ATTOCUBE SYSTEMS AG

attoDRY Interface Documentation

Labview and DLL

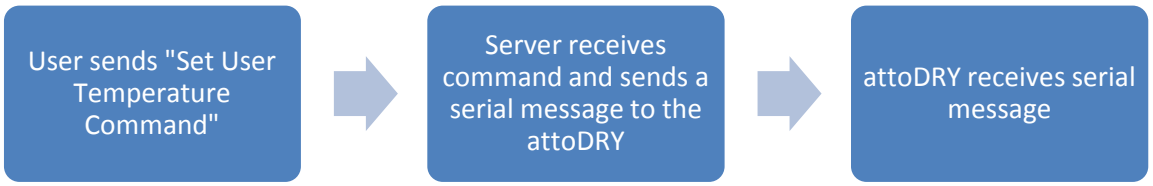
Contents

1. Introduction

The attoDRY Interface software is written in LabVIEW and designed to allow the user to remotely operate the attoDRY. There are 2 different formats, all based on the same source code:

- The attoDRY Interface Program. This option provides a Graphical User Interface for communicating with an attoDRY
- The attoDRY Interface DLL. This option allows the user to write their own programs in any language that can load DLLs written in C to interface with the attoDRY. This can be useful if the user wishes to write a test program, for example. The DLL can also be imported to LabVIEW (via the 'import-shared library' function in LabVIEW). This allows the user to write their own program in LabVIEW. Please note that the binary outputs of the LabVIEW VIs are replaced with integer outputs in the DLL.

The attoDRY interface is design to work asynchronously, that is, the program does not pause when the user sends a command. The user could be a person sitting at the computer, pressing buttons on the Graphical User Interface or, a computer program. When the user starts the program, a server is started that sits between the user and the attoDRY. It processes commands received from the user, and data from the attoDRY. The user never communicates directly with the attoDRY. For example, every time the user issues a command, it is added to the server's queue of commands to process. The server will convert the command into a serial message, which is passed on to the attoDRY. The attoDRY will respond with a confirmation when it receives a command. See Figure 1 for an example.



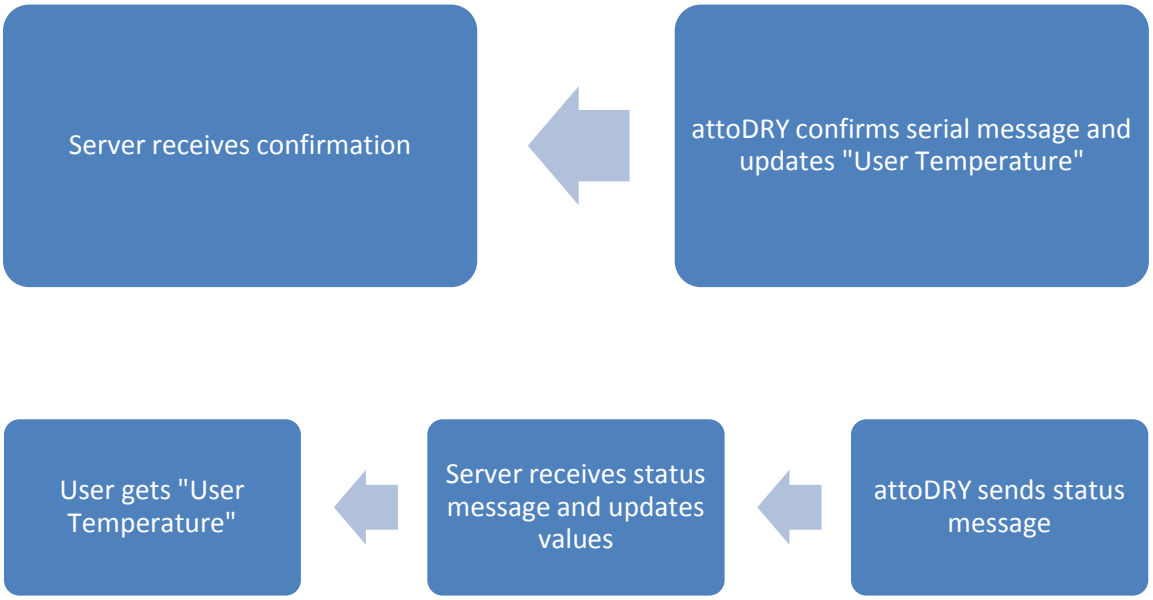


Figure 1 Interaction between the user, the server, and the attoDRY

Every second, the attoDRY sends a status message to the computer. This status message is received by the server running on the computer, and the values held by the server are updated. The user needs to request this data from the server. For some parameters, the values are not sent in the status message. The user needs to send a query to the attoDRY, to which attoDRY will respond with a confirmation, along with the data. This value will be updated in the server, and the user needs to get it from the server.

Once you have finished interfacing with the attoDRY, the “Disconnect” command needs to be run, followed by the “End” command. This will close the program completely.

a. Different Devices

The interface software supports more than one device. When you call the “Begin” function, you need to specify which device you are using. In the DLL, there is an Enumeration, in LabVIEW there is a Type Definition, and there are currently different Graphical User Interfaces for each device. For each device, there are some device specific functions. These cannot be run on the other devices without receiving an error.

b. Requirements

In order to use the interface software, the following must be installed:

- LabVIEW 2016 32 bit Runtime Engine
- LabVIEW VISA Runtime Engine. Use the latest version that is compatible with LabVIEW 2016 and your operating system.

2. Interface Program

The attoDRY Interface program comes tailored to each type of attoDRY. There are seven main sections to each program. They are discussed in more detail ion the following sections. A quick tool tip description of what each button does can be seen by hovering the mouse cursor over a button.

a. Connection Panel

To begin using the interface program, you must first connect to the attoDRY using this panel. Once the attoDRY is connected to the computer with a USB cable, select the COM Port used by the attoDRY by clicking the drop down box. Press 'Connect', the interface program will connect to the attoDRY and display a message saying that he program is waiting for the attoDRY to initialise. NOTE: THE ATTODRY WILL RESTART WHEN YOU PRESS CONNECT. MAKE SURE YOU ARE NOT RUNNING ANY EXPERIMENTS.

Once connected, other buttons will become available to use. Press the 'Disconnect' button to disconnect from the attoDRY.

b. Status Panel

The status panel shows the current value of the attoDRY parameters. This includes temperature, PID values, and error messages. The only button is the 'Lower Error', this lowers a raised error.

c. Logging Panel

This panel allows data to be logged to a text file. A new or existing text file should be specified by the 'Logging Path' box and the logging frequency should be chosen by selecting an option in the drop down 'Logging Frequency' box. By clicking on the 'overwrite/Append' button, the logging behaviour can be chosen. If overwrite is chosen and the logging file exists, it will be overwritten. If the append option is chosen and the logging file exists, data will be appended to the file but, no header will be appended.

d. Control Panel

This panel allows values to be set and actions to be started. The buttons behave as their counterparts on the touch screen behave. The 'Cancel' button is used to cancel an action or, to respond negatively to an action message. The 'Confirm' button is used to respond positively to an action message.

e. Heater Setting Panel

This panel allows values corresponding to the sample heater to be used. They are useful if a custom heater is used. The heater power is calculated using voltage and resistance; hence the resistance of the heater wires is required.

When setting a value, ensure that it is set by clicking on the corresponding 'Get' button. It can take a few seconds for the set and get commands to work.

f. Expert Mode Panel

This panel provides the same functionality as the 'Expert Mode' on the display, along with a couple of other features.

- A constant Sample Heater power can be set
- Sample temperature control can be enabled without changing any pressures.

g. Temperature Sensor Calibration Panel

This panel allows the sample temperature sensor calibration curve to be uploaded and downloaded. This section is useful if, a new sample temperature sensor is used.

The attoDRY expects a .crv calibration curve file. This is not a common format and so, a converter is provided to convert a .340 calibration file to .crv. When a .340 file is selected and the 'Convert' button is pressed, a pop up appears. The pop up requests confirmation from the user that the sensor type read from the .340 file is correct. If you are unsure, select you sensor type from the drop down box and press 'Apply default settings', the fields will be filled out. Press confirm to convert the file, the new file will be placed in the same folder as the original file.

A .crv file can then be uploaded. Select the file using the 'Sample Temperature Sensor Calibration Curve Path' box and press 'Upload'. The progress bar will show upload progress. Once uploaded, the curve should be downloaded to check that it matches the uploaded curve. Do this by specifying a save path in the 'Sample Temperature Sensor Calibration Curve Save Path' box and pressing download. While the curve is downloading, the indicator will be lit.

3. Description of the functions

The following sections describe the functions available for the user. They are shown in a LabVIEW format. Section e describes some DLL specific points.

a. LabVIEW Error inputs and outputs

Every LabVIEW VI has error input and output connectors. In the interest of saving space, they are described here. Section 3.a.xcvii covers error return values in the DLL.

error in - error in can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

status - status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

code - code is the error or warning code.

Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source - source describes the origin of the error or warning.

Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

error out - error out can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

status - status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

code - code is the error or warning code.

Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source - source describes the origin of the error or warning.

Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

a. attoDRY800 Specific Functions

i. attoDRY_Interface.lvlib:getBreakVac800Valve

ATTODRY800 ONLY. Gets the current status of the break-vacuum valve. True is opened, false is closed.

valve status The status of the valve. True if the valve is open, false if it is closed.

ii. attoDRY_Interface.lvlib:getPump800Valve.vi

ATTODRY800 ONLY. Gets the current status of the pump valve. True is opened, false is closed.

valve status The status of the valve. True if the valve is open, false if it is closed.

iii. attoDRY_Interface.lvlib:getSampleSpace800Valve.vi

ATTODRY800 ONLY. Gets the current status of the sample-space valve. True is opened, false is closed.

TF **valve status** The status of the valve. True if the valve is open, false if it is closed.

iv. attoDRY_Interface.lvlib:toggleSampleSpace800Valve.vi

ATTODRY800 ONLY. Toggles the sample-space valve. If it is closed, it will open and if it is open, it will close.

v. attoDRY_Interface.lvlib:togglePump800Valve.vi

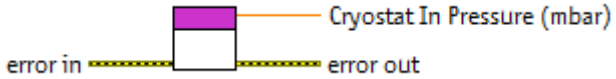
ATTODRY800 ONLY. Toggles the pump valve. If it is closed, it will open and if it is open, it will close.

vi. attoDRY_Interface.lvlib:toggleBreakVac800Valve.vi

ATTODRY800 ONLY. Toggles the break-vacuum valve. If it is closed, it will open and if it is open, it will close.

vii. attoDRY_Interface.lvlib:getPressure800.vi

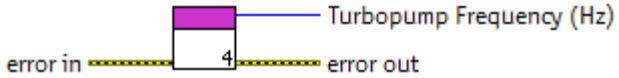
ATTODRY800 ONLY. Gets the pressure at the cryostat inlet, in mbar.



SGL **Pressure (mbar)** The pressure in mbar

viii. attoDRY_Interface.lvlib:getTurbopumpFrequ800.vi

ATTODRY800 ONLY. Gets the current frequency of the turbo pump.



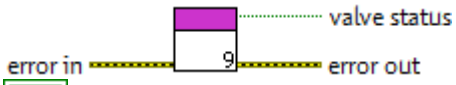
U16 **Turbo pump Frequency (Hz)** The frequency at which the turbo pump is spinning, in Hertz.

ix.

b. attoDRY1100 Specific Functions

x. attoDRY_Interface.lvlib:getHeliumValve.vi

ATTODRY1100 ONLY. Gets the current status of the helium valve. True is opened, false is closed.



valve status The status of the valve. True if the valve is open, false if it is closed.

xi. attoDRY_Interface.lvlib:getInnerVolumeValve.vi

ATTODRY1100 ONLY. Gets the current status of the inner volume valve. True is opened, false is closed.

valve status The status of the valve. True if the valve is open, false if it is closed.

xii. attoDRY_Interface.lvlib:getOuterVolumeValve.vi

ATTODRY1100 ONLY. Gets the current status of the outer volume valve. True is opened, false is closed.

valve status The status of the valve. True if the valve is open, false if it is closed.

xiii. attoDRY_Interface.lvlib:getPressure.vi

ATTODRY1100 ONLY. Gets the current pressure in the valve junction block, in mbar.

Pressure (mbar) The pressure in mbar

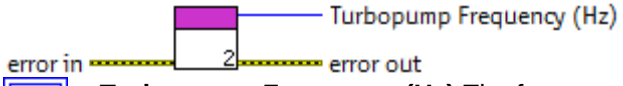
xiv. attoDRY_Interface.lvlib:getPumpValve.vi

ATTODRY1100 ONLY. Gets the current status of the pump valve. True is opened, false is closed.

valve status The status of the valve. True if the valve is open, false if it is closed.

xv. attoDRY_Interface.lvlib:getTurbopumpFrequency.vi

ATTODRY1100 ONLY. Gets the current frequency of the turbo pump.



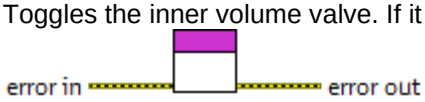
Turbo pump Frequency (Hz) The frequency at which the turbo pump is spinning, in Hertz.

xvi. attoDRY_Interface.lvlib:toggleHeliumValve.vi

ATTODRY1100 ONLY. Toggles the helium valve. If it is closed, it will open and if it is open, it will close.

xvii. attoDRY_Interface.lvlib:toggleInnerVolumeValve.vi

ATTODRY1100 ONLY.
Toggles the inner volume valve. If it is closed, it will open and if it is open, it will close.

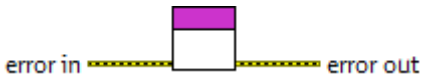


xviii. attoDRY_Interface.lvlib:toggleOuterVolumeValve.vi

ATTODRY1100 ONLY.
Toggles the outer volume valve. If it is closed, it will open and if it is open, it will close.

xix. attoDRY_Interface.lvlib:togglePumpValve.vi

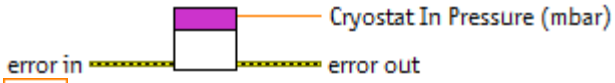
ATTODRY1100 ONLY. Toggles pump valve. If it is closed, it will open and if it is open, it will close.



c. attoDRY2100 Specific Functions

xx. attoDRY_Interface.lvlib:getCryostatInPressure.vi

ATTODRY2100 ONLY. Gets the pressure, in mbar, at the Cryostat Inlet.



SGL **Cryostat In Pressure (mbar)** The pressure at the Cryostat Inlet, in mbar

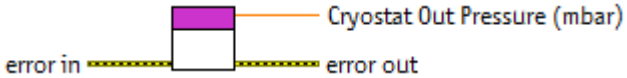
xxi. attoDRY_Interface.lvlib:getCryostatInValve.vi

ATTODRY2100 ONLY. Gets the current status of the Cryostat Inlet valve.

TF **valve status** The status of the valve. True is open, false is closed.

xxii. attoDRY_Interface.lvlib:getCryostatOutPressure.vi

ATTODRY2100 ONLY. Gets the current Cryostat Outlet pressure, in mbar.



SGL **Cryostat Out Pressure (mbar)** The pressure, in mbar, at the Cryostat Outlet

xxiii. attoDRY_Interface.lvlib:getCryostatOutValve.vi

ATTODRY2100 ONLY. Gets the current status of the Cryostat Outlet valve.

valve status The status of the valve. True is open, false is closed.

xxiv. attoDRY_Interface.lvlib:getDumpInValve.vi

ATTODRY2100 ONLY. Gets the current status of the Dump Inlet valve.

valve status The status of the valve. True is open, false is closed.

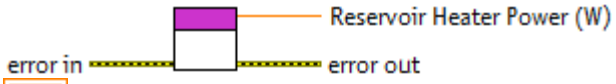
xxv. attoDRY_Interface.lvlib:getDumpOutValve.vi

ATTODRY2100 ONLY. Gets the current status of the Dump Outlet valve.

valve status The status of the valve. True is open, false is closed.

xxvi. attoDRY_Interface.lvlib:getReservoirHeaterPower.vi

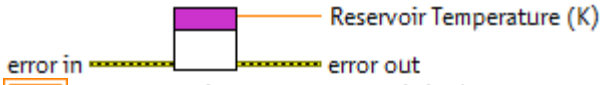
ATTODRY2100 ONLY. Gets the current power of the Liquid Helium Reservoir, in Watts.



Reservoir Heater Power (W) The current power, in Watts, being used by the Reservoir Heater

xxvii. attoDRY_Interface.lvlib:getReservoirTemperature.vi

ATTODRY2100 ONLY. Gets the current temperature of the Liquid Helium Reservoir, in Kelvin.



Reservoir Temperature (K) The current temperature of the Helium Reservoir, in Kelvin

xxviii. attoDRY_Interface.lvlib:toggleCryostatInValve.vi

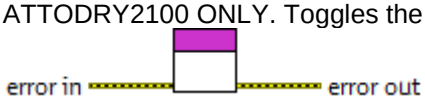
ATTODRY2100 ONLY. Toggles the Cryostat In valve. If it is closed, it will open and if it is open, it will close.

xxix. attoDRY_Interface.lvlib:toggleCryostatOutValve.vi

ATTODRY2100 ONLY. Toggles the Cryostat Outlet valve. If it is closed, it will open and if it is open, it will close.

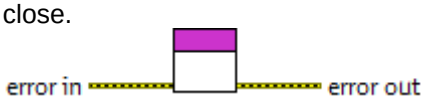
xxx. attoDRY_Interface.lvlib:toggleDumpInValve.vi

ATTODRY2100 ONLY. Toggles the Dump In valve. If it is closed, it will open and if it is open, it will close.



xxxi. attoDRY_Interface.lvlib:toggleDumpOutValve.vi

ATTODRY2100 ONLY. Toggles the Dump Outlet valve. If it is closed, it will open and if it is open, it will close.



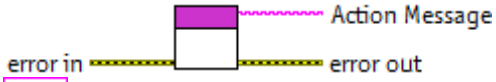
d. General Functions

xxxii. attoDRY_Interface.lvlib:Device.ctl

Use this Enumeration to configure the program/DLL for the specified attoDRY

xxxiii. attoDRY_Interface.lvlib:getActionMessage.vi

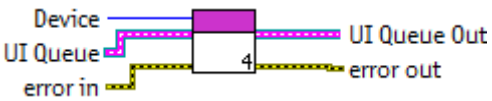
Gets the current action message. If an action is being performed, it will be shown here. It is similar to the pop ups on the display.




Action Message The current action message.

xxxiv. attoDRY_Interface.lvlib:begin.vi

Starts the server that communicates with the attoDRY and loads the software for the device specified by Device. This VI needs to be run before commands can be sent or received.




Device Configures the program or DLL for use with the specified attoDRY


UI Queue Queue for interfacing with a User Interface. When using from a DLL, enter NULL for this parameter


UI Queue Out Queue for interfacing with a User Interface. When using from a DLL, enter NULL for this parameter

xxxv. attoDRY_Interface.lvlib:Cancel.vi

Sends a 'Cancel' Command to the attoDRY. Use this when you want to cancel an action or respond negatively to a pop up.

xxxvi. attoDRY_Interface.lvlib:Confirm.vi

Sends a 'Confirm' command to the attoDRY. Use this when you want to respond positively to a pop up.

xxxvii. attoDRY_Interface.lvlib:Connect.vi

Connects to the attoDRY using the specified COM Port.

COM Port The COM Port that the attoDRY uses e.g. COM6

xxxviii. attoDRY_Interface.lvlib:Disconnect.vi

Disconnects from the attoDRY, if already connected. This should be run before the End function.

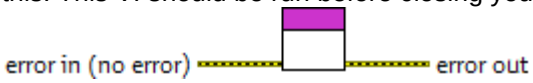
xxxix. attoDRY_Interface.lvlib:downloadSampleTemperatureSensorCalibrationCurve.vi

Starts the download of the <B>Sample Temperature Sensor Calibration Curve</B>. The curve will be saved to <B>Save Path</B>

Save Path The absolute file path for the file you wish to save the curve to.

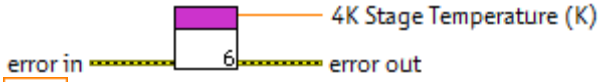
xl. attoDRY_Interface.lvlib:end.vi

Stops the server that is communicating with the attoDRY. The Disconnect command should be run before this. This VI should be run before closing your program.



xli. attoDRY_Interface.lvlib:get4KStageTemperature.vi

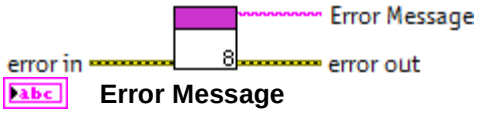
Returns the temperature of the 4 Kelvin Stage



4K Stage Temperature (K)

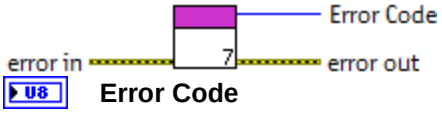
xlii. attoDRY_Interface.lvlib:getAttodryErrorMessage.vi

Returns the current error message.



xliii. attoDRY_Interface.lvlib:getAttodryErrorStatus.vi

Returns the current error code

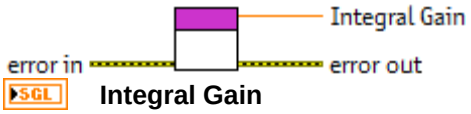


xliv. attoDRY_Interface.lvlib:getDerivativeGain.vi

Gets the Derivative gain. The gain retrieved depends on which heater is active:
- If no heaters are on or the sample heater is on, the **Sample Heater** gain is returned
- If the VTI heater is on and a sample temperature sensor is connected, the **VTI Heater** gain is returned
- If the VTI heater is on and no sample temperature sensor is connected, the **Exchange Heater** gain is returned

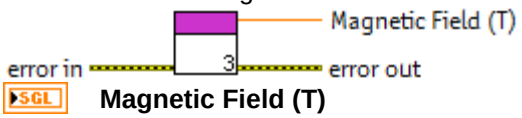
xlv. attoDRY_Interface.lvlib:getIntegralGain.vi

Gets the Integral gain. The gain retrieved depends on which heater is active:
- If no heaters are on or the sample heater is on, the **Sample Heater** gain is returned
- If the VTI heater is on and a sample temperature sensor is connected, the **VTI Heater** gain is returned
- If the VTI heater is on and no sample temperature sensor is connected, the **Exchange Heater** gain is returned



xlvi. attoDRY_Interface.lvlib:getMagneticField.vi

Gets the current magnetic field



xlvii. attoDRY_Interface.lvlib:getMagneticFieldSetPoint.vi

Gets the current magnetic field set point

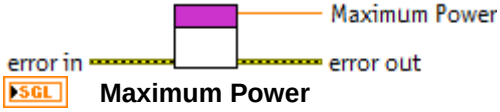
xlviii. attoDRY_Interface.lvlib:getProportionalGain.vi

Gets the Proportional gain. The gain retrieved depends on which heater is active:
- If no heaters are on or the sample heater is on, the **Sample Heater** gain is returned
- If the VTI heater is on and a sample temperature sensor is connected, the **VTI Heater** gain is returned
- If the VTI heater is on and no sample temperature sensor is connected, the **Exchange Heater** gain is returned

xlix. attoDRY_Interface.lvlib:getSampleHeaterMaximumPower.vi

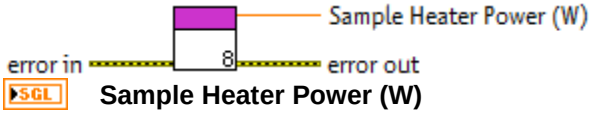
Gets the maximum power limit of the sample heater in Watts. This value, is the one stored in memory on the computer, not the one on the attoDRY. You should first use the appropriate **query VI** to request the value from the attoDRY.

The output power of the heater will not exceed this value. It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off.



l. attoDRY_Interface.lvlib:getSampleHeaterPower.vi

Gets the current Sample Heater power, in Watts



li. attoDRY_Interface.lvlib:getSampleHeaterResistance.vi

Gets the resistance of the sample heater in Ohms. This value, is the one stored in memory on the computer, not the one on the attoDRY. You should first use the appropriate **query VI** to request the value from the attoDRY.

This value, along with the heater wire resistance, is used in calculating the output power of the heater. It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off.

Power = Voltage^2/((HeaterResistance + WireResistance)^2) * HeaterResistance

lii. attoDRY_Interface.lvlib:getSampleHeaterWireResistance.vi

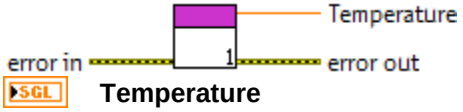
Gets the resistance of the sample heater wires in Ohms. This value, is the one stored in memory on the computer, not the one on the attoDRY. You should first use the appropriate **query VI** to request the value from the attoDRY.

This value, along with the heater resistance, is used in calculating the output power of the heater. It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off.

Power = Voltage^2/((HeaterResistance + WireResistance)^2) * HeaterResistance

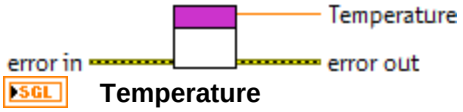
liii. attoDRY_Interface.lvlib:getSampleTemperature.vi

Gets the sample temperature in Kelvin. This value is updated whenever a status message is received from the attoDRY.



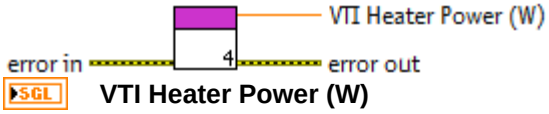
liv. attoDRY_Interface.lvlib:getUserTemperature.vi

Gets the user set point temperature, in Kelvin. This value is updated whenever a status message is received from the attoDRY.



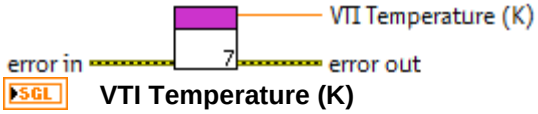
lv. attoDRY_Interface.lvlib:getVtiHeaterPower.vi

Returns the VTI Heater power, in Watts



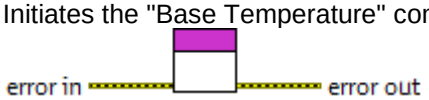
lvi. attoDRY_Interface.lvlib:getVtiTemperature.vi

Returns the temperature of the VTI



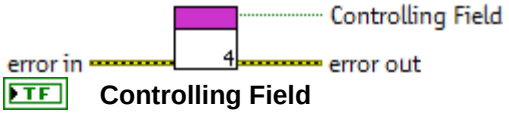
lvii. attoDRY_Interface.lvlib:goToBaseTemperature.vi

Initiates the "Base Temperature" command, as on the touch screen



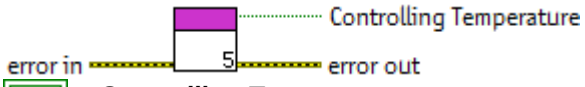
lviii. attoDRY_Interface.lvlib:isControllingField.vi

Returns 'True' if magnetic filed control is active. This is true when the magnetic field control icon on the touch screen is orange, and false when the icon is white.



lix. attoDRY_Interface.lvlib:isControllingTemperature.vi

Returns 'True' if temperature control is active. This is true when the temperature control icon on the touch screen is orange, and false when the icon is white.



Controlling Temperature

lx. attoDRY_Interface.lvlib:isDeviceInitialised.vi

Checks to see if the attoDRY has initialised. Use this VI after you have connected and before sending any commands or getting any data from the attoDRY

Initialised If the device is initialised, True will be returned. If it is not initialised, False is returned.

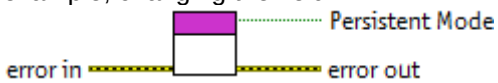
lxi. attoDRY_Interface.lvlib:isGoingToBaseTemperature.vi

Returns 'True' if the base temperature process is active. This is true when the base temperature button on the touch screen is orange, and false when the button is white.

Exchanging Sample

lxii. attoDRY_Interface.lvlib:isPersistentModeSet.vi

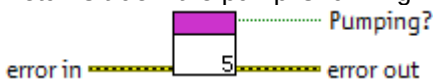
Checks to see if persistent mode is set for the magnet. Note: this shows if persistent mode is set, it does not show if the persistent switch heater is on. The heater may be on during persistent mode when, for example, changing the field.



Persistent Mode

lxiii. attoDRY_Interface.lvlib:isPumping.vi

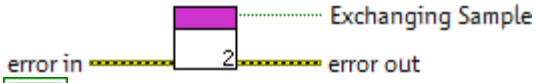
Returns true if the pump is running



Pumping?

lxiv. attoDRY_Interface.lvlib:isSampleExchangeInProgress.vi

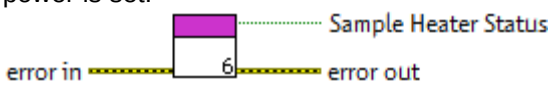
Returns 'True' if the sample exchange process is active. This is true when the sample exchange button on the touch screen is orange, and false when the button is white.



Exchanging Sample

lxv. attoDRY_Interface.lvlib:isSampleHeaterOn.vi

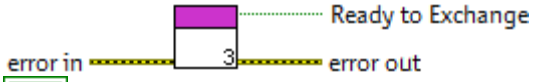
Checks to see if the sample heater is on. 'On' is defined as PID control is active or a constant heater power is set.



Sample Heater Status The status of the sample heater. On returns 'True', off returns 'False'.

lxvi. attoDRY_Interface.lvlib:isSampleReadyToExchange.vi

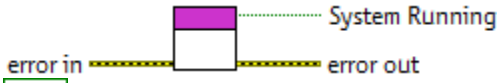
This will return true when the sample stick is ready to be removed or inserted.



Ready to Exchange

lxvii. attoDRY_Interface.lvlib:isSystemRunning.vi

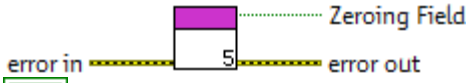
Checks to see if the system is running, that is, if the compressor is running etc



System Running

lxviii. attoDRY_Interface.lvlib:isZeroingField.vi

Returns 'True' if the "Zero Field" process is active. This is true when the "Zero Field" button on the touch screen is orange, and false when the button is white.



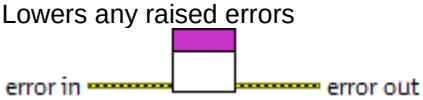
Zeroing Field

lxix. attoDRY_Interface.lvlib:Logging Frequency.ctl

The frequency at which data will be logged. This is an enumeration.

lxx. attoDRY_Interface.lvlib:lowerError.vi

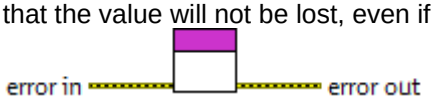
Lowers any raised errors



lxxi. attoDRY_Interface.lvlib:querySampleHeaterMaximumPower.vi

Requests the maximum power limit of the sample heater in Watts from the attoDRY. After running this command, use the appropriate **get VI** to get the value stored on the computer.

The output power of the heater will not exceed this value. It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off.

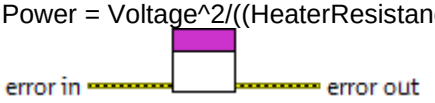


lxxii. attoDRY_Interface.lvlib:querySampleHeaterResistance.vi

Requests the resistance of the sample heater in Ohms from the attoDRY. After running this command, use the appropriate **get VI** to get the value stored on the computer.

This value, along with the heater wire resistance, is used in calculating the output power of the heater. It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off.

Power = Voltage^2/((HeaterResistance + WireResistance)^2) * HeaterResistance

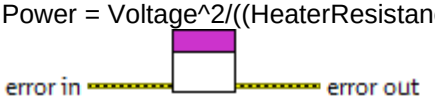


lxxiii. attoDRY_Interface.lvlib:querySampleHeaterWireResistance.vi

Requests the resistance of the sample wires heater in Ohms from the attoDRY. After running this command, use the appropriate **get VI** to get the value stored on the computer.

This value, along with the heater resistance, is used in calculating the output power of the heater. It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off.

Power = Voltage^2/((HeaterResistance + WireResistance)^2) * HeaterResistance



lxxiv. attoDRY_Interface.lvlib:setDerivativeGain.vi

Sets the Derivative gain. The controller that is updated depends on which heater is active:
- If no heaters are on or the sample heater is on, the **Sample Heater** gain is set
- If the VTI heater is on and a sample temperature sensor is connected, the **VTI Heater** gain is set
- If the VTI heater is on and no sample temperature sensor is connected, the **Exchange Heater** gain is set

Derivative Gain

lxxv. attoDRY_Interface.lvlib:setIntegralGain.vi

Sets the Integral gain. The controller that is updated depends on which heater is active:
- If no heaters are on or the sample heater is on, the **Sample Heater** gain is set
- If the VTI heater is on and a sample temperature sensor is connected, the **VTI Heater** gain is set
- If the VTI heater is on and no sample temperature sensor is connected, the **Exchange Heater** gain is set

Integral Gain

lxxvi. attoDRY_Interface.lvlib:setProportionalGain.vi

Sets the Proportional gain. The controller that is updated depends on which heater is active:
- If no heaters are on or the sample heater is on, the **Sample Heater** gain is set
- If the VTI heater is on and a sample temperature sensor is connected, the **VTI Heater** gain is set
- If the VTI heater is on and no sample temperature sensor is connected, the **Exchange Heater** gain is set

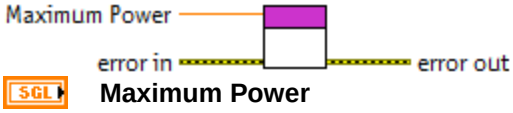
Proportional Gain

lxxvii. attoDRY_Interface.lvlib:setSampleHeaterMaximumPower.vi

Sets the maximum power limit of the sample heater in Watts. After running this command, use the appropriate **request** and **get** VIs to check the value was stored on the attoDRY.

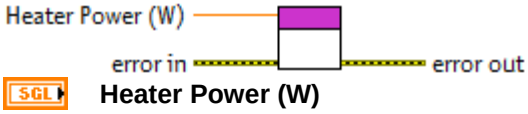
The output power of the heater will not exceed this value.

It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off. Note: the non-volatile memory has a specified life of 100,000 write/erase cycles, so you may need to be careful about how often you set this value.



lxxviii. attoDRY_Interface.lvlib:setSampleHeaterPower.vi

Sets the sample heater value to the specified value



lxxix. attoDRY_Interface.lvlib:setSampleHeaterResistance.vi

Sets the resistance of the sample heater in Ohms. After running this command, use the appropriate **request** and **get** VIs to check the value was stored on the attoDRY.

This value, along with the heater wire resistance, is used in calculating the output power of the heater. It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off.

Power = Voltage^2/((HeaterResistance + WireResistance)^2) * HeaterResistance

It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off. Note: the non-volatile memory has a specified life of 100,000 write/erase cycles, so you may need to be careful about how often you set this value.

lxxx. attoDRY_Interface.lvlib:setSampleHeaterWireResistance.vi

Sets the resistance of the sample heater wires in Ohms. After running this command, use the appropriate **request** and **get** VIs to check the value was stored on the attoDRY.

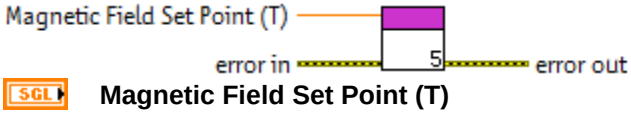
This value, along with the heater resistance, is used in calculating the output power of the heater. It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off.

Power = Voltage^2/((HeaterResistance + WireResistance)^2) * HeaterResistance

It is stored in non-volatile memory, this means that the value will not be lost, even if the attoDRY is turned off. Note: the non-volatile memory has a specified life of 100,000 write/erase cycles, so you may need to be careful about how often you set this value.

lxxxi. attoDRY_Interface.lvlib:setUserMagneticField.vi

Sets the user magntic field. This is used as the set point when field control is active



lxxxii. attoDRY_Interface.lvlib:setUserTemperature.vi

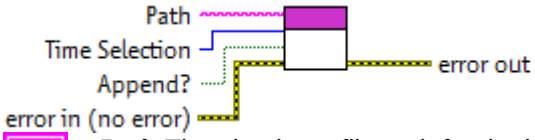
Sets the user temperature. This is the temperature used when temperature control is enabled.

Temperature

lxxxiii. attoDRY_Interface.lvlib:startLogging.vi

Starts logging data to the file specifed by **Path**.

If the file does not exist, it will be created.



Path The absoloute file path for the log file

Time Selection An enum that specifies how often to log data.

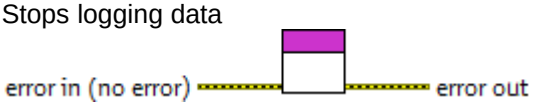
Append? Specifies whether to overwrite the file, if it exists, or to append. If True, the data will be appended to the end of the file (without a header). If False, the file will be overwritten.

lxxxiv. attoDRY_Interface.lvlib:startSampleExchange.vi

Starts the sample exchange procedure

lxxxv. attoDRY_Interface.lvlib:stopLogging.vi

Stops logging data



lxxxvi. attoDRY_Interface.lvlib:sweepFieldToZero.vi

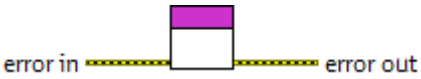
Initiates the "Zero Field" command, as on the touch screen

lxxxvii. attoDRY_Interface.lvlib:toggleFullTemperatureControl.vi

Toggles temperature control, just as the thermometer icon on the touch screen.

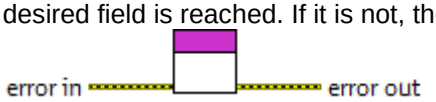
lxxxviii. attoDRY_Interface.lvlib:toggleMagneticFieldControl.vi

Toggle magnetic field control, just as the magnet icon on the touch screen



lxxxix. attoDRY_Interface.lvlib:togglePersistentMode.vi

Toggles persistant mode for magnet control. If it is enabled, the switch heater will be turned off once the desired field is reached. If it is not, the switch heater will be left on.



xc. attoDRY_Interface.lvlib:togglePump.vi

Starts and stops the pump. If the pump is running, it will stop it. If the pump is not running, it will be started.

xci. attoDRY_Interface.lvlib:toggleSampleTemperatureControl.vi

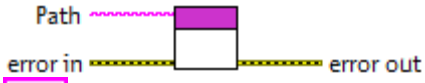
This command only toggles the sample temperature controller. It does not pump the volumes etc. Use **toggleFullTemperatureControl.vi** for behaviour like the temperature control icon on the touch screen.

xcii. attoDRY_Interface.lvlib:toggleStartUpShutdown.vi

Toggles the start up/shutdown procedure. If the attoDRY is started up, the shut down procedure will be run and vice versa

xciii. attoDRY_Interface.lvlib:uploadSampleTemperatureCalibrationCurve.vi

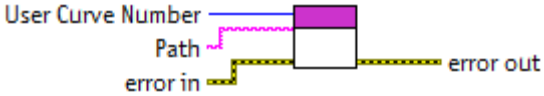
Starts the upload of a **.crv calibration curve file** to the **sample temperature sensor**



Path

xciv. attoDRY_Interface.lvlib:uploadTemperatureCalibrationCurve.vi

Starts the upload of a **.crv calibration curve file** to the specified **User Curve Number** on the temperature monitor. Use a curve number of 1 to 8, inclusive



User Curve Number

Path

e. Shared Library Specific Information

The shared library provides the same functionality as the LabVIEW Vis but, with the ability to use them in programs written in other languages. There are a few differences the user needs to be aware of.

xcv. How to retrieve a value

The definition for retrieving the Sample Temperature is

```
int32_t __cdecl GetSampleTemperature(float *Temperature);
```

The function takes a pointer to a float and returns a signed 32 bit integer. The float to which the given pointer points to will be updated with the Sample Temperature stored in the server. For example,

```
// initialise everything and connect before doing this

float sampleTemp; // a variable for storing user temperature

// get the sample temperature.
e = GetSampleTemperature(&sampleTemp);

if (sampleTemp > 4){
        //do something
}

//disconnect and end
```

Retrieving a string requires a little more information. The function prototype for retrieving the error message is:

```
int32_t __cdecl GetAttodryErrorMessage(char ErrorMessage[], int32_t LNge);
```

It takes a character array and a signed 32 bit integer, and returns a signed 32 bit integer. The LabVIEW server, needs to know the size of the character buffer you provide so that it does not write data past the end of the array. Use this function as follows:

```
// initialise everything and connect before doing this

const int32_t errorMsgLength = 500; // maximum length of the array for
storing the error message.
                                    //When dealing with strings, you must
give labview the maximum length
char errorMsg[errorMsgLength];      // array for storing error message

// get the error message.
GetAttodryErrorMessage(errorMsg, errorMsgLength);

//disconnect and end
```

If the error message is longer than the specified buffer length, it will simply truncate the message.

xcvi. Header Files

Header files (.h files) define the functions and constants available for use in their corresponding .c or .cpp file. The attoDRY Interface header file declares the functions and Enumerations available for use in the shared library and, it must be included in your source code. Included with the DLL, are some other LabVIEW header files. These also need to be included.

xcvii. Error Codes

The way error codes are handled in the Shared Library is different compared to LabVIEW. Section 3.a covers error codes in LabVIEW. Each function returns a signed 32 bit integer, this is the equivalent of the code parameter in a LabVIEW error cluster. How you handle these errors, is up to the person programming. A complete list of error codes can be found here http://zone.ni.com/reference/en-XX/help/371361J-01/lverror/misc_lv_error_codes/ .

xcviii. Binary Outputs

Please note that the binary outputs of the LabVIEW VIs are replaced with integer outputs in the DLL.