daqINSIGHT™ Primer

Version 1.0 - September 15th, 2024

Martin Hebel
SelmaWare Solutions, LLC

Contents

About This Primer

daqINSIGHT is an extremely versatile software package for bringing in serial ASCII data for plotting, logging, metering and, if desired, control of serial devices such as microcontrollers.

The help files are very extensive but can be overwhelming. This Quick Start Guide discusses many key aspects of daqINSIGHT while keeping to the basics of what should be understood to start using the software for your purposes.

While it discusses and provides examples of key concepts, the help references provides much more information on the in-depth details where needed. With an understanding of the material of what is in this guide, it will probably be much that is needed to be understood for the vast majority of users in both using serial data and designing your own GUI's for data.

For this guide, as in the help-files, we use the Arduino and Arduino code where it is necessary to show some microcontroller code.

Overview Of daqINSIGHT

daqINSIGHT is an application for accepting ASCII serial data in typically standard defined formats in being used to analog plotting, digital plotting, metering of data and logging the data. daqINSIGHT accepts standard string formats which defines if data is analog, digital, message or instructions.

While we tend to focus on data coming from serial devices, such as microcontrollers where the strings formats can be programmatically defined, there are also common devices that send serial that cannot be programmed. Through the Data Formatter window, incoming data be modified or parsed to pull access what is necessary to use it with daqINSIGHT.

A key principle with daqINSIGHT is that it operates off single lines of data, either coming in serially or using the daqINSIGHT scripting code, for performing tasks, such as using it as analog, digital, message or instructions. All lines of serial data must end with a Carriage Return (CR, \r or ASCII 13) a New Line (NL, \n or ASCII 10).

As strings of ASCII data arrive, upon receiving a CR or NL, the data is placed in the bottom of the processing queue. These lines of data will be processed top to bottom performing various operations. It is possible to force data to be placed on top to queue is that is processed next using the **!queAddTop** instruction.

With daqINSIGHT, you can quickly design your own interfaces to use incoming data or for other purposes. daqINSIGHT allows rapid placement of controls, changing control properties, updating control values and writing code for operations a control needs to perform, such as a button click.

Note, most all values in daqINSIGHT are of the variant variety in that they can be used as text or as value in math and such. Quotation marks for strings in daqINSIGHT are NOT used – which sometimes is an issue in flexibility. daqINSIGHT is very comma-delimited in its operations and use of comma in strings is not recommended except where specified, nor are semi-colons as they are used to separate instructions.

Interfaces

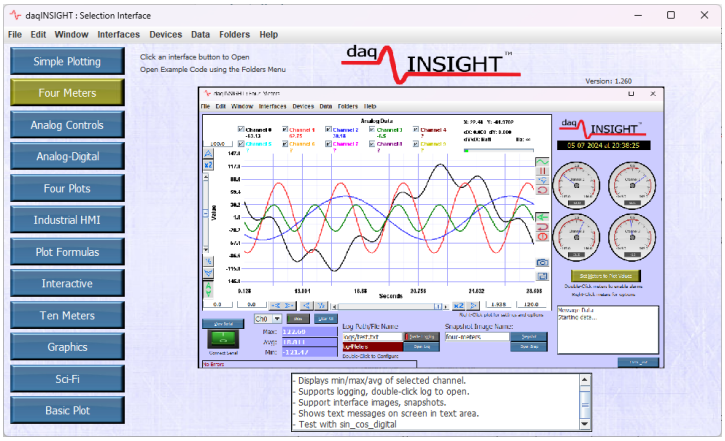
Interface is the term used to refer to the windows through which the user may monitor or interact with daqINSIGHT. There are 2 main Interface types:

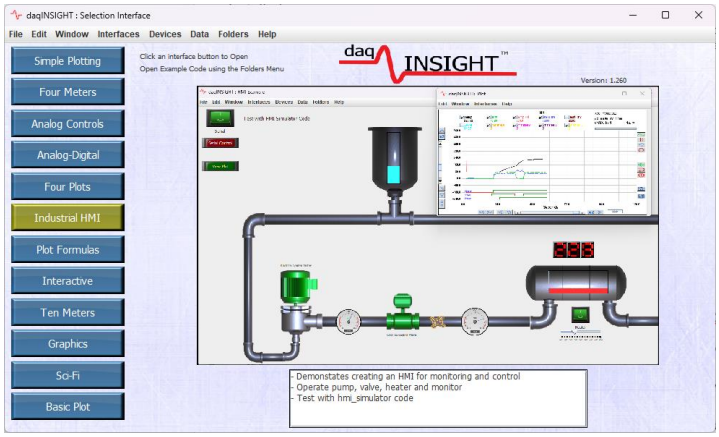
Main Interface – This is the main window of daqINSIGHT.

Child Interfaces – These are one or more windows that can be created in addition to Main Interface.

Installed

There are many interfaces that have been pre-designed for daqINSIGHT and are installed. The selection interface allows viewing a sample image of the interface, notes on use and example code, and the ability to load that interface by clicking a selection.





The Selection Interface may be returned to through the **File → Open Selection Interface**. daqINSIGHT also installs with some example code for the Arduino and Parallax Propeller 2 to test the various interfaces using the **Folders→Open Micro Code Folder**. Simply load the appropriate example code to your microcontroller and connect serially through an interface at the proper baud rate to test.

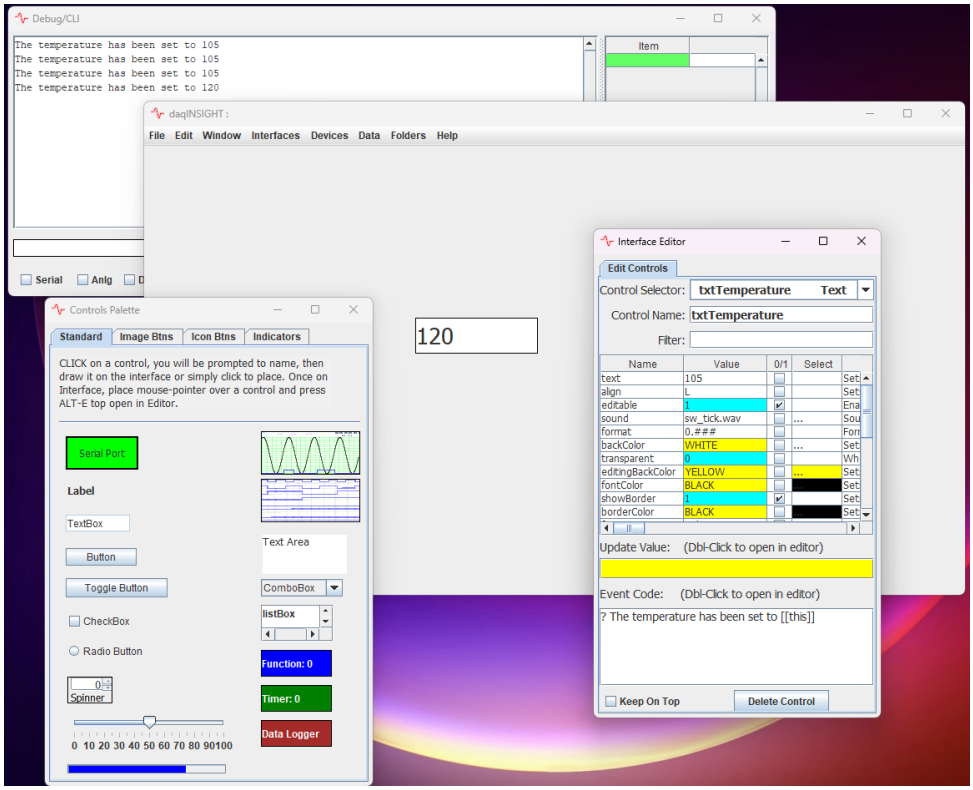
If an installed interface meets your needs, simply do what is necessary to format data from your own application into a suitable string for daqINSIGHT and you are ready to go.

Beyond being examples GUI for ready use, they are also great examples when it comes time to design your own GUI or HMI by looking at the configuration of the controls as example of for your own use.

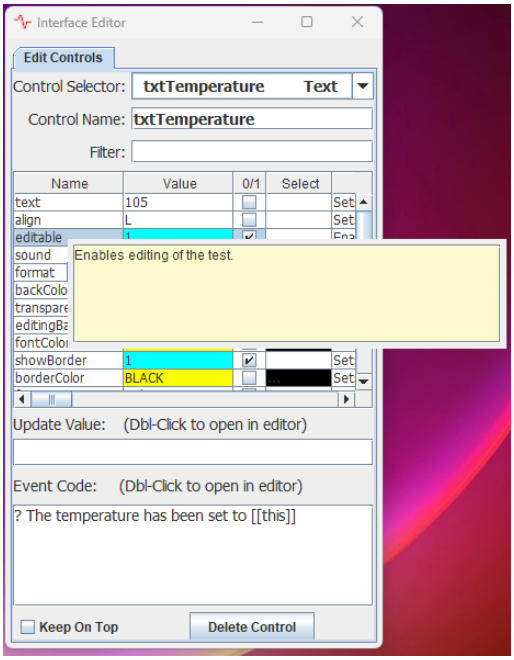
Designing

daqINSIGHT allows rapid development of interface for your specialized needs. Starting your own interface or modifying another is a simple task to get started.

- Use **File→New Interface** to start from a clean slate if desired.
- Press **F9** to enter Edit Mode.
- Press **F2** to open the Controls Palette select a control to use.
- Click the desired control.
- Draw it on the Interface.
- Answer and questions daqINSIGHT may ask, such as a name for the control, one word, typically with a prefix of the control type such as txtTemperaure, metPower, sldDrive, etc.
- Set and properties to change the look, feel or settings of the control. This window will open automatically but it can be opened manually by placing the mouse cursor over the control and pressing **F3**.
- If you'd like it to automatically update its value from data, set the Update Value to a value, such as **[ain0]** for analog data 0.
- If the control will perform some action, such as button being clicked, write daqINSIGHT scripting code for it in the Event Code.
- Once done making changes, press **F9** to exit Edit Mode and you are ready to test.
- For the example shown, open the **Debug/CLI Window** to see results (**F4**).



Note that selecting and then right-clicking a property will provide a help box for it (the same help that is to the right of the property). For Boolean values (0 or 1), highlighted in Blue, the check box may be used to toggle the state. For colors (yellow) and images, the … in Select column will being up additional windows.



There's a bit of understanding needed for the above steps, so please keep reading as we cover key concepts.

Debug/CLI Window

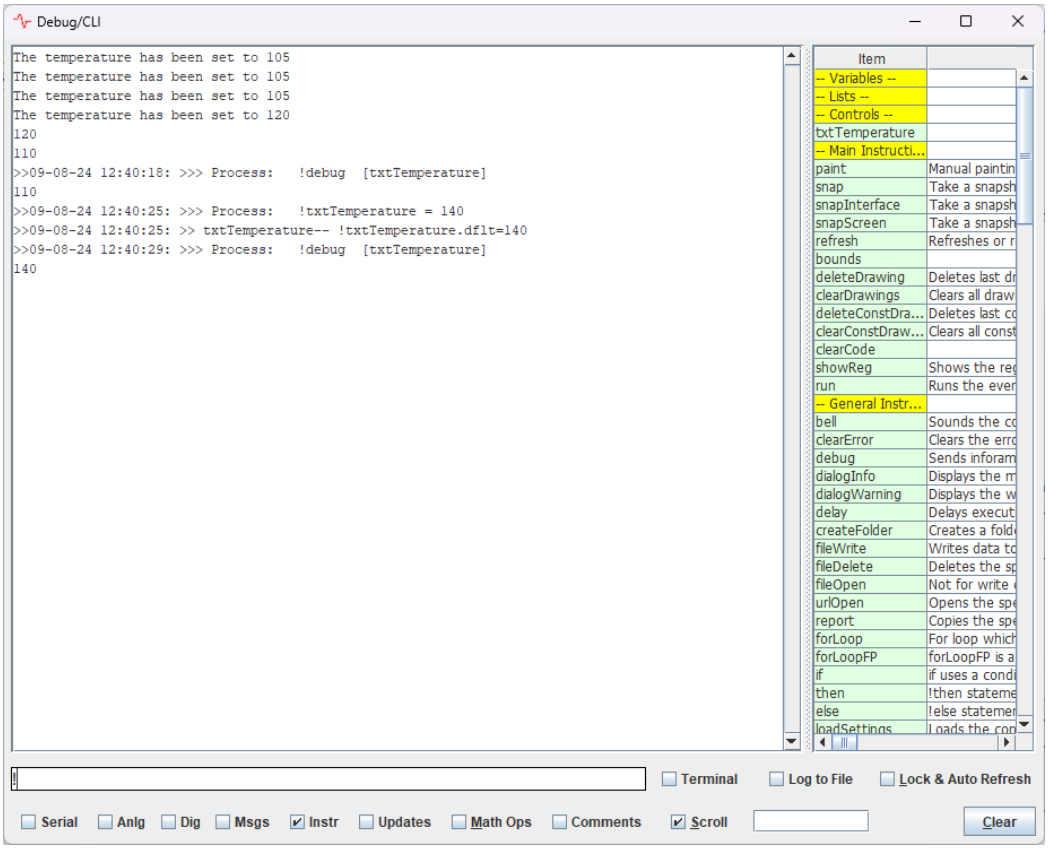
The **Debug/CLI Window** can be a great tool for testing and monitoring. Press **F4** to open the window. CLI means Command Line Interface, which is a means to perform all operations of daqINSIGHT through text instead of using various windows or using serial data for data string.

In this window, an errors encountered by daqINSIGHT will be shown, so while testing an interface be sure to open to monitor it. Various operations being performed can be using the check boxes at the bottom and it allows filtering of data shown to monitor operation.

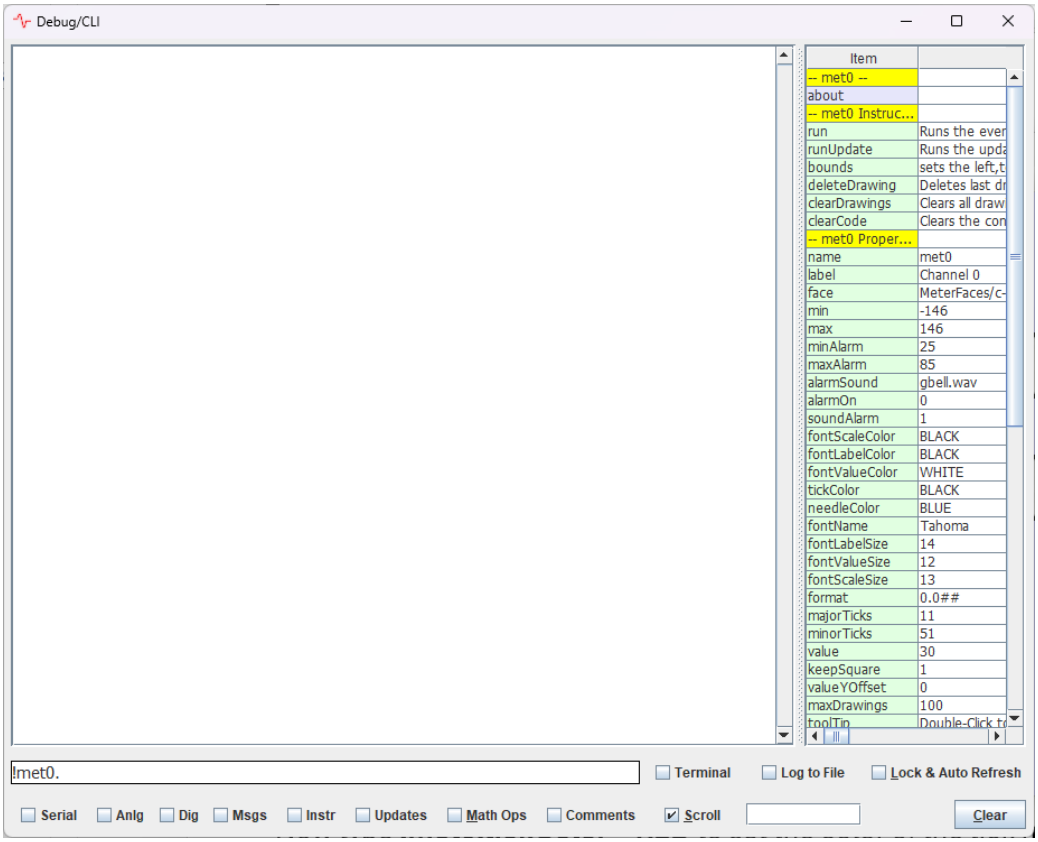
Using the Entry Text box at the bottom, daqINSIGHT instruction and properties may be tested for debugging or monitoring values. This window uses contextual help to access variables, controls, instructions, properties and values.

For example, from the Selection Interface, open Four Meters and open the Debug/CLI.

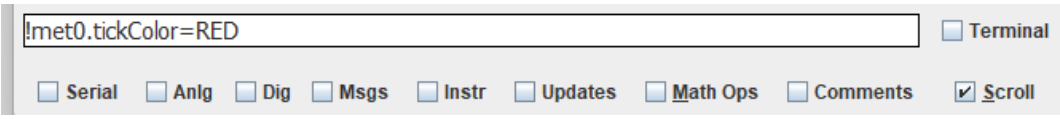
Type **!**, which indicates an instruction will be performed. In the listing all control that can be access will be shown as well as various instructions that can be performed and main properties that can be changed.



Type **!met** to filter the list down to controls with ‘met’ in them. Type CTRL-SPACE to accept the first choice, then type dot (period) which will show all instructions and properties which can be accessed.

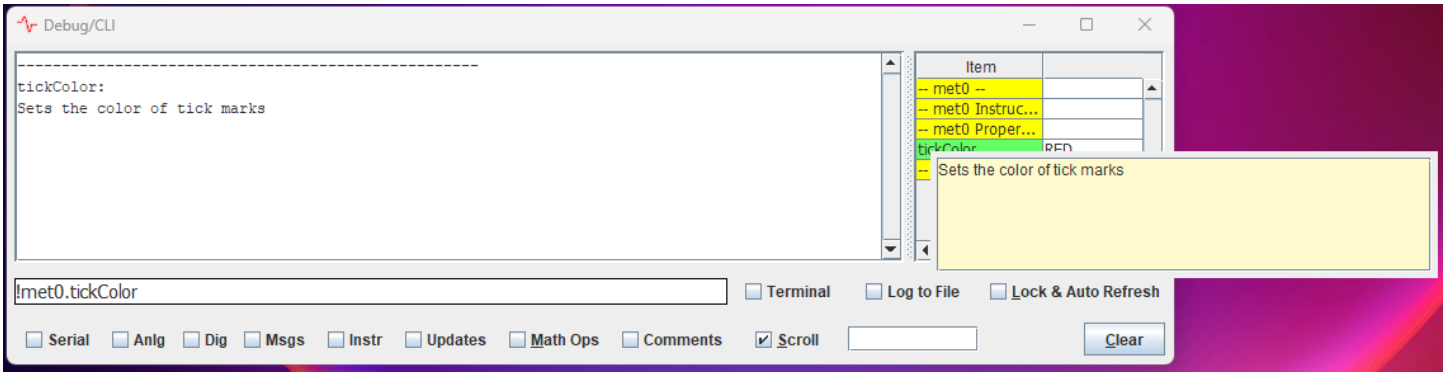


Now type **!met0.tickColor = RED** to set the color of the tick marks on met0. View the interface and confirm.



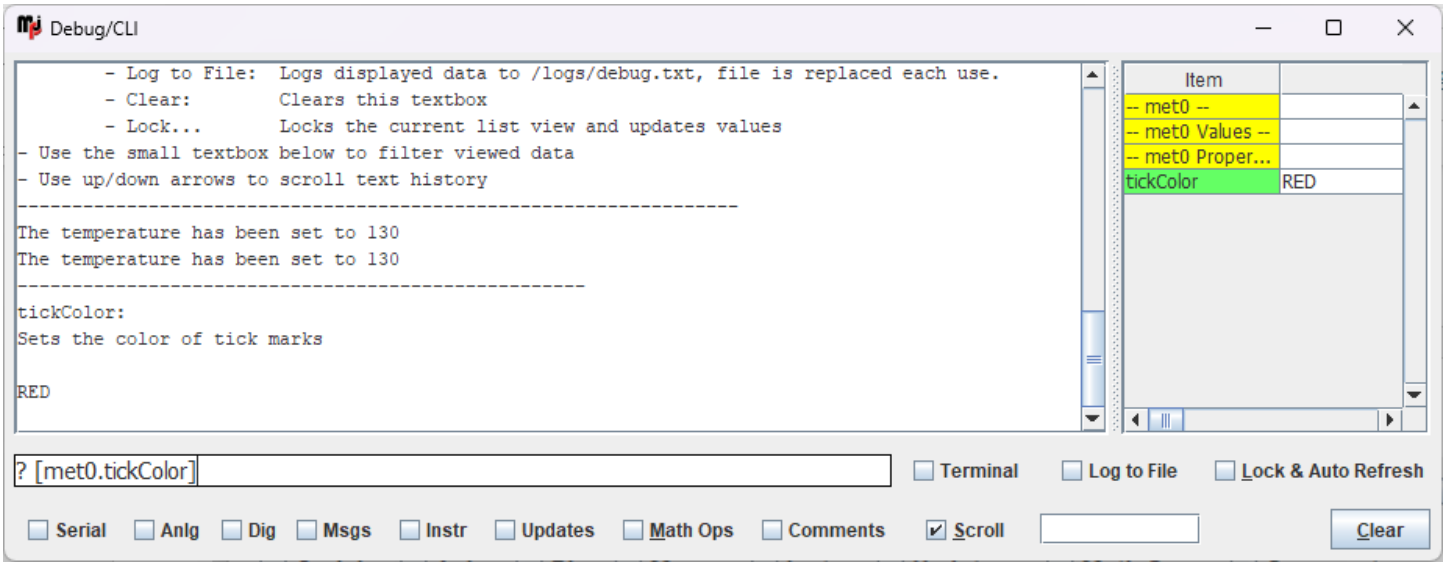
Note, that when something is in the list, it may be double-click to insert it into your text.

Also, selecting and right-clicking an item in the list will show help for it, or from the text entry, hit F1 once typed.



In a similar fashion, values and properties may be accessed. By enclosing in brackets, [ ], the values used in code, such as printing it to the Debug/CLI using ?

? [met0.tickColor]



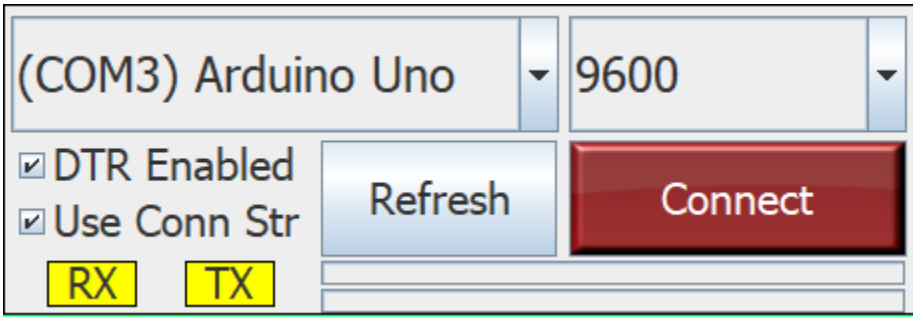
The Lock & Auto Refresh will continually update the values shown in the item list for monitoring.

We will cover more things you can test in the later discussion such as Math operations and Variables, but the Debug/CLI is a great tool to monitor parameters, to test code, and to perform operations using text.

Serial Data

Serial Control

A serial control is used to connect to the serial COM ports and the device.



DTR Enabled allow enabling the DTR line which often causes a controller reset on connect.

Use Conn Str issues a string upon connection, by default it is **!plot.reset** to reset the commonly named plot on the interface.

Multiple Serial Controls may be used. Please see the reference files for issues involved,

While not required for all uses, the main way of bringing data into daqINSIGHT is through the serial ports using the Serial Control. daqINSIGHT has standard string types it uses to identify and correctly use the data:

- Analog Strings
- Digital Strings
- Message Strings
- Instructions

Note that all these strings can be from the serial port, through event code in daqINSIGHT or entered in the Debug/CLI Window for testing or operation.

The **Data Formatter Window** (Under Data menu) may be used in instances where you cannot format the ASCII data to meet daqINSIGHT's expectations. It may allow you to process and use non-standard daqINSIGHT strings.

No matter the format, all strings MUST end with a carriage return or new line character (ASCII 13 or 10).

Analog Data

Analog data is simply a series of comma separated values beginning with a numeric values, such as:

10,20,30

This data string will be processed with and the first value will be placed into the value of **[ain0],** the second into **[ain1]**, third into **[ain2]** and so on. Up to 100 values may be sent in a single string. These values may be used for plotting, update values for control, or in any daqINSIGHT code.

Only the first the value needs to be numeric; the following will successfully parse the data into **[ain0]** to **[ain2]**, though of course only [ain0] can be used as numeric data. The values of **[ain1]** and **[ain2]** could be assigned to be data values for labels for example.

10,Pump On, Heater Off

Analog Plots are configured by default to plot [ain0] to [ain9], though this may be modified in the plot's settings window (right-click the plot for settings).

With the Arduino, data can be formatted and sent with code such as:

```
Serial.print(temperature);
Serial.print(",");
Serial.print(pressure);
Serial.print(",");
Serial.println(flow);  // Note println for end of string
```

Digital Strings

Digital Data begins with % and is followed by sequential 1's and 0's.

%1010

The data, from RIGHT (LSB) to LEFT (MSB) is parsed as **[din0]** to **[din3]**. Up to 100 digital values can be sent, **[din99]**. These values may be used for plots, Update Values for controls, or anywhere in daqINSIGHT code.

The digital plot or the analog plot when enabled for digital data will by default use **[din0]** to **[din9]** for plotting 10 channels, though these may be modified using the plot's settings.

Typical Arduino code to send the digital data is:

```
Serial.print("%");
Serial.print(pumpState);
Serial.print(heaterState);
Serial.println(mixerState); // end of string
```

Message Strings

Message Strings are those not being identified as other string types – analog, digital, or instructions (starting with ! @ or ?).

Hello World, I'm using daqINSIGHT!

The string will be listed in the Messages Window. It can be accessed in its entirety as **[messageString]**, or comma parse into pieces and accessed as **[str0]** to **[str99]**. These values can be used as Update Values for controls or anywhere in daqINSIGHT code.

Instructions

Instructions perform some operation on daqINSIGHT. They start with !. Note some drawing instruction types start with @, and that ? is shorthand to send data to the Debug/CLI Window.

Instructions can be used to set some property of daqINSIGHT, such as:
!met0.tickColor=GREEN

Or to have an interface or control perform some operation, such a clearing a text area:
!areaMessages.clear

*Both the **Debug/CLI** and the **Code Editor** use contextual help to correctly identify control, variables, lists, properties and operation instructions. Select and right-click on an instruction in the list for pop-up help, or type an instruction and press F1 for listed help. Use CTRL-SPACE to complete an instruction being written with the one on the top of the list.*

Note that instructions can come from the Debug/CLI Window, Event Code of daqINSIGHT, from a text file, or from the serial port sent by your controller. An daqINSIGHT Interface file (.daqINSIGHTi) is simply a serial of instructions to construct and configure and interface. An interface could be constructed totally from serial data if desired.

For example, a virtual LED on the interface (named ledHeaterOn) can be controlled on and off with string sent by your controller:

!ledHeaterOn=0

!ledHeaterOff=1

Sending Data to the Controller

Controls values and properties may be read by the microcontroller as serial ASCII data. Once means is to use daqINSIGHT code to send a value, such as the value of a slider for the controller to accept and use:
!send [sldDrive]

Also, in a more controlled fashion, the controller could request data from daqINSIGHT with a serial string and accept it:

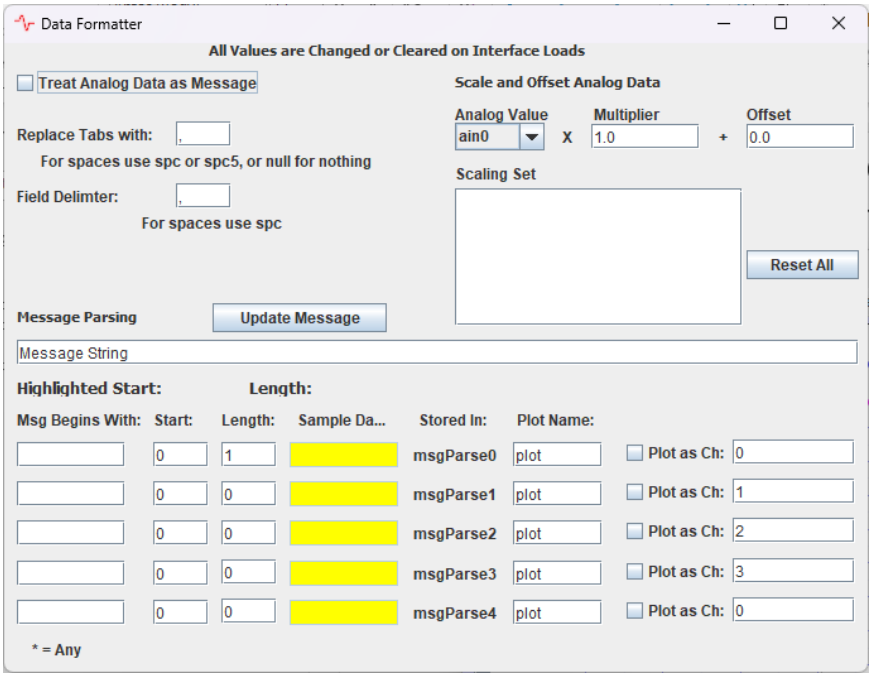
```
Serial.println("!read [sldDrive]");  // Request a value from daqINSIGHT
drive = Serial.parseInt();   // Accept and store the returned value
```

The installed interactive control Arduino sketch has example code to read multiple daqINSIGHT values at once and parse them into different values – Please see **Folders→Micro Code Folder**

Data Formatter Window

The Data Formatter Window allows special parsing of strings by a delimiter or defining the position in a string. It also allows incoming analog data to be scaled prior to processing.

With message parsing, data can be accessed as **[msgParse0]** to **[msgParse4]** or automatically plotted by defining the plot name and channel.



Controls

daqINSIGHT has a large assortment of controls that can be placed on interfaces for monitoring and control. For those supporting images, unique images can be assigned beyond what is distributed with daqINSIGHT.

Creating

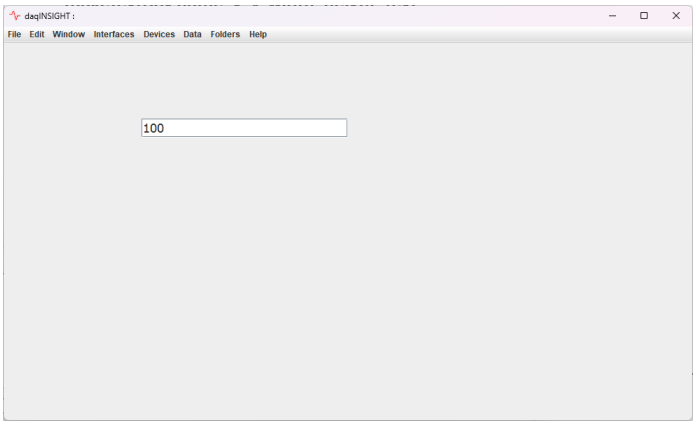
The easiest method to create your controls is to:

- Enter Edit Mode (F9).
- Open the Controls Palette (F2).
- Click a Control and place it on the Interface.
- Set its Properties, Update Value and Code as desired (F3).
- Exit Edit Mode (F9).

But control and entire interface can be made through instructions from the Debug/CLI or from the microcontroller serially, such as:

!makeTextBox name, x, y, width, height, text
Interfaces are 0,0 to 100, 100 from upper left to lower right.

!makeTextBox txtValue,20,20,30,5,100



Once created, the control can have its properties set, such as from the Debug/CLI or from the controller:
!txtValue=50

Properties

Properties of an interface or control consist of appearance, media and operational aspects such as what delimiter to use for data. There are hundreds of properties with each control having specific ones.

Using the Interface Editor window, the properties may be easily viewed or changed. Mouse over a control and type **F3**. This will open the properties for that control. Note that properties of different types – values, Boolean (1/0) highlighted blue, media files and colors (highlighted in yellow),

Boolean properties may be changed by click the check box to set (1) or clear (0). Media and colors have a … for select, which clicking will open selection windows. There is help for the property in the right column, but also you may select and right-click a property to see the help in a pop-up.

Note that with many colors the alpha or transparency may also be set in the color chooser window for RGB or HSV color tabs.

In code (from the controller, in Event Code, or in the Debug/CLI), properties may be set with either:
!propertyName = value (for Main interface)
or
!controlName.propertyName=value (for Child Interfaces and controls).

All controls have default properties when the property does not need to be defined in code:
!txtValue=10

As no one, including us, can remember all the properties available, as discussed in the section on the Debug/CLI Window, contextual help can be used to help format an instruction string. This help is also available in the Code Editor Window.

Properties and values in code can be accessed by enclosing in brackets.

[propertyName] for Main Interface
[controlName.propertyName] for Child Interfaces and controls.

!txtData = [plot.nameCh0] is [plot.ch0]

The Debug/CLI and Code Editor are again great ways for contextual help in formatting your strings.

Values

Like properties, many controls and interfaces have values that can be read and not written to, such as:

[plot.ch0]

[plot.timeCh0]

[systemTime]

Instructions

Beyond setting properties, controls also have more formal instructions that perform some task, such as resetting a plot. These, of course, can come from the Debug/CLI, Event Code or serially from a controller. Contextual help in the Debug/CLI or Code Editor will assist on using the instructions.
!plot.reset

Multiple instructions can be on a single line by separating with a semi-colon.
plot.reset; !bell

Update Values

Update Values allow control values to be updated in the background without explicit code or use of the Processing Queue where data is normally processed for use. Simply place the value you wish to use in the Update Value for a control, such as **[ain0]**, to begin updating. Note: While in Edit Mode, updates are disabled.

The default property of a control is the property updated by default, such as the text for a textbox. To update a different property, identify the property:

.backColor=[ain0]

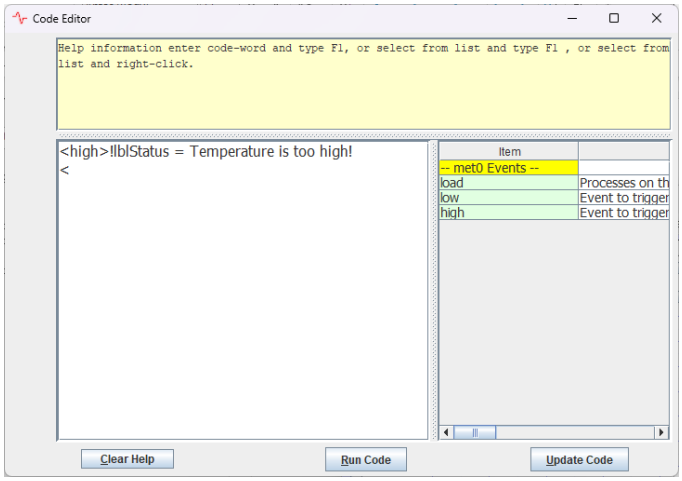
Also, math, enclosed in { }, may be used in the Update Value to modify a value, such as:
{ [ain0] * [txtSpan] + [txtOffset] – 0.5 }

Event Code

Event Code are daqINSIGHT instructions that are placed at the top of the Processing Queue when an event takes place, such as clicking a button or changing a value.

Many controls and the interfaces allow specific code for when an event takes place, such as a click, double-click, analog data arrive, digital data arriving. The basic format per line is:
<event>!instructions;!instruction

Using the Code Editor (double-click inside the Event Code text box) allows contextual help, such as for a meter control, listing the available events for that control type:



The **<load>** event defines code that will run when the control is loaded to allow some specific parameter to be set or some action to be taken, such as creating a variable.

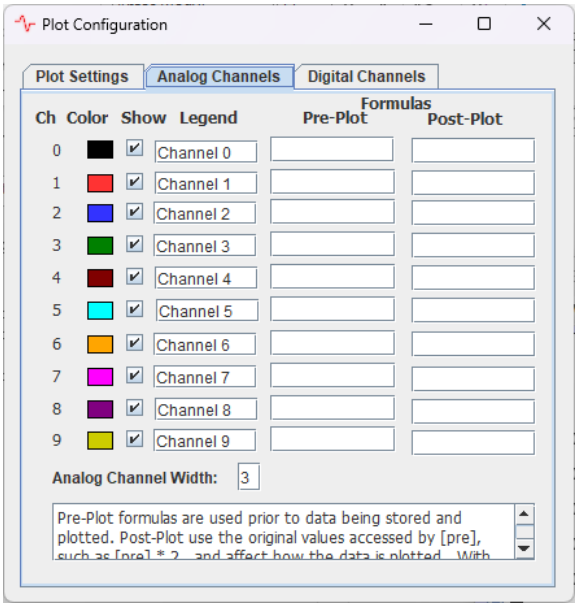
In Event Code, generic code can be written for the control in which it exists, using 'this':

!this.backColor=RED // Has this control perform some instruction
[this] // accesses the name of the control.
[[this]] // accesses the value of the control's default property.
[this.backColor] // accesses the defined property of the control.

Analog Plots

The Analog Plot Controls have an abundance of features the could double the size of this guide, so just a few things will be covered, but be sure to right-click the plot and become familiar with the various option available.

The Settings Window is highly important, so let's discuss that in a bit of detail. Right-Click the plot and click Settings, go the Analog Channels Tab.



Here you may select the color, visibility (as you can in the legend) and name of the channel.

By default, the analog plot will plot [ain0] for channel 0, [ain1] for channel 1, and so on. For the Pre-Plot formula, you may enter the specific analog value you wish to plot, such as [ain15], for a channel, or **off** if you don't wish that channel to be plotted or stored. You may also perform math, such as:
[ain1] + [ain2] /2 (note that { } are not required for math).

The post plot allows changes to data that has been plotted already, such as using [pre] + 10 will add 10 to each point. This may be deleted and return to pre-plot values. Initially setting a post-plot value may take several seconds to accomplish its task.

Also note the used of post-fix notation math for formulas, such as:
:: [ain1] [ain2] + 2 /
These operate much quicker than infix (standard) math.

Once plotted, there a numerous values concerning the plotted data that can be accessed. Please use the Debug/CLI (**F4**) and enter **[plot.** to see what is available, such as averages and times of plotted data.

Note that through the pop-up menu, the analog plot can plot digital data as well for mixed-mode plotting.

Math Operations

daqINSIGHT has an abundance of math operators using both infix (standard) math operations and post-fix for quicker operations, string and number-base conversions.

Standard Math (Infix)

The basic format of Standard Math is enclosing in { } and used standard math operators and functions as well as daqINSIGHT values, such as:

? { 2 * (10 + 5) }

? { [ain0] * [txtSpan] + [txtOffset] }

!lblValue= The value is: { [ain1] * 1000 } volts

Please see the help files for a full listing of math operators and functions available.

Post-Fix

Post -Fix Notation operates quicker than standard math and has capabilities to process strings and perform conversion between number bases. It is denoted with {:: }
? {:: abcdefghijklmnopqrstuvwxyx 5 3 subString } // returns fgh

Note: Strings with spaces must use :: between terms and operators.
? {:: hello world how are you?::6::5::subString } // return world

? {:: 25 dec2Hex } // returns 19

Please see the help files for a full list of operators and use.

Variables

daqINSIGHT allows the creation of variables to hold data. They are of the variant variety in that they can be treated as strings or numeric values. Once created they are treated as properties or values.

!makeVar name=value
!makeVar varData=0
!varData = { [ain0] * 2 }

? [varData]

Variables allow pre- or post- decrementing and incrementing.
? [++varData] // increment before use
? [varData++] // increment after use
? [--varData] // decrement before use
? [varData--] // decrement after use

An indexed variable can be created, such as:
!makeVarIndexed varIdxTest,5
Which will create **varIdxTest0** to **varIdxTest4**

These could be used with data to index them:
!varIdxTest[ain2]=[ain3]
Such that is ain2 were 1, **varIdxTest** would hold the value of ain3. Another use may be to use the index value of a dropbox to index with.

Another form of indexing are List Arrays which can hold large amounts of data, perform manipulation of the data, and be treated as a stack. Please see the references.

Paths, Logging and Reporting

Paths

daqINSIGHT has multiple paths that can be used with instructions to locate directories quickly.

[path]	Path the Interface's Directory
[pathApp]	Path to the application's directory
[pathMedia]	Path to the application's Media Directory
[pathDocs]	Path the Window's Document directory for daqINSIGHT

When used in an instruction, the / is included such that an example such as **[pathDocs]myData.txt** can be used.

By default, most instructions start at the application directory so the **logs/mylog.txt** would access the log file. Some instructions use the Interface's directory as the home path unless otherwise specified.

Folders may be created using !**createFolder**:

!createFolder [pathDocs]MyData

A folder may be opened with **!fileOpen path**

!fileOpen [pathDocs]

Logging Data

The logging control handles logging of data with numerous features. Please see the control and the reference files for it.

The plots allow for automatic snapshots of the plot or interface prior to resetting at maximum time or shifting at maximum time. These may be accessed through the plot's popup menu. Note that the interface cannot be minimized – if minimized and one of these is enabled, the plot will return to normal size. It does not need to be the top window.

Writing Data

Data may be written manually using the **fileWrite** instruction. Note that there are restrictions to where a file can be written, such as the Interface's directory or the Windows document directory for daqINSIGHT. Also, EXE's and JAR's cannot be written to.

!fileWrite path/filename, data

!fileWrite [pathDocs]myData.txt, [systemDateTime] Temperature = [ain0]

Successive writes will append to the file.

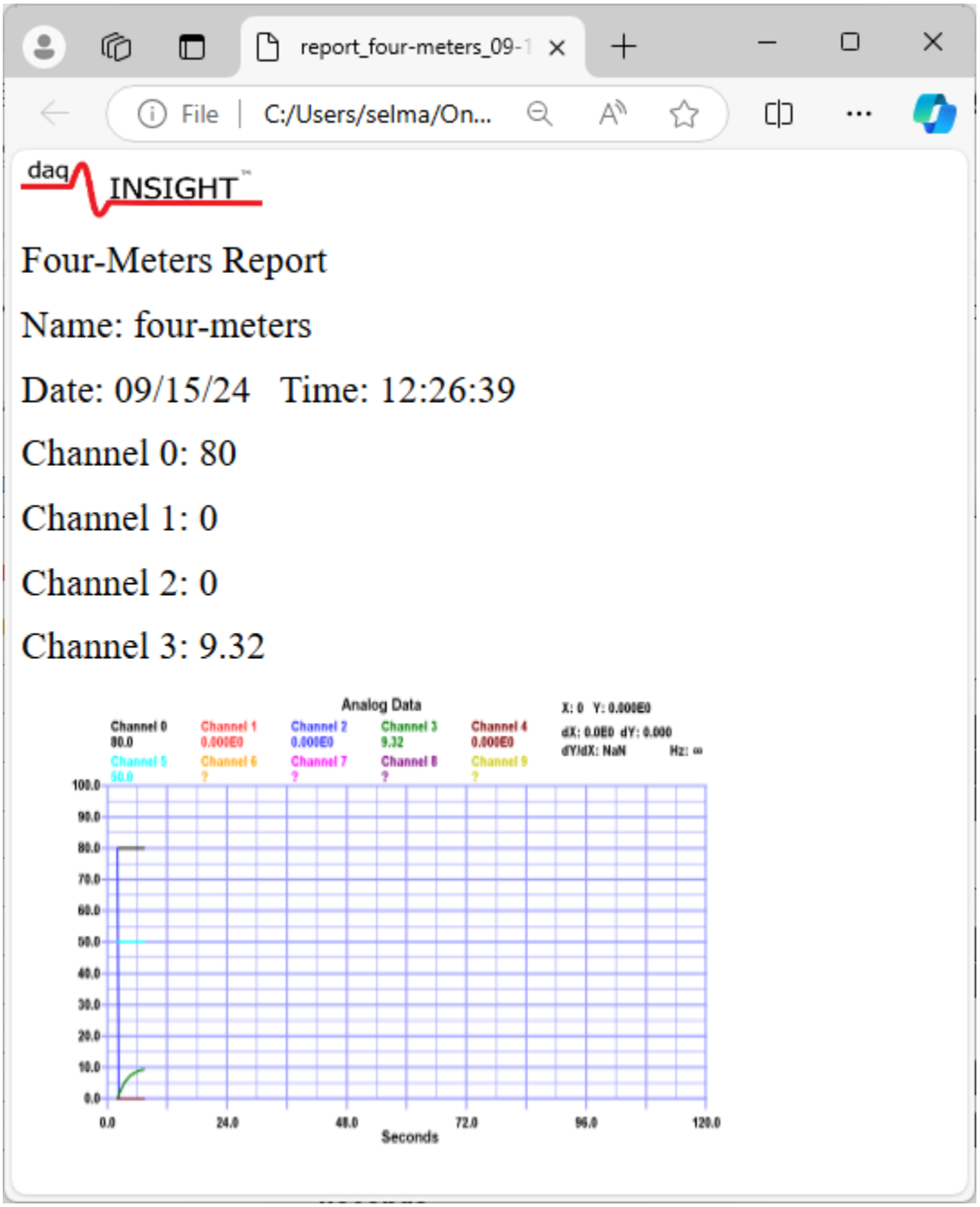
Making Reports

Reports from ASCII file types may be created and saved. File types cannot be binary files, but can be ASCII text such as CSV, TXT, HTML, etc. Some older versions of DOC files MAY work.

!report template.txt, path/filename.txt

Note the file to be used as a template should reside in the interface's directory.

The Four Meters interface has such an example by using an HTML file to generate a report and save it to Window's document directory of daqINSIGHT\Four Meters.



The original HTML file in the Interfaces\four_meters directory is the following:

** **
** <img border="0" src="[pathMedia]system\dl_logo.png" width="200" height="50" > **
**[title] Report **
** Name: [txtSnapName] **
** Date: [systemDate]   Time: [systemTime] **
** [plot.nameCh0]: [plot.ch0] **
** [plot.nameCh1]: [plot.ch1] **
** [plot.nameCh2]: [plot.ch2] **
** [plot.nameCh3]: [plot.ch3] **
** <img border="0" src="[plot.lastSnapName]" width="600" height="400" > **

The ASCII file is read, daqINSIGHT values are replaced such as [systemTime], and written to the specified file.

The code to take snapshots and generate the data is as follows:

```
!createFolder [pathDocs][title] // create new folder in documents
!plot.snap [pathDocs][title]/[txtSnapName]_[systemDateTimeDash].png // take a plot snapshot and store in the folder
!delay 250 // delay to allow snapshot to finalize
!report report.htm,[pathDocs][title]/report_[txtSnapName]_[systemDateTimeDash].htm // generate a report to the folder
!fileOpen [pathDocs][title]/report_[txtSnapName]_[systemDateTimeDash].htm // open the report
```

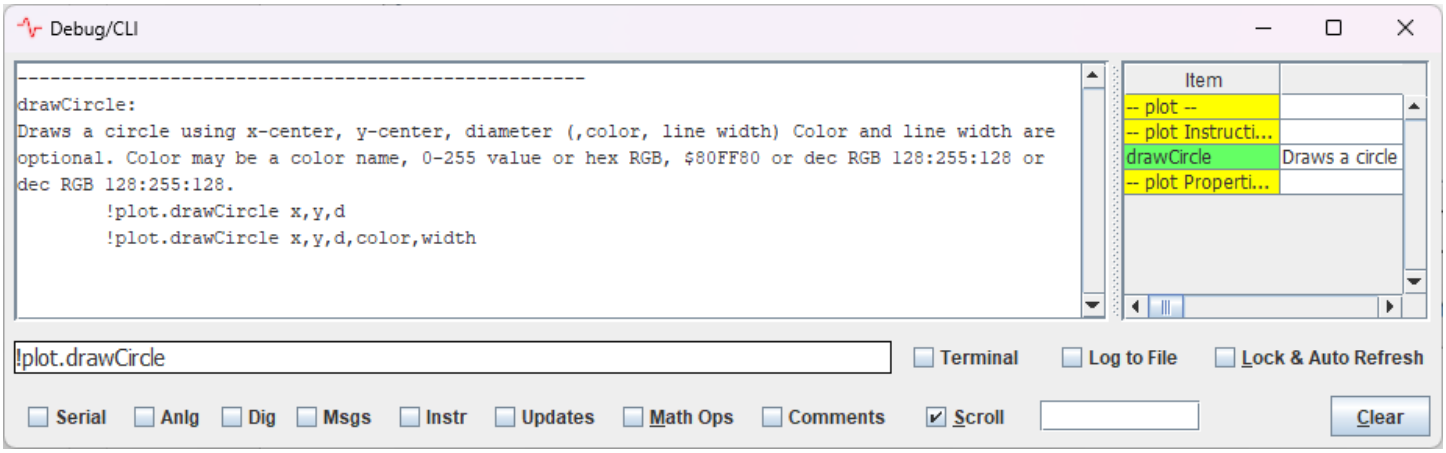
Drawing Instructions

Drawing instructions can be a powerful tool in displaying information in graphical formats and for creating interesting graphics for other uses, such as active controls to indicate data. By incorporating data

With drawing instructions various graphics can be drawn on plots, the interface backgrounds and on the canvas control. Drawing instructions include lines, arcs, rectangles, images, etc.

Drawing instruction can make use of data, such as [ain1], or values such as the number of seconds into the plot, **[plot.plotTimeSeconds]**, or the time in seconds of the prior plotted value, **[plot.lastTimeCh0]**, allowing a powerful way to created specialized graphics.

For help with formatting of the instructions, use the Debug/CLI window, enter an instruction with contextual help and press **F1**.



Colors

Colors can be defined 4 different ways:

- Name, such as RED, GREEN, BLUE, etc
- Decimal value, 0 to 16777215 defining the amount of red, green and blue (not recommended)
- Hexadecimal RGB $000000 to $FFFFFF for the amount of red, green and blue ($RRGGBB).
- Decimal RGB 00:00:00 to 255:255:255 as red:green:blue. Note that a 4th value may be used to set the amount of transparency (alpha) from 0 (fully transparent) to 255 (fully opaque). 128:00:00:128 would be half opaque red.

Analog & Digital Plots

Analog plots support all drawing instructions.

!plot.drawCircle 50,50,5,RED

Coordinates are based on the X and Y axis unless suffixed with an ‘a’, such as 0a, which defines an **absolute coordinates** where 0a,0a is always lower-left and 100a,100a is always upper-right of the plot area. Note, the size for some instructions may also be absolute.

!plot.drawCircle 50a,50a,5a,RED

When using Full instructions, such as drawFullCircle, the drawing can exceed the boundaries of the plotted area.

!plot.drawFullCircle 100a,105a,5,RED

All drawings on the plot can be cleared with **!plot.clearDrawings** or the last one made with **!plot.deleteDrawing**.

Unless made as constant drawings, all drawings are deleted on a plot reset.

Constant Drawings

Constant Drawings survive a plot reset and can be saved as part of the interface file when saved if **constFileSave** is enabled. This allow drawing to be performed that can be part of the plot area for more indications or information.

Constant Drawings are indicating by starting with @ instead of !
@plot.drawLine 0,0,100,100,RED

Constant drawings can be cleared with **!plot.clearConstDrawings** or the last deleted with **!plot.deleteConstDrawing**.

Canvas Controls

Canvas controls support drawing, with 0,0 in upper-left to 100,100 in lower-right. They also support constant drawings using @ and allowing saving to the interface file (See the Plot drawing previously).

By laying out a constant drawing and using normal drawing over it, indicators with backgrounds can be constructed.

Interfaces

Both the main interface and child interfaces allow drawing on the backgrounds, with 0,0 in upper-left to 100,100 in lower-right. They also support constant drawings using @ and allowing saving to the interface file (See the Plot drawing previously).

By laying out a constant drawing and using normal drawing over it, indicators with backgrounds can be constructed.

Unlike drawing on a child interface or control, for the main interface simply using the drawing instruction:

!drawLine 0,0,100,100,RED

Saving

Interfaces

Saving an interface saves all data to construct the interface, settings (properties) of the controls and Event Code. The file type is a .**dqii** extension.

Settings

Saving settings only saves the properties of the controls so that configurations of control settings can be saved. The file type is a **.dqis** extension.

Sharing

If you use only the installed media, you can simply save the interface file with other. If you use your own media for backgrounds, images and buttons, etc., it is recommended you create a folder containing all the elements needed and saving your interface files to that folder, then sharing the folder as a .zip.

In Closing

This primer scratches the surface of the capabilities and potential of daqINSIGHT. Please look through the references and use our discussion board accessible from the Help menu if you have questions.