

Instrument Control Toolbox™ Release Notes



MATLAB® & SIMULINK®



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Instrument Control Toolbox™ Release Notes

© COPYRIGHT 2005–2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

R2022a

New interface for connection to instruments using IVI and VXIplug&play drivers	1-2
New UDP Explorer app	1-2
TCP/IP Client Interface: Specify transfer delay options	1-3
Serial blocks: New parameter to specify input format	1-3
Modbus functionality moved to Industrial Communication Toolbox	1-3
Functionality being removed or changed	1-4
Instrument driver functions will be removed	1-4
IVI-C class-compliant wrappers will be removed	1-4
Instrument object functions will be removed	1-4
visadev interface no longer supports callback functions	1-5
serial function will be removed	1-5
Bluetooth function will be removed	1-5
tcpip function will be removed	1-6
udp function will be removed	1-6
visa function will be removed	1-6
gpib function will be removed	1-7

R2021b

New Serial Explorer and TCP/IP Explorer apps	2-2
TCP/IP Client Interface: Simulink blocks have expanded functionality	2-2
Functionality being removed or changed	2-2
gpib function will be removed	2-2

R2021a

TCP/IP Server Interface: New functions and properties	3-2
VISA Interface: New functions and properties	3-2
Serial Interface: Simulink blocks have expanded functionality	3-2
Functionality being removed or changed	3-3
tcpip function will be removed	3-3
visa function will be removed	3-3
serial function will be removed	3-3
Parity parameter no longer supports mark or space in Serial Configuration block	3-4

R2020b

Bluetooth Interface: New functions and properties with improved performance	4-2
Performance Improvements	4-2
TCP/IP Client Interface: New functions and properties with improved performance	4-4
Performance Improvements	4-4
UDP Interface: New functions and properties with improved performance	4-5
Performance Improvements	4-5
Serial Port Interface: Improved performance	4-6
Functionality being removed or changed	4-6
Bluetooth function will be removed	4-6
tcpip function will be removed	4-7
udp function will be removed	4-7

R2020a

Generate C and C++ Code with MODBUS Objects	5-2
--	------------

R2019b

Serial Port Interface: New functions and properties	6-2
Quick-Control Interfaces: Return and set oscilloscope probe attenuation value	6-2
Functionality being removed or changed	6-2
serialist function is not recommended	6-2
serial function is not recommended	6-2

R2019a

Modbus Explorer App	7-2
Aardvark Adaptor Not Available for Mac for R2019a	7-2

R2018b

Keysight Instruments: Improved control and ease of use with updated drivers from Keysight	8-2
Instrument Control Toolbox Support Package for Ocean Optics Spectrometers Is Being Removed	8-2
Aardvark Adaptor Not Available for Mac	8-2

R2018a

VISA Support: Communicate with instruments using Rohde & Schwarz R&S VISA	9-2
UDP and TCP/IP Communication: Send and receive data with more flexibility in Simulink using the expanded configuration options found in the UDP and TCP/IP client blocks	9-2
Keysight Vendor Name in Code	9-3
Enhanced warnings and troubleshooting information for read warning messages	9-3

Aardvark Adaptor Not Available for Mac for R2018a	9-3
--	------------

R2017b

Quick Control RF Signal Generator Interface: Use a simple interface to easily download signals and control RF signal generators	10-2
GPIB Support for ADLINK: Communicate with instruments using ADLINK GPIB interface hardware	10-2

R2017a

MODBUS Protocol Support: Communicate with hardware using the MODBUS protocol over TCP/IP and Serial	11-2
Serial discovery function	11-2

R2016b

Hardware Support: Connect to GPIB instruments using Measurement Computing interface hardware	12-2
Ocean Optics Spectrometers Support Package supported on Mac	12-2
Advanced tab completion for functions	12-2
Port sharing for UDP interface	12-4
Support Package Installation via Add-Ons Explorer: Install, manage, and update support packages	12-4

R2016a

Hardware Support: Acquire and analyze data from newly supported National Instruments devices	13-2
Bluetooth Support: Use Bluetooth devices that support Serial Port Profile on Mac OS X	13-2

Hardware Support: Connect to instruments with IVI-C drivers using examples for the 13 IVI instrument classes	13-2
Removal of support for IVI-COM drivers and some GPIB adaptors	13-2
Functionality Being Removed or Changed	13-3

R2015b

Support for NI VISA on the Mac platform	14-2
New support package for Keysight IO Libraries and VISA Interface ...	14-2
Removal of 32-bit MATLAB and IVI-COM	14-2
Troubleshooting Guide Added to Documentation in R2015a	14-2

R2015a

New Support Package for National Instruments VISA and ICP interfaces	15-2
Troubleshooting Guide Added to Documentation	15-2

R2014b

SPI support for National Instruments NI-845x devices	16-2
Support for National Instruments NI-Switch devices via the Support Package Installer	16-2
Support Package for Aardvark I2C/SPI Interface	16-2

R2014a

Support for National Instruments NI-8451 and NI-8452 I2C interface via the Support Package Installer	17-2
---	-------------

Support for National Instruments NI-DCPwr devices via the Support Package Installer	17-2
Support for National Instruments NI-DMM devices via the Support Package Installer	17-2

R2013b

SPI protocol support using Total Phase Aardvark SPI interface hardware	18-2
I2C communication protocol now available in the Instrument Control app	18-2
Support for Ocean Optics spectrometers via the Support Package Installer	18-2
Support for NI-SCOPE oscilloscopes via the Support Package Installer	18-2
Support for NI-FGEN function generators via the Support Package Installer	18-3

R2013a

Windows 64-bit support for Tektronix VISA	19-2
Windows 64-bit support for ICS Electronics GPIB interface devices ...	19-2
Windows 64-bit support for NI-SCOPE and NI-FGEN modular instruments	19-2
Support for 10 additional NI-SCOPE devices	19-2

R2012b

Bluetooth support in Test & Measurement Tool	20-2
Quick-Control Function Generator now supports Agilent instruments	20-2

R2012a

I2C Support	21-2
Quick-Control Oscilloscope Now Supports Tektronix Scopes	21-2
Quick-Control Function Generator	21-2
New IVI Class Supported	21-2
IVI-C Class Compliant Wrappers in Test & Measurement Tool	21-3

R2011b

Bluetooth Support	22-2
Quick-Control Oscilloscope	22-2
Generic VISA Support	22-2
VISA Node and Generic VISA Support in the Test & Measurement Tool	22-2
Conversion of Error and Warning Message Identifiers	22-3

R2011a

Support for Three New IVI Classes	23-2
IVI-C Class Compliant Wrapper	23-2
Server Sockets Support	23-2
Known Issue - TCP/IP Endian Default	23-2

R2010b

Support for IVI Class for Digitizers	24-2
Support for IVI Class for Switches	24-2

Support for National Instruments NI-SCOPE Driver Software	24-2
Support for National Instruments NI-FGEN Driver Software	24-2
Enhanced TCP/IP Performance for fread and binblockread Functions	24-2
Two New UDP Packet Size Properties	24-2

R2010a

Expanded spoll Function	25-2
Enhanced Test and Measurement Tool	25-2
Support for Agilent 64-Bit GPIB on Windows 64	25-2

R2009b

New Class-Compliant Interface for IVI-COM Drivers	26-2
Support for Agilent 64-Bit VISA on Windows 64	26-2
Warning Added for Future Deprecation of UDP Binblock Support	26-2

R2009a

Expanded Platform Support Added for Instrument Control Toolbox ...	27-2
Using the Status Function with an IVI Driver Can Cause Namespace Conflicts	27-2
Expanded Platform Support Added for NI-VISA and GPIB Interfaces ..	27-2
IVI-COM Driver Support	27-2

R2008b

Enhanced IVI-COM Driver Support	28-2
Enhanced Functionality of the Test and Measurement Tool	28-3
Enhanced IVI-C Driver Support	28-3
Expanded Instrument Control Toolbox Demos	28-3
Expanded spoll Documentation	28-3

R2008a

Instrument Control Toolbox Block Library	29-2
Improved Throughput	29-2
Instrument Drivers	29-2
Enhanced Capability of the TM Tool	29-2
Support for Agilent IO Libraries Suite 15.0	29-2
fprintf Timeout Errors	29-2

R2007b

Instrument Control Toolbox Block Library	30-2
Instrument Drivers	30-2
spoll GPIB Status Information	30-2
Obsolete Functions	30-2
New Error Message	30-2

R2007a

Confirmation Dialog Preference Settings Moved	31-2
--	-------------

R2006b

No New Features or Changes

R2006a

Generic Instrument Drivers Supported	33-2
LeCroy Driver	33-2
Agilent Instrument Simulations	33-2

R14SP3

No New Features or Changes

R14SP2

Advantech GPIB Supported	35-2
---------------------------------------	-------------

R2022a

Version: 4.6

New Features

Bug Fixes

Version History

New interface for connection to instruments using IVI and VXIplug&play drivers

Connect to and communicate with instruments including oscilloscopes, function generators, signal generators, and spectrum analyzers using industry-standard IVI® and VXI*plug&play* drivers from instrument vendors such as Keysight™ and Tektronix®. Support for these drivers is available through the Instrument Control Toolbox Support Package for IVI and VXIplug&play Drivers.

To use the new interface, you must first install the IVI Shared Components, IVI Compliance Package, and LabWindows™/CVI from the IVI and NI™ websites. Then, download the Instrument Control Toolbox Support Package for IVI and VXIplug&play Drivers from the MATLAB® Add-On Explorer.

Get started with the new interface by viewing a list of installed drivers using `ividriverlist` and connected hardware using `ividevlist`. Then, use the `ividev` function to create a connection to your instrument. You can explore the instrument's attributes and capabilities by clicking the links from the object output display.

For more information about the Instrument Control Toolbox Support Package for IVI and VXIplug&play Drivers, see “IVI and VXIplug&play Drivers”.

Version History

The instrument driver functions `makemid`, `midedit`, and `midtest` will be removed in a future release. For IVI and VXI*plug&play* drivers created with `makemid`, use `ividev` with one of the supported drivers in the Instrument Control Toolbox Support Package for IVI and VXIplug&play Drivers. The apps launched by `midedit` and `midtest` are not needed for the `ividev` interface.

The following IVI-C class-compliant wrapper interfaces will be removed in a future release:

- `instrument.ivic.IviDCPwr`
- `instrument.ivic.IviDmm`
- `instrument.ivic.IviFgen`
- `instrument.ivic.IviPwrMeter`
- `instrument.ivic.IviRFSigGen`
- `instrument.ivic.IviSpecAn`
- `instrument.ivic.IviSwtch`
- `instrument.ivic.IviScope`

To connect to IVI-C class-compliant instruments, use `ividev` instead.

New UDP Explorer app

The **UDP Explorer** app provides a user interface to create a UDP socket and communicate over networks using UDP protocol.

Launch this app from the **Apps** tab, under the **Test and Measurement** section. You can also call the `udpExplorer` command in the Command Window.

You can use this app to perform the following operations on your UDP socket.

-
- Configure byte and datagram connection and communication properties.
 - Write binary or ASCII-terminated string data.
 - Read binary or ASCII-terminated string data.
 - Receive and read multicast data.
 - Broadcast data over a network.
 - Plot data in a separate figure window.
 - Analyze data by viewing it in the **Signal Analyzer** app.
 - Export data to the MATLAB workspace.
 - Generate a MATLAB script for app interactions that uses the `udpport` interface.

For more information about this app, see **UDP Explorer**.

TCP/IP Client Interface: Specify transfer delay options

You can now enable or disable a transfer delay to allow delayed acknowledgement from the connected server for `tcpclient` objects and in the **TCP/IP Explorer** app. The transfer delay is enabled by default. Enabling the delay turns on Nagle's algorithm, which causes the client to collect small segments of outstanding data and send them in a single packet when acknowledgement (ACK) arrives from the server. Disabling it turns off Nagle's algorithm, which immediately sends data to the network.

For the `tcpclient` interface, you can set the `EnableTransferDelay` property as a name-value argument during object creation. For **TCP/IP Explorer**, you can select **Transfer Delay** options during connection configuration.

For more information about this functionality, see `EnableTransferDelay` and “Configure Connection in TCP/IP Explorer”.

Serial blocks: New parameter to specify input format

The Serial Receive block has a new parameter **Input format** to specify the format of data read as column major or row major.

For more information about this parameter, see `Input Format`.

Modbus functionality moved to Industrial Communication Toolbox

The `modbus` interface and the **Modbus Explorer** app are now in Industrial Communication Toolbox™ in R2022a. You can still use all the same functionality if you have Industrial Communication Toolbox installed.

Version History

To continue using Modbus functionality in MATLAB, you must install Industrial Communication Toolbox. If you have been a licensed Instrument Control Toolbox user prior to this release, you might be eligible to continue using Modbus functionality as described in `Opt-In Offer for Instrument Control Toolbox Modbus Users`.

Functionality being removed or changed

Instrument driver functions will be removed

Still runs

The instrument driver functions `makemid`, `midedit`, and `midtest` will be removed in a future release. Although these functions do not have any direct replacement functionality, their workflows are supported by the `ividev` interface.

For existing SCPI-based MATLAB instrument drivers created with `makemid`, the `icdevice` function still works. For IVI and `VXIplug&play` drivers, use `ividev` with one of the supported drivers in the Instrument Control Toolbox Support Package for IVI and `VXIplug&play` Drivers.

The `midedit` and `midtest` functions launch apps to edit and test MATLAB instrument drivers. For existing instrument driver files, you can use a text editor to make edits, if necessary. These apps and their functionalities are not needed for the `ividev` interface.

For more information about the Instrument Control Toolbox Support Package for IVI and `VXIplug&play` Drivers, see “IVI and `VXIplug&play` Drivers”.

IVI-C class-compliant wrappers will be removed

Still runs

The following IVI-C class-compliant wrapper interfaces will be removed in a future release:

- `instrument.ivic.IviDCPwr`
- `instrument.ivic.IviDmm`
- `instrument.ivic.IviFgen`
- `instrument.ivic.IviPwrMeter`
- `instrument.ivic.IviRFSigGen`
- `instrument.ivic.IviSpecAn`
- `instrument.ivic.IviSwth`
- `instrument.ivic.IviScope`

To connect to IVI-C class-compliant instruments, use the `ividev` function in the Instrument Control Toolbox Support Package for IVI and `VXIplug&play` Drivers instead.

This example shows how to connect to an IVI-C class-compliant instrument using the recommended functionality.

Functionality	Use This Instead
<pre>% myScope is the logical name in the iviconfiguration ivicscope = instrument.ivic.IviScope(); ivicscope.init('myScope',true,true);</pre>	<pre>ivicscope = ividev('IviScope','myScope','IDQuery',true);</pre>

For more information about the Instrument Control Toolbox Support Package for IVI and `VXIplug&play` Drivers, see “IVI and `VXIplug&play` Drivers”.

Instrument object functions will be removed

Still runs

The instrument object functions `instrfind`, `instrfindall`, `instrreset`, `instrnotify`, `instrhelp`, and `instrcallback` will be removed in a future release. These functions do not have any direct replacement functionality.

The functions being removed are used by the `serial`, `Bluetooth`, `tcpip`, `udp`, `visa`, `gpib` interfaces. These interfaces will also be removed in a future release, as previously announced. Use the following interfaces and their functions instead.

Functionality	Use This Instead
<code>serial</code>	<code>serialport</code>
<code>Bluetooth</code>	<code>bluetooth</code> (case-sensitive)
<code>tcpip</code>	<code>tcpclient</code> or <code>tcpserver</code>
<code>udp</code>	<code>udpport</code>
<code>visa</code>	<code>visadev</code>
<code>gpib</code>	VISA-GPIB interface with <code>visadev</code>

The recommended interfaces have additional capabilities and improved performance. For more information about the recommended interfaces, see “Interface-Based Instrument Communication”.

visadev interface no longer supports callback functions

Behavior change

Starting in R2022a, callback functions are no longer supported for the `visadev` interface. The `configureCallback` function and the `NumBytesAvailable`, `BytesAvailableFcnMode`, `BytesAvailableFcnCount`, and `BytesAvailableFcn` properties and are no longer supported for `visadev` objects.

serial function will be removed

Warns

`serial` and its object properties will be removed in a future release. Use `serialport` and its properties instead.

This example shows how to connect to a serial port device using the recommended functionality.

Functionality	Use This Instead
<pre>s = serial("COM1"); s.BaudRate = 115200; fopen(s)</pre>	<pre>s = serialport("COM1",115200);</pre>

For more information about updating your code to use the recommended functionality, see “Transition Your Code to serialport Interface”.

Bluetooth function will be removed

Warns

`Bluetooth` and its object properties will be removed in a future release. Use `bluetooth` and its properties instead.

This example shows how to connect to a Bluetooth® device using the recommended functionality.

Functionality	Use This Instead
<code>b = Bluetooth("NXT",3); fopen(b)</code>	<code>b = bluetooth("NXT",3);</code>

For more information about updating your code to use the recommended functionality, see “Transition Your Code to bluetooth Interface”.

tcpip function will be removed

Warns

`tcpip` and its object properties will be removed in a future release. Use `tcpclient` or `tcpserver` and its properties instead.

This example shows how to connect to a TCP/IP client using the recommended functionality.

Functionality	Use This Instead
<code>t = tcpip("127.0.0.1",3030,"NetworkRole","client"); fopen(t)</code>	<code>tcpclient("127.0.0.1",3030);</code>

For more information about updating your code to use the recommended functionality, see “Transition Your Code to tcpclient Interface”.

This example shows how to create a TCP/IP server using the recommended functionality.

Functionality	Use This Instead
<code>t = tcpip("192.168.2.15",3030,"NetworkRole","server"); fopen(t)</code> This binds to host "0.0.0.0" (internally) and port 3030 and only accepts client connections coming from "192.168.2.15". MATLAB is blocked until a client connection is established.	<code>t = tcpserver("0.0.0.0",3030);</code> This binds to "0.0.0.0" and port 3030. "0.0.0.0" means that it accepts any incoming client connection requests on port 3030. MATLAB is not blocked.

For more information about updating your code to use the recommended functionality, see “Transition Your Code to tcpserver Interface”.

udp function will be removed

Warns

`udp` and its object properties will be removed in a future release. Use `udpport` and its properties instead.

This example shows how to connect to UDP using the recommended functionality.

Functionality	Use This Instead
<code>u = udp; fopen(u)</code>	<code>u = udpport("byte");</code> <code>u = udpport("datagram");</code>

For more information about updating your code to use the recommended functionality, see “Transition Your Code to udpport Interface”.

visa function will be removed

Warns

`visa` and its object properties will be removed in a future release. Use `visadev` and its properties instead.

This example shows how to connect to a VISA device using the recommended functionality.

Functionality	Use This Instead
<pre>v = visa("ni", "GPIB0::1::0::INSTR"); fopen(v)</pre>	<pre>v = visadev("GPIB0::1::0::INSTR");</pre>

For more information about updating your code to use the recommended functionality, see “Transition Your Code to `visadev` Interface”.

gpib function will be removed

Warns

`gpib` and its object properties will be removed in a future release. Use the VISA-GPIB interface with `visadev` and its properties instead.

This example shows how to connect to a GPIB instrument using the recommended functionality.

Functionality	Use This Instead
<pre>g = gpib('ni', 0, 1, 'SecondaryAddress', 0); fopen(g)</pre>	<pre>v = visadev("GPIB0::1::0::INSTR");</pre>

For more information about updating your code to use the recommended functionality, see “Transition Your Code to VISA-GPIB Interface”.

R2021b

Version: 4.5

New Features

Bug Fixes

Version History

New Serial Explorer and TCP/IP Explorer apps

Two new apps offer functionality for communicating with your device, instrument, or server:

- The **Serial Explorer** app provides a user interface to connect to and communicate with a serial port device on your machine.
- The **TCP/IP Explorer** app provides a user interface to create a TCP/IP client that communicates with a TCP/IP server.

Launch these apps from the **Apps** tab, under the **Test and Measurement** section. You can also call the `serialExplorer` and `tcpipExplorer` commands in the Command Window.

You can use the apps to perform the following operations on your serial port device or TCP/IP client.

- Configure connection and communication properties.
- Write binary or string data.
- Read binary or string data.
- Plot data in a separate figure window.
- Analyze data by viewing it in the **Signal Analyzer** app.
- Export data to the MATLAB workspace.
- Generate a MATLAB script for app interactions that uses the `serialport` or `tcpclient` interface.

For more information about these apps, see **Serial Explorer** and **TCP/IP Explorer**.

TCP/IP Client Interface: Simulink blocks have expanded functionality

Use the TCP/IP Client Simulink® blocks TCP/IP Receive and TCP/IP Send to receive and send binary and ASCII data to a remote host using TCP/IP communication. In addition to the existing functionality, you can now perform the following with these blocks:

- Generate C/C++ code using Simulink Coder™.
- Specify the **Source Data type** parameter of the TCP/IP Receive block as `int64` or `uint64`.
- Specify the **Custom Terminator** parameter of the TCP/IP Receive block as a numeric vector, which can be an ASCII special character.

For more information about these blocks, see TCP/IP Receive and TCP/IP Send.

Functionality being removed or changed

gpiib function will be removed

Still runs

`gpiib` and its object properties will be removed. Use the VISA-GPIB interface with `visadev` and its properties instead.

This example shows how to connect to a GPIB instrument using the recommended functionality.

Functionality	Use This Instead
<pre>g = gpib('ni',0,1,'SecondaryAddress',0); fopen(g)</pre>	<pre>v = visadev("GPIB0::1::0::INSTR");</pre>

For more information about updating your code to use the recommended functionality, see [Transition Your Code to VISA-GPIB Interface](#).

R2021a

Version: 4.4

New Features

Bug Fixes

Version History

TCP/IP Server Interface: New functions and properties

The TCP/IP server interface has a new set of functions and properties. The existing functionality still runs, but will be removed in a future release. Using the new functions and properties is recommended.

Get started with the new interface by creating a `tcpserver` object that connects to a TCP/IP client. Then, send data to the client using the `server` object and read the data from the `client` object.

```
server = tcpserver(4000);
client = tcpclient("localhost",4000);
write(server,1:5,"uint8");
read(client,5);
```

For more information, see TCP/IP Interface.

Version History

For more information about updating your code to use the recommended functionality, see Transition Your Code to `tcpserver` Interface.

VISA Interface: New functions and properties

The VISA interface has a new set of functions and properties. The existing functionality still runs, but will be removed in a future release. Using the new functions and properties is recommended.

Get started with the new interface by viewing a list of all available VISA resources on your computer using `visadevlist`. Then, create a `visadev` object, write data to the resource, and read from it.

```
resourceList = visadevlist;
v = visadev("COM1");
write(v,1:5);
read(v,5);
```

For more information, see VISA Interface (Includes VXI, PXI, USB).

Version History

For more information about updating your code to use the recommended functionality, see Transition Your Code to `visadev` Interface.

Serial Interface: Simulink blocks have expanded functionality

Use the serial Simulink blocks Serial Configuration, Serial Send, and Serial Receive to send and receive binary data over a serial port. In addition to the existing functionality, you can now perform the following with these blocks:

- Generate C/C++ code using Simulink Coder.
- Specify the **Header** and **Custom Terminator** parameters of the Serial Send and Serial Receive blocks as special characters.

For more information about these blocks, see Serial Configuration, Serial Receive, and Serial Send.

Version History

The mark and space options for the **Parity** parameter are no longer supported in the Serial Configuration block. Valid values for **Parity** are none (default), even, and odd.

Functionality being removed or changed

tcpip function will be removed

Still runs

tcpip with the NetworkRole property set to 'server' and its object properties will be removed. Use tcpserver and its properties instead.

This example shows how to create a TCP/IP server using the recommended functionality.

Functionality	Use This Instead
<pre>t = tcpip("192.168.2.15",3030,"NetworkRole" fopen(t)</pre> <p>This binds to host "0.0.0.0" (internally) and port 3030 and only accepts client connections coming from "192.168.2.15". MATLAB is blocked until a client connection is established.</p>	<pre>t = tcpserver("0.0.0.0",3030);</pre> <p>This binds to "0.0.0.0" and port 3030. "0.0.0.0" means that it accepts any incoming client connection requests on port 3030. MATLAB is not blocked.</p>

For more information about updating your code to use the recommended functionality, see Transition Your Code to tcpserver Interface.

visa function will be removed

Still runs

visa and its object properties will be removed. Use visadev and its properties instead.

This example shows how to connect to a VISA device using the recommended functionality.

Functionality	Use This Instead
<pre>v = visa("ni","GPIB0::1::0::INSTR"); fopen(v)</pre>	<pre>v = visadev("GPIB0::1::0::INSTR");</pre>

For more information about updating your code to use the recommended functionality, see Transition Your Code to visadev Interface.

serial function will be removed

Still runs

serial and its object properties will be removed. Previously, serial and its object properties were not recommended. Use serialport and its properties instead.

This example shows how to connect to a serial port device using the recommended functionality.

Functionality	Use This Instead
<pre>s = serial("COM1"); s.BaudRate = 115200; fopen(s)</pre>	<pre>s = serialport("COM1",115200);</pre>

For more information about updating your code to use the recommended functionality, see [Transition Your Code to serialport Interface](#).

Parity parameter no longer supports mark or space in Serial Configuration block

Errors

The **mark** and **space** options for the **Parity** parameter are no longer supported in the Serial Configuration block. Valid values for **Parity** are **none** (default), **even**, and **odd**.

If you try to open an existing model that has the **Parity** value set to **mark** or **space**, MATLAB returns a warning and changes the parameter to the default value **none**.

R2020b

Version: 4.3

New Features

Bug Fixes

Version History

Bluetooth Interface: New functions and properties with improved performance

A new set of MATLAB functions and properties provides support for communicating with Bluetooth devices. The existing functionality still runs, but will be removed in a future release. Using the new functions and properties is recommended. You can use the new interface without Instrument Control Toolbox.

Get started with the new interface by viewing a list of all Bluetooth devices on your computer using `bluetoothlist`. In this example, an HC-06 Bluetooth module is paired to the computer.

```
list = bluetoothlist
```

```
list=1x4 table
      Name          Address      Channel      Status
      _____  _____  _____  _____
      "HC-06"      "98D331FB3B77"      1      "Ready to connect"
```

Then, create a `bluetooth` object, write data to the device, and read from it. In this example, the HC-06 is configured as a loopback device.

```
b = bluetooth("HC-06");
write(b,1:10);
read(b,10);
```

For more information, see Bluetooth Communication (MATLAB).

Performance Improvements

The new `bluetooth` interface shows faster connection times compared to the existing Bluetooth interface. For example, this code for connecting and disconnecting a Bluetooth device with the new `bluetooth` object is about 8.4x faster than the code for connecting and disconnecting a Bluetooth device with the existing Bluetooth object on a Windows® test system and about 8.3x faster on a macOS system.

```
function timingTest
    b = Bluetooth("SPPDemo", 6);
    fopen(b);
    fclose(b);
    delete(b);
end

function timingTest
    bluetooth("SPPDemo", 6);
end
```

The approximate execution times on Windows are:

- Bluetooth: 13.14 s
- `bluetooth`: 1.57 s

The approximate execution times on macOS are:

- Bluetooth: 21.23 s

-
- `bluetooth`: 2.55 s

The code was timed on a Windows 10, Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60 GHz test system and a macOS 10.14.3 (Mojave), Mac mini Intel Core i7 CPU @ 2.3 GHz test system using the `timeit` function:

```
timeit(@timingTest)
```

Both tests were done using an MSP-EXP432P401R LaunchPad kit with CC2564 Booster Pack module, programmed with the SPPDemo example and loopback enabled.

The new `bluetooth` interface also shows improved performance over the existing `Bluetooth` interface. For example, this code for writing and reading data with the new `bluetooth` object is about 6x faster than the code for writing and reading data with the existing `Bluetooth` object on a Windows test system and about 34x faster on a macOS system.

```
function timingTest
    loops = 10;
    count = 1000;
    data = ones(1, count);
    b = Bluetooth("SPPDemo", 6);
    b.InputBufferSize = count;
    b.OutputBufferSize = count;
    fopen(b);

    tic
    for k = 1:loops
        fwrite(b, data, "uint8");
        fread(b, count, "uint8");
    end
    toc/loops

    fclose(b);
    delete(b);
end
```

```
function timingTest
    loops = 10;
    count = 1000;
    data = ones(1, count);
    b = bluetooth("SPPDemo", 6);

    tic
    for k = 1:loops
        write(b, data, "uint8");
        read(b, count, "uint8");
    end
    toc/loops
end
```

The approximate execution times on Windows are:

- `Bluetooth`: 0.76 s
- `bluetooth`: 0.13 s

The approximate execution times on macOS are:

- Bluetooth: 1.7 s
- bluetooth: 0.05 s

The code was timed on a Windows 10, Intel(R) Xeon(R) W-2133 CPU @ 3.60 GHz test system and a macOS 10.14.3 (Mojave), Mac mini Intel Core i7 CPU @ 2.3 GHz test system by calling the function `timingTest`. Both tests were done using an MSP-EXP432P401R LaunchPad kit with a CC2564 Booster Pack module, programmed with the `SPPDemo` example and loopback enabled.

Version History

For more information about updating your code to use the recommended functionality, see [Transition Your Code to bluetooth Interface \(MATLAB\)](#).

TCP/IP Client Interface: New functions and properties with improved performance

The TCP/IP client interface has a new set of functions and properties. The existing functionality still runs, but will be removed in a future release. Using the new functions and properties is recommended.

Get started with the new interface by creating a `tcpclient` object connected to a TCP/IP echo server, writing data to it, and reading from it.

```
echotcpip("on",3030)
t = tcpclient("localhost",3030)
write(t,1:5,"uint8")
read(t,5);
```

For more information, see [TCP/IP Interface](#).

Performance Improvements

The `tcpclient` interface shows improved performance over the `tcpip` interface. For example, this code for writing and reading data with the new `tcpclient` object is about 1.5x faster than the code for writing and reading data with the existing `tcpip` object.

```
% t is a tcpip object
function timingTest(t,bytecount)
fwrite(t,1:bytecount,"uint8");
fread(t,bytecount,"uint8");
end
```

```
% t is a tcpclient object
function timingTest(t,bytecount)
write(t,1:bytecount,"uint8");
read(t,bytecount,"uint8");
end
```

The approximate execution times are:

- `tcpip`: 4.8 ms
- `tcpclient`: 2.9 ms

The code was timed on a Windows 10, Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function:

```
bytecount = 1000;
timeit(@()timingTest(t,bytecount))
```

Both tests were done using a TCP/IP echo server.

Version History

For more information about updating your code to use the recommended functionality, see [Transition Your Code to tcpclient Interface](#).

UDP Interface: New functions and properties with improved performance

A new set of MATLAB functions and properties supports communicating with UDP. The existing functionality still runs, but will be removed in a future release. Using the new functions and properties is recommended.

Get started with the new interface by creating a `udpport` object connected to a UDP echo server, writing data to it, and reading from it.

```
echoudp("on",8000)
u = udpport;
write(u,ones(1,1000),"uint8","127.0.0.1",8000)
read(u,1000,"uint8");
```

For more information, see [UDP Interface](#).

Performance Improvements

The `udpport` interface shows improved performance over the `udp` interface. For example, this code for writing and reading data with the new `udpport` object is about 5x faster than the code for writing and reading data with the existing `udp` object.

```
% u is a udp object
function timingTest(u,dataToWrite,bytecount)
fwrite(u,dataToWrite,"uint8");
fread(u,bytecount,"uint8");
end

% u is a udpport object
function timingTest(u,dataToWrite,bytecount)
write(u,dataToWrite,"uint8","127.0.0.1",8000);
read(u,bytecount,"uint8");
end
```

The approximate execution times are:

- `udp`: 8.3 ms
- `udpport`: 1.6 ms

The code was timed on a Windows 10, Intel(R) Xeon(R) W-2133 CPU @ 3.60 GHz test system using the `timeit` function:

```
dataToWrite = ones(1,1000);
bytecount = 1000;
timeit(@()timingTest(u,dataToWrite,bytecount))
```

Both tests were done using a UDP echo server.

Version History

For more information about updating your code to use the recommended functionality, see [Transition Your Code to udpport Interface](#).

Serial Port Interface: Improved performance

The `serialport` interface shows improved performance over the `serial` interface. For example, this code for writing and reading data with the `serialport` object is about 1.1x faster than the code for writing and reading data with the `serial` object with the default baud rate of 9600.

```
% s is a serial object
function timingTest(s,bytecount)
fwrite(s,1:bytecount,"uint8");
fread(s,bytecount,"uint8");
end

% s is a serialport object
function timingTest(s,bytecount)
write(s,1:bytecount,"uint8");
read(s,bytecount,"uint8");
end
```

The approximate execution times for different baud rates follow:

	s.BaudRate			
	9600	19200	56000	115200
<code>serial</code>	120 ms	68 ms	31 ms	23 ms
<code>serialport</code>	109 ms	55 ms	21 ms	11 ms

The code was timed on a Windows 10, Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function:

```
bytecount = 100;
timeit(@()timingTest(s,bytecount))
```

The tests were done using a serial loopback connector.

For more information, see [Serial Port Interface](#).

Functionality being removed or changed

Bluetooth function will be removed

Still runs

Bluetooth and its object properties will be removed. Use `bluetooth` and its properties instead.

This example shows how to connect to a Bluetooth device using the recommended functionality.

Functionality	Use This Instead
<code>b = Bluetooth("NXT",3); fopen(b)</code>	<code>b = bluetooth("NXT",3);</code>

See Transition Your Code to `bluetooth` Interface (MATLAB) for more information about using the recommended functionality.

tcpip function will be removed

Still runs

`tcpip` with the `NetworkRole` property set to 'client' and its object properties will be removed. Use `tcpclient` and its properties instead.

This example shows how to connect to a TCP/IP client using the recommended functionality.

Functionality	Use This Instead
<code>t = tcpip("127.0.0.1",3030,"NetworkRole","client"); fopen(t)</code>	<code>tcpclient("127.0.0.1",3030);</code>

See Transition Your Code to `tcpclient` Interface for more information about using the recommended functionality.

udp function will be removed

Still runs

`udp` and its object properties will be removed. Use `udpport` and its properties instead.

This example shows how to connect to UDP using the recommended functionality.

Functionality	Use This Instead
<code>u = udp; fopen(u)</code>	<code>u = udpport("byte");</code>
	<code>u = udpport("datagram");</code>

See Transition Your Code to `udpport` Interface for more information about using the recommended functionality.

R2020a

Version: 4.2

New Features

Bug Fixes

Generate C and C++ Code with MODBUS Objects

The `modbus`, `read`, `write`, `writeRead`, and `maskWrite` functions support C and C++ code generation using MATLAB Coder.

R2019b

Version: 4.1

New Features

Bug Fixes

Version History

Serial Port Interface: New functions and properties

The serial port interface has a new set of functions and properties. The existing functionality still runs, but new function names and properties are recommended. The new interface has improved performance.

Get started with the new interface by viewing a list of all serial ports on your computer using `serialportlist`.

```
list = serialportlist

list =
    1×4 string array
    "COM1"    "COM3"    "COM4"    "COM8"
```

Then, create a `serialport` object, write data to the device, and read from it.

```
s = serialport("COM8",115200);
write(s,1:5,"uint32")
read(s,5,"uint32");
```

For more information, see [Serial Port Interface](#).

Version History

For more information about updating your code to use the recommended functionality, see [Transition Your Code to serialport Interface](#).

Quick-Control Interfaces: Return and set oscilloscope probe attenuation value

The `configureChannel` function now accepts `'ProbeAttenuation'` as an input argument or as a name-value pair parameter name. Use `'ProbeAttenuation'` to return or set the value of an oscilloscope's probe attenuation.

```
value = configureChannel(myScope,'Channel1','ProbeAttenuation')
configureChannel(myScope,'Channel1','ProbeAttenuation',100)
```

Functionality being removed or changed

serialist function is not recommended

Still runs

`serialist` is not recommended. Use `serialportlist` instead.

See [Transition Your Code to serialport Interface](#) for more information about using the recommended functionality.

serial function is not recommended

Still runs

`serial` and its object properties are not recommended. Use `serialport` and its properties instead.

This example shows how to connect to a serial port device using the recommended functionality.

Functionality	Use This Instead
<pre>s = serial("COM1"); s.BaudRate = 115200; fopen(s)</pre>	<pre>s = serialport("COM1",115200);</pre>

See [Transition Your Code to serialport Interface](#) for more information about using the recommended functionality.

R2019a

Version: 4.0

New Features

Bug Fixes

Version History

Modbus Explorer App

The Modbus Explorer offers a user interface to easily set up read and write operations, and a live plot to see the values. The **Read Registers** table allows you to easily organize and manage read operations for multiple addresses, such as different sensors or switches on a PLC.

You can read and write to coils and registers in the Modbus Explorer app. The app supports a subset of the MATLAB Modbus functionality. You can do the following in the Modbus Explorer:

- Read coils, inputs, registers, and holding registers. This is the functionality of the Modbus read function.
- Write to coils and holding registers. This is the functionality of the Modbus write function.

To launch the Modbus Explorer, choose one of these options:

- In the MATLAB Apps tab, under **Test & Measurement**, select **MODBUS Explorer**.
- At the MATLAB command line, type `modbusExplorer`.

For more information, see:

- Configure a Connection in the Modbus Explorer
- Read Coils, Inputs, and Registers in the Modbus Explorer
- Write to Coils and Holding Registers in the Modbus Explorer
- Control a PLC Using the Modbus Explorer

Aardvark Adaptor Not Available for Mac for R2019a

For R2019a, R2018b, and R2018a you cannot use the Aardvark adaptor for the I2C or SPI interfaces on the macOS platform. You can still use it on Windows in all releases, and on Linux® for R2018a and R2019a. For releases prior to R2018a, you can use it on all three platforms, including the Mac.

Version History

For R2019a, R2018b, and R2018a you cannot use the Aardvark adaptor for the I2C or SPI interfaces on the macOS platform. To use Aardvark on the Mac platform, use a release prior to R2018a, or a future release, once available.

R2018b

Version: 3.14

New Features

Bug Fixes

Version History

Keysight Instruments: Improved control and ease of use with updated drivers from Keysight

The performance when using Keysight instruments with the Instrument Control Toolbox is improved due to updated instrument drivers from Keysight. To get improved performance, install the new driver using the Download MATLAB Drivers section on the Keysight website.

Instrument Control Toolbox Support Package for Ocean Optics Spectrometers Is Being Removed

The Instrument Control Toolbox Support Package for Ocean Optics Spectrometers is being removed in R2018b. Instrument Control Toolbox still supports use of Ocean Optics hardware. Required MATLAB files are installed with the toolbox. Required Ocean Optics files, such as instrument drivers, need to be installed separately from Ocean Optics.

Version History

You can still use Ocean Optics instruments with the toolbox. To do so, install any required instrument drivers from Ocean Optics, if you do not already have them installed. They are no longer provided in a support package.

Required files from Ocean Optics are available from the MathWorks website: <https://www.mathworks.com/hardware-support/ocean-optics-spectrometers.html>.

Aardvark Adaptor Not Available for Mac

For R2018b and R2018a, you cannot use the Aardvark adaptor for the I2C or SPI interfaces on the macOS platform. It also does not work on Linux for 2018b. You can still use it on Windows. For releases prior to R2018a, you can use it on all three platforms, including the Mac.

Version History

For R2018b and R2018a, you cannot use the Aardvark adaptor for the I2C or SPI interfaces on the macOS platform. To use Aardvark on the Mac platform, use a release prior to R2018a, or a future release, once available.

R2018a

Version: 3.13

New Features

Bug Fixes

Version History

VISA Support: Communicate with instruments using Rohde & Schwarz R&S VISA

You can now use the Instrument Control Toolbox VISA interface with Rohde & Schwarz R&S VISA.

These vendors are supported for the VISA interface:

- Keysight (formerly Agilent Technologies®)
- Rohde & Schwarz
- Tektronix
- National Instruments™

UDP and TCP/IP Communication: Send and receive data with more flexibility in Simulink using the expanded configuration options found in the UDP and TCP/IP client blocks

The TCP/IP Send, TCP/IP Receive, UDP Send, and UDP Receive blocks have additional options to expand configuration in Simulink models. The options are available when you open the Block Parameters dialog box from the block.

The TCP/IP blocks now offer settings to configure these properties:

- New **Transfer Delay** option (TransferDelay property)
- New data type ASCII under **Source Data type** (new value for the DataType property)
- New **ASCII format string** option, which is enabled when you choose ASCII as your data type
- New **Terminator** option, which is enabled when you choose ASCII as your data type (Terminator property)

The UDP blocks now offer settings to configure these properties:

- New **Enable local port sharing** option (EnablePortSharing property)
- New **UDP packet size** option (OutputDatagramPacketSize property)
- New **Output latest data** option
- New **Local address** option (LocalHost property)
- New data type ASCII under **Source Data type** (new value for the DataType property)
- New **ASCII format string** option, which is enabled when you choose ASCII as your data type
- New **Terminator** option, which is enabled when you choose ASCII as your data type (Terminator property)

For information about these new options, open the Block Parameters dialog box by double-clicking the block in your model, and click the **Help** button.

TCP/IP Blocks Renamed

In the Instrument Control Toolbox Block Library, the TCP/IP Receive block is now called the TCP/IP Client Receive block, and the TCP/IP Send block is now called the TCP/IP Client Send block. Note that the internal name has not been changed, so there is no compatibility issue if you have scripts or

models with these block names. Both of the blocks are for use as TCP/IP clients. They do not work as TCP/IP servers.

Keysight Vendor Name in Code

The use of Agilent as a value for the `vendor` property (`agilent`) is now replaced by Keysight (`keysight`). Note that if you have `agilent` in scripts, they will continue to work.

For example, if you used Agilent VISA to create the `visa` object, as shown here:

```
v = visa('agilent', 'GPIB0::1::30::INSTR');
```

you should now use Keysight as the vendor name:

```
v = visa('keysight', 'GPIB0::1::30::INSTR');
```

Enhanced warnings and troubleshooting information for read warning messages

If you receive a warning for a read attempt where either no data was read or not all data was read, the warning links to troubleshooting information to help with the specific read problem. This information is available for the following interfaces:

- Bluetooth
- Serial Port
- GPIB
- UDP
- TCP/IP
- VISA

The troubleshooting information covers reading ASCII, binary, and binblock data for each one.

To see the troubleshooting info, click on the Help link in the warning message. You can also access all of this information in the "Troubleshooting Read Warning Messages" section in the Troubleshooting Guide. In the Instrument Control Toolbox Doc Center, select "Troubleshooting" to access them.

Aardvark Adaptor Not Available for Mac for R2018a

For R2018a, you cannot use the Aardvark adaptor for the I2C or SPI interfaces on the macOS platform. You can still use it on Windows and Linux. For releases prior to R2018a, you can use it on all three platforms, including the Mac.

Version History

For R2018a, you cannot use the Aardvark adaptor for the I2C or SPI interfaces on the macOS platform. To use Aardvark on the Mac platform, use a release prior to R2018a.

R2017b

Version: 3.12

New Features

Bug Fixes

Quick Control RF Signal Generator Interface: Use a simple interface to easily download signals and control RF signal generators

The Quick Control interface for RF Signal Generators joins the Quick-Control Instruments family of Quick-Control Oscilloscope and Quick-Control Function Generator. You can use this new interface for simplified RF signal generator control and waveform generation.

Create the Quick-Control RF Signal Generator object using the Instrument Control Toolbox `rfsiggen` function. It provides simplified control of signal generators and performs arbitrary waveform generation without dealing with the underlying drivers. You can use it with RF signal generators that use IVI-C drivers.

For information about prerequisites, see [Quick-Control RF Signal Generator Requirements](#).

For information about functions that work with the Quick-Control RF Signal Generator, see [Quick-Control RF Signal Generator Functions](#).

For information about properties that work with the Quick-Control RF Signal Generator, see [Quick-Control RF Signal Generator Properties](#).

For examples of creating the RF Signal Generator object, downloading a waveform, and generating signal and modulation output, see [Download and Generate Signals with RF Signal Generator](#).

GPIB Support for ADLINK: Communicate with instruments using ADLINK GPIB interface hardware

You can now use adaptor boards from ADLINK for the GPIB interface.

Communication with instruments over the GPIB interface requires an adaptor board and driver from one of these vendors:

- Keysight (formerly Agilent®)
- ICS Electronics™
- Measurement Computing™ Corporation (MCC)
- ADLINK Technology
- National Instruments

R2017a

Version: 3.11

New Features

Bug Fixes

MODBUS Protocol Support: Communicate with hardware using the MODBUS protocol over TCP/IP and Serial

The Instrument Control Toolbox now supports the MODBUS interface, over TCP/IP or Serial RTU. You can use it to communicate with MODBUS servers, such as controlling a PLC (Programmable Logic Controller), communicating with a temperature controller, controlling a stepper motor, sending data to a DSP, reading bulk memory from a PAC controller, and monitoring temperature and humidity on a MODBUS probe.

Using the MODBUS interface, you can do the following tasks, which correspond to the MODBUS function codes listed in the table.

Functionality	MODBUS Function Code
Read and write coils	1, 5, 15
Read discrete inputs	2
Read and write holding registers	3, 6, 16
Read input registers	4
Perform mask writes on holding registers	22
Perform write/read (in one operation) on holding registers	23

You use the `modbus` function to create a MODBUS object, and then perform reads and writes using communication functions. You can perform read operations from four types of target addressable areas:

- Coils
- Inputs
- Input registers
- Holding registers

You can write data to coils and holding registers. You can also perform a combination of one write operation and one read operation on groups of holding registers in a single MODBUS transaction.

For information about using the MODBUS interface, including examples for all of the functions and the list of properties you can use, see [MODBUS Communication](#).

Serial discovery function

You can use the `seriallist` function to discover serial ports you can access. The function returns a list of all serial ports on a system. The list includes virtual serial ports provided by USB-to-serial devices and Bluetooth Serial Port Profile devices.

For example:

```
seriallist
ans =
    1x2 string array
```

"COM1" "COM3"

In this case the system has ports COM1 and COM3 available to use. You could then use one of the ports when creating a `serial` object to communicate over the serial port interface.

Note On Linux systems, the `serialist` function does not show ports that are in use. On Windows and Mac systems, it shows both available and in-use ports. But on Linux it shows only available ports.

R2016b

Version: 3.10

New Features

Bug Fixes

Hardware Support: Connect to GPIB instruments using Measurement Computing interface hardware

You can now use adaptor boards from Measurement Computing Corporation (MCC) for the GPIB interface. These were unsupported in R2016a due to the deprecation of 32-bit Windows, but you can now use them on 64-bit Windows. Communication with instruments over the GPIB interface requires an adaptor board and driver from one of these vendors:

- Keysight (formerly Agilent)
- ICS Electronics
- Measurement Computing Corporation (MCC)
- National Instruments

Ocean Optics Spectrometers Support Package supported on Mac

You can use Instrument Control Toolbox to communicate with Ocean Optics USB spectrometers. Ocean Optics manufactures a broad line of USB-powered spectrometers covering the visible, near IR, and UV portions of the spectrum. You can use these spectrometers from MATLAB on Windows and Macintosh platforms.

To use the Ocean Optics support, you must install the Instrument Control Toolbox Support Package for Ocean Optics Spectrometers.

Advanced tab completion for functions

To get a list of options you can use on the function, press the **Tab** key after entering a function on the MATLAB command line. The list expands, and you can scroll to choose a property or value. For example, when you create a `gpiB` object, you can get a list of installed vendors:

```
g = gpiB('
```

When you press **Tab** after the parentheses and single quote, as shown here, the list of installed GPIB vendors displays, such as `keysight`, `ics`, `mcc`, and `ni`.

The format for the GPIB object constructor function is:

```
g = gpiB('vendor',boardindex,primaryaddress)
```

When you press **Tab** where a field should appear, you get the list of options for that field. The other interface objects, such as Bluetooth, Serial, TCP/IP, etc., also include this capability on their object constructor functions.

You can also get the values for property-value pairs. For example, to get the possible terminator values when creating a serial object, type:

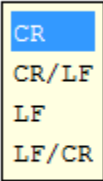
```
s = serial('COM1','Terminator','
```

Press **Tab** after typing the single quote after `Terminator` to get the possible values for that property, as shown here.

```

>>
>>
>>
>>
>>
>>
>>
fx >> s = serial('COM1','Terminator','

```



Many of the other toolbox functions also have tab completion. For example, when using the `fread` function you can specify the precision type using tab completion.

```
data = fread(s,256,'
```

Press **Tab** after typing the single quote after the size (256 values in this example), since precision is the next argument the `fread` function takes, to get the possible values for the precision types, such as `'double'`, `'int16'`, etc.

```

>>
>> s = serial('COM1')

Serial Port Object : Serial-COM1

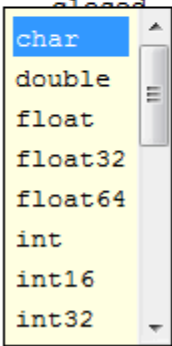
Communication Settings
  Port:          COM1
  BaudRate:     9600
  Terminator:   'LF'

Communication State
  Status:       closed
  RecordStatus:

Read/Write State
  TransferStatus:
  BytesAvailable:
  ValuesReceived:
  ValuesSent:

fx >> data = fread(s,256,'

```



When the list of possible values is long, a scroll bar appears in the pop-up window, as shown in this example.

Port sharing for UDP interface

UDP ports can now be shared by other applications to allow for multiple applications to listen to the UDP datagrams on that port. You can bind a UDP object to a specific `LocalPort` number, and in another application bind a UDP socket to that same local port number so both can receive UDP broadcast data.

This allows for the ability to listen to UDP broadcasts on the same local port number in both MATLAB and other applications. You can enable and disable this capability with a new property of the UDP object called `EnablePortSharing`.

The `EnablePortSharing` property allows you to control UDP port sharing, and the possible values are `on` and `off`. The default value is `off`.

EnablePortSharing Values	Result
'on'	Allows other UDP sockets to bind to the UDP object's <code>LocalPort</code> .
'off' (default)	Prevents other UDP sockets from binding to the UDP object's <code>LocalPort</code> .

Note that you need to set this property before calling `fopen` on the UDP object, or you will get an error.

This example shows creating a UDP object, assigning the local port, enabling port sharing, then opening the connection.

```
u = udp();
u.LocalPort = 5000;
u.EnablePortSharing = 'on';
fopen(u)
```

You can now do read and write operations, and other applications can access the port since port sharing is enabled.

Support Package Installation via Add-Ons Explorer: Install, manage, and update support packages

You can now install support packages using the MATLAB **Add-Ons** menu. You can also use **Add-Ons** to update installed support package software or update the firmware on third-party hardware.

Installing a Support Package

To install an Instrument Control Toolbox support package:

On the MATLAB **Home** tab, in the **Environment** section, click **Add-Ons > Get Hardware Support Packages**.

In the Add-On Explorer, scroll to the **Hardware Support Packages** section, and click **show all** to find your support package.

Uninstalling a Support Package

To uninstall support packages:

On the MATLAB **Home** tab, in the **Environment** section, click **Add-Ons > Manage Add-Ons**.

Updating a Support Package

To update existing support packages:

On the MATLAB **Home** tab, in the **Environment** section, click **Add-Ons > Check for Updates > Hardware Support Packages**.

For more information about using Add-On Explorer, see [Get Add-Ons](#).

R2016a

Version: 3.9

New Features

Bug Fixes

Version History

Hardware Support: Acquire and analyze data from newly supported National Instruments devices

NI Scope now supports NI PXIe-5105 and the NI PXIe-5114 Oscilloscope/Digitizer devices.

Bluetooth Support: Use Bluetooth devices that support Serial Port Profile on Mac OS X

The Instrument Control Toolbox Bluetooth support now includes the Mac OS X.

You can connect to devices over the Bluetooth interface and transmit and receive ASCII and binary data. Instrument Control Toolbox supports the Bluetooth Serial Port Profile (SPP). You can identify any SPP Bluetooth device and establish a two-way connection with that device.

The Bluetooth interface is supported on these platforms:

- Mac OS X
- Microsoft® Windows 64-bit

Hardware Support: Connect to instruments with IVI-C drivers using examples for the 13 IVI instrument classes

The toolbox now includes new examples for using IVI-C for all 13 IVI instrument classes.

- AC Power (IviACPwr)
- Counter (IviCounter)
- DC Power (IviDCPwr)
- Digitizer (IviDigitizer)
- Digital Multimeter (IviDmm)
- Down Converter (IviDownconverter)
- Function Generator (IviFgen)
- Power Meter (IviPwrMeter)
- Up Converter (IviUpconverter)
- RF Signal Generator (IviRFSigGen)
- Oscilloscope (IviScope)
- Spectrum Analyzer (IviSpecAn)
- Switch (IviSwtch)

Find the new examples under the **Examples** section of the Instrument Control Toolbox Documentation Center.

Removal of support for IVI-COM drivers and some GPIB adaptors

In R2016a, 32-bit MATLAB is no longer supported. For Instrument Control Toolbox, that means IVI-COM drivers and some GPIB adaptors are no longer supported, since they do not support 64-bit. You will receive errors if you try to use them.

Version History

IVI-COM Drivers

In R2016a, IVI-COM is no longer supported and you will receive the following error if you try to use IVI-COM functionality.

Error: IVI-COM drivers are no longer supported in MATLAB. IVI drivers are supported in MATLAB using the driver's IVI-C interface.

GPIB Adaptors

Also in R2016a, some GPIB adaptors, including Capital Equipment Corporation (CEC), CONTEC, IOtech®, and Measurement Computing Corporation (MCC), are no longer supported. You will receive the following error if you try to use a GPIB adaptor that is not supported.

Error: The specified VENDOR adaptor could not be found. Consider upgrading your GPIB adaptor to one of the 64-bit compliant adaptors listed on the GPIB supported hardware page, <https://www.mathworks.com/products/instrument/supported/gpib.html>. Contact MathWorks technical support to consider support for this hardware in future releases.

Functionality Being Removed or Changed

Functionality	Result	Use Instead	Compatibility Considerations
IVI-COM drivers	error	IVI-C drivers	See "IVI-COM Drivers" in "Removal of support for IVI-COM drivers and some GPIB adaptors" on page 13-2
GPIB adaptors: Capital Equipment Corporation (CEC), CONTEC, IOtech, and Measurement Computing Corporation (MCC)	error	A 64-bit compliant GPIB adaptor	See "GPIB Adaptors" in "Removal of support for IVI-COM drivers and some GPIB adaptors" on page 13-2

R2015b

Version: 3.8

New Features

Bug Fixes

Version History

Support for NI VISA on the Mac platform

The VISA interface, version 14.0 or newer, is now supported on the Mac platform. This includes all supported types of VISA, such as VISA-GPIB, as well as the Quick Control Oscilloscope and Quick Control Function Generator interfaces. You must install National Instruments VISA 14.0 or newer on your Mac computer.

New support package for Keysight IO Libraries and VISA Interface

The Instrument Control Toolbox Support Package for Keysight IO Libraries and VISA Interface simplifies the use of Keysight (formerly Agilent) VISA by installing the necessary software components, such as the IO libraries and VISA shared components.

Removal of 32-bit MATLAB and IVI-COM

In R2016a, 32-bit MATLAB will no longer be supported. For Instrument Control Toolbox, that means in R2016a, IVI-COM drivers and some GPIB adaptors will no longer be supported, since they do not support 64-bit.

In R2015b, you will receive a warning that these features will be unavailable in the following release. In R2016a, you will receive errors if you try to use them.

Version History

IVI-COM Drivers

In this release, you will receive the following warning when you use IVI-COM support.

Warning: IVI-COM functionality will no longer be available starting in MATLAB R2016a. 32-bit and 64-bit IVI-C drivers are supported on both 32-bit and 64-bit MATLAB, respectively. It is recommended to use the IVI-C driver instead of the IVI-COM driver.

GPIB Adaptors

Also in R2016a, some GPIB adaptors, including Capital Equipment Corporation (CEC), CONTEC, and IOtech, will no longer be supported. In this release, you will receive the following warning when you use a GPIB adaptor that will not be supported in R2016a.

Warning: This GPIB adaptor will no longer be supported in MATLAB R2016a. Consider upgrading your GPIB adaptor to one of the 32-bit/64-bit compliant adaptors listed on the GPIB supported hardware page, <https://www.mathworks.com/products/instrument/supported/gpib.html>. Contact MathWorks technical support to consider support for this hardware in future releases.

Troubleshooting Guide Added to Documentation in R2015a

A troubleshooting guide covering all of the supported interfaces was added to the Instrument Control Toolbox documentation in R2015a. It contains general information outlining things to try if you have trouble connecting or communicating with your instrument. It also has interface-specific information for all of the supported interfaces.

See Troubleshooting in Instrument Control Toolbox for the contents of the Troubleshooting Guide.

R2015a

Version: 3.7

New Features

Bug Fixes

New Support Package for National Instruments VISA and ICP interfaces

The Instrument Control Toolbox Support Package for National Instruments VISA and ICP Interfaces allows you to use the Quick Control Oscilloscope and Quick Control Function Generator interfaces.

After you download and install the support package, you can use the Quick Control interfaces to communicate with oscilloscopes and function generators. You can perform the following tasks.

- Configure oscilloscope properties
- Read waveforms from oscilloscopes
- Generate standard waveforms with function generators
- Generate arbitrary waveforms with function generators

The support package simplifies the use of these interfaces by installing the necessary software components, such as VISA, VISA shared components, and the IVI compliance package. For more information about the Quick Control interfaces, see Quick Control Oscilloscope Requirements and Quick Control Function Generator Requirements.

Troubleshooting Guide Added to Documentation

A troubleshooting guide covering all of the supported interfaces has been added to the Instrument Control Toolbox documentation. It contains general information outlining things to try if you have trouble connecting or communicating with your instrument. It also has interface-specific information for all of the supported interfaces.

See Troubleshooting in Instrument Control Toolbox for the contents of the Troubleshooting Guide.

R2014b

Version: 3.6

New Features

Bug Fixes

SPI support for National Instruments NI-845x devices

The Instrument Control Toolbox support of SPI communication now includes the use of a NI-845x adaptor board, in addition to the Aardvark adaptor support.

For information on using the SPI interface, see [Transmitting Data Over the SPI Interface](#).

To use the SPI interface with the NI-845x adaptor, you must download the Instrument Control Toolbox Support Package for NI-845x I2C/SPI Interface to obtain the latest driver, if you do not already have the driver installed. If you already have the latest driver installed, you do not need to download this Support Package. To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the NI-845x I2C/SPI Interface from the list.

Support for National Instruments NI-Switch devices via the Support Package Installer

You can use the Instrument Control Toolbox to communicate with NI-Switch switches. You can control a switch, such as a NI PXI-2586, to perform tasks such as switching power signals and loads in control applications and devices.

To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the NI-Switch from the list. For more information on installing this support package, see [Install the NI-Switch Support Package](#).

Support Package for Aardvark I2C/SPI Interface

For the Instrument Control Toolbox I2C and SPI interfaces, you can use either a Total Phase Aardvark host adaptor or an NI-845x adaptor. To use the I2C or SPI interface with the Aardvark adaptor, you must download this Hardware Support Package to obtain the driver, if you do not already have the driver installed. If you already have the driver installed, you do not need to download this Support Package.

To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the Aardvark I2C/SPI Interface from the list. For more information on installing this support package, see [Install the Aardvark Support Package](#).

The support package downloads and installs the Total Phase Aardvark host adaptor driver file on your host computer. Examples of using the Aardvark adaptor with the I2C interface can be found in the Instrument Control Toolbox documentation. For more information on using Aardvark with the I2C and SPI interfaces, see [I2C Interface Overview](#) and [SPI Interface Overview](#).

R2014a

Version: 3.5

New Features

Bug Fixes

Support for National Instruments NI-8451 and NI-8452 I2C interface via the Support Package Installer

The Instrument Control Toolbox support of I2C communication now includes the use of a NI-845x adaptor board, in addition to the Aardvark adaptor support.

For information on using the I2C interface, see [Transmitting Data Over the I2C Interface](#).

To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the NI845x I2C Driver from the list. For more information on installing this support package, see [Installing the NI-845x I2C Driver Support Package](#).

Support for National Instruments NI-DCPwr devices via the Support Package Installer

You can use the Instrument Control Toolbox to communicate with NI-DCPOWER power supplies. You can control and take digital measurements from the power supply, for example, a NI PXI 4011 triple-output programmable DC power supply.

To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the NI -DCPOWER from the list. For more information on installing this support package, see [Installing the NI-DCPOWER Support Package](#).

Support for National Instruments NI-DMM devices via the Support Package Installer

You can use the Instrument Control Toolbox to communicate with NI-DMM digital multimeters. You can control and take measurements from a digital multimeter, for example, measuring voltage or resistance.

To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the NI -DMM from the list. For more information on installing this support package, see [Installing the NI-DMM Support Package](#).

R2013b

Version: 3.4

New Features

Bug Fixes

SPI protocol support using Total Phase Aardvark SPI interface hardware

SPI, or Serial Peripheral Interface, is a synchronous serial data link standard that operates in full duplex mode. Instrument Control Toolbox SPI support lets you open connections with individual chips and to read and write over the connections to individual chips using an Aardvark host adaptor.

The primary uses for the `spi` interface involve the `write`, `read`, and `writeAndRead` functions for synchronously reading and writing binary data. To identify SPI devices in the Instrument Control Toolbox, use the `instrhwinfo` function on the SPI interface, called `spi`.

For information about supported platforms, see SPI Interface Overview.

For information on using this feature, see Configuring SPI Communication and Transmitting Data Over the SPI Interface.

I2C communication protocol now available in the Instrument Control app

In an earlier release, the toolbox introduced support for I2C. In R2013b, this support is extended to the Test & Measurement Tool.

To use the I2C support in the Test & Measurement Tool, select the **I2C** node in the instrument tree and right-click **Scan for I2C adaptors**.

For more information on the I2C interface, see I2C Interface Overview. For information on using it in the Test & Measurement Tool, see the Help within the tool by selecting the **I2C** node in the tree.

Support for Ocean Optics spectrometers via the Support Package Installer

Support for Ocean Optics spectrometers is available via the Support Package Installer. You can use the Instrument Control Toolbox to communicate with Ocean Optics USB spectrometers. You can acquire data from the spectrometer and control it. Ocean Optics manufactures a broad line of USB powered spectrometers covering the visible, near IR and UV portions of the spectrum. These spectrometers can be used from MATLAB on Windows, Linux, and Mac platforms.

To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the **Ocean Optics Spectrometers** from the list. For more information on installing this support package, see Installing the Ocean Optics Spectrometers Support Package.

Support for NI-SCOPE oscilloscopes via the Support Package Installer

Support for NI-SCOPE oscilloscopes is available via the Support Package Installer. You can use the Instrument Control Toolbox to communicate with NI-SCOPE oscilloscopes. You can acquire waveform data from the oscilloscope and control it.

To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the **NI-SCOPE** from the list. For more information on installing this support package, see Installing the NI-SCOPE Support Package.

Support for NI-FGEN function generators via the Support Package Installer

Support for NI-FGEN function generators is available via the Support Package Installer. You can use the Instrument Control Toolbox to communicate with NI-FGEN function generators. You can control and configure the function generator, and perform tasks such as generating sine waves.

To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the NI - FGEN from the list. For more information on installing this support package, see [Installing the NI-FGEN Support Package](#).

R2013a

Version: 3.3

New Features

Bug Fixes

Windows 64-bit support for Tektronix VISA

The VISA adaptor support now includes 64-bit Tektronix VISA. This works the same as any other VISA support, including VISA-TCPIP, VISA-Serial, or VISA-USB. For more information, see VISA Overview.

It is important to note that if you are using 64-bit Tektronix VISA on a machine with VISA implementations from multiple vendors, (e.g., you have installed drivers from Tektronix and another vendor such as Agilent), it is required that Tektronix VISA be configured as the primary VISA for it to be usable with Instrument Control Toolbox. Most 64-bit VISA implementations include a utility that allows you to select the primary and preferred VISA implementations. Use the VISA utility to set Tektronix VISA to be the primary VISA implementation on your machine. This step can be accomplished at any time, regardless of the order of installation of the VISA drivers.

Windows 64-bit support for ICS Electronics GPIB interface devices

The ICS Electronics GPIB support now includes Windows 64-bit support.

Windows 64-bit support for NI-SCOPE and NI-FGEN modular instruments

In addition to the 32-bit support, you can now use Instrument Control Toolbox with instruments using the National Instruments NI-SCOPE version 3.6 or higher and NI-FGEN version 2.7.2 or higher driver software on 64-bit Windows systems.

Support for 10 additional NI-SCOPE devices

As a result of supporting the latest NI-SCOPE driver (see previous section), the following devices are now supported.

- PXI-5153
- PCI-5153
- PXI-5154
- PCI-5154
- PXIe-5185
- PXIe-5186
- PXI-5620
- PXI-5621
- PXIe-5622
- PXI-5900

R2012b

Version: 3.2

New Features

Bug Fixes

Bluetooth support in Test & Measurement Tool

In R2011b, the toolbox introduced support for Bluetooth. In R2012b, this support is extended to the Test & Measurement Tool.

To use the Bluetooth support in the Test & Measurement Tool, select the **Bluetooth** node in the instrument tree and right-click **Scan for bluetooth devices**.

For more information on the Bluetooth interface, see Controlling Instruments Using Bluetooth. For information on using it in the Test & Measurement Tool, see the Help within the tool by selecting the **Bluetooth** node in the tree.

Quick-Control Function Generator now supports Agilent instruments

In R2012a, the toolbox introduced a new method to communicate with function generators, the Quick-Control Function Generator, for use with function generators using IVI-C drivers. In R2012b, the Quick-Control Function Generator is extended for use with Agilent function generators using SCPI commands.

Create the Quick-Control Function Generator object using the Instrument Control Toolbox `fgen` function. It provides simplified controlling of function generators and performs arbitrary waveform generation without dealing with the underlying drivers.

For information on prerequisites, functions that work with the Quick-Control Function Generator, and a full work flow example, see “Quick-Control Function Generator”.

R2012a

Version: 3.1

New Features

Bug Fixes

I2C Support

The toolbox now supports I2C communication, which is Inter-Integrated Circuit communication. Instrument Control Toolbox I2C support lets you open connections with individual chips and read and write over the connections to individual chips.

The I2C interface lets you do chip-to-chip communication using an Aardvark host adaptor. Some applications of this interface include communication with SPD EEPROMs and NVRAM chips, communication with SMBus devices, accessing low-speed DACs and ADCs, changing settings on color monitors using the display data channel, changing sound volume in intelligent speakers, reading hardware monitors and diagnostic sensors, and turning on or off the power supply of system components.

To identify I2C devices in the Instrument Control Toolbox, use the `instrhwinfo` function on the I2C interface, called `i2c`.

For information on using the I2C interface, see [Controlling Instruments Using I2C](#).

Quick-Control Oscilloscope Now Supports Tektronix Scopes

In R2011b the toolbox introduced a new method to communicate with oscilloscopes, the Quick-Control Oscilloscope, for use with scopes using IVI-C or IVI-COM drivers. In R2012a, the Quick-Control Oscilloscope is extended for use with Tektronix scopes.

Create the Quick-Control Oscilloscope object using the Instrument Control Toolbox `oscilloscope` function. It simplifies controlling oscilloscopes and performs waveform acquisitions without dealing with the underlying drivers.

Quick-Control Function Generator

A new method of communicating with instruments, Quick-Control Instruments, was introduced in R2011b with the Quick-Control Oscilloscope. In R2012a, a second instrument class is introduced - the Quick-Control Function Generator. You can use this new function generator, or `fgen`, for simplified `fgen` control and waveform acquisition.

Create the Quick-Control Function Generator object using the Instrument Control Toolbox `fgen` function. It provides simplified controlling function generators and performs arbitrary waveform acquisitions without dealing with the underlying drivers. You can use it with function generators using IVI-C drivers.

For information on prerequisites, functions that work with the Quick-Control Function Generator, and a full work flow example, see [Using Quick-Control Function Generator](#).

New IVI Class Supported

You can now use the following IVI instrument class with Instrument Control Toolbox:

- `IviACPwr` - IVI AC Power instrument class.

See [Using MATLAB IVI Wrappers](#) for more information on using the classes. See [Getting Started with IVI Drivers](#) for a list of supported classes.

IVI-C Class Compliant Wrappers in Test & Measurement Tool

In R2010b the toolbox introduced support for IVI-C class compliant wrappers. The IVI-C wrappers provide an interface to MATLAB for instruments running with IVI-C class compliant drivers, including 64-bit support. In R2012a, this support is extended to the Test & Measurement Tool.

View the IVI-C nodes by setting a preference in MATLAB. In **File > Preferences**, go to **Instrument Control**. Then select the **Show IVI Instruments in TMTTool** option in the **IVI Instruments** section. Then when you start the Test & Measurement Tool, the new **IVI Instruments** node appears under **Instrument Drivers**.

For more information, see Using IVI-C Class-Compliant Wrappers. For information on using it in the Test & Measurement Tool, see the Help within the tool by selecting the **IVI Instruments** node in the tree.

R2011b

Version: 3.0

New Features

Bug Fixes

Version History

Bluetooth Support

The toolbox now supports Bluetooth devices. Instrument Control Toolbox supports the Bluetooth serial port profile (SPP).

The Instrument Control Toolbox Bluetooth interface lets you connect to devices over the Bluetooth interface and to transmit and receive ASCII and binary data. You can identify any SPP Bluetooth device and establish a two-way connection with that device.

Bluetooth is an open wireless technology standard for exchanging data over short distances using short wavelength radio transmissions from fixed and mobile devices using a packet-based protocol. Bluetooth provides a secure way to connect and exchange information between devices such as Lego Mindstorm NXT robots, USB Bluetooth adaptors (dongles), wireless sensors, mobile phones, faxes, laptops, computers, printers, GPS receivers, etc.

To identify Bluetooth devices in the Instrument Control Toolbox, use the `instrhwinfo` function on the Bluetooth interface, called `Bluetooth`.

For information on using the Bluetooth interface, see [Controlling Instruments Using Bluetooth](#).

Quick-Control Oscilloscope

A new family of instrument support, Quick-Control Instruments, is being introduced. For R2011b, the Quick-Control Oscilloscope is available. You can use this new oscilloscope function for simplified oscilloscope control and waveform acquisition.

Create the Quick-Control Oscilloscope object using the Instrument Control Toolbox `oscilloscope` function. It provides a simplified way to control oscilloscopes and perform waveform acquisitions without dealing with the underlying drivers. It can be used with scopes using IVI-C or IVI-COM drivers.

For information on prerequisites, functions that work with the Quick-Control Oscilloscope, and a full work flow example of using it, see [Using Quick-Control Oscilloscope](#).

Generic VISA Support

In both the command-line and the Test & Measurement Tool, a generic VISA interface is now supported. In the Test & Measurement Tool, generic devices appear in the **More** node under the **VISA** node. In the command-line toolbox, they are available as a type `'generic'`.

For example, if you have a generic VISA device that is made by National Instruments, you could use the `instrhwinfo` function to see it:

```
instrhwinfo('visa','ni','generic')
```

You can use this generic support to communicate over open VISA sockets, USB Raw, etc.

For more information, see [Working with the Generic Interface](#).

VISA Node and Generic VISA Support in the Test & Measurement Tool

In the Test & Measurement Tool, instruments that use the VISA interface now appear under the **VISA** node in the instrument tree. For example, if you are using a TCP/IP instrument with the VISA

interface, instead of a **TCP/IP - VISA** node in the tree, you will see a **VISA** node, with a **TCP/IP** node under it. It is easier to see what protocols are using the VISA interface with the **VISA** node.

Also, the generic VISA interface is now supported. Generic devices appear in a **More** node under the **VISA** node in the instrument tree. If your instrument is recognizable as a type such as 'gpib' or 'tcpip', it appears in that type-specific node. For example, a TCP/IP instrument appears in the **TCPIP** node under the **VISA** node. But if it is a generic instrument, it appears in the **More** node.

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in the Instrument Control Toolbox software.

Version History

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the <'illustrative:old:ID'> identifier has changed to <'new:similar:ID'>. If your code checks for <'illustrative:old:ID'>, you must update it to check for <'new:similar:ID'> instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable MSGID.

To determine the identifier for an error, run the following command just after you see the error:

```
exception = MException.last;  
MSGID = exception.identifier;
```

Note: Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so it runs warning-free.

R2011a

Version: 2.12

New Features

Bug Fixes

Support for Three New IVI Classes

You can now use the following three IVI instrument classes with the Instrument Control Toolbox software:

- Upconverter
- Downconverter
- Timercounter

See *Using MATLAB IVI Wrappers* for more information on using the classes. See *Getting Started with IVI Drivers* for a list of supported classes.

IVI-C Class Compliant Wrapper

The eight supported classes of IVI instruments are now supported for IVI-C using class compliant wrappers. This includes 64-bit support for IVI-C drivers.

The IVI-C wrappers provide an interface to MATLAB for instruments running on IVI-C class compliant drivers.

For more information, see *Using IVI-C Class-Compliant Wrappers*.

Server Sockets Support

Support for Server Sockets is available, using the `NetworkRole` property on the TCP/IP interface. This support is for a single remote connection. You can use this connection to communicate between a client and MATLAB, or between two instances of MATLAB.

For example, you might collect data such as a waveform into one instance of MATLAB, and then want to transfer it to another instance of MATLAB.

For more information, see *Using TCP/IP Server Sockets*.

Known Issue - TCP/IP Endian Default

For the TCP/IP and UDP interfaces, the Endian value of the `ByteOrder` property defaults to `bigEndian`. This may cause problems if you need to use `littleEndian`. You should match the byte order of the machine you are connecting to.

R2010b

Version: 2.11

New Features

Bug Fixes

Support for IVI Class for Digitizers

You can now use the IviDigitizer class with the Instrument Control Toolbox software. See Using MATLAB IVI Wrappers for more information.

Support for IVI Class for Switches

You can now use the IviSwitc class with the Instrument Control Toolbox software. See Using MATLAB IVI Wrappers for more information.

Support for National Instruments NI-SCOPE Driver Software

You can now use Instrument Control Toolbox with instruments using the NI-SCOPE version 3.4 driver software.

Support for National Instruments NI-FGEN Driver Software

You can now use Instrument Control Toolbox with instruments using the NI-FGEN version 2.7.2 driver software.

Enhanced TCP/IP Performance for fread and binblockread Functions

fread and binblockread functions are now enhanced with better TCP/IP throughput.

Two New UDP Packet Size Properties

The InputDatagramPacketSize and OutputDatagramPacketSize properties allow you to control the size of the datagram packets and work with larger packets.

R2010a

Version: 2.10

New Features

Bug Fixes

Expanded spoll Function

You can now use the `spoll` function to execute a serial poll on VISA objects.

Enhanced Test and Measurement Tool

The Test and Measurement Tool interface is enhanced as follows:

- Enhanced desktop environment
- Two new menu items, **Desktop** and **Window**, which give you more control of the tool's size, location and layout

Support for Agilent 64-Bit GPIB on Windows 64

You can now use Instrument Control Toolbox with an Agilent 64-bit GPIB interface on a Windows 64-bit platform.

R2009b

Version: 2.9

New Features

Bug Fixes

New Class-Compliant Interface for IVI-COM Drivers

Instrument Control Toolbox includes a class-compliant interface for IVI-COM drivers that lets you switch among instruments with different interfaces that use a class-compliant IVI-COM driver.

Support for Agilent 64-Bit VISA on Windows 64

You can now use Instrument Control Toolbox with an Agilent 64-bit VISA interface on a Windows 64-bit platform.

Warning Added for Future Deprecation of UDP Binblock Support

A warning has been added for the deprecation of binblock read and write operations over an UDP network in a future release of the toolbox. You will receive a warning message if you read or write to an instrument over an UDP network.

R2009a

Version: 2.8

New Features

Bug Fixes

Expanded Platform Support Added for Instrument Control Toolbox

You can now install the Instrument Control Toolbox software on all platforms.

Using the Status Function with an IVI Driver Can Cause Namespace Conflicts

If you are creating an IVI driver, use the `InstrumentStatus` function to set the status group for the vendor-specific driver status. Using `Status` can cause a namespace conflict. If you are using a MATLAB provided driver, you can continue to use the `Status` function.

Expanded Platform Support Added for NI-VISA and GPIB Interfaces

You can now use Instrument Control Toolbox with a National Instruments VISA or a GPIB interface on a Windows Vista™ 64 platform.

IVI-COM Driver Support

This version of the Instrument Control Toolbox software offers expanded IVI-COM driver coverage.

R2008b

Version: 2.7

New Features

Bug Fixes

Version History

Enhanced IVI-COM Driver Support

The Instrument Control Toolbox devices can now correctly initialize collections and items.

Version History

IVI-COM drivers generated using MAKEMID in the Instrument Control Toolbox, Version 2.7 (R2008b) will not work with versions prior to Version 2.7 (R2008b), and will throw an error. If you need to use the drivers created with Instrument Control Toolbox Version 2.7, make sure you update your license to the latest release of the MATLAB software.

After you install Instrument Control Toolbox Version 2.7, if you regenerate the MDD file, you will need to update your code. To update your code, change the way you call your collections and items. Collections are now spelled as a plural noun and items are spelled as a singular noun. For example, if you had an item called `Measurements` and a collection called `Measurement`, you need to change the code so that the item name reads `Measurement` and the collection name reads `Measurements`. The following table shows you an example of the interface and the corresponding methods as it should look once you have updated your code. The example uses the item `Measurement` and the collection, `Measurements`. The example also compares the model with the Agilent `RFPowerMeter` driver model.

Interface	Instrument Control Toolbox Version 2.7 driver model methods	Agilent RFPowerMeter driver methods
Measurement	ClearLimitFails ConfigureRelative Fetch Configure ConfigureTimeGated Measure ConfigureAnalogOutput ConfigureTTLOutput Read ConfigureLimits EnableAnalogOutput ConfigureOffset EnableTTLOutput	ClearLimitFails ConfigureRelative Fetch Configure ConfigureTimeGated Measure ConfigureAnalogOutput ConfigureTTLOutput Read ConfigureLimits EnableAnalogOutput ConfigureOffset EnableTTLOutput
Measurements	Abort IsOperationComplete Initiate	Abort IsOperationComplete Initiate

Here is another example showing the corresponding properties of the two sample interfaces, `Measurement` and `Measurements`. This example also compares the model with the Agilent `RFPowerMeter` driver model.

Interface	Instrument Control Toolbox Version 2.7 driver model properties	AgilentRFPowerMeter driver model properties
Measurement	Feed1Channel Feed1Gate Feed1Type Feed2Channel Feed2Gate Feed2Type LimitAutoClear LimitEnabled LimitFails LimitLower LimitUpper Offset OffsetEnabled Operation RatioUnits RelativeEnabled Resolution Units	Feed1Channel Feed1Gate Feed1Type Feed2Channel Feed2Gate Feed2Type LimitAutoClear LimitEnabled LimitFails LimitLower LimitUpper Offset OffsetEnabled Operation RatioUnits RelativeEnabled Resolution Units
Measurements	Count Name Item	Count Name Item

Enhanced Functionality of the Test and Measurement Tool

Test and Measurement Tool now has advanced interaction between the hardware interfaces and the instrument driver.

Enhanced IVI-C Driver Support

Instruments that require a reset will now work with IVI-C drivers in the Instrument Control Toolbox product.

Expanded Instrument Control Toolbox Demos

The Instrument Control Toolbox product now has a new demo that uses an IVI-COM driver with a mixed signal oscilloscope.

Expanded spoll Documentation

The low-level GPIB spoll function now has more documentation to help you work with serial poll.

R2008a

Version: 2.6

New Features

Bug Fixes

Instrument Control Toolbox Block Library

There are three new Simulink blocks that can send and receive data via a serial port in Simulink.

- **Serial Configuration** — Configure the parameters for a serial port that you can use to send and receive data.
- **Serial Receive** — Receive binary data over a serial port.
- **Serial Send** — Send binary data over a serial port.

Improved Throughput

Throughput is improved in the Instrument Control Toolbox Serial interface and TCP/IP Receive blocks.

Instrument Drivers

Several new instrument drivers have been added to MATLAB Central.

Enhanced Capability of the TM Tool

The capability for discovery of VISA-USB and VISA TCP/IP (VXI-11) instruments in the TM TOOL is now enhanced.

Support for Agilent IO Libraries Suite 15.0

The Instrument Control Toolbox software now supports the Agilent IO Libraries Suite 15.0.

fprintf Timeout Errors

You will no longer see sporadic timeouts when you write to the serial port using `fprintf`.

R2007b

Version: 2.5

New Features

Bug Fixes

Version History

Instrument Control Toolbox Block Library

There are four new Simulink blocks that can send and receive data over TCP/IP and UDP in Simulink.

- **TCP/IP Receive** — Receive data over a TCP/IP network from a specified remote machine.
- **TCP/IP Send** — Send data over a TCP/IP network to a specified remote machine.
- **UDP Receive** — Receive data over an UDP network from a specified remote machine.
- **UDP Send** — Send data over an UDP network to a specified remote machine.

Instrument Drivers

- Additional instrument drivers are now available for Agilent, LeCroy®, Tabor, and Tektronix instruments.
- Support for Keithley® version 8.3 GPIB drivers is updated.

spoll GPIB Status Information

Additional GPIB status information is now provided with the `spoll` function.

Obsolete Functions

The following functions are obsolete as of this release. Any attempt to use these functions now results in an error message.

Obsolete Function Name	New Function Name
<code>freeserial</code>	<code>fclose</code>
<code>instrcomm</code>	<code>tmtool</code>
<code>instrcreate</code>	<code>tmtool</code>

Version History

In the previous release, `freeserial` was nonoperational and generated a warning. Now any call to this function generates an error.

In previous releases, `instrcomm` and `instrcreate` opened their own graphical user interfaces. The functionality of these interfaces is available in the `tmtool` GUI.

New Error Message

The `fwrite` and the `fprintf` functions will return an error message if the `flowcontrol` property is set to `hardware` and a hardware connection is not detected. Previously, MATLAB software would fail to respond while it waited for the connection.

Version History

In the previous release, MATLAB software would become unresponsive if the `flowcontrol` property was set to `hardware` and a hardware connection was not detected. The `fwrite` and the `fprintf` functions will now return an error message.

R2007a

Version: 2.4.2

New Features

Version History

Confirmation Dialog Preference Settings Moved

The preferences for setting whether a dialog box confirms that you want the MATLAB Instrument Driver Editor and the MATLAB Instrument Driver Testing Tool to create new files are in a new location. You now access them in the **General > Dialog Confirmations** node of the MATLAB Preferences dialog box.

For details on these Instrument Control preference settings, see **General Preferences for Instrument Control** in the **Instrument Control User's Guide** documentation.

Version History

In previous versions of MATLAB and Instrument Control Toolbox software, these dialog box preferences were found under the **Instrument Control** node of the Preferences dialog box.

R2006b

Version: 2.4.1

No New Features or Changes

R2006a

Version: 2.4

New Features

Generic Instrument Drivers Supported

This release includes support for generic instrument drivers that allow Instrument Control Toolbox software to communicate with devices or software that do not use industry-standard drivers or protocols.

For more information, see [Using Generic Instrument Drivers](#).

LeCroy Driver

This release includes support for LeCroy instrument drivers.

Several drivers ship with the toolbox. You can find these drivers by looking in the directory

```
matlabroot\toolbox\instrument\instrument\drivers
```

where *matlabroot* is the MATLAB installation directory, as seen when you type

```
matlabroot
```

in the MATLAB Command Window.

Many other drivers are available on the MathWorks Web site at

```
https://www.mathworks.com/matlabcentral/fileexchange
```

including drivers specifically for Instrument Control Toolbox software under the Test and Measurement category.

Agilent Instrument Simulations

Instrument Control Toolbox software includes simulations of the Agilent 33120a function generator and Agilent e3648 DC power supply. These simulations are available as instrument drivers.

You can see the drivers shipped with the release by looking in the directory

```
matlabroot\toolbox\instrument\instrument\drivers
```

where *matlabroot* is the MATLAB installation directory, as seen when you type

```
matlabroot
```

in the MATLAB Command Window.

The Agilent simulations are provided by the drivers

```
generic_agilent_33120a.mdd
```

```
generic_agilent_e3648a.mdd
```

R14SP3

Version: 2.3

No New Features or Changes

R14SP2

Version: 2.2

New Features

Advantech GPIB Supported

Supported GPIB interfaces now include Advantech® GPIB.

For further information, type

```
instrhelp gpib
```