths for your spectrum.

Start wavelength 500

End wavelength 800

Wavelength interval 25

OK Cancel

The UI will give an indication as to what wavelength is currently being tested and how far through that test it is. There is also the also the option to check the data file directory to see which files have been processed.

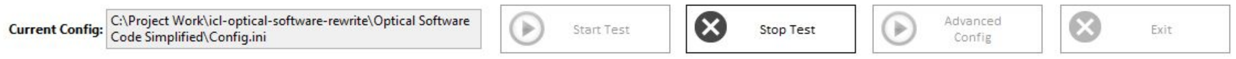
The process, once started, requires little operator interaction as stages of the test will happen automatically until there are no more wavelengths to be tested. The Acquisition Complete message will be displayed when all tests have completed.

Stopping a Test

Stopping the test can be done by selecting the ***Stop Test*** button on the UI. This will often be the only button enabled whilst a test is running and can pressed at any point regardless of what test in being run.

If the current test is an A-Test, pressing the ***Stop Test*** button will stop the A-Test and move onto the B-Test. The B-Test will then run the same number of averages that the A-Test reached when it was stopped. The post-processing calculations will then be executed as normal.

The Stop Test button is not limited to which test it can stop and can stop both tests if needed.

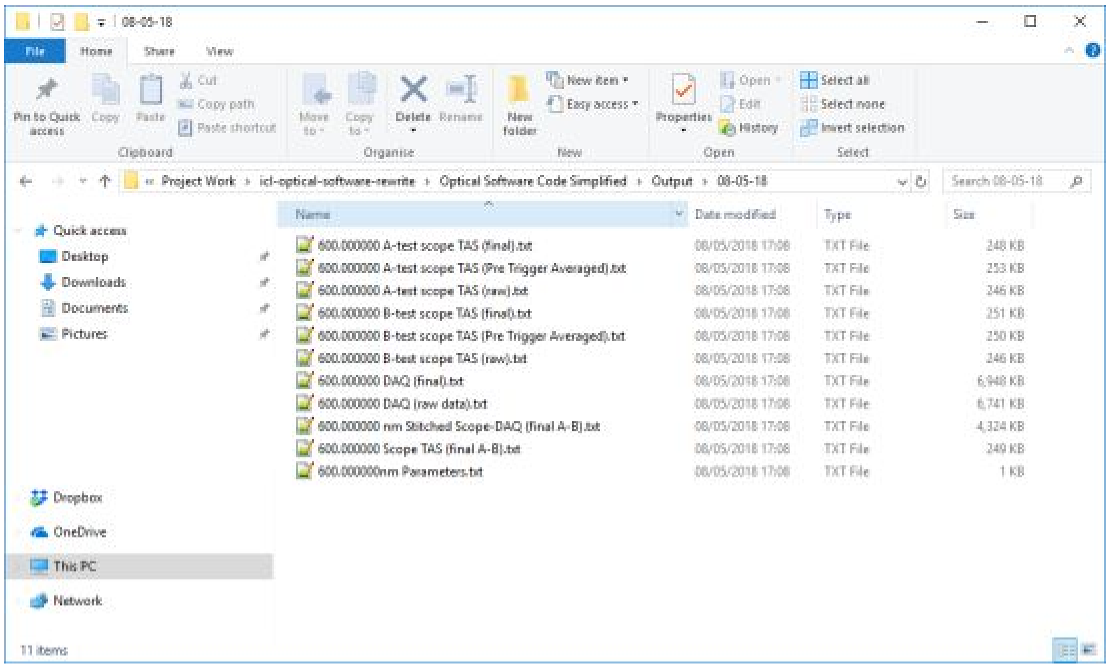

Current Config: C:\Project Work\icl-optical-software-rewrite\Optical Software Code Simplified\Config.ini

File I/O - Data Files

Throughout the testing procedure the raw and final data files of both the DAQ and Scope are generated and written to the location specified by the operator. There is also a parameters file that lists the key parameters set during this test - this is different to the application configuration file.

During a test there will be some temporary files located here. This is for performance reasons and means data that will be used at a later stage in the test is saved to a file rather than the application holding it in memory. In the event of a PC failure, this data will not be lost and could provide explanation as to any fault.

An overview of the File I/O from a coding perspective can be found in the Appendix of this document.


« Project Work › icl-optical-software-rewrite › Optical Software Code Simplified › Output › 08-05-18

Name	Date modified	Type	Size
600.000000 A-test scope TAS (final).txt	08/05/2018 17:08	TXT File	248 KB
600.000000 A-test scope TAS (Pre Trigger Averaged).txt	08/05/2018 17:08	TXT File	253 KB
600.000000 A-test scope TAS (raw).txt	08/05/2018 17:08	TXT File	246 KB
600.000000 B-test scope TAS (final).txt	08/05/2018 17:08	TXT File	251 KB
600.000000 B-test scope TAS (Pre Trigger Averaged).txt	08/05/2018 17:08	TXT File	250 KB
600.000000 B-test scope TAS (raw).txt	08/05/2018 17:08	TXT File	246 KB
600.000000 DAQ (final).txt	08/05/2018 17:08	TXT File	6,948 KB
600.000000 DAQ (raw data).txt	08/05/2018 17:08	TXT File	6,741 KB
600.000000 nm Stitched Scope-DAQ (final A-B).txt	08/05/2018 17:08	TXT File	4,324 KB
600.000000 Scope TAS (final A-B).txt	08/05/2018 17:08	TXT File	249 KB
600.000000nm Parameters.txt	08/05/2018 17:08	TXT File	1 KB

11 items

File I/O - Error Log

The application has a dedicated Error Module that will write any errors to an error log. The location of this is defined in the Advanced Configuration screen. It is not possible to view this log through the application itself but the file can be opened by notepad.

Code Overview

LabVIEW Project File

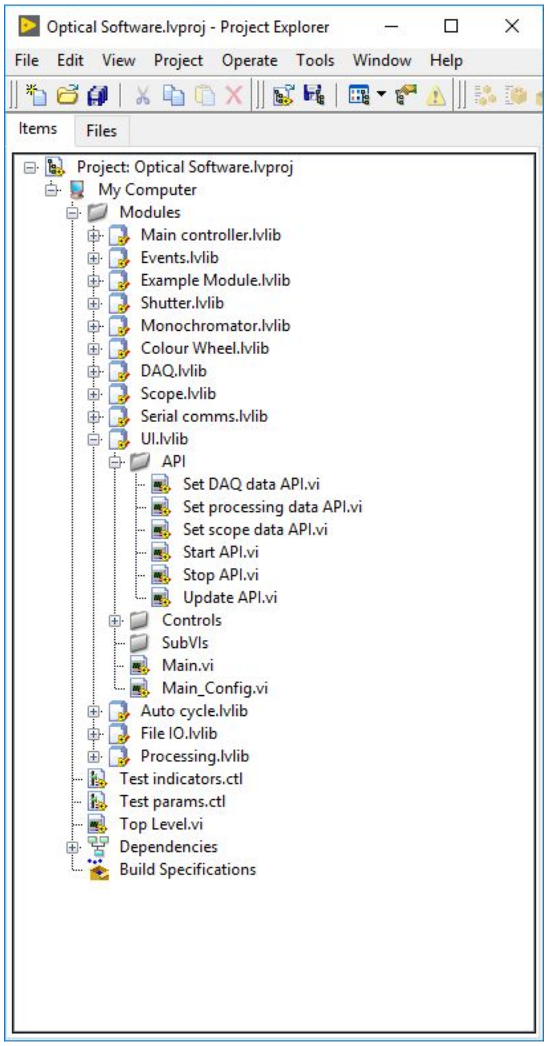
All the files used in this application are now contained within a LabVIEW project file (.lvproj). Utilising a LabVIEW project file provides a way of organising and managing an application as it grows in size and complexity.

As an application grows, the number of modules and SubVIs will also increase. Having these VI's and dependencies collated in an easy to read format means any developers or users involved in this project know where to look should modifications need to be made.

This environment can be used to open the Top Level VI and any module contained within it. In the case of this application, each module is responsible for a certain area of functionality.

Each module has its own LabVIEW Library (.lvlib). The benefit of using a LabVIEW Library is that all files associated with that module are contained within that library, from the APIs (used for sending messages to this module from other modules) to SubVI's (blocks of code that have been wrapped to contain certain functionality or purpose).

In the image to the right, it can be seen that there is a 'UI' library. This contains 'Main.VI'. This is the VI that contains the SubVI's and is the receiver of any messages sent to it using the API's.


Optical Software.lvproj - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files

```
Project: Optical Software.lvproj
  My Computer
    Modules
      Main controller.lvlib
      Events.lvlib
      Example Module.lvlib
      Shutter.lvlib
      Monochromator.lvlib
      Colour Wheel.lvlib
      DAQ.lvlib
      Scope.lvlib
      Serial comms.lvlib
      UI.lvlib
        API
          Set DAQ data API.vi
          Set processing data API.vi
          Set scope data API.vi
          Start API.vi
          Stop API.vi
          Update API.vi
        Controls
        SubVIs
        Main.vi
        Main_Config.vi
      Auto cycle.lvlib
      File IO.lvlib
      Processing.lvlib
    Test indicators.ctl
    Test params.ctl
    Top Level.vi
    Dependencies
    Build Specifications
```

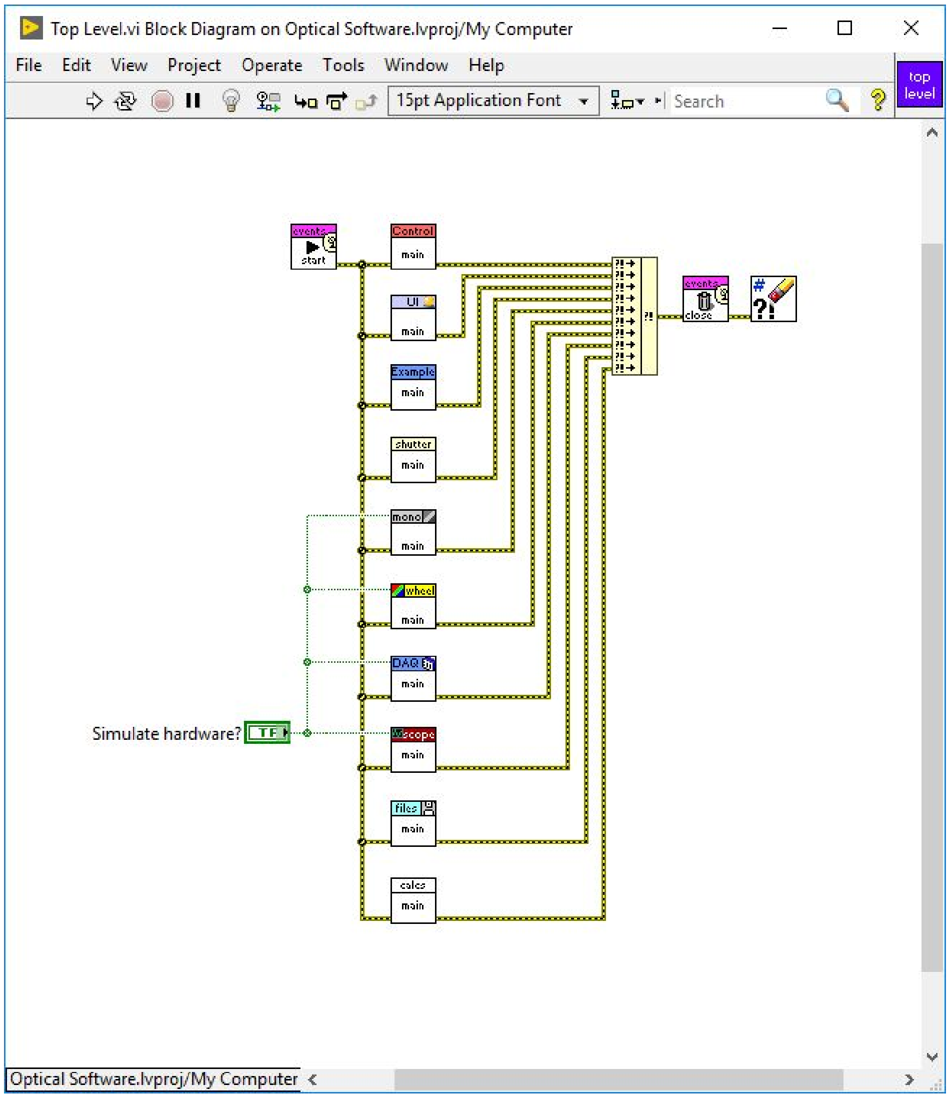
In this application the DAQ module sends information to the UI module. Therefore, 'Set DAQ data API.vi' is used to send this data. If we wanted to send a confirmation from the UI module back to the DAQ module, we would need to use an API from the DAQ library.

In addition to this being a more intuitive layout for the developer it also provides benefits when it comes to debugging. Should an issue arise relating to a certain device in the system, it can be easily identified where, in the application, this issue is likely to be originating from. For example, if there is an issue with DAQ data being read it is highly probable the problem lies in the DAQ Module. From here, various debug techniques can be used to deduce what the problem is.

More information and features of the Project Explorer can be found in the following white paper on the National Instruments website: http://www.ni.com/white-paper/7197/en/.

Top Level VI Overview

The application is run from the Top Level VI. This must be opened from Project Explorer and can be seen below. The block diagram of the Top Level VI is seen in below.



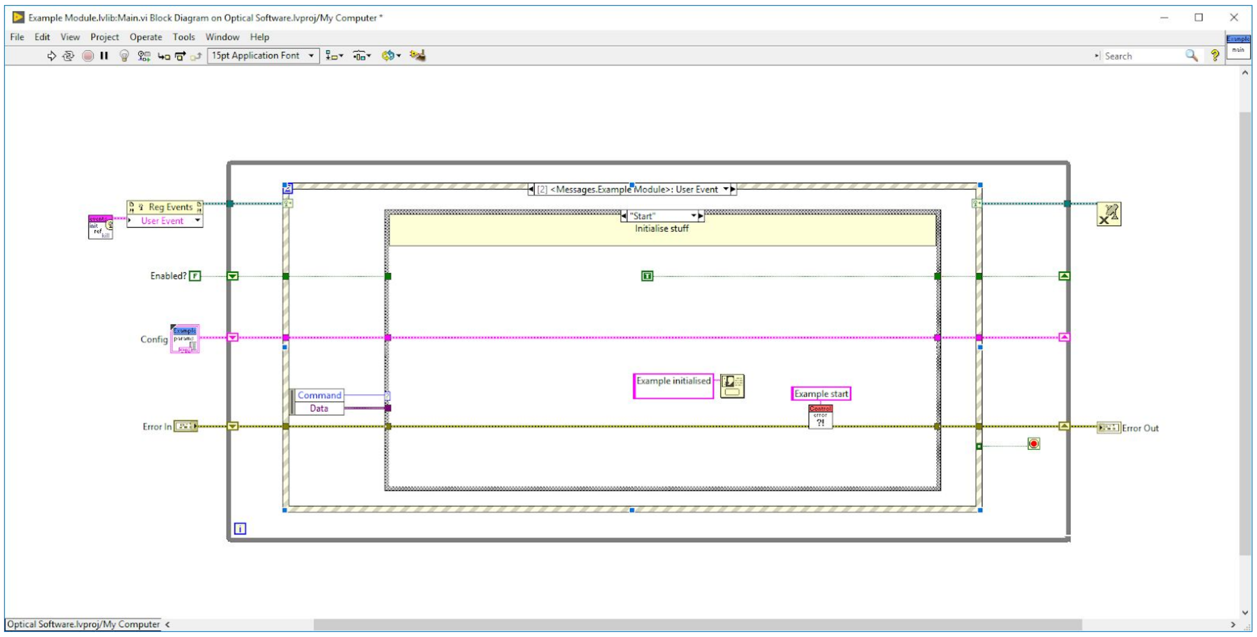
This provides an indication of how the architecture has been split into modules. Each module runs in parallel and provides a different purpose. Existing modules include: Main Controller Module, User Interface Module, Shutter Module, Monochromator Module, Colour Wheel Module, DAQ Module, Scope Module, File IO Module and Post Processing Module.

Breaking it down like this is better for performance and is good practice. Rather than one loop being responsible for all functionality, there are multiple loops sharing the work.
It should be made clear that the Top Level VI ***is not*** the User Interface. The User Interface is a module in itself and is the only module that is configured to show its Front Panel on execution. All other modules (and the Top Level VI) will run in the background and will not show their Front Panels.

Navigating the Architecture

The layout of each module will look similar to an extent. That is based on the fact that they all originate from the same 'Example Module'. This has been left in the project file and the 'Creating a Module' section will explain how it is possible to use this to create new modules.

The block diagram of the Example Module is shown below.



The module uses 'User Events' to interact and transfer messages with other modules. The event structure has dynamically registered for an event and is constantly waiting for a message to be sent to it. As mentioned previously, this message will have been sent using one of the API's from within the Example Module library file. When a message is received by the module it unpacks it and reads the command and data. This command is wired directly to a case structure and will perform whatever task is within that case.

This is a highly expandable method of module development in that many different commands/API's can be created and the module will be able to handle them; as long as a case has been created within that module.

A module could have dozens of different commands that it needs to handle. It is down to the developer as to what this command will do when the module receives it.

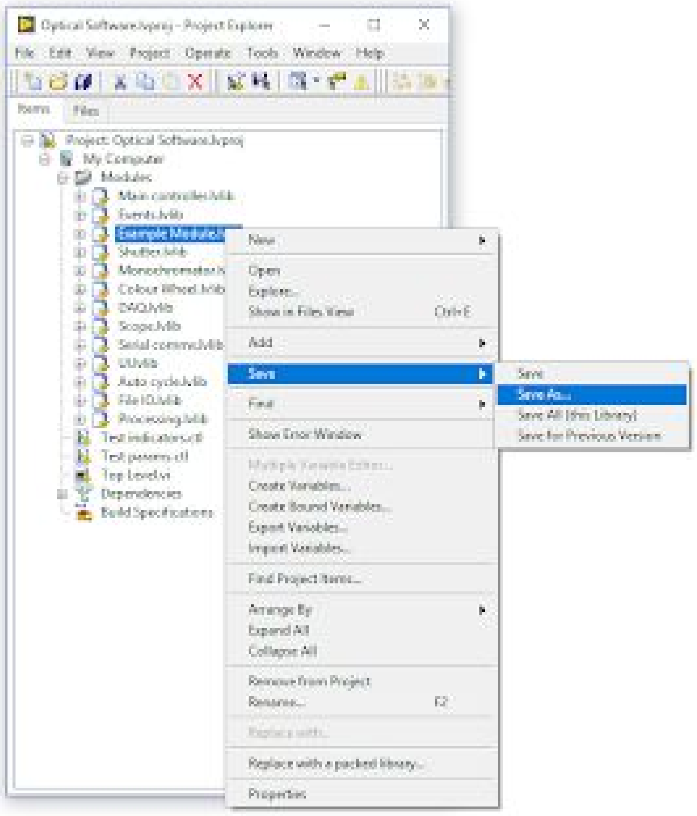
More information on Event-Driven Programming in LabVIEW can be found at the following link: http://www.ni.com/white-paper/3331/en/ and additional information on Dynamically Registering for Events can be found here:
http://zone.ni.com/reference/en-XX/help/371361K-01/lvhowto/dynamic_register_event/

Creating a New Module

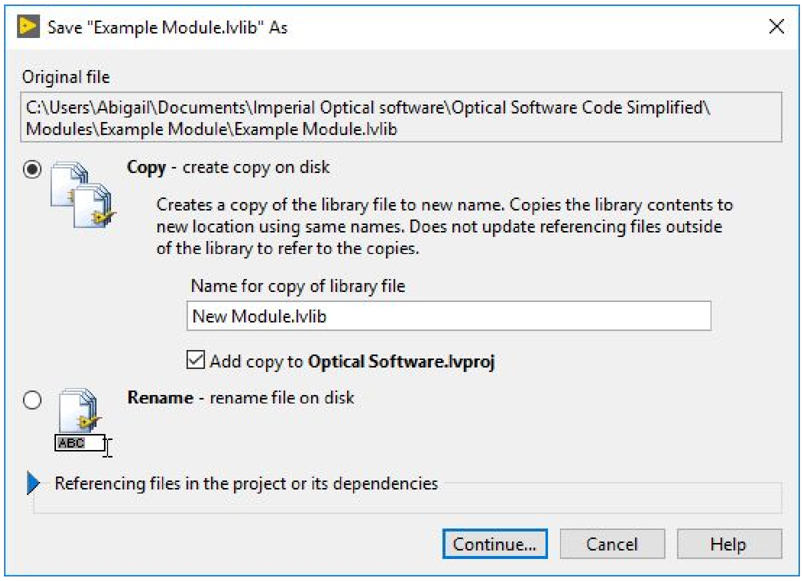
When looking to create a new module it is best to make a duplication of the 'Example Module' library. As it is a LabVIEW Library, all of its contents, layout and structure will be replicated and there will be no name conflicts. This is another benefit of using libraries.

The steps of creating a new module using Project Explorer are listed below:

1. Create a copy of 'Example Module.lvlib' by right clicking on it and selecting Save As.



2. Select a new Name and Select Continue…



Save "Example Module.lvlib" As

Original file

C:\Users\Abigail\Documents\Imperial Optical software\Optical Software Code Simplified\Modules\Example Module\Example Module.lvlib

● **Copy** - create copy on disk

Creates a copy of the library file to new name. Copies the library contents to new location using same names. Does not update referencing files outside of the library to refer to the copies.

Name for copy of library file

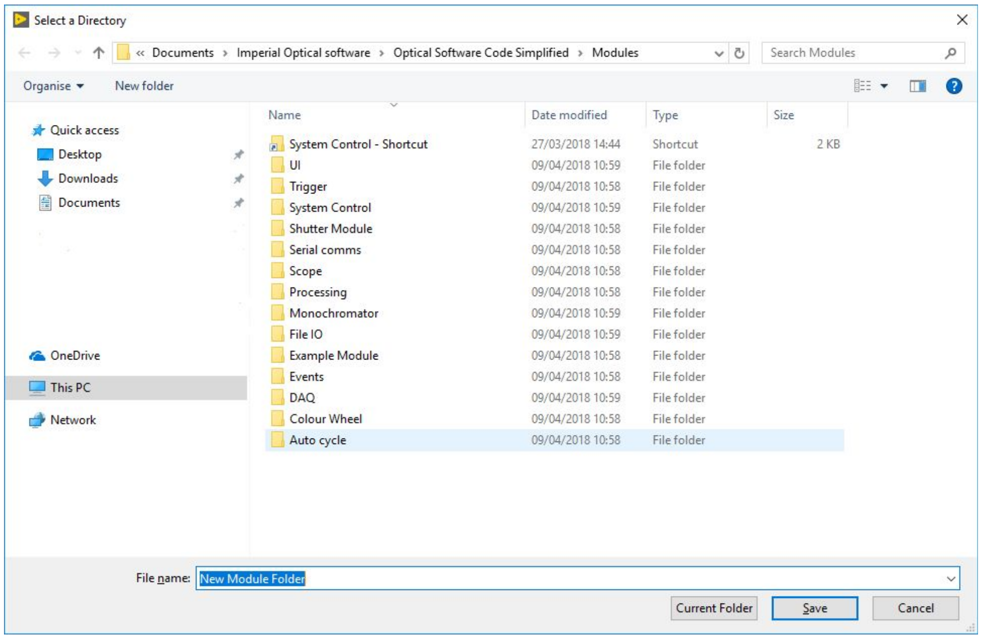
New Module.lvlib

☑ Add copy to **Optical Software.lvproj**

○ **Rename** - rename file on disk

Referencing files in the project or its dependencies

3. Select a directory location and save it.



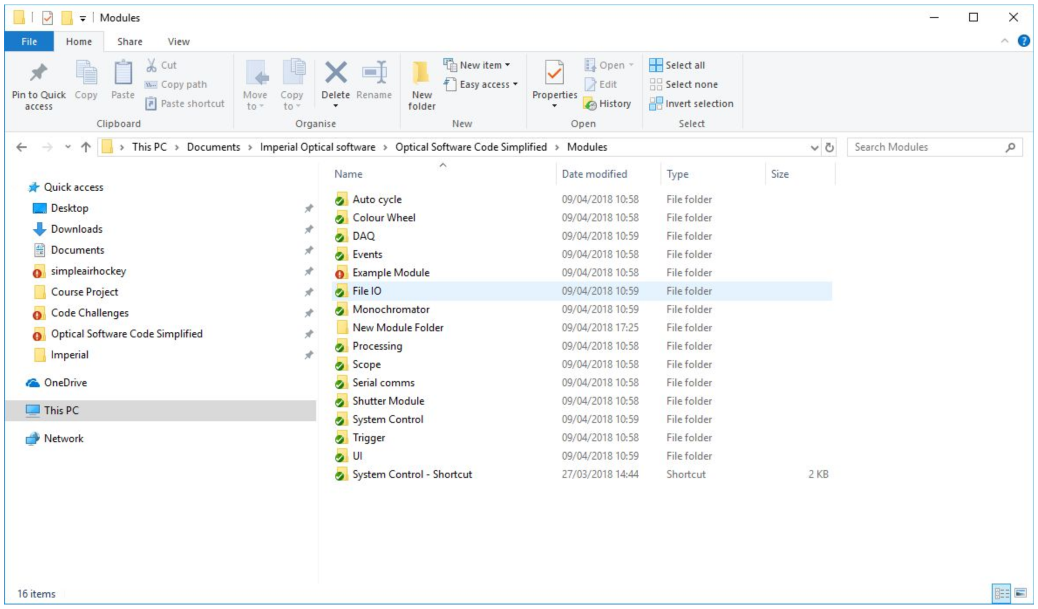
Select a Directory

« Documents › Imperial Optical software › Optical Software Code Simplified › Modules

Name	Date modified	Type	Size
System Control - Shortcut	27/03/2018 14:44	Shortcut	2 KB
UI	09/04/2018 10:59	File folder	
Trigger	09/04/2018 10:58	File folder	
System Control	09/04/2018 10:59	File folder	
Shutter Module	09/04/2018 10:58	File folder	
Serial comms	09/04/2018 10:58	File folder	
Scope	09/04/2018 10:58	File folder	
Processing	09/04/2018 10:58	File folder	
Monochromator	09/04/2018 10:59	File folder	
File IO	09/04/2018 10:59	File folder	
Example Module	09/04/2018 10:58	File folder	
Events	09/04/2018 10:58	File folder	
DAQ	09/04/2018 10:59	File folder	
Colour Wheel	09/04/2018 10:58	File folder	
Auto cycle	09/04/2018 10:58	File folder	

File name: New Module Folder

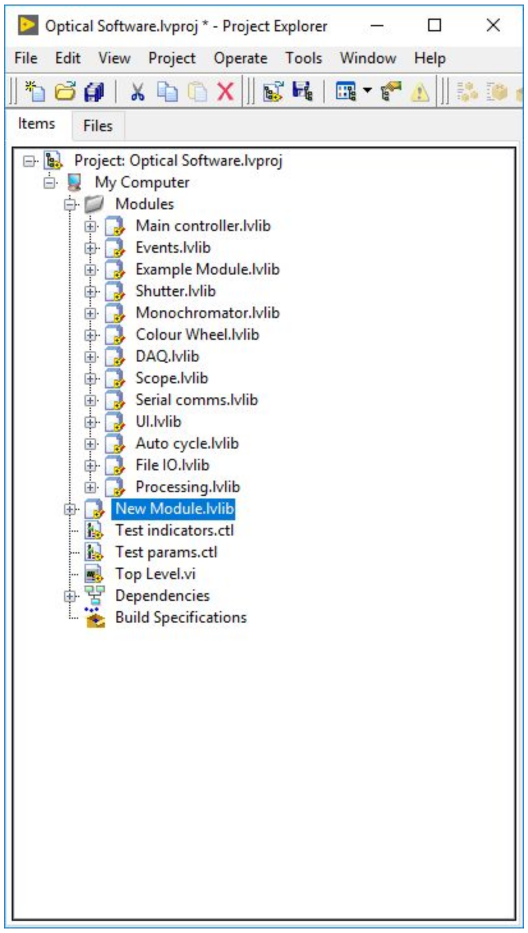
4. Check that the folder and its contents have been saved correctly.



This PC > Documents > Imperial Optical software > Optical Software Code Simplified > Modules

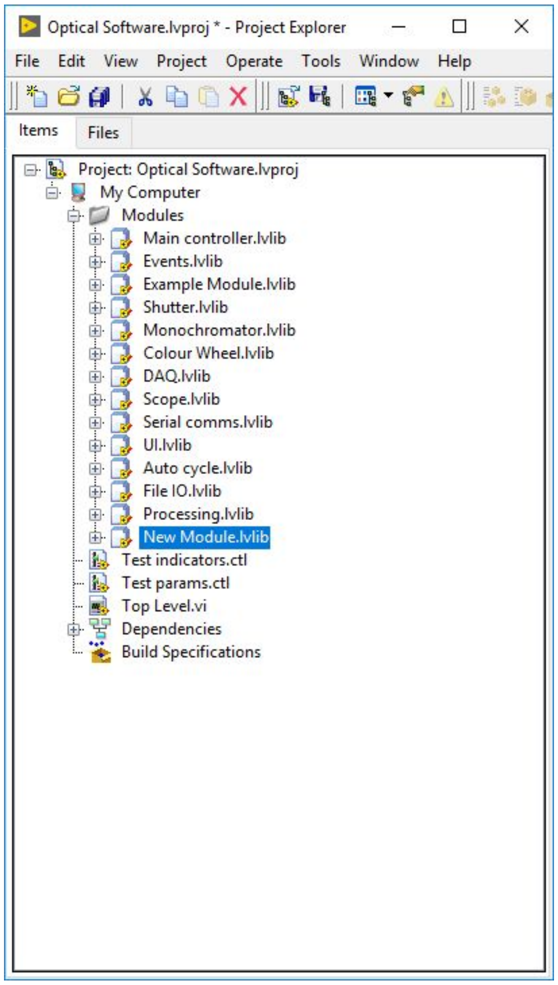
Name	Date modified	Type	Size
Auto cycle	09/04/2018 10:58	File folder	
Colour Wheel	09/04/2018 10:58	File folder	
DAQ	09/04/2018 10:59	File folder	
Events	09/04/2018 10:58	File folder	
Example Module	09/04/2018 10:58	File folder	
File IO	09/04/2018 10:59	File folder	
Monochromator	09/04/2018 10:59	File folder	
New Module Folder	09/04/2018 17:25	File folder	
Processing	09/04/2018 10:58	File folder	
Scope	09/04/2018 10:58	File folder	
Serial comms	09/04/2018 10:58	File folder	
Shutter Module	09/04/2018 10:58	File folder	
System Control	09/04/2018 10:59	File folder	
Trigger	09/04/2018 10:58	File folder	
UI	09/04/2018 10:59	File folder	
System Control - Shortcut	27/03/2018 14:44	Shortcut	2 KB

5. Switch to Project Explorer to view the new library in the project file.



```
Project: Optical Software.lvproj
  My Computer
    Modules
      Main controller.lvlib
      Events.lvlib
      Example Module.lvlib
      Shutter.lvlib
      Monochromator.lvlib
      Colour Wheel.lvlib
      DAQ.lvlib
      Scope.lvlib
      Serial comms.lvlib
      UI.lvlib
      Auto cycle.lvlib
      File IO.lvlib
      Processing.lvlib
    New Module.lvlib
    Test indicators.ctl
    Test params.ctl
    Top Level.vi
    Dependencies
    Build Specifications
```

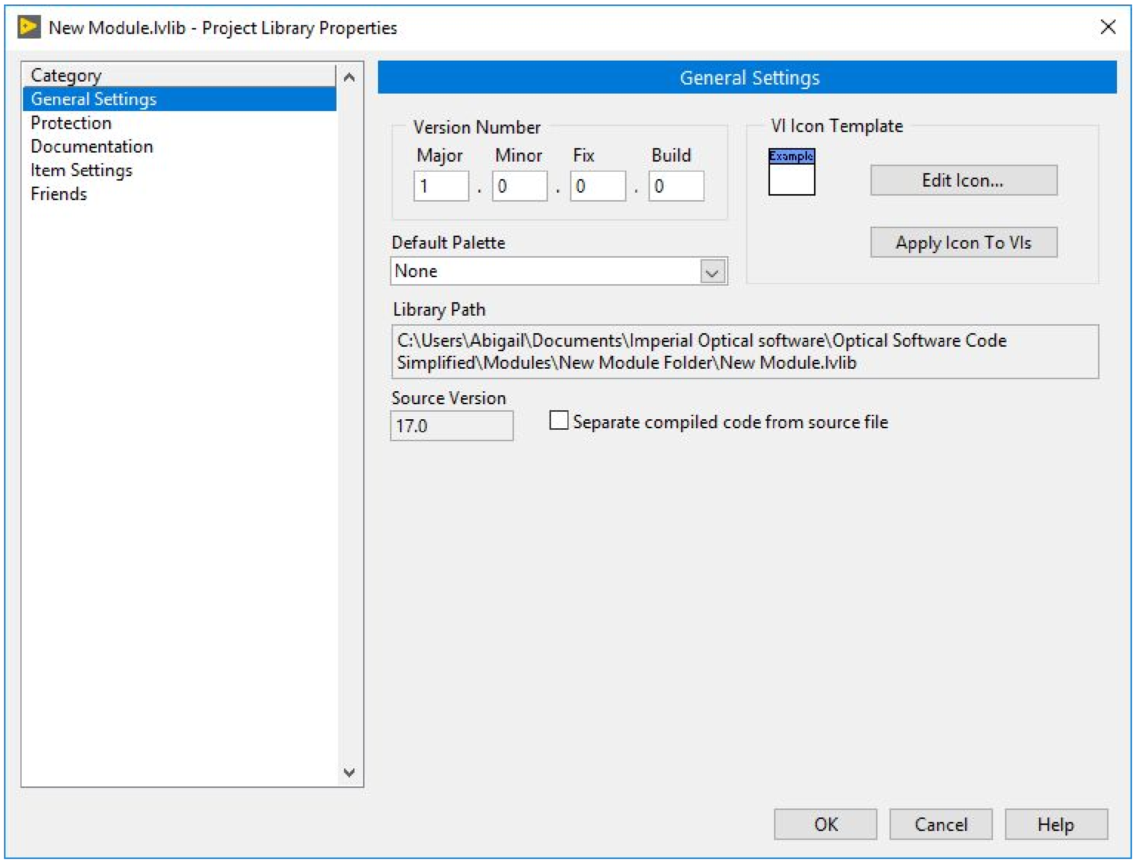
6. For organisation purposes, move this library to the ‘Modules’ folder by clicking it and dragging.


Optical Software.lvproj * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files

Project: Optical Software.lvproj
My Computer
Modules
Main controller.lvlib
Events.lvlib
Example Module.lvlib
Shutter.lvlib
Monochromator.lvlib
Colour Wheel.lvlib
DAQ.lvlib
Scope.lvlib
Serial comms.lvlib
UI.lvlib
Auto cycle.lvlib
File IO.lvlib
Processing.lvlib
New Module.lvlib
Test indicators.ctl
Test params.ctl
Top Level.vi
Dependencies
Build Specifications

7. It is sensible to change the appearance of all VI's within this library. Ie, change the color theme of the icons so sub VI's from this module are easily distinguishable when reviewing the code. In order to do this, right click on the library and select Properties.

8. Go to the General Settings category and select Edit Icon...



Version Number: Major 1 . Minor 0 . Fix 0 . Build 0

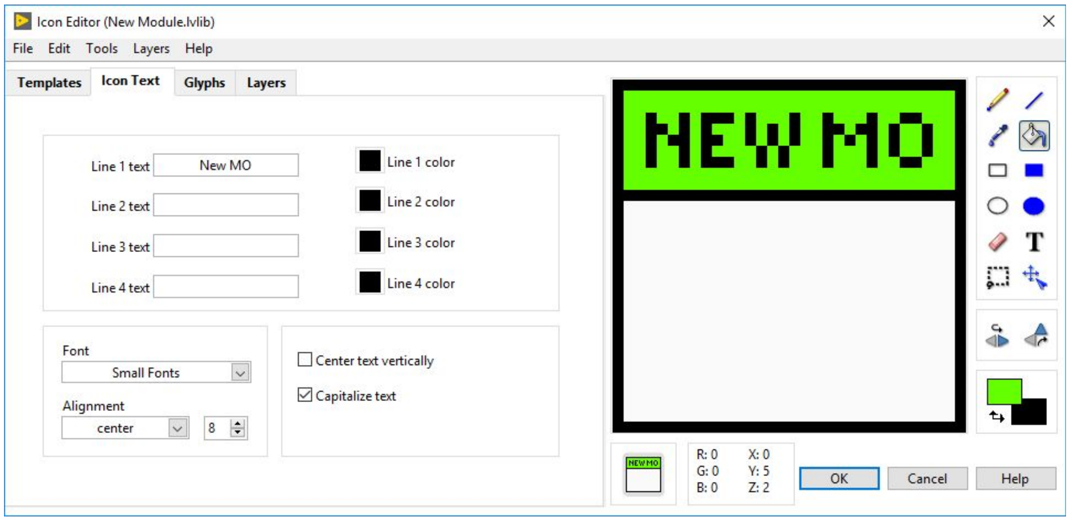
Default Palette: None

Library Path: C:\Users\Abigail\Documents\Imperial Optical software\Optical Software Code Simplified\Modules\New Module Folder\New Module.lvlib

Source Version: 17.0

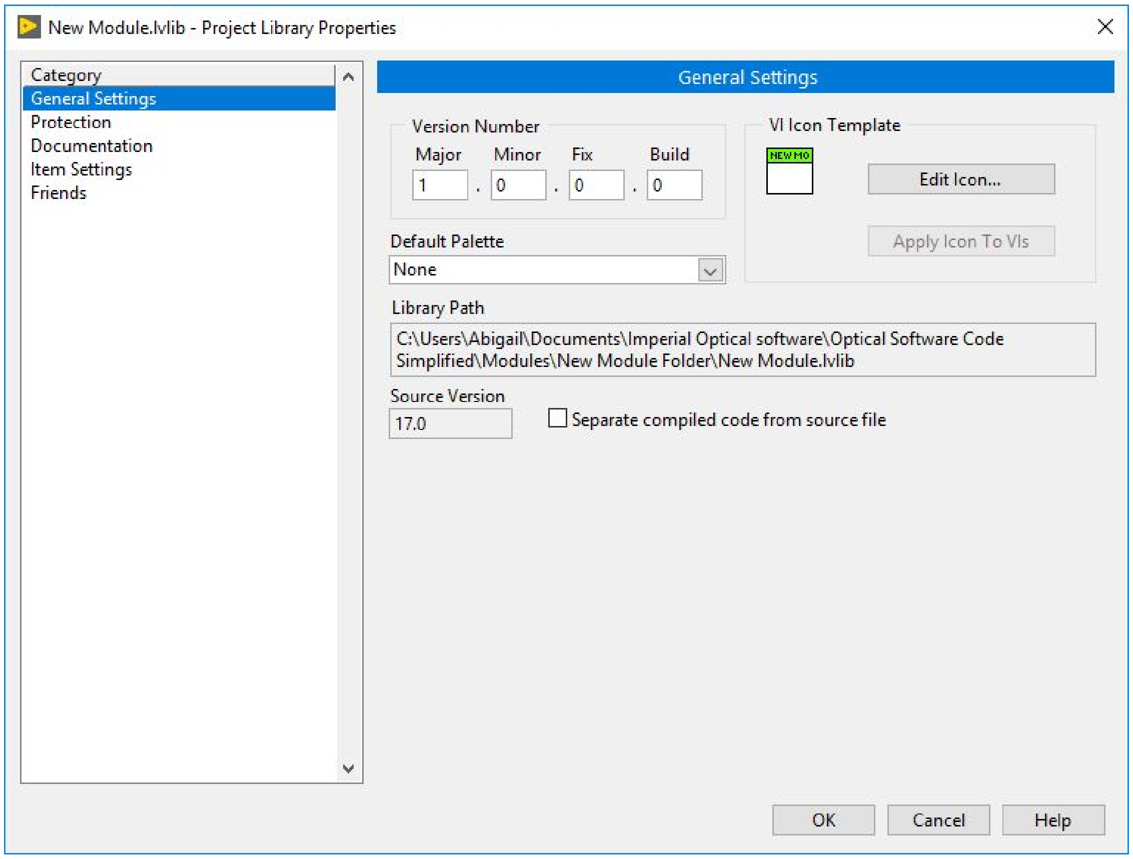
☐ Separate compiled code from source file

9. Using the toolbox palette on the right hand side, change the color of the banner (at the top of the icon) and use the Icon Text tab to change the text that is displayed. Below is an example. It can be whatever theme suits the developer. The important thing is that it is different from the existing modules. When happy with the theme, select OK.


Icon Editor (New Module.lvlib)
File Edit Tools Layers Help
Templates Icon Text Glyphs Layers
NEW MO
OK Cancel Help

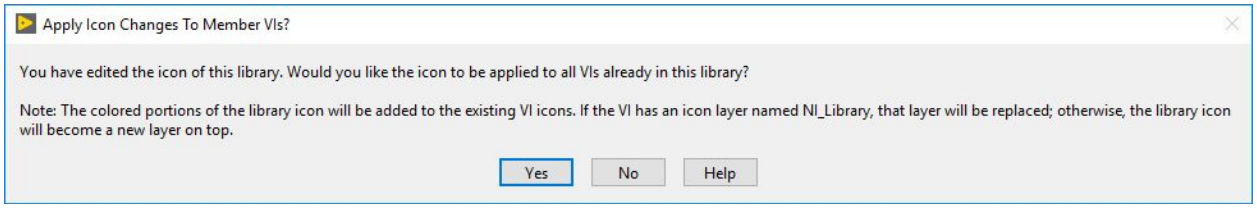
Line 1 text: New MO
Line 2 text:
Line 3 text:
Line 4 text:
Line 1 color
Line 2 color
Line 3 color
Line 4 color
Font: Small Fonts
Alignment: center 8
☐ Center text vertically
☑ Capitalize text
R: 0 G: 0 B: 0
X: 0 Y: 5 Z: 2

10. A preview of how the VI's in this library will appear is now shown. Select OK apply the changes.



Version Number
Major 1 . Minor 0 . Fix 0 . Build 0
VI Icon Template
Default Palette: None
Library Path: C:\Users\Abigail\Documents\Imperial Optical software\Optical Software Code Simplified\Modules\New Module Folder\New Module.lvlib
Source Version: 17.0
☐ Separate compiled code from source file

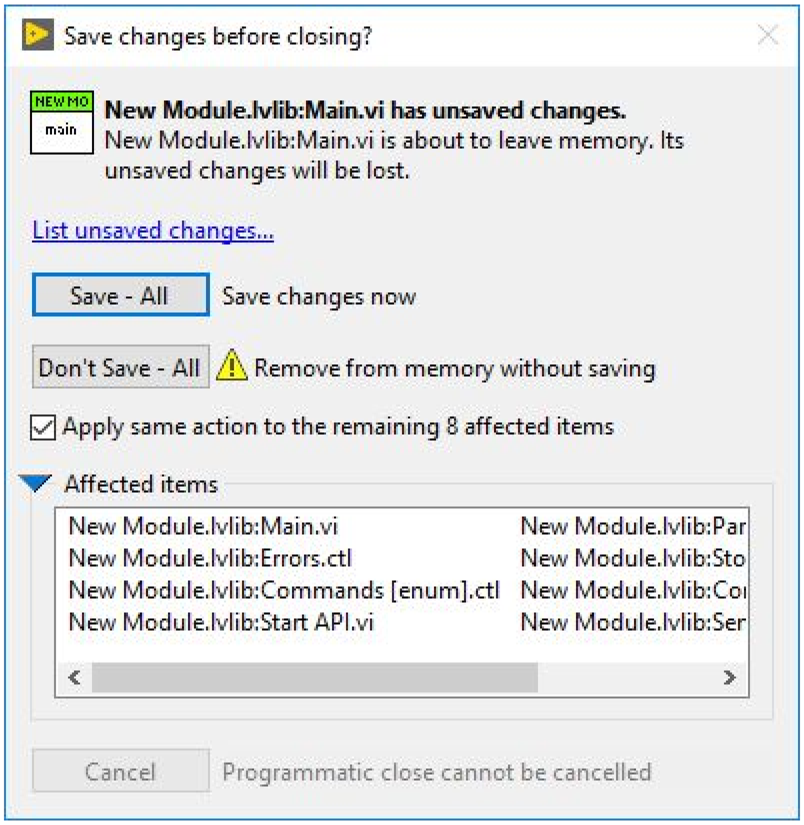
11. A warning will be shown informing that this Icon will be applied to all the VI's in this library and overwrite the existing Icons. Select Yes to confirm this is understood.


Apply Icon Changes To Member VIs?

You have edited the icon of this library. Would you like the icon to be applied to all VIs already in this library?

Note: The colored portions of the library icon will be added to the existing VI icons. If the VI has an icon layer named NI_Library, that layer will be replaced; otherwise, the library icon will become a new layer on top.

12. As these changes affect all VI's in the library, they will need to be saved. When prompted to do this select Save - All.


Save changes before closing?

New Module.lvlib:Main.vi has unsaved changes.
New Module.lvlib:Main.vi is about to leave memory. Its unsaved changes will be lost.

List unsaved changes...

Save - All Save changes now

Don't Save - All Remove from memory without saving

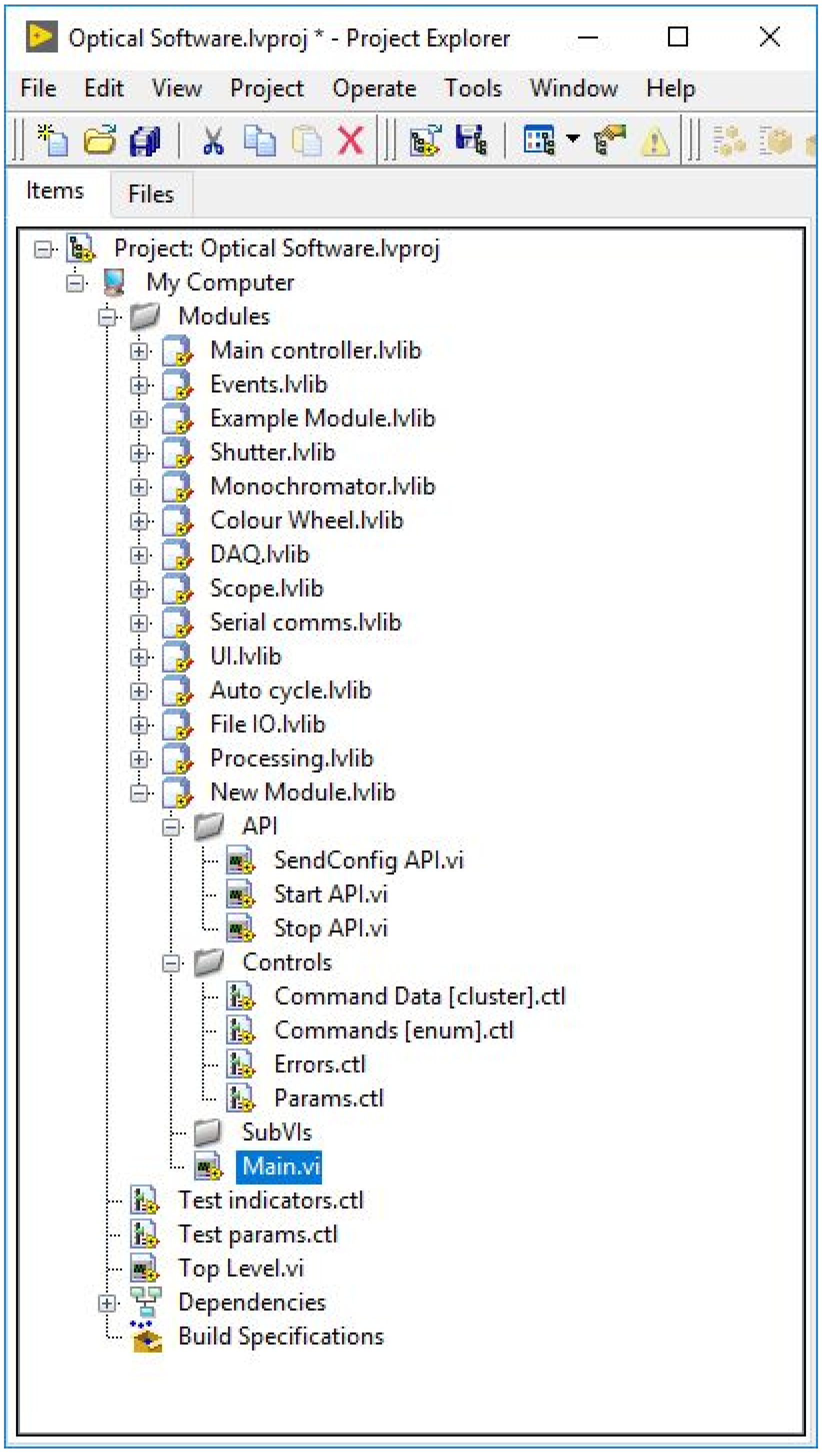
☑ Apply same action to the remaining 8 affected items

Affected items

New Module.lvlib:Main.vi
New Module.lvlib:Errors.ctl
New Module.lvlib:Commands [enum].ctl
New Module.lvlib:Start API.vi
New Module.lvlib:Par
New Module.lvlib:Sto
New Module.lvlib:Co
New Module.lvlib:Ser

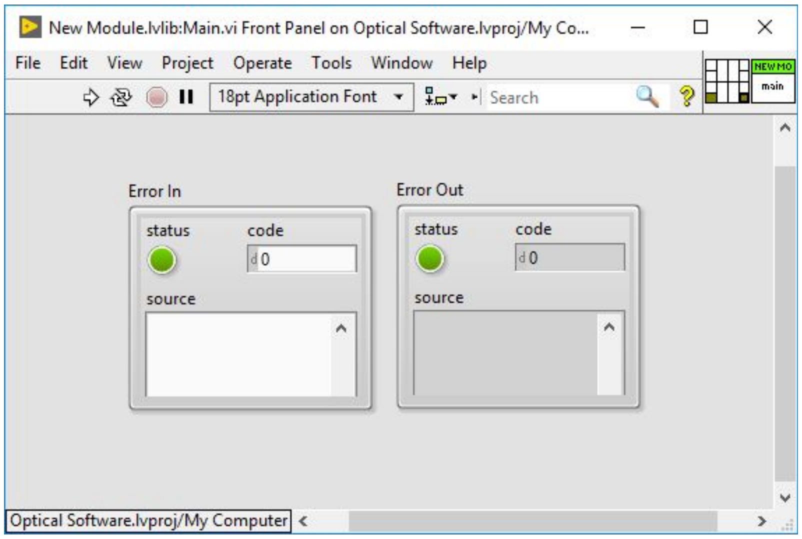
Cancel Programmatic close cannot be cancelled

13. From the Project Explorer, navigate to Main.vi within the new library and double click to open it.

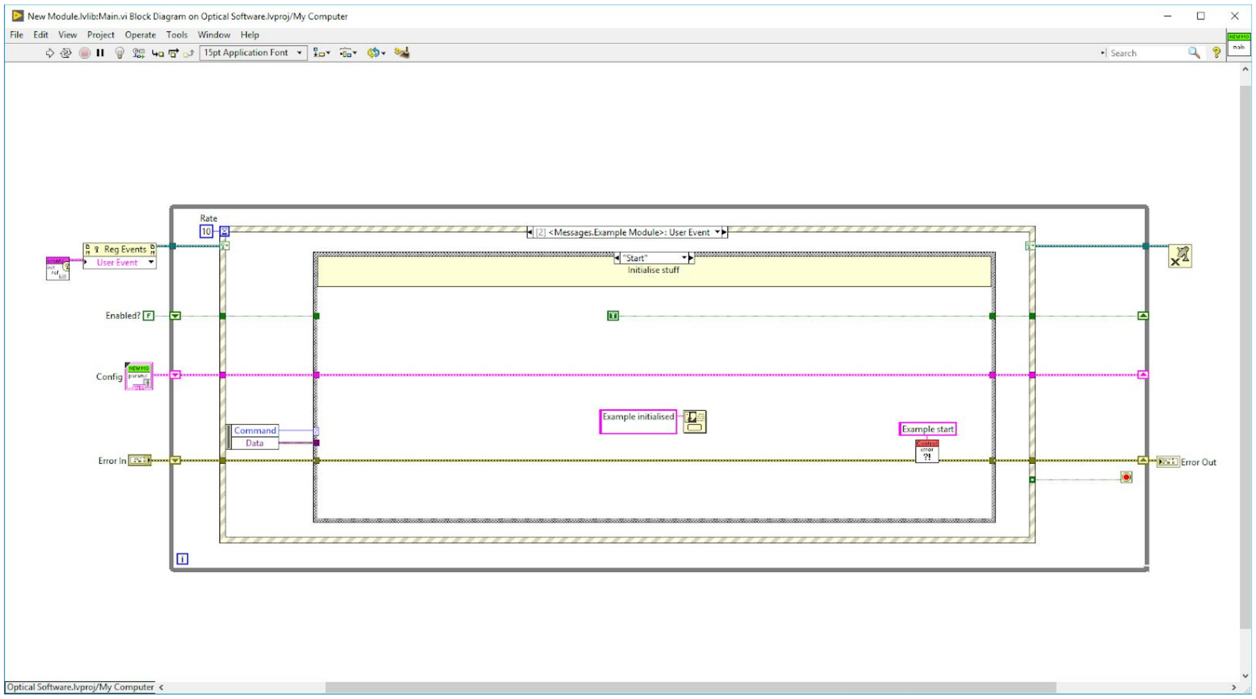

Optical Software.lvproj * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files

```
Project: Optical Software.lvproj
  My Computer
    Modules
      Main controller.lvlib
      Events.lvlib
      Example Module.lvlib
      Shutter.lvlib
      Monochromator.lvlib
      Colour Wheel.lvlib
      DAQ.lvlib
      Scope.lvlib
      Serial comms.lvlib
      UI.lvlib
      Auto cycle.lvlib
      File IO.lvlib
      Processing.lvlib
      New Module.lvlib
        API
          SendConfig API.vi
          Start API.vi
          Stop API.vi
        Controls
          Command Data [cluster].ctl
          Commands [enum].ctl
          Errors.ctl
          Params.ctl
        SubVIs
        Main.vi
    Test indicators.ctl
    Test params.ctl
    Top Level.vi
    Dependencies
    Build Specifications
```

14. In the top right of the window it can that the VI now displays the Icon create in the previous steps.

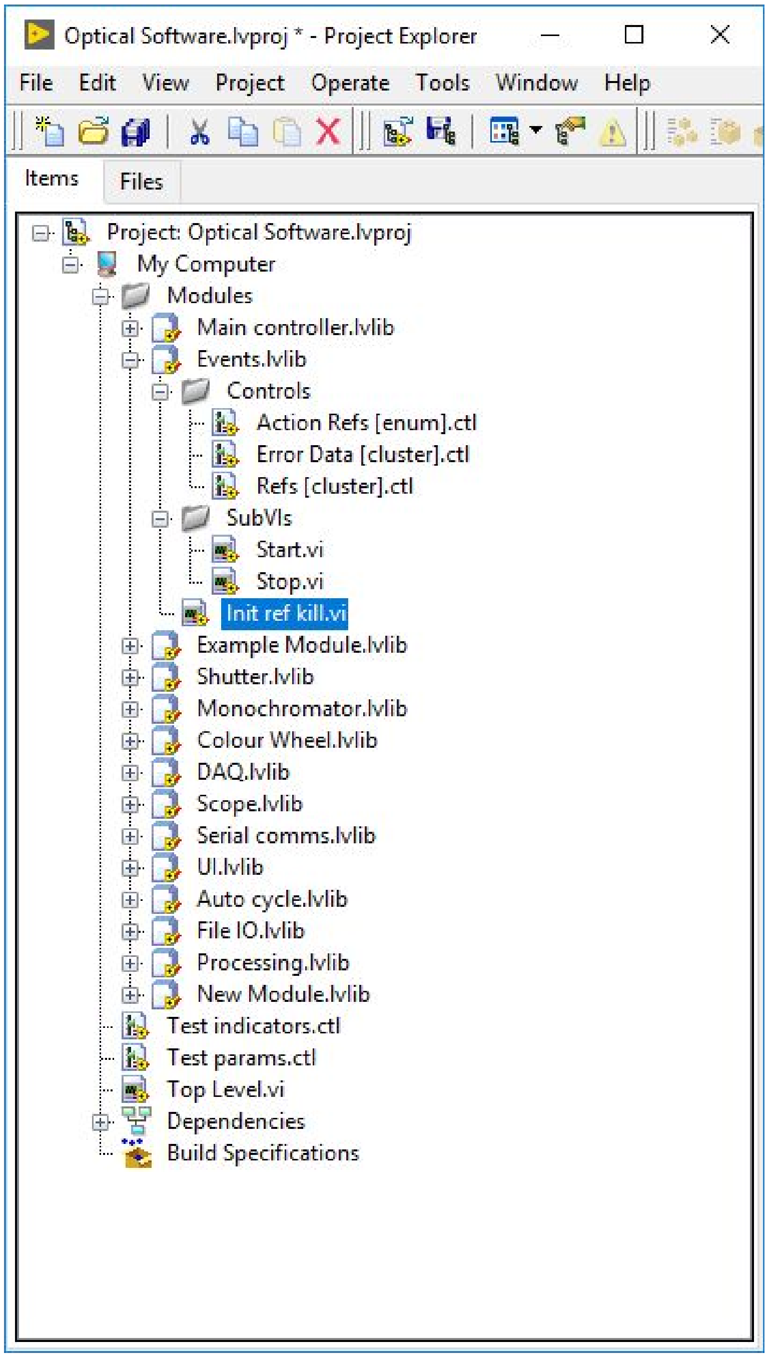

New Module.lvlib:Main.vi Front Panel on Optical Software.lvproj/My Co...
File Edit View Project Operate Tools Window Help
18pt Application Font
Search
NEW MO
main
Error In
status
code
source
Error Out
status
code
source
Optical Software.lvproj/My Computer

15. Pressing Ctrl + E will display the Block Diagram of this VI. This is the shell that will receive messages sent from other modules and execute functionality that is contained within it.

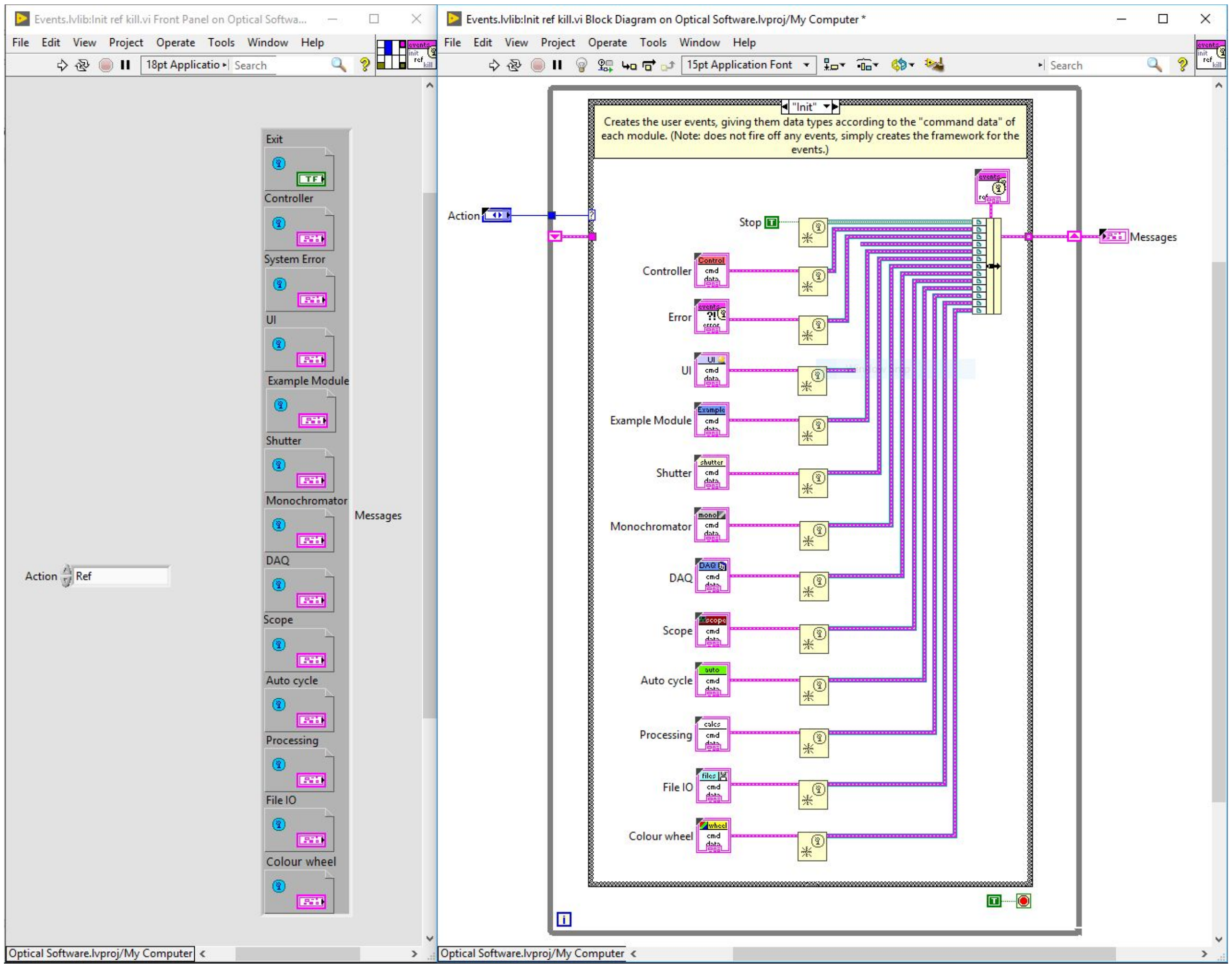

New Module.lvlib:Main.vi Block Diagram on Optical Software.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Rate
10
Reg Events
User Event
[2] <Messages.Example Module>: User Event
"Start"
Initialise stuff
Enabled?
Config
Command
Data
Example initialised
Example start
Error In
Error Out
Optical Software.lvproj/My Computer

Adding a New Module

16. Register Module for Events. It is now important that we Register this Module so that it can receive messages sent to it. This was explained in more detail earlier on in the document. Navigate to the Events library in the Project Explorer and Open Init Ref Kill.vi. As the name suggests this vi is responsible for initialising, referencing and killing the references used for passing the event based messages around the application.



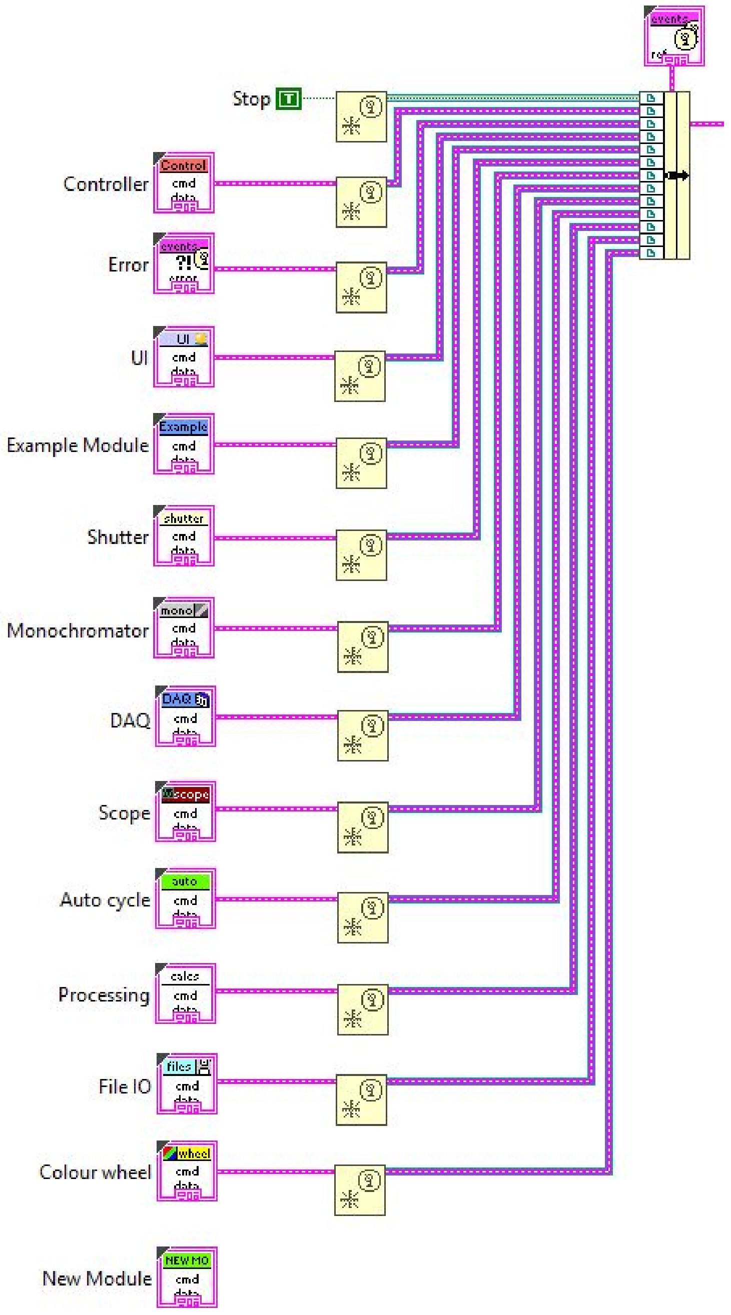
17. Press Ctrl + E to view the Block Diagram of the VI.


Events.lvlib:Init ref kill.vi Front Panel on Optical Softwa...
Events.lvlib:Init ref kill.vi Block Diagram on Optical Software.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
18pt Applicatio
15pt Application Font
Search
Exit
Controller
System Error
UI
Example Module
Shutter
Monochromator
DAQ
Scope
Auto cycle
Processing
File IO
Colour wheel
Messages
Action
Ref
"Init"
Stop
Error
Optical Software.lvproj/My Computer

Creates the user events, giving them data types according to the "command data" of each module. (Note: does not fire off any events, simply creates the framework for the events.)

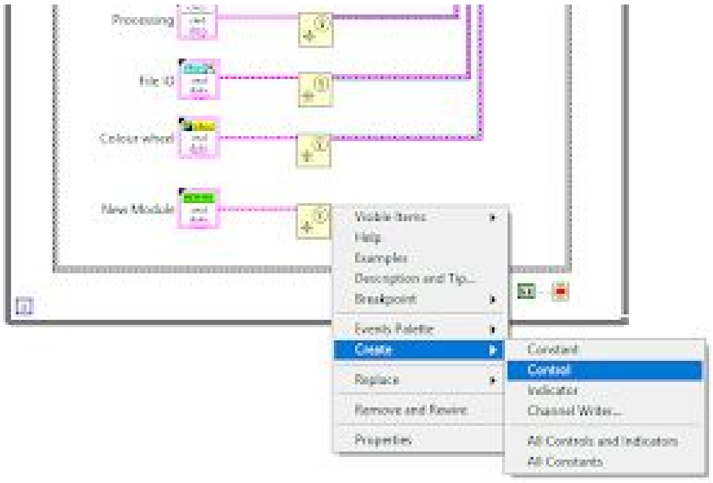
18. Right Click on the block diagram and select Select a VI from the menu. Navigate to Modules/New Module Folder/Controls/Command Data [cluster].ctl and select OK.

19. This will place one of the controls previous created dynamically when the new library was created. This control is a cluster specific to the library and contains the data that will be passed around within every message that is sent. The cluster consists of an enum and variant. This is generic enough to be adaptable to any message and data that needs to be sent.

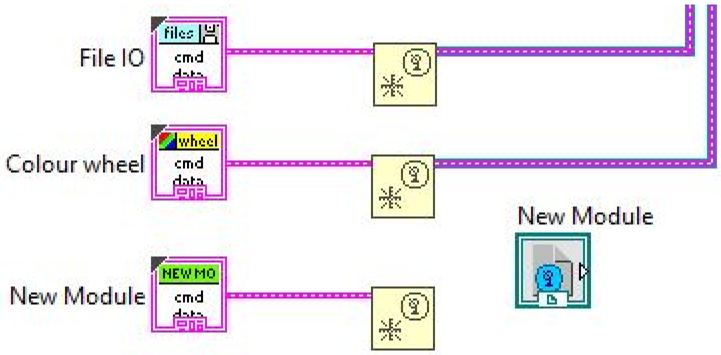


20. Right click on the Block Diagram and using the search function search for Create User Event. When found, drag this onto the Block Diagram. Alternatively, copy and paste one of the Create User Event functions that is already on the Block Diagram. Wire the New Module control to the User Event Data Type input on the Create User Event function.

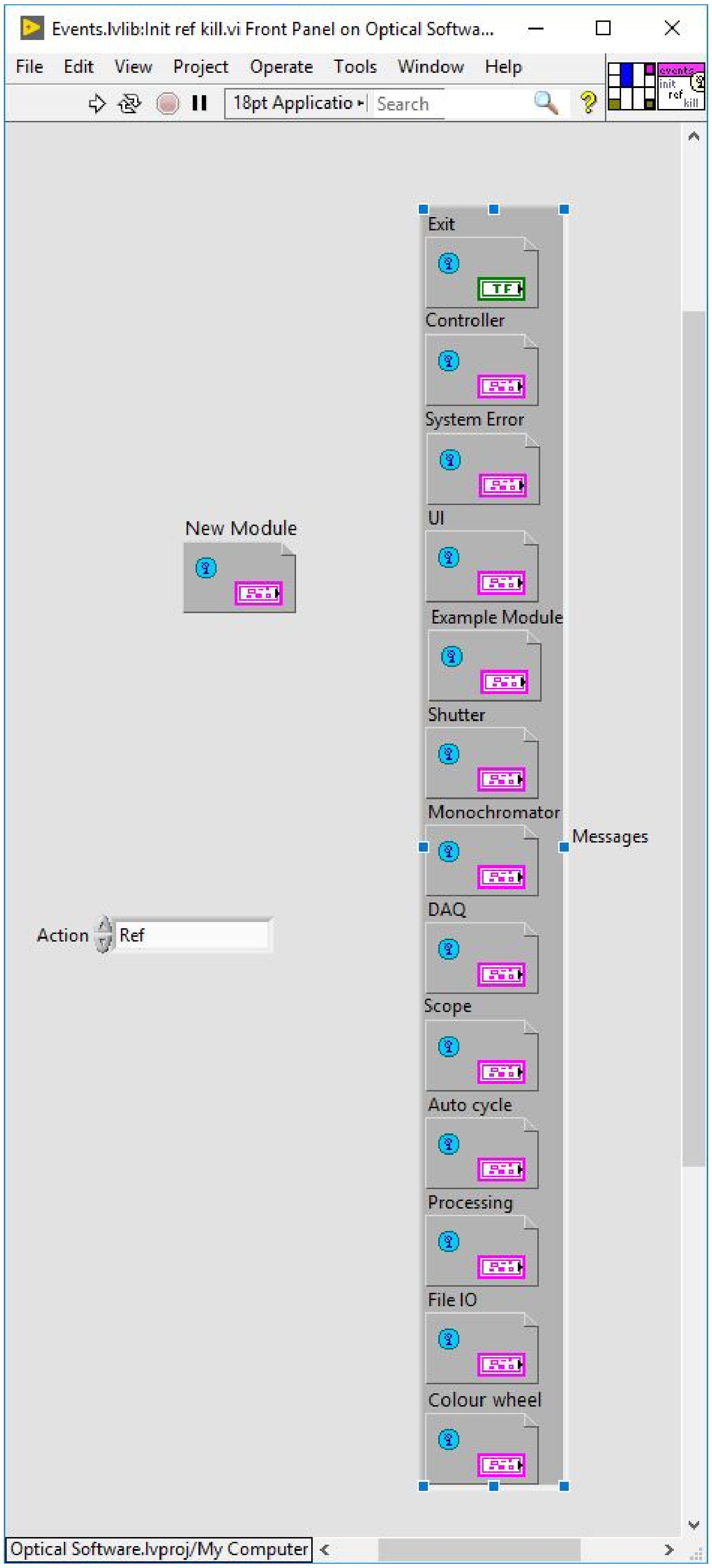
21. RIght click on the User Event Output of the Create User Event function and create a control.



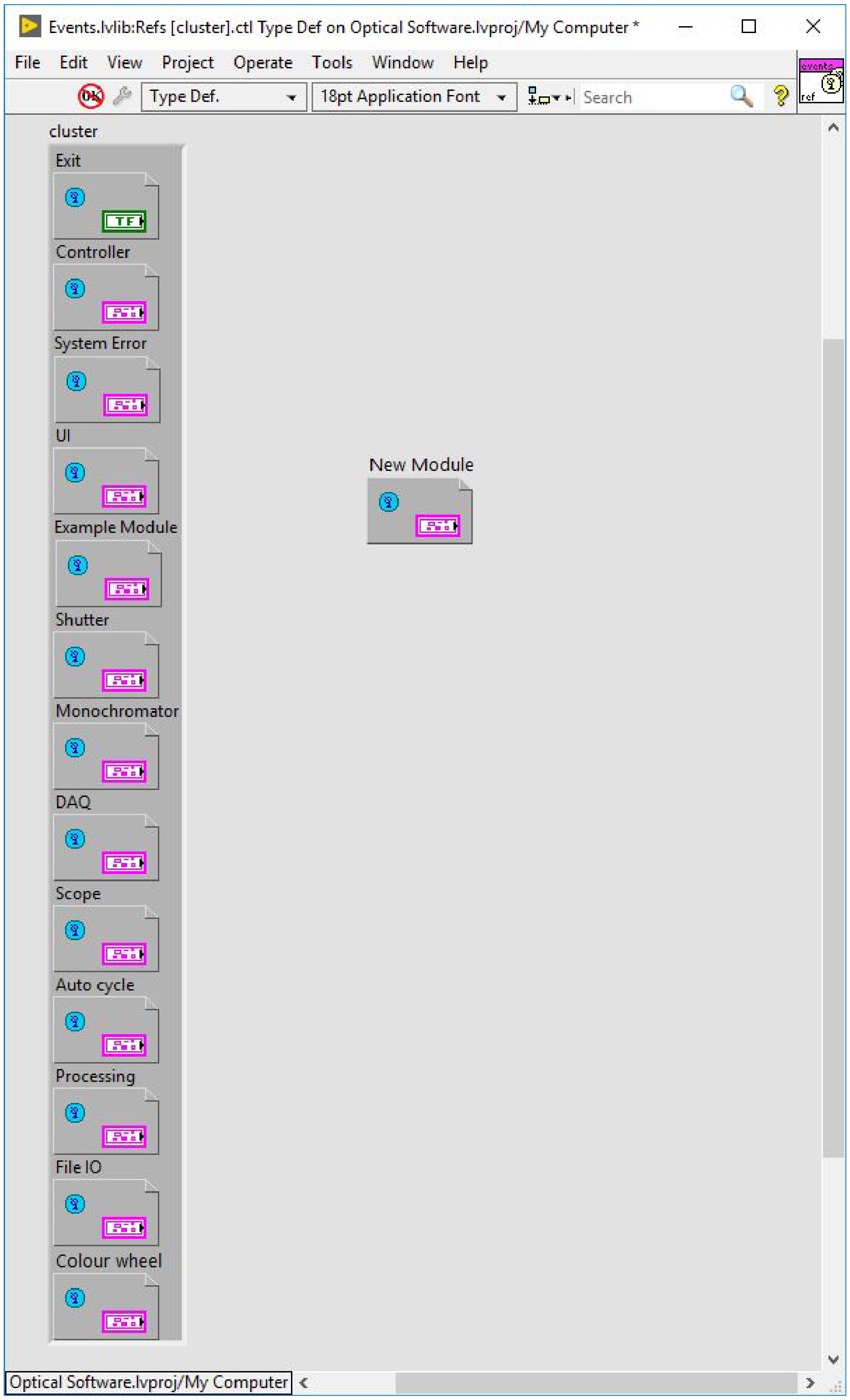
22. A control will be placed on the Block Diagram.



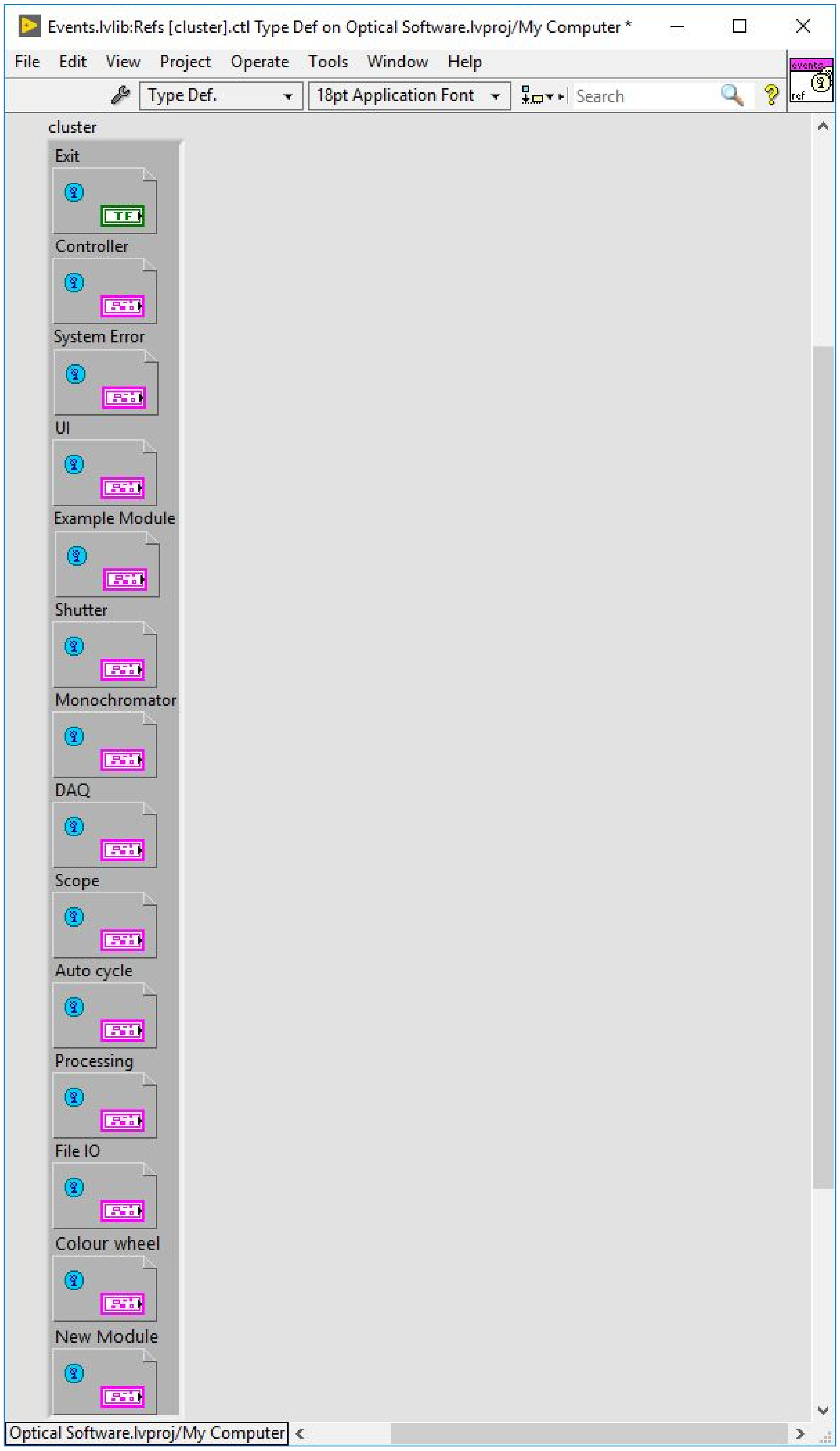
23. Double click this to find it on the Front Panel.



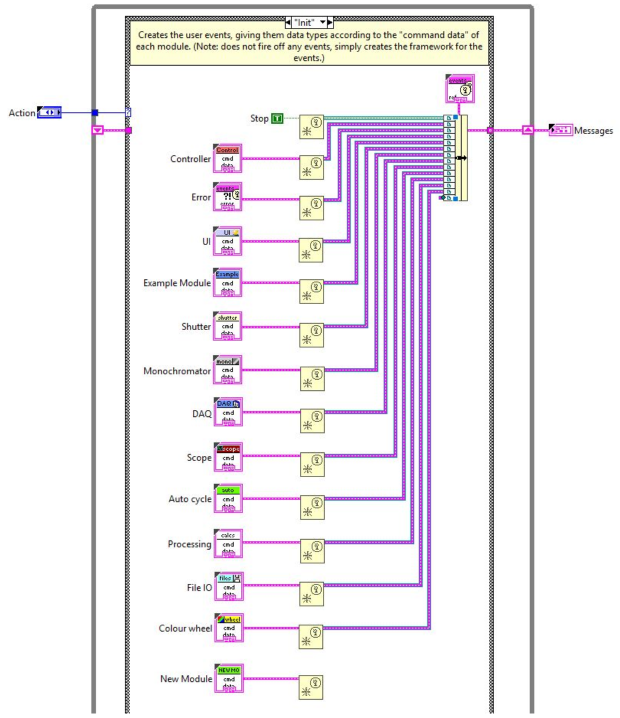
24. This now needs to be added to the Type Definition that contains all of the existing Event References. On the Front Panel, select the New Module reference and press Ctrl + X to add it to the clipboard. Right click on the Type Definition containing all the existing Event References and select Open Type Def. Press Ctrl + V to paste the New Module reference onto the Front Panel of this Type Definition.



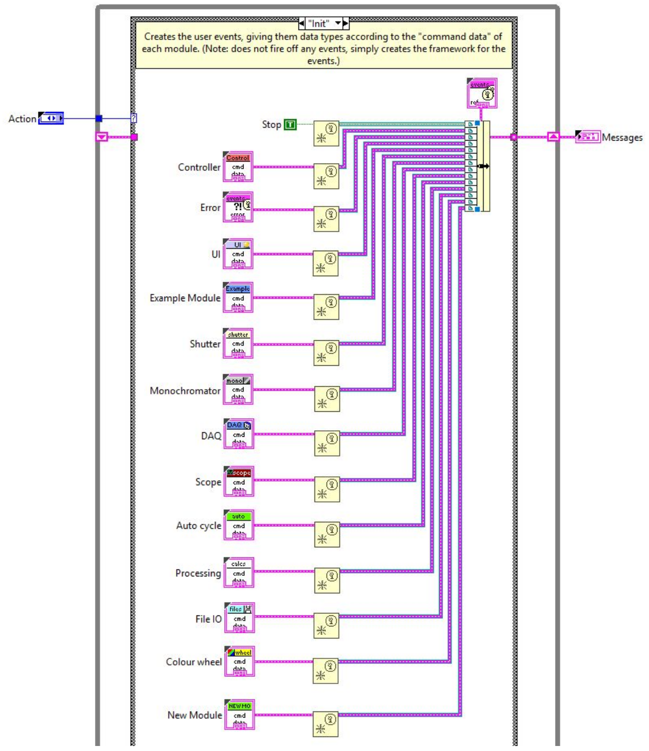
25. Drag the new reference into the cluster containing all the existing Event References. Save this and close the window.



26. Go back to the Block Diagram of the Init Ref Kill.vi. The bundle function will now have an additional input that requires a wire.


"Init"
Creates the user events, giving them data types according to the "command data" of each module. (Note: does not fire off any events, simply creates the framework for the events.)
Action
Stop
Controller
Error
UI
Example Module
Shutter
Monochromator
DAQ
Scope
Auto cycle
Processing
File IO
Colour wheel
New Module
Messages

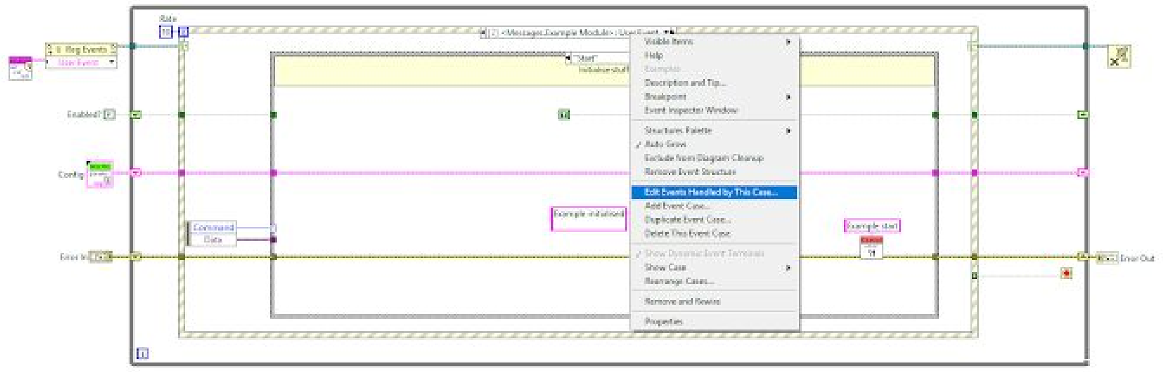
27. Link the Event Reference output from the Create New User function to this input.


"Init"
Creates the user events, giving them data types according to the "command data" of each module. (Note: does not fire off any events, simply creates the framework for the events.)
Action
Stop
Controller
Error
UI
Example Module
Shutter
Monochromator
DAQ
Scope
Auto cycle
Processing
File IO
Colour wheel
New Module
Messages

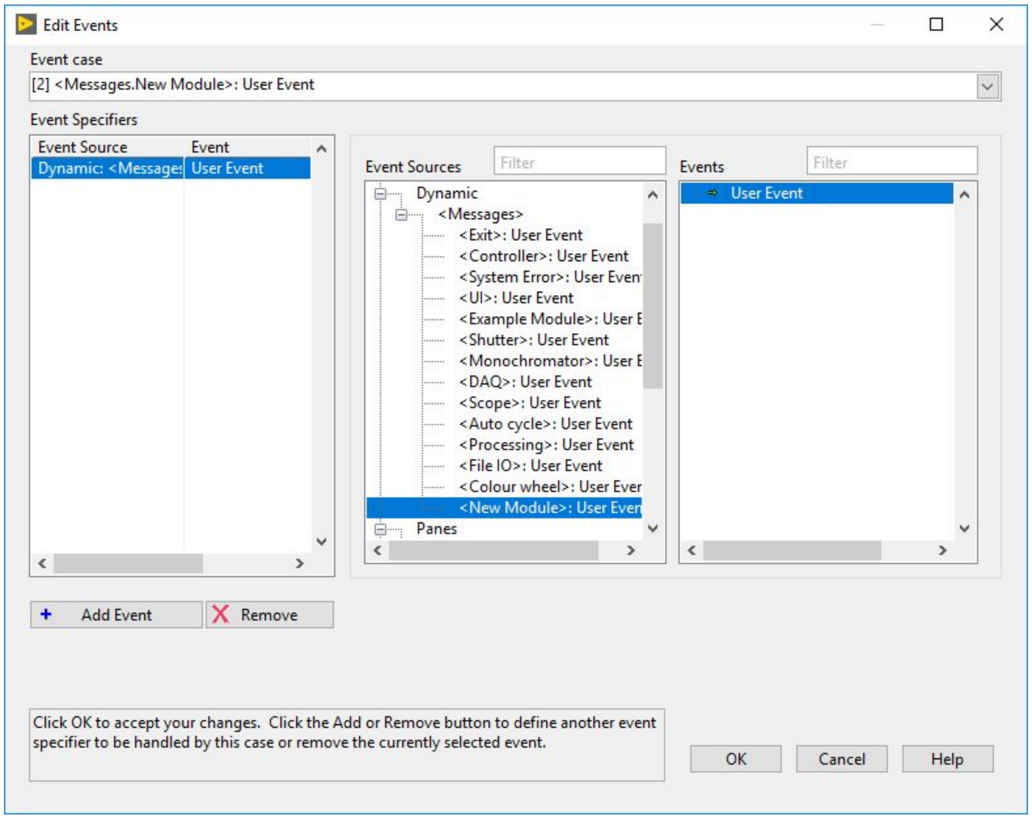
28. Save this VI and close the window.

29. Go back to Main.vi that is within the New Module library.

30. Switch to the User Event case of the Event Structure and right click on the drop down menu of it. Select Edit Events Handled by This Case…



31. Change the User Event being received by this module by selecting <New Module>: User Event from the Event Sources list. Select Ok to apply this.


Edit Events

Event case

[2] <Messages.New Module>: User Event

Event Specifiers

Event Source	Event
Dynamic: <Messages	User Event

Event Sources

- Dynamic
  - <Messages>
    - <Exit>: User Event
    - <Controller>: User Event
    - <System Error>: User Even
    - <UI>: User Event
    - <Example Module>: User E
    - <Shutter>: User Event
    - <Monochromator>: User E
    - <DAQ>: User Event
    - <Scope>: User Event
    - <Auto cycle>: User Event
    - <Processing>: User Event
    - <File IO>: User Event
    - <Colour wheel>: User Ever
    - <New Module>: User Even
- Panes

Events

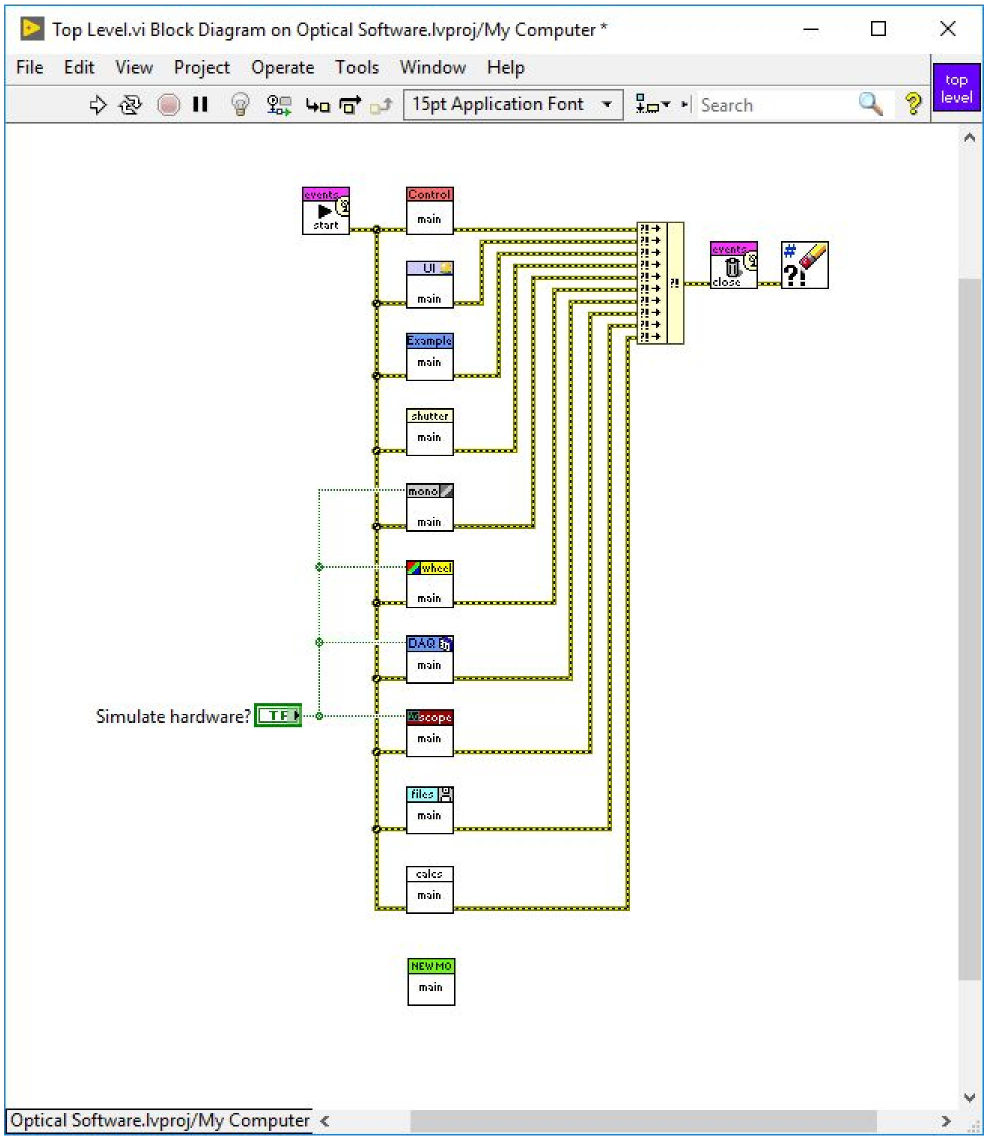
- User Event

Click OK to accept your changes. Click the Add or Remove button to define another event specifier to be handled by this case or remove the currently selected event.

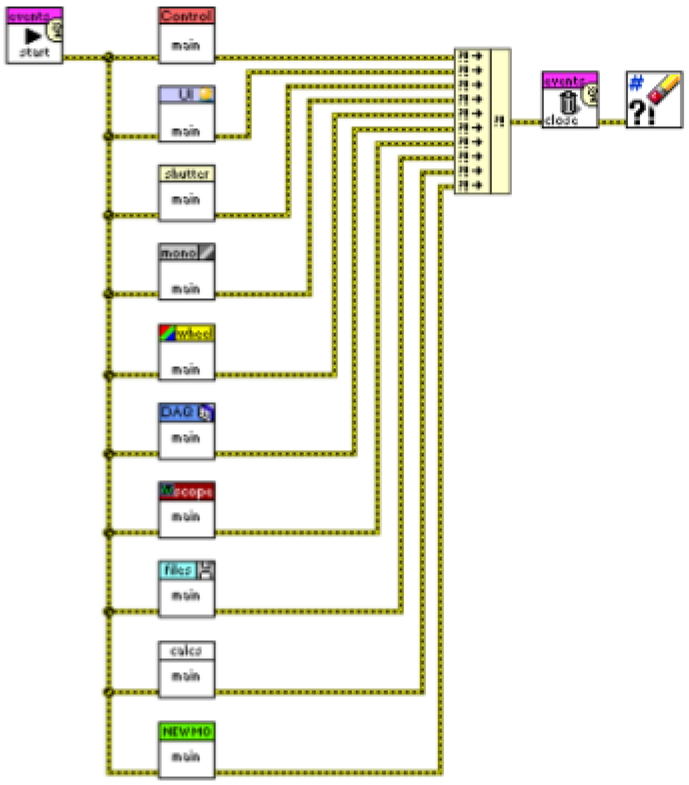
32. Save Main.vi and close the window.

33. From the Project Explorer, open Top Level.vi. This displays all the Modules currently used by the application.

34. In order to place the New Module in the Top Level vi, right click on the Block Diagram and select Select a VI from the menu. Navigate to Modules/New Module Folder/Main.vi and Select Ok.



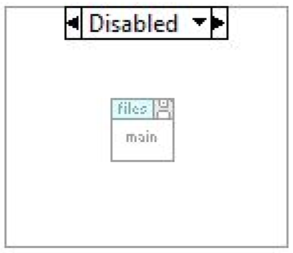
35. As with the existing Modules, wire the error cluster output of the Start Events vi to the error cluster input of the New Module Main.vi. Wire the error cluster output of the New Module Main.vi to the Merge Errors node. Expand the Merge Errors if necessary.



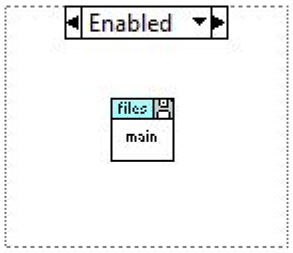
Removing/Disabling a Module

In order to remove a module it can simply be deleted from the Block Diagram of the code. It won't break the code but, from that point on, any API calls made to that module will do nothing and get no response.

A more subtle, and easily reversible, method would be to disable that section of the code. Using quick drop (Ctrl + Spacebar) and searching for Diagram Disable Structure will provide a structure that can be dragged around any part of the code that the developer wants to be able to view but not be included during runtime.



Right-clicking on the Diagram Disable Structure and selecting 'Enable this Subdiagram' will bring the code back into play and be included during runtime.



Right-clicking on the Diagram Disable Structure and selecting 'Disable this Subdiagram' will revert it back.

More information on the Diagram Disable Structure can be found here:
http://zone.ni.com/reference/en-XX/help/371361P-01/glang/diagram_disable_structure/

Testing an Isolated Module

The modularity of the architecture means that each module can be run on its own. Ie, if one aspect of the system needs to be tested it can be done so by select the module from the project explorer and executing the code. This should be treated as a debugging tool to debug a certain but of the application rather than a permanent method of controlling one of the devices.

A key thing to remember when doing this is that the config is no longer being sent from the File IO module. Therefore, the developer will need to input this in themselves.

Further Reading

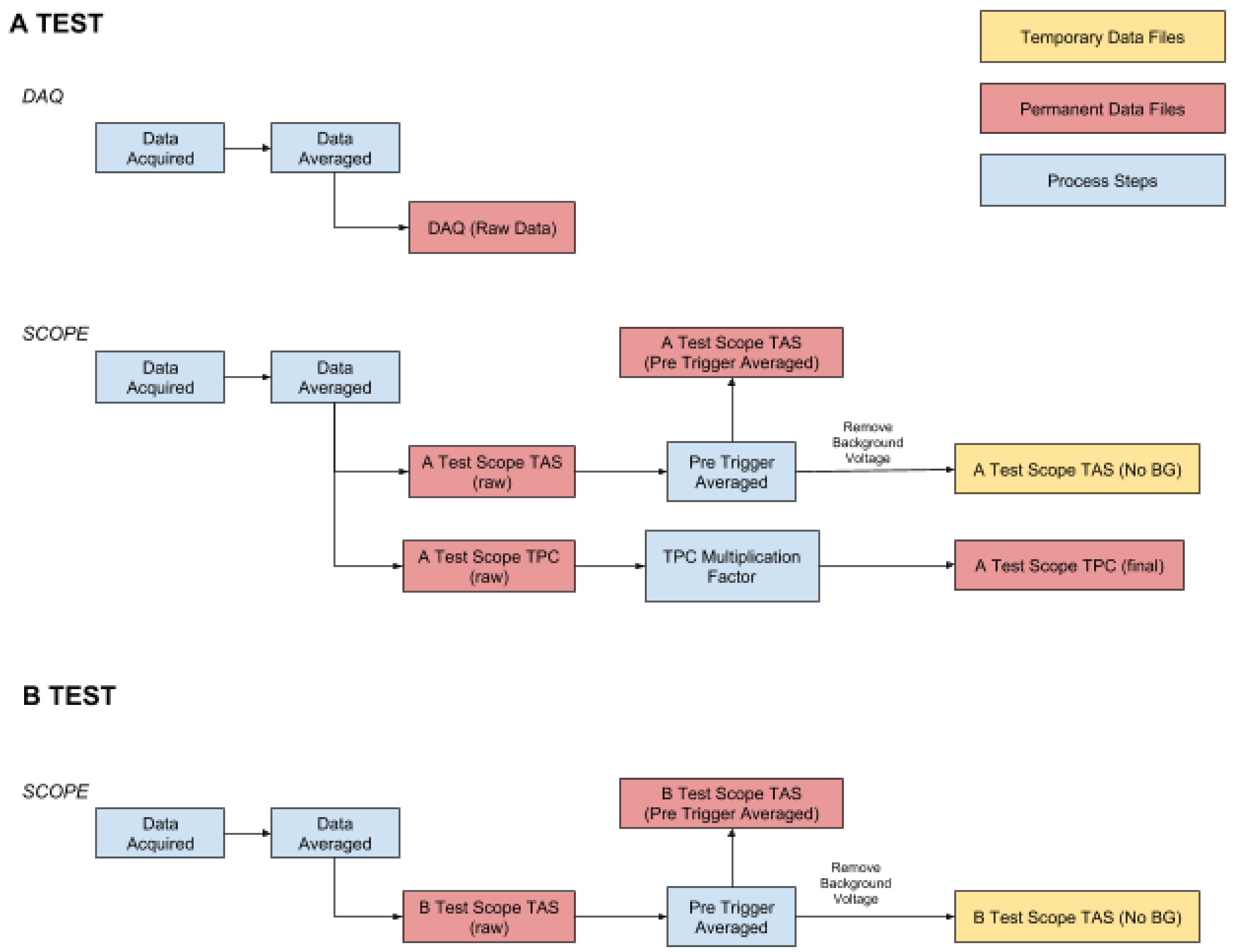
Debugging Techniques in LabVIEW: http://www.ni.com/tutorial/14130/en/

Event-Driven Programming in LabVIEW: http://www.ni.com/white-paper/3331/en/

Dynamically Registering for Events:
http://zone.ni.com/reference/en-XX/help/371361K-01/lvhowto/dynamic_register_event/

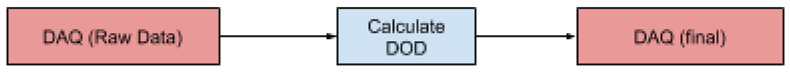
Diagram Disable Structure:
http://zone.ni.com/reference/en-XX/help/371361P-01/glang/diagram_disable_structure/

Appendix - File I/O Handling

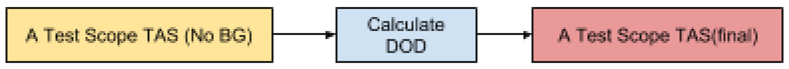


POST PROCESSING

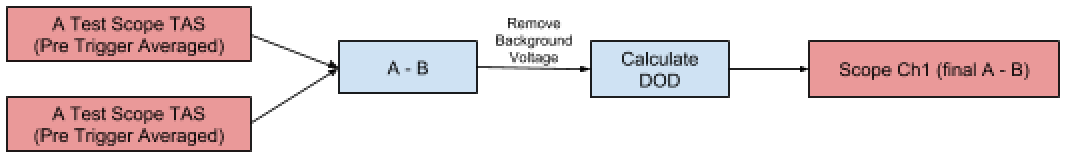
DAQ



Scope



A-B


A Test Scope TAS (Pre Trigger Averaged)
A Test Scope TAS (Pre Trigger Averaged)
A - B
Remove Background Voltage
Calculate DOD
Scope Ch1 (final A - B)