Using Data Translation® Measurement Instruments with MATLAB® through IVI-COM

Introduction

You can access the functionality of Data Translation's measurement instruments (TEMPpoint, VOLTpoint, and MEASURpoint) using the MATLAB Instrument Control Toolbox. The Instrument Control Toolbox communicates with DT measurement instruments through a MATLAB driver to the instruments IVI-COM driver. Both USB and Ethernet (LXI) versions of the instruments are supported.

To use the MATLAB Instrument Control Toolbox with a measurement instrument, you need to install the IVI-COM driver (DtxMeasurement) provided by Data Translation.

This document describes how to access the functionality of a DT measurement instruments through the MATLAB Instrument Control Toolbox.

Installing the DtxMeasurement IVI-COM Driver and the MATLAB Driver for Measurement Instruments

To install the IVI-COM driver and MATLAB driver, perform the following steps:

1. Insert the MEASURpoint Software CD (supplied with your instrument) into your CD-ROM or DVD drive.
   The installation program should automatically start, and the installation program should appear.
2. If the installation program does not automatically start, double-click **Setup.exe** from the CD.
   The installation program appears.
3. Click **Install from Web (recommended)** to get the latest version of the software or **Install from CD** to install the software from the CD.
4. If you are installing from the web, perform these steps:
   a. Click **MEASURpoint Software** and follow the prompts to install the software (including the MEASURpoint Framework application, IVI-COM driver, and IVI shared components) and related documentation.
   b. Click **MATLAB Driver for Measurement Instruments** to install DtxMeasurement_DtxMeasurement.mdd and related examples and documentation.
5. If you are installing from the CD, perform these steps:
   a. Click **Install Measurement Software**.
   b. Ensure that **Measurement (Software & Application)** is selected.
   c. Click **Install Selected Features** and follow the prompts to install the software (including the Measurement application, IVI-COM driver, and IVI shared components) and related documentation.
   d. When you are finished with the Instrument Omni CD, click **Quit Installer**.

Note: The MATLAB driver for Measurement Instruments (DtxMeasurement_DtxMeasurement.mdd) is also available for download from the MATLAB Central website.

Using the DtxMeasurement IVI-COM Driver in MATLAB

Once you have installed the DtxMeasurement IVI-COM driver and the MATLAB driver for Measurement Instruments, start MATLAB and ensure that the

DtxMeasurement_DtxMeasurement.mdd file is in the current directory window. If this file is not in the current directory window, browse to the directory in which you downloaded the DtxMeasurement_DtxMeasurement.mdd file, and save this file to your working directory.

You can access measurement instruments from the MATLAB command line using IVI-COM methods and properties. The following sections describe typical operations when writing a MATLAB script.

Note: The MATLAB Instrument Control Toolbox provides the graphical Test & Measurement Tool that allows you to access measurement instruments without writing MATLAB scripts; the tool generates the MATLAB script for you. See www.mathworks.com/tmtool for more information.

Step 1. Create an Object for the Instrument

Create an object for the measurement instrument using the following command:

```
>> dev =
icdevice('DtxMeasurement_DtxMeasurement.mdd',
RsrcName);
```

where, *RsrcName* is an IVI logical name or an instrument-specific string, such as a VISA resource descriptor, that identifies the address of the measurement instrument, and *dev* is the object that is created.

USB Instruments

For USB instruments, specify USB::*InstrumentName* for *RsrcName*, where *InstrumentName* is the name of the instrument in the Open Layers Control Panel. An example follows:

```
>> dev =
icdevice('DtxMeasurement_DtxMeasurement.mdd',
'USB::DT987x(01)');
```

To determine, and optionally edit, the name of your USB instrument, from the Windows Start menu, click **Settings** -> **Control Panel** -> **Open Layers Control Panel**. The name of your device is listed. Ensure that you use this name after the USB:: prefix.

Ethernet Instruments

For Ethernet (LXI) instruments, specify TCPIP::*Address*::INSTR for *RsrcName*, where *Address* is the IP address of the instrument on the network. An example follows:

```
>> dev =
icdevice('DtxMeasurement_DtxMeasurement.mdd',
  'TCPIP::192.43.218.69::INSTR'; or
  'TCPIP::192.43.218.69::SOCKET' *
* use of SOCKET does not require VISA I/O Library
```

You can determine the IP address of your instrument on the TCP/IP network using an LXI discovery tool, such as the Data Translation Eureka Discovery Utility, automatically installed with the Instrument Omni CD. To use the Eureka Discovery Utility, perform the following steps:
From the Windows Start menu, click **Programs** -> **Data Translation, Inc** -> **Instrument Support** -> **Eureka LXI Instrument Discovery**

Step 2. Connect to the Instrument

Once you have created an object for the Measurement instrument, connect to the instrument using the following command:

```
>> connect(dev);
```

This command opens a session with the instrument, enabling all subsequent communication with the instrument.

Step 3. Configure and Control the Instrument

To configure and control a Measurement instrument, open the MATLAB Instrument Driver Editor (see ***Figure 1***) using the following command:

```
>> midedit 'DtxMeasurement_DtxMeasurement.mdd'
```

The MATLAB Instrument Driver Editor window appears.

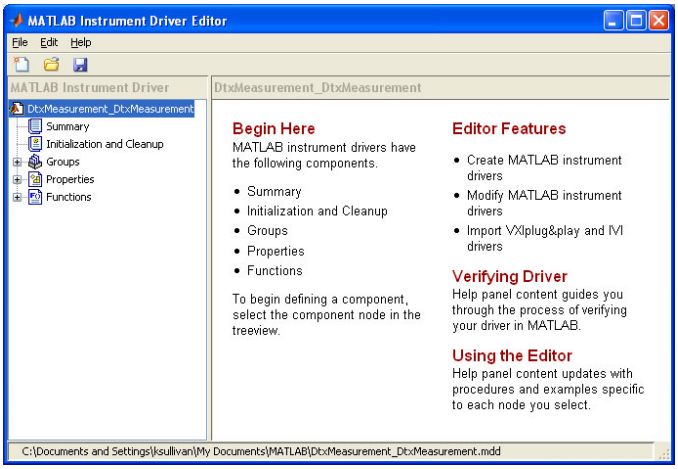

Figure 1: MATLAB Instrument Driver Editor

This window allows you to see the available IVI-COM properties and methods (functions) that you can access from MATLAB.

Accessing an Interface

From MATLAB, you can access the following interface, properties, and methods of the DtxMeasurement IVI-COM driver using the **get** command:

- Identity (inherited from IIviDriver) – Provides properties for returning information about the instrument, including the instrument model, manufacturer, firmware revision, and so on.
- Trigger – Provides properties for controlling the scan rate and the number of scans to acquire on the instrument.
- Acquisition – Provides methods and properties for initiating, retrieving, and aborting measurements on an instrument.
- DigitalIn – Provides methods for reading the digital input port of an instrument.
- DigitalOut – Provides methods for updating the digital output port of an instrument.
- System – Provides methods for locking and unlocking an instrument and properties for returning information about the instrument, such as the instrument serial number.
- DriverOperation (inherited from IIviDriver) – Provides properties and methods relating to the operation of the driver, including instrument simulation.
- Utility (inherited from IIviDriver) – Provides common methods for the instrument, including querying errors and resetting the instrument.
- Initialized property (inherited from IIviDriver) – Returns whether the instrument was initialized successfully.
- Close method (inherited from IIviDriver) – Closes the I/O session to the instrument.
- Initialize method (inherited from IIviDriver) – Opens the I/O session to the instrument.

For example, to verify that the instrument is connected, use the **get** command to create a group (called *comobj* in this example) to access the Identity interface of the IVI-COM driver:

```
>> comobj = get(dev, 'Identity');
```

You can then access various properties of this group. This example returns the instrument model information from the Identity interface:

```
propertyValue = get(comobj, 'InstrumentModel');
```

Executing a Function

To execute a function, use the *invoke* command, as shown in this example:

```
>> comobj = get(dev, 'Channels')

>> invoke(comobj, 'Configure', channels);
```

This particular example configures the list of channels specified by the *channels* parameter by invoking the **Configure** function of the Channel interface.

Getting Help about a Property or Function

To access help about a property or function, use this command:

>> instrhelp

>> instrhelp (Interface, 'Method_Property')

where *Interface* specifies the interface and *Method_Property* specifies the method or property for which to get help. The following example returns help on the Configure method of the Channels interface:

>> instrhelp (Channels, 'Configure')

You can also refer to the online help for the DtxMeasurement IVI-COM driver for detailed information about each interface, method, and property.

Example

The following example shows how to access a Data Translation measurement instrument from the MATLAB Instrument Control Toolbox using the DtxMeasurement IVI-COM driver:

```
%% Measure analog input channels on the Measurement(DT987x or DT887x) instrument
%
%  Copyright (C) 2012 DataTranslation Inc.


%% Open connection to instrument.
% RsrcName is an IVI logical name or an instrument specific string that
% identifies the address of the instrument, such as a VISA resource descriptor
% string. For USB Measurement instruments, specify USB::InstrumentName,
% where InstrumentName is the name of the Measurement instrument in the
% Open Layers Control Panel. USB::DT987x(00) is an example of a ResourceName
% for the DT987x.

RsrcName = 'TCPIP::192.43.218.113::SOCKETS';
dev = icdevice('DtxMeasurement_DtxMeasurement.mdd', RsrcName);

try
    connect(dev);

    % Enable protected commands to function
    comobj = get(dev,'System');
    invoke(comobj, 'EnableProtectedCommands', 'admin');

    % Get the instrument identity
    comobj = get(dev, 'Identity');
    propertyValue = get(comobj, 'InstrumentModel');
    str = strcat ('InstrumentModel= ',propertyValue);
    disp(str);
    propertyValue = get(comobj, 'InstrumentManufacturer');
    str = strcat ('InstrumentManufacturer= ',propertyValue);
    disp(str);
    propertyValue = get(comobj, 'InstrumentFirmwareRevision');
    str = strcat ('InstrumentFirmwareRevision= ',propertyValue);
    disp(str);
    propertyValue = get(comobj, 'Description');
    str = strcat ('Description= ',propertyValue);
```

```
    disp(str);
    propertyValue = get(comobj, 'Identifier');
    str = strcat ('Identifier= ',propertyValue);
    disp(str);
    propertyValue = get(comobj, 'Vendor');
    str = strcat ('Vendor= ',propertyValue);
    disp(str);
    propertyValue = get(comobj, 'Revision');
    str = strcat ('Revision= ',propertyValue);
    disp(str);

    % If the first channel type is thermocouple, set its thermocouple type
    % to K
    chanType = get(dev.Channel(1), 'ChannelType');
    if strcmp(chanType,'DtxMeasurementChannelTypeThermocouple')
        set(dev.Channel(1), 'ThermocoupleType', 'DtxMeasurementThermocoupleTypeK');
    end

    % If the first channel type is RTD, set its RTD type to PT100
    if strcmp(chanType,'DtxMeasurementChannelTypeRtd')
        set(dev.Channel(1), 'RtdType', 'DtxMeasurementRtdTypeAmericanPT100');
    end

    % If the first channel type is MultiRange, set the RangeType type to
    % Bipolar 10 Volts
    if strcmp(chanType,'DtxMeasurementChannelTypeMultiRange')
        set(dev.Channel(1), 'RangeType', 'DtxMeasurementRangeTypeBip10Volts');
    end

    % Enable channel 0,1 and 2 for scanning
    comobj = get(dev, 'Channels');
    comobj = comobj(1);
    channels = int32([0;1;2]);
    invoke(comobj, 'Configure', channels);

    % Specify the rate at which to scan the list of enabled channels (in seconds).
    comobj = get(dev, 'Trigger');
    set(comobj, 'TimerInterval', .2);

    % Initiate the measurements
    comobj = get(dev, 'Acquisition');
    invoke(comobj, 'Initiate');
    pause(2);
    isRunning = 1;

    % Read 10 scans every 2 seconds for 1 minute
    RequestedScansToRead = 10;
    RequestedScansIndex = 0;

    for timeInc = 1:30
        [ScansIndex, State] = invoke(comobj, 'GetStatus', 0, 0);

        if (State ==1)
            [ActualScansIndex, ActualScansRead, StartTimeInSeconds,
StartTimeInMilliSeconds, Samples] = invoke(comobj, 'Fetch', int32(RequestedScansIndex),
int32(RequestedScansToRead), int32(0), int32(0), int32([0;0]), int32([0;0]),
double([0;0]));
```

```
            RequestedScansIndex = ActualScansIndex+ActualScansRead;
            disp(['Actual Scans Index: ',num2str(ActualScansIndex) ,'  Actual Scans Read:
', num2str(ActualScansRead)]);
            disp(Samples);
            pause(2);
        else
            break;
        end
    end

    invoke(comobj, 'Abort');

    % Disable protected commands
    comobj = get(dev,'System');
    invoke(comobj, 'DisableProtectedCommands', 'admin');

catch DtxMeasurementerror
    disp(['Error id: ', DtxMeasurementerror.identifier]);
    disp(['Error Message: ',DtxMeasurementerror.message]);
end

% Close the connection with instrument
disconnect(dev);
delete(dev);
```

Conclusion

To access Data Translation measurement instruments from MATLAB, you need the MATLAB Instrument Control Toolbox, DtxMeasurement IVI-COM driver, and MATLAB driver for Measurement Instruments. By combining the precise measurement capability of Data Translation with the powerful analytical capability of MATLAB, developers can create robust applications that solve difficult measurement problems with ease.

For More Information

Overview of Measurement Instruments:
www.datatranslation.com/products/instruments/

Overview of MATLAB software:
www.mathworks.com/matlab

Overview of MATLAB Instrument Control Toolbox:
www.mathworks.com/products/instrument