



## ***Using Data Translation® DT8824 Instrument Modules with MATLAB® through IVI-COM***

RTF-A

### ***Introduction***

You can access the functionality of Data Translation's DT8824 instrument modules using the MATLAB Instrument Control Toolbox. The Instrument Control Toolbox communicates with DT8824 instrument modules through a MATLAB driver (wrapper) to the DT8824 IVI-COM driver.

To use the MATLAB Instrument Control Toolbox with a DT8824 instrument module, you need to install the DT8824 IVI-COM driver provided by Data Translation. The DT8824 IVI-COM driver setup program also installs the MATLAB driver for the DT8824 (DT8824\_DT8824.mdd) and example programs that illustrates how to use MATLAB with the IVI-COM driver.

This document describes how to access the functionality of a DT8824 instrument module through the MATLAB Instrument Control Toolbox.

### ***Prerequisite Software***

You must first install VISA (Virtual Instrument Software Architecture) software before installing the DT8824 IVI-COM driver.

To install VISA, do the following:

1. Go to **www.agilent.com**, enter **IO Libraries Suite** in the search field, and select **Agilent IO Libraries Suite 15.0** from the search results.
2. Follow the instructions on Agilent's web site to download and install the Agilent IO Libraries, which include VISA support, VISA COM support, and the Agilent Connection Expert tool.

To use the MATLAB Driver for the DT8824, ensure that you have installed version R2009b or higher of MATLAB and the Instrument Control Toolbox.

# ***Installing the DT8824 IVI-COM Driver and the MATLAB Driver for the DT8824***

To install the DT8824 IVI-COM driver and MATLAB driver for DT8824, perform the following steps:

1. Insert the DT8824 CD (supplied with your DT8824 instrument module) into your CD-ROM or DVD drive.  
*The installation program should automatically start, and the DT8824 installation program should appear.*
2. If the installation program does not automatically start, double-click **Setup.exe** from the CD.  
*The DT8824 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, click **DT8824 Software** to install the DT8824 software (including the DT8824 Getting Started Application, IVI-COM driver, IVI shared components, Eureka Discovery Utility, DT8824 SCPI Support, DT8824 Calibration Utility) and related documentation.
5. If you are installing from the DT8824 CD, click **Install Features**. When you are finished with the DT8824 CD, click **Quit Installer**.

---

**Note:** The MATLAB driver for DT8824 (DT8824\_DT8824.mdd) is available for download from the MATLAB Central website ([www.mathworks.com](http://www.mathworks.com)).

---

# Using the DT8824 IVI-COM Driver in MATLAB

Once you have installed the DT8824 IVI-COM driver and the MATLAB driver for DT8824, start MATLAB and ensure that the DT8824\_DT8824.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 DT8824\_DT8824.mdd file, and save this file to your working directory. Alternatively, you can change the MATLAB search path. The Path Browser allows you to add directories that MATLAB searches through. Refer to your MATLAB documentation for more information.

You can access DT8824 instrument modules from the MATLAB command line using IVI-COM methods and properties. For more information on IVI concepts, you can access the IVI-COM Driver Reference. From the Windows Start menu, click **Programs -> IVI -> DT8824 -> Documentation**. 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 DT8824 instrument modules without writing MATLAB scripts; the tool generates the MATLAB script for you. See [www.mathworks.com/tmtool](http://www.mathworks.com/tmtool) for more information.

---

## Create an Object for the DT8824

Create an object for the DT8824 instrument module using the following command:

```
>> devObj = icdevice('DT8824_DT8824', devRsrcName);
```

where, *RsrcName* is an IVI logical name or an instrument-specific string, such as a VISA resource descriptor, that identifies the address of the DT8824 instrument module, and *dev* is the object that is created.

Specify `TCPIP::Address::INSTR` for *RsrcName*, where *Address* is the IP address of the instrument on the network. An example follows:

```
>> dev = icdevice('DT8824_DT8824',  
    '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 DT8824 CD. To use the Eureka Discovery Utility, perform the following steps: From the Windows Start menu, click **Programs -> Data Translation, Inc -> Instruments -> Eureka LXI Instrument Discovery**.

## Examples

There are three example programs available for download with the DT8824 MATLAB driver:

1. This example demonstrates how to synchronize the trigger on two devices using the trigger bus and perform continuous analog input on both devices.
2. This example demonstrates how to perform analog input measurements.
3. This example demonstrates how to write a value to the digital output port and verify the value written by reading back the value from the same port.

The following example shows how to access a DT8824 instrument from the MATLAB Instrument Control Toolbox using the DT8824 IVI-COM driver:

```
%% Analog input synchronization using Trigger Bus on Two DT8824
%% instruments
%
% Copyright (C) 2010 DataTranslation Inc.
%

%% Introduction
% This example demonstrates how to synchronize the trigger on two
% devices using the trigger bus and perform continuous analog input on
% both devices.

%% Create a device object for each 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.

masterDevRsrcName = 'TCPIP::192.43.218.135::INSTR';
slaveDevRsrcName = 'TCPIP::192.43.218.136::INSTR';

masterDeviceObj = icdevice('DT8824_DT8824.mdd', masterDevRsrcName);
slaveDeviceObj = icdevice('DT8824_DT8824.mdd', slaveDevRsrcName);

try
    %% Connect the device object to the instrument

    connect(masterDeviceObj);
    connect(slaveDeviceObj);

    %% Get the identity for each instrument

    % Get master instrument identity
    comobj = get(masterDeviceObj, '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);

% Get slave instrument identity
comobj = get(slaveDeviceObj, '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);

%% Configure the master instrument

% Enable analog input channel 1 for scanning
set(masterDeviceObj.AnalogInputChannel(1), 'Enabled', 'on');

```

```

    % Disable wrapping, so that data does not get overwritten in the
    % hardware FIFO when the FIFO is full
    set(masterDeviceObj.Analoginputacquisition(1), 'WrapEnabled',
'off');

    % Set reference clock source to internal
    set(masterDeviceObj.Referenceoscillator(1),
'ReferenceClockSource', 'DT8824ReferenceOscillatorSourceInternal');

    % Enable the master instrument to drive the reference clock on the
    % trigger bus
    set(masterDeviceObj.Referenceoscillator(1), 'OutputEnabled',
'on');

    % Set the clock frequency
    set(masterDeviceObj.Analoginputacquisition(1), 'SampleRate',
1000);

    % Set the trigger source to start streaming when the instrument
    % receives the Initiate command
    set(masterDeviceObj.Analoginputtrigger(1), 'Source',
'IMMEDIATE');

    % Set the master trigger bus line1 (LXI0) state to be driven. This
    % MUST be done prior to setting the slave trigger source to avoid
    % false triggers.
    set(masterDeviceObj.Triggerbusline(1), 'Enabled', 'on');

    %% Configure the slave instrument

    % Enable channel 1 for scanning
    set(slaveDeviceObj.Analoginputchannel(1), 'Enabled', 'on');

    % Disable wrapping, so that data does not get overwritten in the
    % hardware FIFO when the FIFO is full
    set(slaveDeviceObj.Analoginputacquisition(1), 'WrapEnabled',
'off');

    % Set the clock source to LXI line 7 on the trigger bus
    set(slaveDeviceObj.Referenceoscillator(1),
'ReferenceClockSource', 'DT8824ReferenceOscillatorSourceLXI7');

    % Set the clock frequency
    set(slaveDeviceObj.Analoginputacquisition(1), 'SampleRate',
1000);

    % Set the slave trigger source to the value that corresponds to
    % masterDeviceObj.Triggerbusline(1) where
    % masterDeviceObj.Triggerbusline(1) = 'LXI0'

```

```

% masterDeviceObj.Triggerbusline(2) = 'LXI1'
% masterDeviceObj.Triggerbusline(3) = 'LXI2'
% masterDeviceObj.Triggerbusline(4) = 'LXI3'
% masterDeviceObj.Triggerbusline(5) = 'LXI4'
% masterDeviceObj.Triggerbusline(6) = 'LXI5'
set(slaveDeviceObj.Analoginputtrigger(1), 'Source', 'LXI0');

%% Initiate the acquisition on the slave instrument.

% Now, the slave leaves the idle state and waits for the trigger on
% LXI0 trigger bus line
groupObj = get(slaveDeviceObj, 'Analoginputacquisition');
slaveAnalogInputAcquisitionObj = groupObj(1);
invoke(slaveAnalogInputAcquisitionObj, 'Arm');
invoke(slaveAnalogInputAcquisitionObj, 'Initiate');

%% Initiate the acquisition on the master instrument.

% Now, the slave leaves the 'waiting for trigger' state and starts
% acquiring data
masterGroupObj = get(masterDeviceObj, 'Analoginputacquisition');
masterAnalogInputAcquisitionObj = masterGroupObj(1);
invoke(masterAnalogInputAcquisitionObj, 'Arm');
invoke(masterAnalogInputAcquisitionObj, 'Initiate');

isRunning = false;
ScanIndex = 0;
RequestedScansToRead = 100;

%% Process data for the master instrument

while (isRunning == false || ScanIndex < (RequestedScansToRead-1))
    [ScanIndex, isRunning, isArmed, isTriggered, isADSyncDetected,
    isADFifoOverflow] = invoke(masterAnalogInputAcquisitionObj,
    'GetStatus', 0, 0, 0, 0, 0, 0);
end

% Read 100 scans from the master instrument
RequestedScansIndex = 0;
[ActualScansIndex, ActualScansRead, StartTimeInSeconds,
StartTimeInMilliseconds, masterSamples] =
invoke(masterAnalogInputAcquisitionObj, 'Fetch',
int32(RequestedScansIndex), int32(RequestedScansToRead), int32(0),
int32(0), int32(0), int32(0), [0;0]);

```

```

        RequestedScansIndex = ActualScansIndex+ActualScansRead;
        disp(['Master: Actual Scans Index: ', num2str(ActualScansIndex) , '
Actual Scans Read: ', num2str(ActualScansRead)]);
        disp(masterSamples);

        %% Process data for the slave instrument

        isRunning = false;
        ScanIndex = 0;

        while (isRunning == false && ScanIndex < (RequestedScansToRead-1))
            [ScanIndex, isRunning, isArmed, isTriggered, isADSyncDetected,
            isADFifoOverflow] = invoke(slaveAnalogInputAcquisitionObj,
            'GetStatus', 0, 0, 0, 0, 0, 0, 0);
        end

        % Read 100 scans from the slave
        RequestedScansIndex = 0;
        [ActualScansIndex, ActualScansRead, StartTimeInSeconds,
        StartTimeInMilliseconds, slaveSamples] =
        invoke(slaveAnalogInputAcquisitionObj, 'Fetch',
        int32(RequestedScansIndex), int32(RequestedScansToRead), int32(0),
        int32(0), int32(0), int32(0), [0;0]);

        RequestedScansIndex = ActualScansIndex+ActualScansRead;
        disp(['Master: Actual Scans Index: ', num2str(ActualScansIndex) , '
Actual Scans Read: ', num2str(ActualScansRead)]);

        %% Plot the data

        disp(slaveSamples);
        plot(masterSamples);
        hold on;
        plot(slaveSamples, 'green');
        hold off

        %% Stop the acquisition

        invoke(masterAnalogInputAcquisitionObj, 'Abort');
        invoke(slaveAnalogInputAcquisitionObj, 'Abort');

catch DT8824error
    disp(['Error id: ', DT8824error.identifier]);
    disp(['Error Message: ', DT8824error.message]);
end

%% Disconnect the device objects from the instruments and remove them
%% from memory

```



```
disconnect(masterDeviceObj);  
delete(masterDeviceObj);  
  
disconnect(slaveDeviceObj);  
delete(slaveDeviceObj);
```

## ***Conclusion***

To access DT8824 instruments from MATLAB, you need the MATLAB Instrument Control Toolbox, DT8824 IVI-COM driver, and MATLAB driver for DT8824. By combining the precise measurement capability of Data Translation with the powerful analytical capability of MATLAB, developers can create robust applications that solve difficult temperature measurement problems with ease.

## ***For More Information***

1. Overview of DT8824 Instrument Module:  
[www.datatranslation.com/products/dataacquisition/ethernet/DT8824.asp](http://www.datatranslation.com/products/dataacquisition/ethernet/DT8824.asp)
2. Overview of MATLAB software:  
[www.mathworks.com/matlab](http://www.mathworks.com/matlab)
3. Overview of MATLAB Instrument Control Toolbox:  
[www.mathworks.com/products/instrument](http://www.mathworks.com/products/instrument)

MATLAB® is a registered trademark of The MathWorks, Inc.