DATA TRANSLATION®
A MEASUREMENT COMPUTING COMPANY

UM-22024-N

Data Translation DAQ Adaptor for MATLAB®

Trademark and Copyright Information

Measurement Computing Corporation, InstaCal, Universal Library, and the Measurement Computing logo are either trademarks or registered trademarks of Measurement Computing Corporation. Refer to the Copyrights & Trademarks section on mccdaq.com/legal for more information about Measurement Computing trademarks. Other product and company names mentioned herein are trademarks or trade names of their respective companies.

© 2016 Measurement Computing Corporation. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form by any means, electronic, mechanical, by photocopying, recording, or otherwise without the prior written permission of Measurement Computing Corporation.

Notice

Measurement Computing Corporation does not authorize any Measurement Computing Corporation product for use in life support systems and/or devices without prior written consent from Measurement Computing Corporation. Life support devices/systems are devices or systems that, a) are intended for surgical implantation into the body, or b) support or sustain life and whose failure to perform can be reasonably expected to result in injury. Measurement Computing Corporation products are not designed with the components required, and are not subject to the testing required to ensure a level of reliability suitable for the treatment and diagnosis of people.

Table of Contents

About this Manual 5
Intended Audience 5
How this Manual is Organized 5
Conventions Used in this Manual 5
Where To Get Help 5

Chapter 1: Overview 7
Overview 8
Session-Based or Legacy Interface 9
Related Information 10

Chapter 2: Session-Based DAQ Adaptor for MATLAB 11
Overview 12
Requirements 13
Installation 14
Hardware Discovery and Session Setup 15
Analog Input Operations 16
Analog Input Channel Functions, Events, and Properties 16
Setting up an Analog Input Operation 17
Adding Other Channels in the Analog Input Stream 19
Performing a Single-Value Input Operation 20
Performing a Foreground Acquisition 20
Performing a Background Acquisition 21
Triggering Analog Input Operations 22
Analog Output Operations 24
Analog Output Channel Functions, Events, and Properties 24
Setting up an Analog Output Operation 24
Adding a Digital Output Port in the Analog Output Stream 26
Performing a Single-Value Output Operation 27
Performing a Foreground Output Operation 27
Performing a Background Output Operation 28
Triggering Analog Output Operations 29
Analog Input and Output Operations at the Same Time 31
Performing I/O Operations at the Same Time in the Foreground 31
Performing I/O Operations at the Same Time in the Background 32
Digital I/O Operations 34
Performing Digital Input Operations 34
Performing a Digital Output Operation 34

Chapter 3: Legacy DAQ Adaptor for MATLAB 35
Requirements 36
Installation 37
Setting up an I/O Operation 38
Analog Input and Output Subsystem Properties 39
Setting the Input Type 40
Setting the Synchronization Mode 40
Setting the Filter Type 40
Channel Properties 41
Configuring a Channel for a Voltage Input 43
Configuring a Channel for a Current Measurement 44
Configuring a Channel for an IEPE (Accelerometer) Input 44
Configuring a Channel for a Thermocouple Input 45
Configuring a Channel for an RTD Input 45
RTD Support for MEASURpoint Instruments 45
RTD Support for the DT9829 and Other DT-Open Layers Devices 46
Configuring a Channel for a Thermistor Input 48
Configuring a Channel for a Resistance Measurement 49
Configuring a Channel for a Strain Gage Input 49
Configuring a Channel for a Bridge-Based Sensor 50
Configuring the Gain of a Channel 50
Utility Methods in MATLAB 51
olDaReadBridgeSensorVirtualTeds 51
olDaReadStrainGageVirtualTeds 53
VoltstoMicroStrain 55
VoltstoBridgeBasedSensor 56
MATLAB Property Notes 57
Multiple Subsystems of a Single Type 57
Analog Output Operation Notes 58
Using Hardware Triggers 58

Index 59

About this Manual

This manual describes how to use the Data Translation DAQ Adaptor for MATLAB to program Data Translation DT-Open Layers devices using MATLAB® from the Mathworks.

Intended Audience

This document is intended for engineers, scientists, technicians, or others responsible for using and/or programming DT-Open Layers-compliant devices using MATLAB. It is assumed that you have some familiarity with data acquisition principles and that you understand your application.

How this Manual is Organized

The manual is organized as follows:

- Chapter 1, "Overview," describes the Data Translation DAQ Adaptor for MATLAB, describes the session-based and legacy interfaces of the DAQ Adaptor for MATLAB, and provides references to related information.
- Chapter 2, "Session-Based DAQ Adaptor for MATLAB," describes the session-based Data Translation DAQ Adaptor for MATLAB.
- Chapter 3, "Legacy DAQ Adaptor for MATLAB," describes the legacy Data Translation DAQ Adaptor for MATLAB.

Conventions Used in this Manual

The following conventions are used in this manual:

- Notes provide useful information or information that requires special emphasis, cautions provide information to help you avoid losing data or damaging your equipment, and warnings provide information to help you avoid catastrophic damage to yourself or your equipment.
- Code snippets are shown in `courier`.

Where To Get Help

Should you run into problems installing or using the Data Translation DAQ Adaptor for MATLAB, the Data Translation Technical Support Department is available to provide technical assistance. Refer to the Data Translation web site (www.mccdaq.com) or call 508-946-5100. If you are outside the United States or Canada, call your local distributor, whose number is listed on our web site (www.mccdaq.com).



Overview

Overview 8
Session-Based or Legacy Interface 9
Related Information 10

Overview

The Data Translation DAQ Adaptor for MATLAB provides an interface between MATLAB and the Data Acquisition Toolbox and Data Translation's DT-Open Layers architecture. Figure 1 shows the data flow model using the Data Translation DAQ Adaptor for MATLAB.

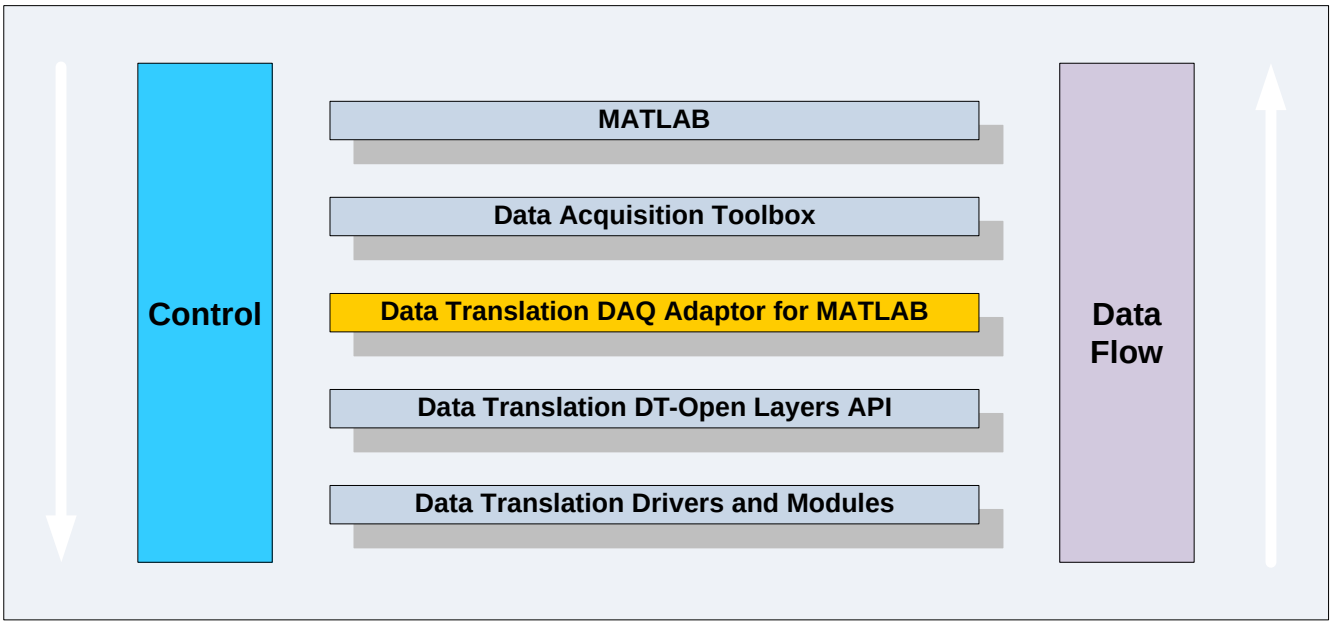

Figure 1: Data Flow Model Using the Data Translation DAQ Adaptor for MATLAB

DT-Open Layers provides an interface to all of Data Translation's analog and digital I/O data acquisition hardware modules. By using the Data Translation DAQ Adaptor for MATLAB with MATLAB and the Data Acquisition Toolbox, you can acquire data from Data Translation modules directly within MATLAB and then use all of MATLAB's features to analyze and display the data.

Session-Based or Legacy Interface

Data Translation provides both a 64-bit session-based interface and a 32-bit legacy interface of the DAQ Adaptor for MATLAB.

If want to use a 64-bit version of MATLAB (release R2016a or greater of MATLAB), you need the session-based DAQ Adaptor for MATLAB.

For all previous versions of MATLAB (32-bit), you need the legacy DAQ Adaptor for MATLAB.

Both interfaces are described in this manual.

Related Information

Refer to your MATLAB documentation, including documentation for the Data Acquisition toolbox, and the user's manuals for your Data Translation modules. For more information on DT-Open Layers, refer to following documents from Data Translation:

- Online help for the Data Acquisition Toolbox for MATLAB: https://www.mathworks.com/help/daq
- *DataAcq SDK User's Manual* (UM-18326). For programmers who are developing their own application programs using the Microsoft C compiler, this manual describes how to use the DT-Open Layers™ DataAcq SDK™ to access the capabilities of Data Translation data acquisition devices.
- *DT-Open Layers for .NET User's Manual* (UM-22161). For programmers who are developing their own application programs using Visual C# or Visual Basic .NET, this manual describes how to use the DT-Open Layers for .NET Class Library to access the capabilities of Data Translation data acquisition devices.



Session-Based DAQ Adaptor for MATLAB

Overview 12
Requirements 13
Installation 14
Hardware Discovery and Session Setup 15
Analog Input Operations 16
Analog Output Operations 24
Analog Input and Output Operations at the Same Time 31
Digital I/O Operations 34

Overview

Data Translation's session-based DAQ Adaptor for MATLAB supports analog and digital I/O operations on Data Translation DT-Open Layers-compliant modules.

Note: For Data Translation modules only, if your device supports digital input, tachometer, or counter operations in the analog input stream, you can read the value of the digital input port, tachometer, or counter channel through the specified analog input subsystem using the session-based DAQ Adaptor for MATLAB.

Similarly, if your Data Translation device supports digital output operations in the analog output stream, you can update the value of the digital output port through the specified analog output subsystem using the session-based DAQ Adaptor for MATLAB.

MEASURpoint instruments are not supported using the session-based DAQ Adaptor for MATLAB.

Requirements

You must install the following before installing the session-based DAQ Adaptor for MATLAB:

- Your Data Translation hardware module(s) and their drivers (from the Data Acquisition OMNI CD that ships with your hardware)
- A 64-bit version of MATLAB - Version R2016a or higher
- Data Acquisition Toolbox Version 3.9 or higher

Install Data Translation components from a recent OMNI CD, or download the software from our web site (www.mccdaq.com). Contact The MathWorks for updates to your MATLAB components.

Installation

You can install the session-based DAQ Adaptor for MATLAB one of the following ways:

From the Data Translation web site:

1. Navigate to the following web page for the Data Translation DAQ Adaptor for MATLAB: http://www.mccdaq.com/Products/Data-Acquisition-Software/MATLAB-Data-Acquisition
2. Click the **Download** tab.
3. Click **Free Download**.
4. Click **Session-Based DAQ Adaptor for MATLAB**, and then click **Download**.
   *The file **Data Translation Data Acquisition Toolbox.mltbx** is downloaded to your computer.*
5. Once it is downloaded, right-click on the file **Data Translation Data Acquisition Toolbox.mltbx**, and click **Install**.
   The Install Data Translation Data Acquisition Toolbox dialog appears.
6. Click **Install**.
   The Data Translation DAQ Adaptor for MATLAB is installed.

From within MATLAB:

1. Click **Add-Ons.**
   The Add-Ons Explorer appears.
2. Search for **Data Translation**.
3. Click **Session-Based DAQ Adaptor for MATLAB**.
4. Click **Add**, and select **Add to MATLAB**.
   The Data Translation DAQ Adaptor for MATLAB is installed.

From the MATLAB Central web site:

1. Navigate to the MATLAB Central main page:
   http://www.mathworks.com/matlabcentral/?s_tid=srchtitle
2. Search for **Data Translation**.
3. Click **Session-Based DAQ Adaptor for MATLAB**.
4. Click **Download Zip**.
   *The file **Data Translation Data Acquisition Toolbox.mltbx** and the PDF file that contains the documentation are downloaded to your computer.*
5. Once downloaded and unzipped, right-click on the file **Data Translation Data Acquisition Toolbox.mltbx**, and click **Install**.
   The Install Data Translation Data Acquisition Toolbox dialog appears.
6. Click **Install**.
   The Data Translation DAQ Adaptor for MATLAB is installed.

Hardware Discovery and Session Setup

Refer to the following web page for information on the functions and properties that are available for hardware discovery and session setup:
http://www.mathworks.com/help/daq/hardware-discovery-and-session-setup.html

Note that to create a data acquisition session for Data Translation devices, use the **daq.createSession** function and specify `'dt'` as the vendor name.

Analog Input Operations

This section describes information about performing analog input operations using the MATLAB session-based interface.

Analog Input Channel Functions, Events, and Properties

Refer to the following web page for information on the functions, events, and properties for setting up analog input channels:
http://www.mathworks.com/help/daq/analog-data-acquisition.html

For Data Translation devices, note the following differences:

- **TerminalConfig** – For Data Translation devices, the terminal configuration (SingleEnded or Differential) applies to all the channels of the analog input subsystem. Therefore, you cannot specify one channel as SingleEnded and another channel as Differential for the same subsystem.
- **Range** – Gains are not specified in MATLAB. Therefore, when specifying the input range for an analog input channel on a Data Translation device, you must specify the effective input range that you desire. For example, assume that your device supports an input range of ±10 V and a gain of 1, which yields an effective input range of ±10 V, and a gain of 2 , which provides an effective input range of ±5 V. For the **Range** property, specify either –10 V to +10 V or –5 V to +5 V for the channel.

  Some devices also allow you to read the value of other input channels, such as the digital input port, tachometer, or counters, through the analog input stream. For these devices, a range appropriate to that channel type, such as 0 to 65535 for the digital input port, is provided. In these cases, the input range applies only to the specified channel type.
- **MeasurementType** – Currently, `'Voltage'` (input) and `'IEPE'` (input) measurement types are available for Data Translation devices that support these capabilities.

Setting up an Analog Input Operation

This section shows an example of setting up an analog input operation for a Data Translation device using the session-based DAQ Adaptor for MATLAB:

1. Discover connected devices using the **daq.getDevices** function. In the following example, the system has several devices installed, including the Data Translation DT9837B device. The DT9837B device has the ID `'DT9837-B(01)'`, which is set in the Open Layers Control Panel (see the documentation for your hardware for more information on the Open Layers Control Panel):

```
devs = daq.getDevices
ans =
Data acquisition devices:
index Vendor Device ID             Description
----- ------ ---------  -------------------------------
1     ni       Dev1    National Instruments USB-9201
2     ni       Dev2    National Instruments USB-9234
3     ni       Dev4    National Instruments USB-4431
4     ni       Dev5    National Instruments USB-9265
5 directsound  Audio0  DirectSound Primary Sound Capture Driver
6 directsound  Audio1  DirectSound Microphone (Realtek High Definition Audio)
7 directsound  Audio2  DirectSound Primary Sound Driver
8 directsound  Audio3  DirectSound Speaker/HP Realtek High Definition Audio)
9    dt      DT9837-B(01) Data Translation DT9837-B(01)
```

2. To get information about the device, click the device ID or specify the index of the device to query. For example, to get information about the Data Translation device with index 9, use the following command:

```
devs(9)
```

   The following information is returned:

```
dt.Data Translation DT9837-B(01) (Device ID:'DT9837-B(01)')
Analog input subsystem supports:
-10 to +10 Volts, -1.0 to +1.0 Volts ranges
Rates from 195.3 to 105469.0 scans/sec
7 channels ('0'-'6')
'Voltage' measurement type
```

3. Create a session for Data Translation devices using the **daq.createSession** function, specifying `'dt'` as the vendor name. An example follows; the default number of scans and the acquisition duration are shown:

```
s = daq.createSession('dt')
s
s =
Data acquisition session using Data Translation hardware:
Will run for 1 second (1000 scans) at 1000.116 scans/second.
No channels have been added.
```

   By default, the acquisition is configured to run for a duration of 1 second to acquire 1000 scans; for this device, the default rate is 1000.116 scans per second.

4. Configure the session, as needed, using the following properties: **NumberofScans**, **DurationinSeconds**, **Rate**, or **IsContinuous**. The following example sets the duration of the acquisition to 10 seconds:

```
s.DurationInSeconds = 10;
s
s =
Data acquisition session using Data Translation hardware:
Will run for 10 seconds (10000 scans) at 1000.116 scans/second.
No channels have been added.
```

5. Add analog input channels to the session using the **addAnalogInputChannel** function. The following example adds analog input channels 0 and 1 of the DT9837-B(01) device to the session; note that these channels are configured for the IEPE measurement type:

```
ch0 = s.addAnalogInputChannel('DT9837-B(01)','0', 'IEPE');
ch1 = s.addAnalogInputChannel('DT9837-B(01)','1', 'IEPE');
s
s=
Data acquisition session using Data Translation hardware: Will run
for 10 seconds (10000 scans) at 1000.116 scans/second.
Number of channels: 2
index Type Device        Channel MeasurementType Range      Name
_____ ____ ____________ _______ _______________ __________ _____
1     ai   DT9837-B(01)  0       IEPE (Diff)  -10 to +10 Volts
2     ai   DT9837-B(01)  1       IEPE (Diff)  -10 to +10 Volts
```

Note: If desired, you can add analog input and analog output channels from one or multiple devices in the same session.

6. Configure the analog input channels, as needed, using the following properties: **Range**, **TerminalConfig**, **ID**, **Coupling**, **ExcitationCurrent,** and/or **ExcitationSource**.

   The following example configures IEPE inputs for analog input channel 0 (*ch* index 1) and analog input channel 1 (*ch* index 2) of the `'DT9837-B(01)'` device. Channel 0 is configured for AC coupling and an internal excitation current value of 4 mA. Channel 1 is configured for DC coupling and no excitation source is used:

```
ch0.Range = [-10 10];
ch0.TerminalConfig='SingleEnded';
ch0.Coupling='AC';
ch0.ExcitationCurrentSource='Internal';
ch0.ExcitationCurrent = 0.004;
ch1.Range = [-10 10];
ch1.TerminalConfig='SingleEnded';
ch1.Coupling='DC';
ch1.ExcitationCurrentSource='None';
```

Adding Other Channels in the Analog Input Stream

If your Data Translation device supports reading the tachometer, digital input port, or counter through the analog input stream, you can add these as channels using the **addAnalogInputChannel** function. Note that you must specify `'Voltage'` as the measurement type using the **addAnalogInputChannel** function; however, in these cases, the value is treated as a raw number rather than a voltage.

Refer to the user's manual for your device to determine whether other channels can be added to the analog input stream as well as their corresponding channel numbers.

The following example adds the tachometer (channel 4) of the DT9837B device with device ID `'DT9837-B(01)'` to the analog input stream:

```
ch = s.addAnalogInputChannel('DT9837-B(01)','4', 'Voltage');
```

Get information about the channel:

```
ch
ans =

Data acquisition analog input voltage '4' on device 'DT9837-B(01)'
Terminal Config: Differential
Range: 0 to 4294967295
Name:"
Device:[1x1 daq.dt.DeviceInfo]
MeasurementType:'Tachometer'
```

Note that the range for the channel depends on the channel type. For example, the range for a 16-bit digital input port is 0 to 65535. The value that is returned represents the value of the channel, so in this example, if the first eight bits are set to 1 and the other bits are set to 0, then a value of 255 would be returned for this channel.

Note that if you add the analog input channels as well as other channels to the input stream, the specified voltage range is used for the analog input channels only.

Performing a Single-Value Input Operation

You can acquire a single analog input value using the **inputSingleScan** function.

The following example shows how to acquire a single analog input value for a Data Translation device using the session-based DAQ Adaptor for MATLAB. A Data Translation DT9837B device with device ID `'DT9837-B(01)'` is used in this example:

1. Create a session:

   ```
   s = daq.createSession('dt');
   ```

2. Add analog input channels 0 and 1 to the session:

   ```
   s.addAnalogInputChannel('DT9837-B(01)','0', 'Voltage');
   s.addAnalogInputChannel('DT9837-B(01)','1', 'Voltage');
   ```

3. Acquire an analog input value for each input channel that was added:

   ```
   data = inputSingleScan(s);
   ```

   One value is acquired for each analog input channel and the operation stops.

4. Read the acquired data. In this example, the value –0.1495 V is acquired from analog input channel 0 and 0.8643 V is acquired from analog input channel 1.

   ```
   data =
       -0.1495    0.8643
   ```

Performing a Foreground Acquisition

When you perform an analog input operation in the foreground, the operation blocks MATLAB until the operation is complete. Therefore, no other MATLAB commands can be used until the operation completes.

The following example shows how to start an analog input operation in the foreground. A Data Translation DT9837B device with device ID `'DT9837-B(01)'` is used in this example:

1. Create a session and add analog input channel 0 to the session:

   ```
   s = daq.createSession('dt');
   ```

2. Add analog input channel 0 to the session:

   ```
   s.addAnalogInputChannel('DT9837-B(01)', 0, 'IEPE');
   ```

3. Set the sample rate to 10 kHz:

   ```
   s.Rate = 10000;
   ```

4. Set the channel's ExcitationCurrent to 4 mA:

   ```
   s.Channels(1).ExcitationCurrent = .004;
   ```

5. Start the analog input session in the foreground and save the data and the acquisition time in the variables *[data, time]*:

   ```
   [data,time] = startForeground(s);
   ```

6. Plot the data:

   ```
   plot(time, data);
   ```

7. When you are finished with the operation, delete the session:

```
delete (s);
```

Performing a Background Acquisition

When you perform an analog input operation in the background, other MATLAB commands can be executed while the analog input operation is in progress. MATLAB is not blocked. A background acquisition depends on events and listeners to allow your code to access data as the hardware acquires it and to react to any errors as they occur.

Refer to the following web page for more information on events and listeners in background operations: http://www.mathworks.com/help/matlab/matlab_oop/events-and-listeners--concepts.html

To start an analog input operation in the background, use the **startBackground** command. Use the **DataAvailable** event to determine when the acquired data is available.

Note: By default the **DataAvailable** event fires when 1/10 second worth of data is available for analysis. If you want to change the frequency of when the **DataAvailable** event is fired, set **IsNotifyWhenDataAvailableExceedsAuto** to false, and modify the value of **NotifyWhenDataAvailableExceeds** as desired. Refer to your MATLAB documentation for more information.

Listeners execute a callback function when notified that the event has occurred. Use **Session.addlistener** to create a listener object that executes your callback function.

By default, the **IsContinuous** property is set to False and the operation stops automatically. If you have a continuous operation, set **IsContinuous** to True and use **stop** when you want to stop the background operation.

If desired, you can use **wait** to block MATLAB execution until the background operation is complete.

This example shows how to acquire data from a Data Translation DT9837B device with device ID `'DT9837-B(01)'` in the background using events and listeners:

1. Create an Data Translation session object:

```
s = daq.createSession('dt');
```

2. Add analog input 0 to the session:

```
addAnalogInputChannel(s,'DT9837-B(01)', '0', 'Voltage');
```

3. Add the listener for the DataAvailable event and assign it to the variable *lh*:

```
lh = addlistener(s,'DataAvailable', @plotData);
```

4. Create a simple callback function to plot the acquired data and save it as *plotData.m* in your working directory:

```
function plotData(src,event)
  plot(event.TimeStamps, event.Data)
end
```

   Here, *src* is the session object for the listener and event is a *daq.DataAvailableInfo* object that contains the data and associated timing information.

5. Start a continuous background operation and see the plot update while MATLAB is running:

```
s.IsContinuous = true;
startBackground(s);
```

6. When you are finished acquiring data, stop the operation:

```
s.stop();
```

7. Delete the listener:

```
delete (lh);
```

8. When you are finished with the operation, delete the session:

```
delete (s);
```

Triggering Analog Input Operations

Note: Refer to the following web page for information on the functions and properties for setting up triggers using MATLAB:
http://www.mathworks.com/help/daq/simultaneous-and-synchronized-operations.html

You can trigger analog input operations based on an external event if your Data Translation device supports an external TTL trigger for the analog input subsystem. Refer to the user's manual for your device to determine its supported capabilities.

To specify an external TTL trigger on a Data Translation device, use the MATLAB **addTriggerConnection** function, and specify the following parameters:

- For the trigger source, specify `'external'`.
- For the trigger destination, specify 'deviceID/EXT_TTL_AI_TRIG', where *deviceID* is the ID of your device as defined in the Open Layers Control Panel. See the documentation for your hardware for more information on pin assignments and the Open Layers Control Panel.
- For the trigger type, specify 'StartTrigger'.

Once you have set up the trigger connection, use the **TriggerCondition** property to set the trigger condition to `'RisingEdge'` or `'FallingEdge'`.

The following example shows how to trigger an analog input operation on a DT9837B device with device ID `'DT9837-B(01)'` using a signal connected to the external TTL trigger input on the device:

1. Create a session and add two analog input channels:

   ```
   s = daq.createSession('dt');
   ch = addAnalogInputChannel(s,'DT9837-B(01)',0:1,'Voltage');
   ```

2. Configure the terminal configuration and range of the channels in the session:

   ```
   ch(1).TerminalConfig = 'SingleEnded';
   ch(1).Range = [-10.0 10.0];
   ch(2).TerminalConfig = 'SingleEnded';
   ch(2).Range = [-10.0 10.0];
   ```

3. Create an external trigger connection and specify that a rising-edge trigger will trigger acquisition:

   ```
   addTriggerConnection(s,'external','DT9837-B(01)/EXT_TTL_AI_TRIG',
     'StartTrigger');
   s.Connections(1).TriggerCondition = 'RisingEdge';
   ```

4. Set the rate and the duration of the acquisition:

   ```
   s.Rate = 50000;
   s.DurationInSeconds = 0.01;
   ```

5. Set up the foreground operation:

   ```
   [data,timestamps] = startForeground(s);
   plot(timestamps,data);
   ```

   When the rising-edge trigger is detected on the external TTL trigger input pin on the DT9837-B(01) device, analog input data is acquired and plotted in the foreground.

Analog Output Operations

This section describes information about performing analog output operations using the MATLAB session-based interface.

Analog Output Channel Functions, Events, and Properties

Refer to the following web page for information on the functions, events, and properties for setting up analog input channels:
http://www.mathworks.com/help/daq/analog-signal-generation.html

For Data Translation devices, note the following differences:

- **TerminalConfig** – For Data Translation devices, the terminal configuration (SingleEnded or Differential) applies to all the channels of the analog output subsystem. This setting is typically not configurable for analog output subsystems but is returned.
- **Range** – One or more analog output voltage ranges may be supported by your Data Translation device. The voltage range applies to all analog output channels of the analog output subsystem.

  Some devices also allow you to update the digital output port through the analog output stream. For these devices, a digital output range, such as 0 to 65535, is provided. The digital output range applies only to the digital output channel in the analog output stream.
- **MeasurementType** – Currently, `'Voltage'` (output) is the only measurement type available for Data Translation devices. You must specify this measurement type even if you are setting up a digital output channel in the analog output stream.

Setting up an Analog Output Operation

This section shows an example of setting up an analog output operation for a Data Translation device using the session-based DAQ Adaptor for MATLAB:

1. Discover connected devices using the **daq.getDevices** function. In the following example, the system has several devices installed, including the Data Translation DT9836-12-2 device. The DT9836-12-2 device has the ID `'dt9836'`, which is set in the Open Layers Control Panel (see the documentation for your hardware for more information on the Open Layers Control Panel):

```
devs = daq.getDevices
ans =
Data acquisition devices:
index Vendor Device ID             Description
----- ------ ---------  -------------------------------
1 dt           dt9836  Data Translation DT9836-12-2
2 directsound  Audio0  DirectSound Primary Sound Capture Driver
3 directsound  Audio1  DirectSound Microphone (Realtek High Definition Audio)
4 directsound  Audio2  DirectSound Primary Sound Driver
5 directsound  Audio3  DirectSound DELL U2414H (Intel(R) Display Audio)
6 directsound  Audio4  DirectSound Speakers/Headphones (Realtek High Def Audio)
7 ni           Dev1    National Instruments USB-9234
8 ni           Dev2    National Instruments PCI-MIO-16XE-50
```

2. To get information about the device, click the device ID or specify the index of the device to query. For example, to get information about the Data Translation device with index 1, use the following command:

   ```
   devs(1)
   ```

   The following information is returned:

   ```
   dt.Data Translation DT9836-12-2 (Device ID:'dt9836')
    Analog input subsystem supports:
     -10 to +10 Volts, -5.0 to +5.0 Volts ranges
     Rates from 0.0 to 225000.0 scans/sec
     23 channels ('0' - '22')
     'Voltage' measurement type

    Analog output subsystem supports:
     -10 to +10 Volts ranges
     Rates from 0.0 to 500000.0 scans/sec
     3 channels ('0','1','2')
     'Voltage' measurement type
   ```

3. Create a session for Data Translation devices using the **daq.createSession** function, specifying '**dt**' as the vendor name. An example follows; the default number of scans and the acquisition duration are shown:

   ```
   s = daq.createSession('dt')
   s
   s =
   Data acquisition session using Data Translation hardware:
   Will run for 1 second (1000 scans) at 1000 scans/second.
   No channels have been added.
   ```

   By default, the operation is configured to run for a duration of 1 second to output If1000 scans; for this device, the default rate is 1000 scans per second.

4. Configure the session, as needed. For analog output operations, use the following properties: **DurationinSeconds**, **Rate**, and/or **IsContinuous**. The following example sets the output rate to 8 kHz:

   ```
   s.Rate = 8000;
   ```

5. Add analog output channels to the session using the **addAnalogOutputChannel** function. The following example adds analog output channels 0 and 1 of the dt9836 device to the session:

   ```
   ch0 = s.addAnalogOutputChannel('dt9836','0', 'Voltage');
   ch1 = s.addAnalogOutputChannel('dt9836','1', 'Voltage');
   ```

Note: If desired, you can add analog input and analog output channels from one or multiple devices in the same session.

Adding a Digital Output Port in the Analog Output Stream

If your Data Translation device supports updating the digital output port through the analog output stream, you can add the digital output port as a channel using the **addAnalogOutputChannel** function. Note that you must specify `'Voltage'` as the measurement type using the **addAnalogOutputChannel** function even though it is a digital output channel. The DAQ Adaptor for MATLAB ignores voltage for this channel, treating the data as a raw value.

The following example adds the digital output port (channel 2) of the DT9836-12-2 device (device ID `'dt9836'`) to the analog output stream:

```
aoutch = s.addAnalogOutputChannel('dt9836','2', 'Voltage');
```

Get information about the channel:

```
aoutch
ans =

Data acquisition analog output voltage '2' on device 'dt9836'
Terminal Config: Differential
Range: 0 to +65535 DigitalOutput
Name:
Device:[1x1 daq.dt.DeviceInfo]
MeasurementType:'DigitalOutput'
```

The range for a digital output channel is 0 to 65535, which represents the 16-bit digital output port. Note that if you add the analog output channels as well as the digital output channel to the output stream, the specified voltage range is used for the analog output channels and the 0 to 65535 range is used for the digital output channel.

Note: The MeasurementType that is returned for the digital output channel in the analog output stream is `'DigitalOutput'`, which indicates the channel type. However, when using the **addAnalogOutputChannel** function, you must specify `'Voltage'` for this channel type.

When you want to update the digital output channel, specify a value that corresponds to the bits of the 16-bit digital port. For example, to set the lower eight bits to 1 and the higher eight bits to 0, specify a value of 255.

Performing a Single-Value Output Operation

You can output a single analog output value using the **outputSingleScan f**unction.

The following example shows how to output a single analog output value to a Data Translation device using the session-based DAQ Adaptor for MATLAB. A Data Translation DT9836-12-2 device (device ID `'dt9836'`) is used in this example:

1. Create a session:

   ```
   s = daq.createSession('dt');
   ```

2. Add analog output channels 0 and 1 to the session:

   ```
   s.addAnalogOutputChannel('dt9836', 0, 'Voltage');
   s.addAnalogOutputChannel('dt9836', 1, 'Voltage');
   ```

3. Create an output value and output a single scan for each output channel that was added.

   ```
   outputSingleScan(s,[1.5 4]);
   ```

   In this example, a value of 1.5 V is output to analog output channel 0 and a value of 4 V is output to analog output channel 1. One value is output for each channel and the operation stops.

Performing a Foreground Output Operation

When you perform an analog output operation in the foreground, the operation blocks MATLAB until the operation is complete. Therefore, no other MATLAB commands can be used until the operation completes.

The following example shows how to start an analog output operation in the foreground. A Data Translation DT9836-12-2 device with device ID `'dt9836'` is used in this example:

1. Create a session:

   ```
   s = daq.createSession('dt');
   ```

2. Add analog output channel 0 to the session:

   ```
   s.addAnalogOutputChannel('dt9836', 0, 'Voltage');
   ```

3. Create the data to output:

   ```
   outputData = linspace(-1, 1, 2200)';
   ```

4. Queue the data:

   ```
   queueOutputData(s,outputData);
   ```

   The duration changes based on the length of the queued data and the specified scan rate. When the session contains output channels, duration and number of scans become read-only properties of the session. The number of scans in a session is determined by the amount of data queued and the duration is determined by **s.ScansQueued** divided by **s.Rate**.

5. Generate the output signal in the foreground; MATLAB returns once the data has been output.

   ```
   startForeground(s);
   ```

6. When you are finished, delete the session:

```
delete (s);
```

Performing a Background Output Operation

When you perform an analog output operation in the background, other MATLAB commands can be executed while the analog output operation is in progress. MATLAB is not blocked. A background output operation depends on events and listeners to allow your code to continue to output data as the queue data is depleted and to react to any errors that may occur.

To start an analog output operation in the background, use the **startBackground** command. Use the **DataRequired** event to determine when more data needs to be queued to the device. This event is fired periodically when a continuous output operation is in progress.

Listeners execute a callback function when notified that the event has occurred. Use **addlistener** to create a listener object that executes your callback function.

Refer to the following web page for more information on events and listeners in background operations: http://www.mathworks.com/help/matlab/matlab_oop/events-and-listeners--concepts.html

By default, the **IsContinuous** property is set to False and the operation stops automatically. If you want to perform a continuous output operation, set **IsContinuous** to True and use **stop** when you want to stop the background operation.

To start an analog output operation in the background, use the **startBackground** command.

This example shows how to output analog data from a Data Translation DT9836-12-2 device (device ID `'dt9836'`) in the background:

1. Create an Data Translation session object:

```
s = daq.createSession('dt');
```

2. Add analog output 0 to the session:

```
addAnalogOutputChannel(s,'dt9836', '0', 'Voltage');
```

3. Set the output rate. In this example, the output rate is set to 8 kHz:

```
s.Rate = 8000;
```

4. Create the data to output:

```
data = linspace(0,pi,s.Rate)';
outputSignal = sin(data*2)*2;
```

5. Queue the output data:

```
queueOutputData(s,outputSignal);
```

6. Set up an output event handler that adds a listener for DataRequired events. In this example, the event handler requeues the output data when the DataRequired event occurs; the listener for the DataRequired event is assigned to the variable *lh*:

```
lh = addlistener(s,'DataRequired', ...
     @(src,event) src.queueOutputData(outputSignal));
```

7. Generate the output signal in the background:

```
s.IsContinuous = true;
startBackground(s);
```

   You can execute other MATLAB commands while the signal generation operation is in progress.

8. When you are finished generating the signal, stop the operation:

```
s.stop();
```

9. Delete the listener:

```
delete (lh);
```

10. When you are finished, delete the session:

```
delete (s);
```

Triggering Analog Output Operations

Note: Refer to the following web page for information on the functions and properties for setting up triggers using MATLAB:
http://www.mathworks.com/help/daq/simultaneous-and-synchronized-operations.html

You can trigger analog output operations based on an external event if the your Data Translation device supports an external TTL trigger for the analog output subsystem. Refer to the user's manual for your device to determine its supported capabilities.

To specify an external TTL trigger for an analog output operation on a Data Translation device, use the MATLAB **addTriggerConnection** function with the following parameters:

- For the trigger source, specify `'external'`.
- For the trigger destination, specify 'deviceID/EXT_TTL_AO_TRIG', where *deviceID* is the ID of your device as defined in the Open Layers Control Panel. See the documentation for your hardware for more information on pin assignments and the Open Layers Control Panel.
- For the trigger type, specify 'StartTrigger'.

Once you have set up the trigger connection, use the **TriggerCondition** property to set the trigger condition to `'RisingEdge'` or `'FallingEdge'`.

The following example shows how to trigger an analog output operation on a DT9836 device (device ID `'dt9836'`) using a signal connected to the external TTL trigger input on the device:

1. Create a session:

```
s = daq.createSession('dt');
```

2. Add analog output channel 0 to the session:

```
s.addAnalogOutputChannel('dt9836', 0, 'Voltage');
```

3. Create an external trigger connection and specify that a rising-edge trigger will trigger the analog output operation:

```
addTriggerConnection(s,'external','dt9836/EXT_TTL_AO_TRIG',
  'StartTrigger');
s.Connections(1).TriggerCondition = 'RisingEdge';
```

4. Create the data to output:

```
outputData = linspace(-1, 1, 2200)';
```

5. Queue the data:

```
queueOutputData(s,outputData);
```

6. Set up the foreground operation:

```
startForeground(s);
```

   When the rising edge is detected on the on external trigger pin of the DT9836 device, the analog output operation is started in the foreground. MATLAB returns once the data has been output.

Analog Input and Output Operations at the Same Time

The DAQ Adaptor for MATLAB supports performing analog input and analog output operations at the same time. For DT-Open Layers-compliant devices that support this feature, the DAQ Adaptor automatically starts both operations at the same time (unless different trigger sources are used for each subsystem).

Performing I/O Operations at the Same Time in the Foreground

The following example shows how to perform analog input and output operations at the same time in the foreground. This examples uses a Data Translation DT9836-12-2 device (device ID `'dt9836'`):

1. Create a Data Translation session object:

   ```
   s = daq.createSession('dt');
   ```

2. Add analog output 0 to the session:

   ```
   addAnalogOutputChannel(s,'dt9836', '0', 'Voltage');
   ```

3. Add analog input 0 to the session:

   ```
   addAnalogInputChannel(s,'dt9836', '0', 'Voltage');
   ```

4. Set the properties of the session. Note that for any given session, acquisition and signal generation run at the same rate. In this example, the acquisition and signal generation rate is set to 8 kHz:

   ```
   s.Rate = 8000;
   ```

5. Create the data to output:

   ```
   data = linspace(0,pi,s.Rate)';
   outputSignal1 = sin(data)*2;
   ```

6. Queue the output data:

   ```
   queueOutputData(s,outputSignal1);
   ```

7. Generate the output signal and acquire data in the foreground:

   ```
   [data,time] = startForeground(s);
   ```

   MATLAB returns once the I/O operations are complete.

8. Plot the data:

   ```
   plot(time, data);
   ```

9. When you are finished, delete the session:

   ```
   delete (s);
   ```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/547150042013006062