
Using DATAQ ActiveX in MATLAB®

Please follow this tutorial step by step to learn how to port previously recorded data into a MATLAB® matrix with the ReadDataqFile ActiveX Control.

For more example programs using the DATAQ ActiveX Controls with MATLAB®, go to: <http://www.dataq.com/develop/MATLAB.php>

Step 1: Download and Install the DATAQ ActiveX Controls.

The latest version of the DATAQ ActiveX Controls can be downloaded for free at: <http://www.dataq.com/develop/index.php?qu=install>

Step 2: Download the ReadDataqFileAnalysis.m example program.

Go to: <http://www.dataq.com/develop/MATLAB.php?file=6>
Click "Download this Code" and save the Zip (dataq_MATLAB_6.zip) file to your local drive, then unzip the file contents.

Zip file contents:

ReadDataqFileAnalysis.m – Main M-File function. This sample code will: create the ReadDataqFile control in a MATLAB® figure window, open a WinDaq file, put the first 200 data points into a matrix, and then plot the data.

controlerror.m – Support M-File event function. If Windows generates an error, this event returns that error to your program.

dataqfileerror.m – Support M-File event function. This event most commonly fires when the specified file does not exist or it is not a WinDaq file.

endoffile.m – Support M-File event function. This event executes with an End of File message when the ReadDataqFile control reaches the end of the WinDaq file.

README.txt

Step 3: Add the directory where the M-files are located to the MATLAB® path.

Use the addpath function from the MATLAB® command prompt. For example, if you unzipped the file to the C:\ drive, type:

```
>> addpath('C:\DATAQMATLABExamples')
```

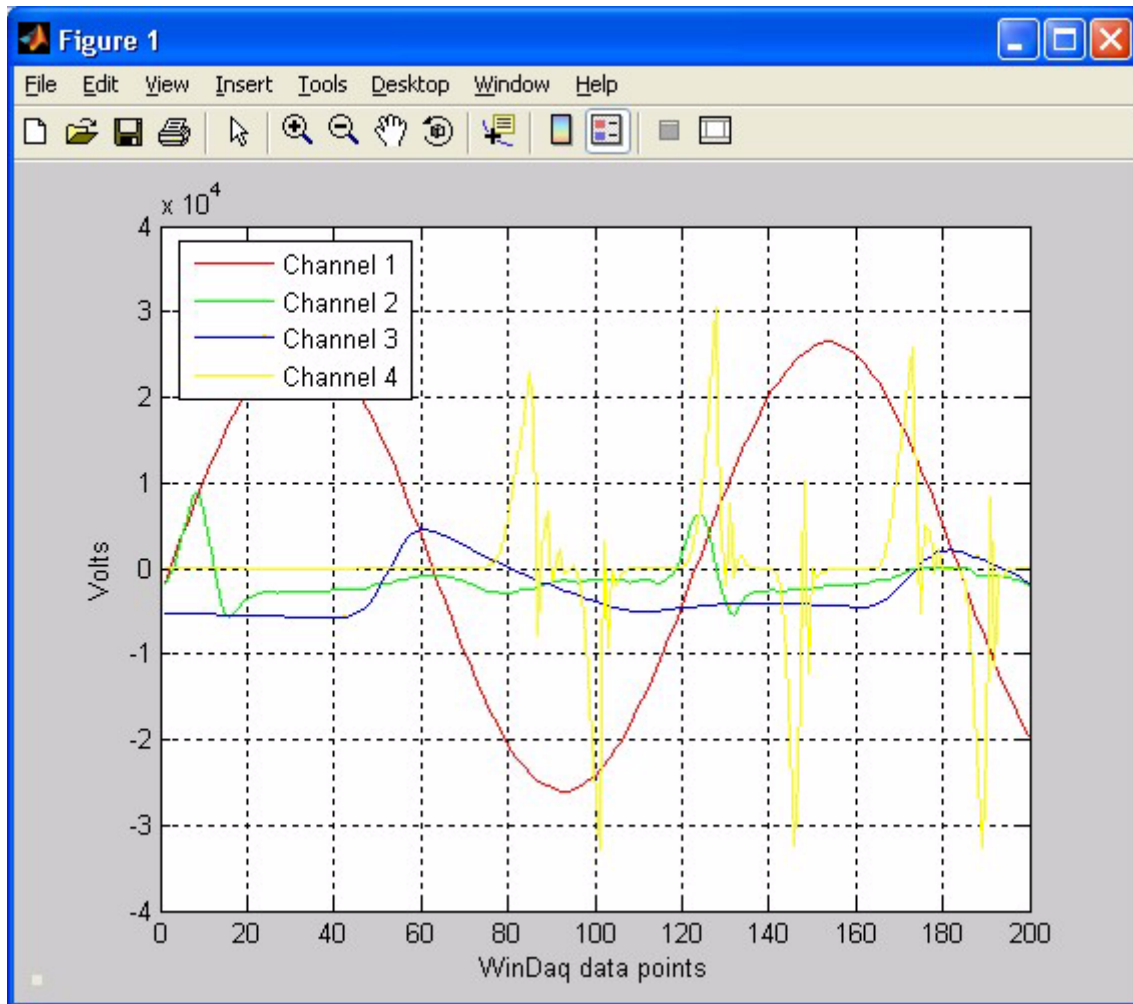
Step 4: Run the example program.

Note: ReadDataqFileAnalysis.m uses the WinDaq file C:\DATAQ\SAMPLE.wdq by default. If you do not have the SAMPLE.wdq file in the DATAQ folder on your C drive, the example program will generate a file error until you change the file name in the code.

To run the example program, type the M-file name at the MATLAB® command prompt:

```
>> ReadDataqFileAnalysis
```

The program should generate a figure window like the one below:



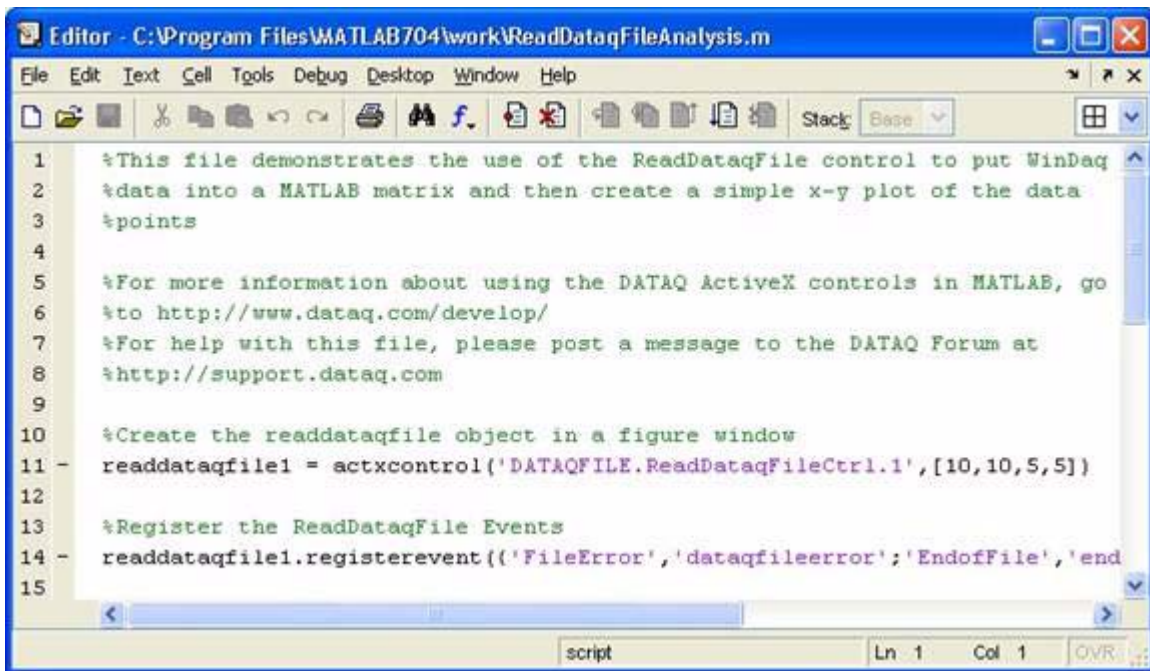
Step 4: Look at the MATLAB® code.

Now that you have the example running in MATLAB®, let's take a look at the code syntax to see what happened when you executed ReadDataqFileAnalysis:

Open the ReadDataqFileAnalysis M-file from MATLAB® by going to File>>Open.

Select the file ReadDataqFileAnalysis.m from the location you unzipped it to and click Open.

This will open the M-file in the MATLAB® Editor.



```

1  %This file demonstrates the use of the ReadDataqFile control to put WinDaq
2  %data into a MATLAB matrix and then create a simple x-y plot of the data
3  %points
4
5  %For more information about using the DATAQ ActiveX controls in MATLAB, go
6  %to http://www.dataq.com/develop/
7  %For help with this file, please post a message to the DATAQ Forum at
8  %http://support.dataq.com
9
10 %Create the readdataqfile object in a figure window
11 - readdataqfile1 = actxcontrol('DATAQFILE.ReadDataqFileCtrl.1',[10,10,5,5])
12
13 %Register the ReadDataqFile Events
14 - readdataqfile1.registerevent({'FileError','dataqfileerror';'EndOfFile','end
15

```

Note: The % character designates a comment in MATLAB®. (%This is what a comment looks like.) The ... characters designate that the code is going on continuously to the next line.

Let's take a look at the code line by line:

MATLAB® has a function called actxcontrol to use ActiveX controls. The Syntax for actxcontrol is:
 dataqcontrol = actxcontrol('progid', position, fig_handle,
 event_handles, ... 'filename')

The returned object (dataqcontrol in the above example) is the default interface for the control in MATLAB®. For a list of DATAQ ActiveX progid's, go to: <http://www.dataq.com/develop/MATLAB-help.php>.

Line 11 creates the ReadDataqFile ActiveX control and assigns it to the variable readdataqfile1:

```
readdataqfile1 = actxcontrol('DATAQFILE.ReadDataqFileCtrl.1',[10,10,5,5])
```

Note: The control is created at a position 10 pixels from the bottom left corner and a size of 5 X 5 (width by height).

Line 12 registers the event handles after the control has been created. ReadDataqFile has three events: FileError, EndOfFile, and Control Error. The code below registers these events and identifies the M-file function they are associated with. For example, when the ControlError event is triggered, the M-file function controlerror.m will execute.

```
readdataqfile1.registerevent({'FileError','dataqfileerror'; 'EndOfFile', ... 'endoffile'; 'ControlError','controlerror'})
```

Line 17 sets the FileName property to the WinDaq file located at C:\DATAQ\SAMPLE.wdq. If the

file does not exist a FileError will be generated.

```
set(readdataqfile1, 'FileName', 'C:\DATAQ\SAMPLE.wdq')
```

Line 20 opens the WinDaq file:

```
readdataqfile1.Open
```

Line 24 sets dqAnalysis as a matrix array equal to the first 200 channel scans to be returned. One scan is defined as one pass through all enabled channels (i.e., one sample per channel).

```
dqAnalysis = readdataqfile1.GetData(200, 0)
```

Syntax: ReadDataqFile1.GetData(*count*, *method*) (where count is an Integer between 1 and 32,767 and method is either 0 (FormatBinary) or 1 (FormatScaled))

Note: dqAnalysis will be displayed on the screen. However, if you end the line with a semicolon, MATLAB® performs the computation but does not display any output.

Line 28 sets dqChannels to the number of columns in the dqAnalysis matrix, and dqDataPts to the number of rows (channels scans).

```
[dqChannels, dqDataPts] = size(dqAnalysis)
```

```
dqChannels =
```

```
4
```

```
dqDataPts =
```

```
200
```

Line 31 creates a time vector, dqTime, of rows from 1 to dqDataPts.

```
dqTime = 1:dqDataPts
```

Line 34 sets the default line style for the graph so that all of the lines are solid.

```
set(0, 'defaultaxeslinestyleorder', '-')
```

Line 35 sets the default line colors so that each channel (for up to eight channels) will show as a different colored line.

```
set(0, 'defaultaxescolororder', [1 0 0; 0 1 0; 0 0 1; 1 1 0; 0 0 0; 1 0 1; 0 1 1])
```

Line 36 plots the dqAnalysis data along a time line of 1 to 200 data points, and labels each of the channels in the graph legend.

```
plot(dqTime, dqAnalysis), legend('Channel 1', 'Channel 2', 'Channel 3', ...  
'Channel 4', 2)
```

Line 37 annotates the x-axis as "WinDaq data points", the y-axis as "Volts", and puts the grid lines on the graph.

```
xlabel('WinDaq data points'), ylabel('Volts'), grid on
```

Step 5: Look at the ReadDataqFile Control properties, methods, and events.

To learn more about the ReadDataqFile control and its properties, methods and events, download the ActiveX help file: <http://www.dataq.com/support/documentation/helpfiles/dataqxc.html.zip>.

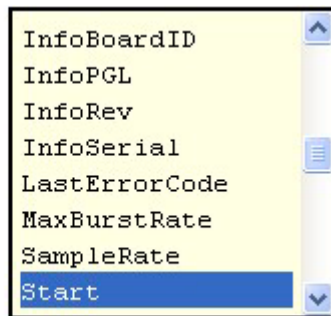
Control Information	MATLAB [®] Syntax
List Control Properties	dataqcontrol.get
List Control Events	dataqcontrol.events
List events attached to listeners	dataqcontrol.eventlisteners
List Control Interfaces	dataqcontrol.interfaces
List Available Methods	dataqcontrol.invoke
Release COM Interface	dataqcontrol.release
Delete COM Control	dataqcontrol.delete

The above table shows some of the ActiveX control information that is easily accessed by MATLAB[®].

For example:

```
>> readdataqfile1.SampleRate
ans =
    250
```

Use the Tab key to display the available control options.



```
>> dataqsdk.st
```

Step 6: Perform MATLAB[®] Calculations.

Now that you understand the example program code, let's add more code to perform calculations on the dqAnalysis matrix of data that we have imported from WinDaq.

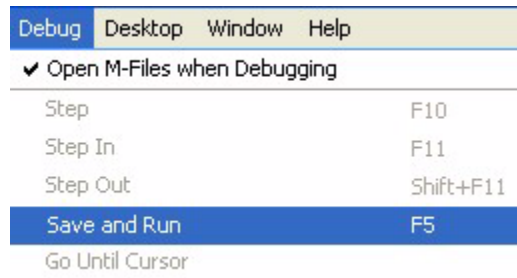
Note: The following calculations can be run from the MATLAB[®] command prompt or added to the ReadDataqFileAnalysis M-file for re-use.

Before we do calculations, let's change our format to FormatScaled so that our data will be depicted as recognizable voltage measurements.

Go to **Line 24** and change the format from 0 to 1 (1 = FormatScaled)

```
dqAnalysis = readdataqfile1.GetData(200, 1)
```

Now execute the program. Go to Debug>>Save and Run, or click the F5 key:



Find the Max, Min, and Mean

Max:

```
>> mx = max(dqAnalysis)
```

This example returns the max for each of the 200 data point rows. To find the largest value in the entire data set use:

```
>> mx = max(dqAnalysis (:))
```

Min:

```
>> min(dqAnalysis (:))
```

Mean (For matrices, mean(A,2) is a column vector containing the mean value of each row):

```
>> mean(dqAnalysis, 2)
```

ans =

```

    0.6443
   -0.2000
   -0.4027
    0.0305

```

For the first 200 data points from the SAMPLE.wdq file, you should get the mean results shown above. (Channel 1 = 0.6443) The mean values are rounded.

To learn how to perform mathematical calculations in MATLAB® and for complete documentation of the MATLAB® programming language, please consult the MATLAB® help file.

If you have questions about this tutorial please post to the DATAQ Forum at: <http://support.dataq.com>.