



Agilent E6630A Wireless Connectivity Test Set (and X-Series Instruments)

Programmer's Guide



Agilent Technologies

Notices

© Agilent Technologies, Inc.
2008-2013

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Trademark Acknowledgements

Microsoft[®] is a US registered trademark of Microsoft Corporation.

Windows[®], MS Windows[®], Windows Vista[®], and Windows 7[®] are US registered trademarks of Microsoft Corporation.

Visual Basic[®], Visual Basic.NET[®], Visual Studio[®], and MS Excel[®] are registered trademarks of Microsoft Corporation.

Adobe Acrobat[®] and Reader[®] are US registered trademarks of Adobe Systems Incorporated.

Java[™] is a US trademark of Sun Microsystems, Inc.

MATLAB[®] is a US registered trademark of Math Works, Inc.

LabVIEW[™] and CVI[™] are trademarks of National Instruments Corporation.

LabWindows is a mark used by National Instruments Corporation under a license from Microsoft Corporation.

Manual Part Number

E6630-90002

Print Date

August 2013

Printed in USA

Agilent Technologies Inc.
1400 Fountaingrove Parkway
Santa Rosa, CA 95403

Warranty

The material contained in this document is provided “as is,” and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a US Government prime contract or subcontract, Software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014

(June 1995), or as a “commercial item” as defined in FAR 2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the US Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING

A **WARNING** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Warranty

This Agilent technologies instrument product is warranted against defects in material and workmanship for a period of one year from the date of shipment. During the warranty period, Agilent Technologies will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Agilent Technologies. Buyer shall prepay shipping charges to Agilent Technologies, and Agilent Technologies shall pay shipping charges to return the product to Buyer. For products returned to Agilent Technologies from another country, Buyer shall pay all shipping charges, duties, and taxes.

Where to Find the Latest Information

Documentation is updated periodically. For the latest information about these products, including instrument software upgrades, application information, and product information, browse to one of the following URLs, according to the name of your product:

<http://www.agilent.com/find/e6630a>

To receive the latest updates by email, subscribe to Agilent Email Updates at the following URL:

<http://www.agilent.com/find/emailupdates>

Information on preventing instrument damage can be found at:

<http://www.agilent.com/find/tips>

Is your product software up-to-date?

Periodically, Agilent releases software updates to fix known defects and incorporate product enhancements. To search for software updates for your product, go to the Agilent Technical Support website at:

<http://www.agilent.com/find/techsupport>

Table of Contents

1 Introduction to Programming X-Series Applications

How to Use this Manual	9
X-Series Programming Options	10
Hardware Connection Formats	11
VISA Driver Options	11
Agilent VEE Pro	13
Programming Documentation Roadmap	14

2 SCPI Programming Fundamentals

SCPI Language Basics	16
Command Keywords, Separators and Syntax	16
Creating Valid Commands	17
Special Characters in Commands	17
Parameters in Commands	19
Variable Parameters	19
Block Program Data	21
Writing Multiple Commands on the Same Line	21
SCPI Termination and Separator Syntax Examples	22
Where to find X-Series SCPI Command Definitions	23
Help System Features for SCPI Programmers	23
Help System Contents Pane	23
Help Topic Content	24
List of SCPI Commands	25
Simple SCPI Communication Methods	27
Connecting to the X-Host Using VISA Connection Strings	27
Left Sub-Instrument Connections	27
Right Sub-Instrument Connections	27
Distinguishing Between the Sub-Instruments	28
Communicating SCPI Using Telnet	28
Determining Instrument IP Address	30
Enabling Telnet in Windows	31
Communicating SCPI using Agilent Connection Expert	31
Techniques for Improving Measurement Performance	34
Turn off Display Updates	34
Use Binary Data Format instead of ASCII	34

Contents

Minimize the Number of Bus Transactions	34
Minimize DUT/instrument Setup changes	35
Avoid Unnecessary Use of *RST	35
Use Video or External Triggering for Single Burst Signals	35
To make Power Measurements on Multiple Bursts or Slots use CALCulate:DATA<n>:COMPRESS?	36
More Hints & Tips	37

3 Developing and Deploying VISA Projects

Programming in Visual Basic 6 with VISA	39
Location of Header Files	39
Programming in C or C++ with VISA	39
Location of Header Files & Libraries	39
Programming with Microsoft .NET and VISA	40
Location of Header Files	40
Requirements for Deploying a VISA Project	41
Multiple VISA DLL Versions	41

4 Program Samples

Where to find Sample Programs	43
Programing Samples	44
Matrix of Program Sample Functionality and Programming Language	44
Visual Basic 6	45
Transfer Screen Images	45
C, C++	46
Serial Poll for Operation Complete	46
Trace Detector/Couple Markers	46
C#.NET & Visual Studio 2010	47
Transfer Screen Images	47
Serial Poll for Sweep Complete	47
Agilent VEE Pro	48
Transfer Screen Images	48
LabVIEW	48
Transfer Screen Images	48

Appendix A References

Documents & Web Sites	49
Developer Resources	51

Agilent Developer Network (ADN)	51
Technical Support	51



1 Introduction to Programming X-Series Applications

How to Use this Manual

This chapter provides overall information regarding remote programming of Agilent X-Series instruments, and how to use the programming documentation provided with the product.

This chapter includes the following topics:

“X-Series Programming Options” on page 10

“Hardware Connection Formats” on page 11

“VISA Driver Options” on page 11

“Agilent VEE Pro” on page 13

“Programming Documentation Roadmap” on page 14

The second chapter, “[SCPI Programming Fundamentals](#)” on page 15, provides an introduction to **Standard Commands for Programmable Instruments (SCPI)**, which is the most popular and versatile protocol for programming X-Series instruments.

The chapter “[Developing and Deploying VISA Projects](#)” on page 39 provides basic information about X-Series programming with the **Virtual Instrument Software Architecture (VISA)**, using various popular programming languages.

The chapter “[Program Samples](#)” on page 43 describes all program samples that are included in the `\progexamples` folder of the X-Series Documentation DVD, and provides information about how to find other X-Series program samples.

X-Series Programming Options

The E6630A Wireless Connectivity Test Set uses the same applications used in Agilent X-Series analyzers and test sets. You can program these instruments using a variety of programming tools, languages and Application Development Environments (ADEs).

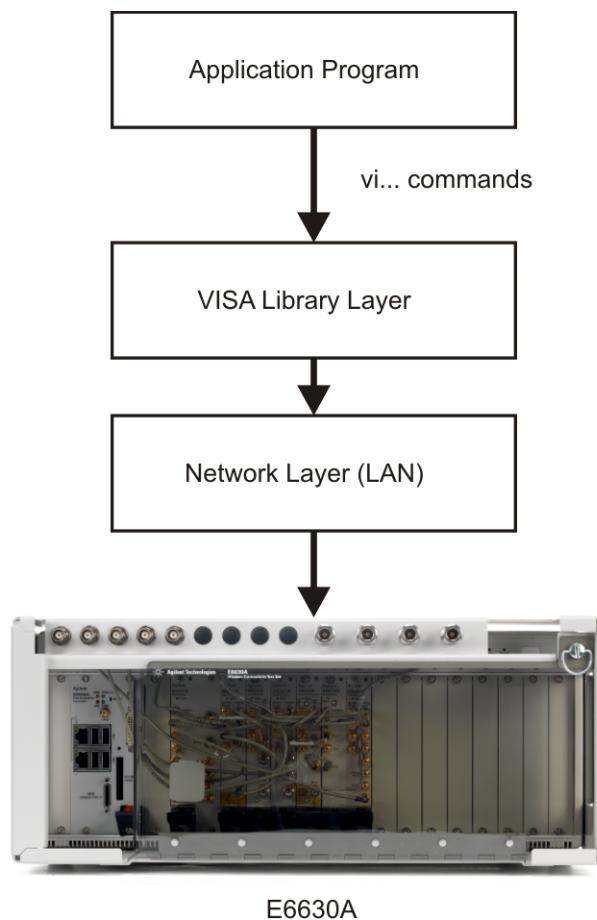
There are also several software driver technologies that you can use to program X-Series instruments, which offer various tradeoffs between programming tool, ADE and driver technology. [Table 1-1](#) explains the relative advantages of each programming method and driver technology. [Figure 1-1 on page 11](#) shows a conceptual overview of the hierarchy of drivers that are available for X-Series programming.

Table 1-1 Programming Options & Driver Technologies for X-Series Instruments

Method	Description	
VISA Drivers	Features	As shown in Figure 1-1 , VISA is a driver technology that operates at a lower level of abstraction than instrument drivers. As such, it offers greater flexibility, at the expense of a longer learning curve. VISA is a generic, industry-wide standard, unlike Instrument Drivers, which are instrument-specific. For more details, see “VISA Driver Options” on page 11 .
	Acquisition & Licensing	Free download from Agilent web site.
	Requires separate ADE?	Yes, but depending on your program development requirements, you may be able to use a free download such as one of the Microsoft Visual Studio Express editions.
	Driver Support	VISA (commands go to the VISA Library Layer), as shown in Figure 1-1 below.
Agilent VEE	Features	An integrated, graphically-oriented, standalone ADE, which supports instruments from Agilent and other manufacturers. For more details, see “Agilent VEE Pro” on page 13 .
	Acquisition & Licensing	License must be purchased from Agilent.
	Requires separate ADE?	No
	Driver Support	Supports both Instrument Drivers and VISA.

Figure 1-1

E6630A Software Driver Hierarchy



Hardware Connection Formats

E6630A supports the TCPIP connection standard (represented by the "Network Layer" in [Figure 1-1](#) above) for LAN and HiSLIP instruments.

In general, modern driver technology hides the details of the hardware connection from the programmer, so your instrument's actual hardware connection is unlikely to have any significant effect on the optimal choice of programming tool, language or ADE.

VISA Driver Options

[Agilent I/O Libraries Suite](#) is a collection of libraries, Application Programming Interfaces (APIs) and utility programs. The I/O libraries (SICL, VISA, and VISA COM) enable instrument communication for a variety of development environments (Agilent VEE Pro, Microsoft Visual Studio, etc.) that are compatible with GPIB, USB, LAN, RS-232, PXI, AXIe, and VXI test instruments from a variety of manufacturers.

The suite's utility programs help you quickly and easily connect instruments to a computer.

The Agilent IO Libraries Suite includes the following libraries:

Item	Library Name	Documentation Location & Notes
1	Agilent Virtual Instrument Software Architecture (VISA) ¹	<p>The VISA API is a programming interface originally developed and standardized by the VXIplug&play Alliance (now the IVI Foundation) as an industry-wide standard for communicating with instruments over various hardware interfaces. The definition includes the standard <code>visa.h</code> header file for use with C and C++, which provides declarations for the <code>visa32.dll</code> library.</p> <p>Additionally, Agilent has developed the header files <code>visa32.cs</code> and <code>visa32.bas</code>, to permit the VISA DLL to be used with C#.NET and Visual Basic.NET respectively.</p> <p>For more information, see the VISA Documentation Help in the Agilent I/O Libraries Suite.</p>
2	VISA for the Common Object Model (VISA COM) ¹	<p>The VISA COM I/O API is a programming interface standardized by the IVI Foundation for communicating with instruments over various hardware interfaces.</p> <p>Agilent Technologies offers an implementation of the VISA COM I/O standard that is compatible with Agilent hardware as well as computer standard I/O interfaces. VISA COM I/O is an update of the older VISA C API to work in and with Microsoft's COM technology.</p> <p>For more information, see the VISA COM Help in the Agilent I/O Libraries Suite.</p>
3	Agilent Standard Instrument Control Library (SICL) ¹	<p>SICL is compatible only with Agilent interfaces, whereas VISA is an industry-wide standard.</p> <p>This library is not described further in this document. For more information, see the SICL Documentation Help in the Agilent I/O Libraries Suite.</p> <p>Note that SICL is supported only for the C and VB6 languages; there is no SICL support for .NET languages.</p>

1. Note that using VISA functions and SICL functions in the same I/O application is not supported.

Agilent VEE Pro

Agilent VEE (Visual Engineering Environment) Pro provides a graphical language and integrated development environment that permits efficient development of measurement and analysis solutions, while requiring minimal custom programming.

You can select and edit objects from pull-down menus or toolbars and connect them to each other by virtual wires to specify the program's data flow, mimicking the order of tasks you want to perform.

Agilent VEE Pro can communicate with any instrument from any vendor, using GPIB, LAN, USB, RS-232, VXI or LXI.

For further details, see the web page for [Agilent VEE Pro](#).

For information about using IVI Instrument Drivers with Agilent VEE, see [Agilent Application Note 1595](#).

Programming Documentation Roadmap

Most X-Series manuals and publications can be accessed via the Additional Documentation page in the instrument Help system, and are also included on the Documentation DVD shipped with the instrument. Exceptions are noted in [Table 1-2](#) below.

All documents can also be found online at the [Agilent X-Series Document Library](#).

Table 1-2 X-Series Documentation Resources

Resource	Description
X-Series Programmer's Guide (This document)	Provides general programming information on the following topics: <ul style="list-style-type: none"> • Introduction to Programming X-Series Applications • SCPI Programming Fundamentals • Program Samples Note that SCPI command descriptions for measurement applications are not in this document, but are in the User's and Programmer's Reference manuals for each measurement application (mode).
User's and Programmer's Reference manuals	Describes all SCPI commands for a measurement application (mode). Note that: <ul style="list-style-type: none"> • Each measurement application has its own User's and Programmer's Reference. • The content of this manual is duplicated in the instrument's Help file. That is, the context-sensitive help content for a key is identical to that in User's and Programmer's Reference manual for the same mode.
Embedded Help in the instrument	Describes all SCPI commands for a measurement application (mode), organized according to the front-panel key and softkey hierarchy. <p>Note that the content that you see in Help when you press a key is identical to that in the User's and Programmer's Reference for the same topic.</p>
Agilent X-Series Signal Analyzer: Getting Started Guide	Provides valuable sections related to programming including: <ul style="list-style-type: none"> • Licensing New Measurement Application Software - After Initial Purchase • Configuring instrument LAN Hostname, IP Address, and Gateway Address • Using the Windows Remote Desktop to connect to the instrument remotely This manual is shipped with the instrument as a printed document.
Agilent Application Notes	Printable PDF versions of pertinent application notes.
Agilent I/O Libraries Suite	The download package includes documentation describing the Agilent Virtual Instrument Software Architecture (VISA) library, and showing how to use it to develop I/O applications and instrument drivers on Windows PCs. <p>Not included on X-Series Documentation DVD.</p>



2 SCPI Programming Fundamentals

This chapter provides overall information on programming X-Series instruments using Standard Commands for Programmable Instruments (SCPI). Sections include:

- “SCPI Language Basics” on page 16
- “Where to find X-Series SCPI Command Definitions” on page 23
- “Simple SCPI Communication Methods” on page 27
- “Techniques for Improving Measurement Performance” on page 34

SCPI Language Basics

This section provides a basic introduction to the SCPI language. For more details about SCPI, see [IEEE Standard 488.2–1992](#).

Topics covered in this section include:

- “Command Keywords, Separators and Syntax” on page 16
- “Creating Valid Commands” on page 17
- “Special Characters in Commands” on page 17
- “Parameters in Commands” on page 19
- “Writing Multiple Commands on the Same Line” on page 21

Command Keywords, Separators and Syntax

Keywords, Parameters & Separators: A typical SCPI command is made up of keywords separated by colons. The keywords are followed by parameters that can be followed by optional units. The parameter list is separated from the command by a space.

Example: `:SENSe:FREQuency:STARt 1.5 MHZ`

Upper- vs. Lower-Case Usage: The instrument does not distinguish between upper and lower case letters. In the documentation, upper case letters indicate the short form of the keyword, whereas lower case letters indicate the long form of the keyword. Either form may be used in the command.

Example:

`:Sens:Freq:Star 1.5 mhz`

This is the same as

`:SENSE:FREQ:start 1.5 MHz`

NOTE The command `:SENS:FREQU:STAR` would **not** be valid because `FREQU` is neither the short, nor the long form of the command. Only the short and long forms of the keywords are allowed in valid commands.

Multiple SCPI commands on the same line: This is permissible if the commands are separated by a semicolon. See “Writing Multiple Commands on the Same Line” on page 21.

Initial Colon: In general, SCPI commands start with a colon, as shown above. You may choose to omit the initial colon, but, if you do so, note that SCPI rules for the interpretation of Compound Headers will be invoked by the command interpreter. For full discussion and examples of Compound Headers, see Appendix A of [IEEE Standard 488.2–1992](#). For examples, see “SCPI Termination and Separator Syntax Examples” on page 22.

Creating Valid Commands

Commands are not case-sensitive, and there are often many different ways of writing a particular command. These are examples of valid commands for a given command syntax:

Command Syntax	Sample Valid Commands
<code>:[SENSe:]BANDwidth[:RESolution] <freq></code>	<p>The following sample commands are all identical. They all cause the same result.</p> <ul style="list-style-type: none"> <code>:Sense:Band:Res 1700</code> <code>:BANDWIDTH:RESOLUTION 1.7e3</code> <code>:sens:band 1.7KHZ</code> <code>:SENS:band 1.7E3Hz</code> <code>:band 1.7kHz</code> <code>:bandwidth:RES 1.7e3Hz</code>
<code>:MEASure:SPECTrum[n]?</code>	<ul style="list-style-type: none"> <code>:MEAS:SPEC?</code> <code>:Meas:spec?</code> <code>:meas:spec3?</code> <p>The number 3 in the last meas example causes it to return different results than the commands above it. See the command description for more information.</p>
<code>[[:SENSe]:DETector[:FUNction] NEGative POSitive SAMPle</code>	<ul style="list-style-type: none"> <code>:DET:FUNC neg</code> <code>:Detector:Func Pos</code>
<code>:INITiate:CONTinuous ON OFF 1 0</code>	<p>The sample commands below are identical.</p> <ul style="list-style-type: none"> <code>:INIT:CONT ON</code> <code>:init:continuous 1</code>

Special Characters in Commands

Special Character	Meaning	Example
	A vertical stroke between parameters indicates alternative choices. The effect of the command is different depending on which parameter is selected.	<p>Command: <code>TRIGger:SOURce EXTernal INTernal LINE</code></p> <p>The choices are external, internal, and line.</p> <p>Ex: <code>TRIG:SOURCE INT</code> is one possible command choice.</p>

Special Character	Meaning	Example
	A vertical stroke between keywords indicates identical effects exist for both keywords. The command functions the same for either keyword. Only one of these keywords is used at a time.	Command: SENSe: BANDwidth BWIDth: OFFSet Two identical commands are: Ex1: SENSE: BWIDTH: OFFSET Ex2: SENSE: BAND: OFFSET
[]	keywords in square brackets are optional when composing the command. These implied keywords will be executed even if they are omitted.	Command: [SENSe:] BANDwidth[: RESolution]: AUT O The following commands are all valid and have identical effects: Ex1: bandwidth: auto Ex2: band: resolution: auto Ex3: sense: bandwidth: auto
< >	Angle brackets around a word, or words, indicates they are not to be used literally in the command. They represent the needed item.	Command: SENS: FREQ <freq> In this command example the word <freq> should be replaced by an actual frequency. Ex: SENS: FREQ 9.7MHz.
{ }	Parameters in braces can optionally be used in the command either not at all, once, or several times.	Command: MEASure: BW <freq>{, level} A valid command is: meas: BW 6 MHz, 3dB, 60dB

Parameters in Commands

There are four basic types of parameters: booleans, keywords, variables and arbitrary block program data.

Type	Description
OFF ON 0 1 (Boolean)	This is a two state boolean-type parameter. The numeric value 0 is equivalent to OFF. Any numeric value other than 0 is equivalent to ON. The numeric values of 0 or 1 are commonly used in the command instead of OFF or ON. Queries of the parameter always return a numeric value of 0 or 1.
keyword	The keywords that are allowed for a particular command are defined in the command syntax description.
Units	Numeric variables may include units. The valid units for a command depend on the variable type being used. See the following variable descriptions. The indicated default units will be used if no units are sent. Units can follow the numerical value with, or without, a space.
Variable	<p>A variable can be entered in exponential format as well as standard numeric format. The appropriate range of the variable and its optional units are defined in the command description.</p> <p>The following keywords may also be used in commands, but not all commands allow keyword variables.</p> <ul style="list-style-type: none"> • DEFault - resets the parameter to its default value. • UP - increments the parameter. • DOWN - decrements the parameter. • MINimum - sets the parameter to the smallest possible value. • MAXimum - sets the parameter to the largest possible value. <p>The numeric value for the function's MINimum, MAXimum, or DEFault can be queried by adding the keyword to the command in its query form. The keyword must be entered following the question mark.</p> <p>Example query: SENSE:FREQ:CENTER? MAX</p>

Variable Parameters

Type	Description
<integer>	An integer value with no units.
<real>	A floating point number with no units.
<freq> <bandwidth>	A positive rational number followed by optional units. The default unit is Hertz. Acceptable units include: Hz, kHz, MHz, GHz.

Type	Description
<time> <seconds>	A rational number followed by optional units. The default units are seconds. Acceptable units include: ks, s, ms, μ s, ns.
<voltage>	A rational number followed by optional units. The default units are Volts. Acceptable units include: V, mV, μ V, nV
<current>	A rational number followed by optional units. The default units are Amperes. Acceptable units include: A, mA, μ A, nA.
<power>	A rational number followed by optional units. The default units are W. Acceptable units include: kW, W, mW, μ W, nW, pW.
<ampl>	A rational number followed by optional units. The default units are dBm. Acceptable units include: dBm, dBmV, dB μ V.
<rel_power> <rel_ampl>	A positive rational number followed by optional units. The default units are dB. Acceptable units include: dB.
<percent>	A rational number between 0 and 100. You can either use no units or use PCT.
<angle> <degrees>	A rational number followed by optional units. The default units are degrees. Acceptable units include: DEG, RAD.
<string>	A series of alpha numeric characters.
<bit_pattern>	Specifies a series of bits rather than a numeric value. The bit series is the binary representation of a numeric value. There are no units. Bit patterns are most often specified as hexadecimal numbers, though octal, binary or decimal numbers may also be used. In the SCPI language these numbers are specified as: <ul style="list-style-type: none">• Hexadecimal, #Hdddd or #hdddd where 'd' represents a hexadecimal digit 0 to 9 and 'a' to 'f'. So #h14 can be used instead of the decimal number 20.• Octal, #Odddddd or #oddddd where 'd' represents an octal digit 0 to 7. So #o24 can be used instead of the decimal number 20.• Binary, #Bdddddddddddddd or #bdddddddddddddd where 'd' represents a 1 or 0. So #b10100 can be used instead of the decimal number 20.

Block Program Data

Some parameters consist of a block of data. There are a few standard types of block data. Arbitrary blocks of program data can also be used.

Type	Description
<trace>	<p>An array of rational numbers corresponding to displayed trace data. See the description of the <code>FORMat:DATA</code> command in the "Programming the Analyzer" chapter of any X-Series Users and Programmers Reference or online Help file for information about available data formats.</p> <p>A SCPI command often refers to a block of current trace data with a variable name such as: <code>Trace1</code>, <code>Trace2</code>, or <code>trace3</code>, depending on which trace is being accessed.</p>
<arbitrary block data>	<p>A block of data bytes. The first information sent in the block is an ASCII header beginning with <code>#</code>. The block is terminated with a semi-colon. The header can be used to determine how many bytes are in the data block. There are no units.</p> <p>A data query returns each block of data in the following format:</p> <pre>#DNNN<nnn binary data bytes>;</pre> <p>where <code>#DNNN</code> is the header. To parse this data:</p> <ol style="list-style-type: none"> 1. Read two characters (<code>#D</code>), where <code>D</code> tells you how many <code>N</code> characters follow the <code>D</code> character, 2. Read <code>D</code> characters, and convert to an integer that specifies the number of data bytes in the block, 3. Read <code>NNN</code> bytes into a real array. <p>Example: Header value = <code>#512320</code></p> <ul style="list-style-type: none"> • The first numeric character/digit (5) tells you how many additional digits there are in the header. • The 12320 means that 12,320 data bytes follow the header. • Divide the number of data bytes by the bytes/data point of the current data format, which is 8 for <code>REAL</code>, 64. Thus, in this example, there are $12320/8 = 1540$ data points in this block.

Writing Multiple Commands on the Same Line

Multiple commands can be written on the same line, reducing your code space requirement. To do this:















- Commands must be separated with a semicolon (;)
- If the commands are in different subsystems, the key word for the new subsystem must be preceded by a colon (:)
- If the commands are in the same subsystem, the full hierarchy of the command key words need not be included. The second command can start at the same key word level as the command that was just executed.

SCPI Termination and Separator Syntax Examples

The following are some examples of valid and invalid commands. The examples are created from a theoretical instrument with the simple set of commands indicated below:

```
[ :SENSe ] :POWer [ :RF ] :ATTenuation 40dB
:TRIGger [ :SEQuence ] :EXTernal [ 1 ] :SLOPe POSitive
[ :SENSe ] :FREQuency :STARt :POWer [ :RF ] :MIXer :RANGe [ :UPPer ]
```

Table 2-1 Examples of Valid and Invalid SCPI Commands

 Invalid Command	Problem
 Valid Command	
 PWR:ATT 40dB	The short form of POWER is POW, not PWR.
 POW:ATT 40dB	
 FREQ:STAR 30MHz;MIX:RANG -20dBm	The MIX:RANG command is in the same :SENSE subsystem as FREQ, but executing the FREQ command puts you back at the SENSE level. You must specify POW to get to the MIX:RANG command.
 FREQ:STAR 30MHz;POW:MIX:RANG -20dBm	
 FREQ:STAR 30MHz;POW:MIX RANG -20dBm	MIX and RANG require a colon to separate them.
 FREQ:STAR 30MHz;POW:MIX:RANG -20dBm	
 :POW:ATT 40dB;TRIG:FREQ:STAR 2.3GHz	:FREQ:STAR is in the :SENSE subsystem, not the :TRIGGER subsystem.
 :POW:ATT 40dB;:FREQ:STAR 2.3GHz	
 :POW:ATT?:FREQ:STAR?	:POW and FREQ are within the same :SENSE subsystem, but they are two separate commands, so they should be separated with a semicolon, not a colon.
 :POW:ATT?;:FREQ:STAR?	
 :POW:ATT -5dB;:FREQ:STAR 10MHz	Attenuation cannot be a negative value.
 :POW:ATT 5dB;:FREQ:STAR 10MHz	

Where to find X-Series SCPI Command Definitions

All X-Series SCPI commands are described in two locations: the Users & Programmers Reference manual for each application (PDF format), and the Embedded Help for each application (HTML Help format).

Reference Type	Usage & More Information
Users & Programmers Reference Manuals	<p>All available PDF manuals are included on the X-Series Spectrum Analyzer Documentation DVD, in the folder <code>\files</code>.</p> <p>You can also download all Users & Programmers Reference manuals from the Agilent web site, by using the hyperlinks in the Additional Documentation section of the instrument's Embedded Help.</p> <p>In the Users & Programmers References, SCPI command descriptions are organized by front-panel functionality, but you can also find a specific command by looking for it in the common or measurement-specific "List of SCPI Commands" chapters.</p>
Embedded Help	<p>The instrument's Embedded Help system contains context-sensitive reference information for each installed measurement application. To see the Help topic for any active function or key, press the green front-panel Help key when the measurement application is open. For more details of how to use Help as a SCPI command reference, see "Help System Features for SCPI Programmers" on page 23.</p> <p>In the Help files, SCPI command descriptions are organized by front-panel functionality, but you can also find a specific command by looking for it in the alphabetized List of SCPI Commands.</p> <p>All available Compiled Help Metafiles (CHMs) are also included on the X-Series Spectrum Analyzer Documentation DVD, in the <code>\help</code> subfolder. The CHM Help file for each measurement application has a name of the form <code><mode_name>.en-us.chm</code>.</p>

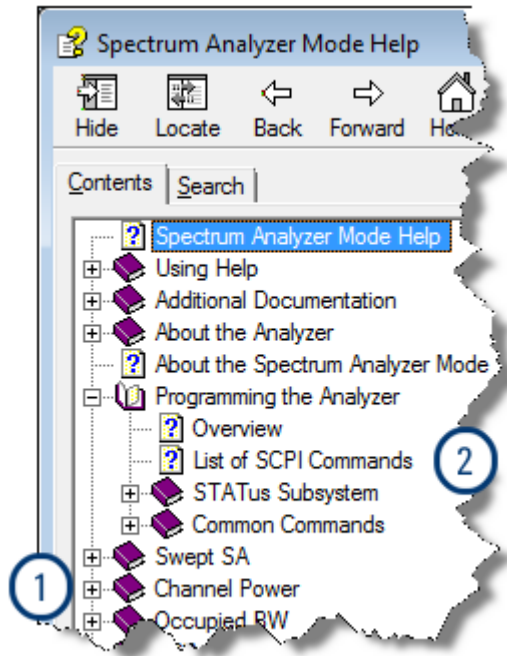
Help System Features for SCPI Programmers

Help System Contents Pane

The features described below are shown in the Help system Contents Pane (see [Figure 2-1](#)).

Figure 2-1

Example Help System Contents Pane





1. "Help Topic Content" on page 24
2. "List of SCPI Commands" on page 25

Help Topic Content

A typical Help topic is shown in Figure 2-2. Each Help topic includes:

- A description of the current active function or key,
- SCPI Command parameters, including limits, presets, variables, and queries,
- Associated Remote-Only commands (if any).

Figure 2-2 Example Help Topic - Scale/Div Topic

To exit, press  For more about Help, press 

Scale / Div

Sets the units per vertical graticule division on the display. This function is only available when Scale Type (Log) is selected and the vertical scale is power. When Scale Type (Lin) is selected, Scale/Div is grayed out.

Key Path	AMPTD Y Scale
Remote Command	<pre>:DISPlay:WINDow[1]:TRACe:Y [:SCALe]:PDIVision <rel_amp1> :DISPlay:WINDow[1]:TRACe:Y [:SCALe]:PDIVision?</pre>
Example	DISP:WIND:TRAC:Y:PDIV 5 DB
Dependencies	Scale/Div is grayed out in linear Y scale. Sending the equivalent SCPI command does change the Scale/Div, though it has no affect while in Lin.

List of SCPI Commands

The List of SCPI Commands is an alphabetically sorted list of all commands in the current measurement application. Each item shown is a hyperlink to the specific Help Topic that contains the command or query. See [Figure 2-3](#) for an example of a List of SCPI Commands.

Figure 2-3 Example List of Commands

**NOTE**

You can query the instrument for all supported SCPI commands in the current mode by sending the ":SYST:HELP:HEAD?" query. For details on how to query the instrument see "[Communicating SCPI Using Telnet](#)" on page 28.

Simple SCPI Communication Methods

This section describes some simple methods that you can use to create SCPI communication sessions between a computer and an X-Series instrument:

[“Connecting to the X-Host Using VISA Connection Strings” on page 27](#)

[“Communicating SCPI Using Telnet” on page 28](#)

[“Communicating SCPI using Agilent Connection Expert” on page 31](#)

Connecting to the X-Host Using VISA Connection Strings

If the E6630A is fully loaded (that is, if it includes both of the two sub-instruments that can be installed), it is necessary to communicate with the "Left" and "Right" sub-instruments separately, even though the physical LAN connection is to a single controller module shared by both.

Left Sub-Instrument Connections

The connection strings for the left sub-instrument are listed below.

Connection Type	Connection String & Usage
VXI-11/SICL	TCPIP0:: <hostname>::inst0::INSTR</hostname>
Socket:	TCPIP0:: <hostname>::5025::SOCKET 5025 is a default SCPI socket port number for the first X-Host instance and is also the port number specified in the config file for the "Left" X-Host instance. The instrument's IP address can be substituted for the hostname. Examples: TCPIP0:: myE6630A.soco.agilent.com::5025::SOCKET TCPIP0::148.5.96.172::5025::SOCKET</hostname>
HiSLIP	TCPIP0:: <hostname>::hislip0::INSTR</hostname>

Right Sub-Instrument Connections

The connection strings for the right sub-instrument are listed below.

Connection Type	Connection String & Usage
VXI-11/SICL	TCPIP0:: <hostname>::inst1::INSTR</hostname>
Socket:	TCPIP0:: <hostname>::5125::SOCKET 5125 is the port number specified in the config file for the "Right" X-Host instance. The instrument's IP address can be substituted for the hostname. Examples: TCPIP0:: myE6630A.soco.agilent.com::5125::SOCKET TCPIP0::148.5.96.172::5125::SOCKET</hostname>
HiSLIP	TCPIP0:: <hostname>::hislip1::INSTR</hostname>

Distinguishing Between the Sub-Instruments

Once connected, you can use the following SCPI commands to identify which xSA instance you are connected to:

Command	Returns
SYSTem:MODule:NAME?	This will return the name of the instance as displayed on the xSA main window. Example: "Left"
SYSTem:MODule:INDex?	This will return the index of this instance. This index is used as the device number in a VISA address ("hislip#"). Example: 0
SYSTem:MODule:MNEMonic?	This will return the mnemonic of the instance as used in the command line "/Process:<mnemonic>" argument. Example: "Left"

Communicating SCPI Using Telnet

You can communicate SCPI using a Telnet connection from a computer to the instrument. The following procedure describes how to connect a computer running Microsoft Windows to the instrument.

To complete the procedure, you will need to know the IP address of the instrument, which you can obtain by ["Determining Instrument IP Address" on page 30](#).

TIP In newer versions of Microsoft Windows (Windows Vista and Windows 7), you may first need to enable the Telnet client. See ["Enabling Telnet in Windows" on page 31](#).

NOTE In addition to the procedure described below, you can open a Telnet connection with the instrument using an internet connection to the instrument's Embedded Web Server. This procedure is described in the [Agilent X-Series Signal Analyzer: Getting Started Guide](#).

To initiate a Telnet session and communicate SCPI using the LAN connection to the instrument:

Step	Action	Notes
1	Obtain the IP address of the instrument	If necessary, you can obtain it via the procedure described in "Determining Instrument IP Address" on page 30 .
2	Ensure that the instrument Telnet socket is On	Press System, I/O Config, SCPI LAN , and make sure SCPI Telnet (Port 5023) is toggled to On .

Step	Action	Notes
3	Enable computer's Telnet client if required	See "Enabling Telnet in Windows" on page 31.
4	Test LAN connection	<ol style="list-style-type: none"> 1. On a Microsoft Windows computer, in the Taskbar select Start, Run, and type "cmd" to open a DOS session. 2. Enter the DOS command "ping", a single space and the IP address of the instrument, and press Enter. The results should resemble those shown in Figure 2-4. If the LAN connection is working, you will see statistics for Packets Sent and Packets Received. 3. In the DOS window, type: "telnet <instrument_IP_address> 5023", then press Enter. A Telnet window opens with a Welcome answerback from the instrument Host Name, and the command prompt changes to "SCPI>". You can enter any valid SCPI command at the prompt and receive responses to queries sent. <p>NOTE Port 5023 in the command given above is for the left sub-instrument; if the E6630A includes both sub-instruments, use 5123 for the right one.</p>

Figure 2-4 Command Window and ping Command results

```

C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings>ping 255.255.252.07

Pinging 255.255.252.07 with 32 bytes of data:

Reply from 255.255.252.07: bytes=32 time=16ms TTL=128
Reply from 255.255.252.07: bytes=32 time<1ms TTL=128
Reply from 255.255.252.07: bytes=32 time=1ms TTL=128
Reply from 255.255.252.07: bytes=32 time=1ms TTL=128

Ping statistics for 255.255.252.07:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 16ms, Average = 4ms

C:\Documents and Settings>

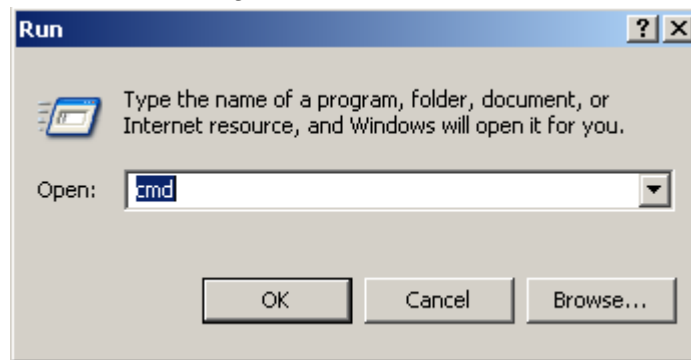
```

Determining Instrument IP Address

1. If necessary, close the Agilent Signal Analyzer application, by selecting **File > Exit** from the front panel and softkey menu, then confirming that you want to close the application.
2. When you can see the Windows desktop, move the cursor to the bottom of the screen using a mouse or the keyboard, to reveal the Windows Taskbar. In the Windows Taskbar, click **Start, Run**.
3. In the Window Run Dialog (shown in [Figure 2-5](#)), type "cmd" then click **OK** or press **Enter** to open a DOS command window.

Figure 2-5

Windows Run Dialog



4. At the DOS command prompt, enter "ipconfig", and press **Enter**.

The results should resemble the window shown in [Figure 2-6](#). The IP Address is listed under Ethernet adapter Local Area Connection.

Figure 2-6 Command Window and ipconfig Results

```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings>ipconfig

Windows IP Configuration

Ethernet adapter Wireless Network Connection:

    Media State . . . . . : Media disconnected

Ethernet adapter {2D7F75871ADCE-4A38-8D96-2D7F75871ADA}:

    Connection-specific DNS Suffix . :
    IP Address . . . . . : 0.0.0.0
    Subnet Mask . . . . . : 0.0.0.0
    Default Gateway . . . . . :

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . : aoilent.com
    IP Address . . . . . : 255.255.252.0
    Subnet Mask . . . . . : 255.255.252.0
    Default Gateway . . . . . : 255.255.252.0

C:\Documents and Settings>

```

Enabling Telnet in Windows

In newer versions of Microsoft Windows (Windows Vista and Windows 7), the Telnet client is disabled by default. To enable the Telnet client, do the following:

Step	Actions
1. Open Windows Control Panel	From the Windows Start menu, select Control Panel .
2. Select Programs	
3. Display Windows Features dialog	In the Programs and Features group, click Turn Windows features on or off . The Windows Features dialog appears.
4. Enable Telnet client	In the listbox, locate Telnet client and check its checkbox. Click OK .

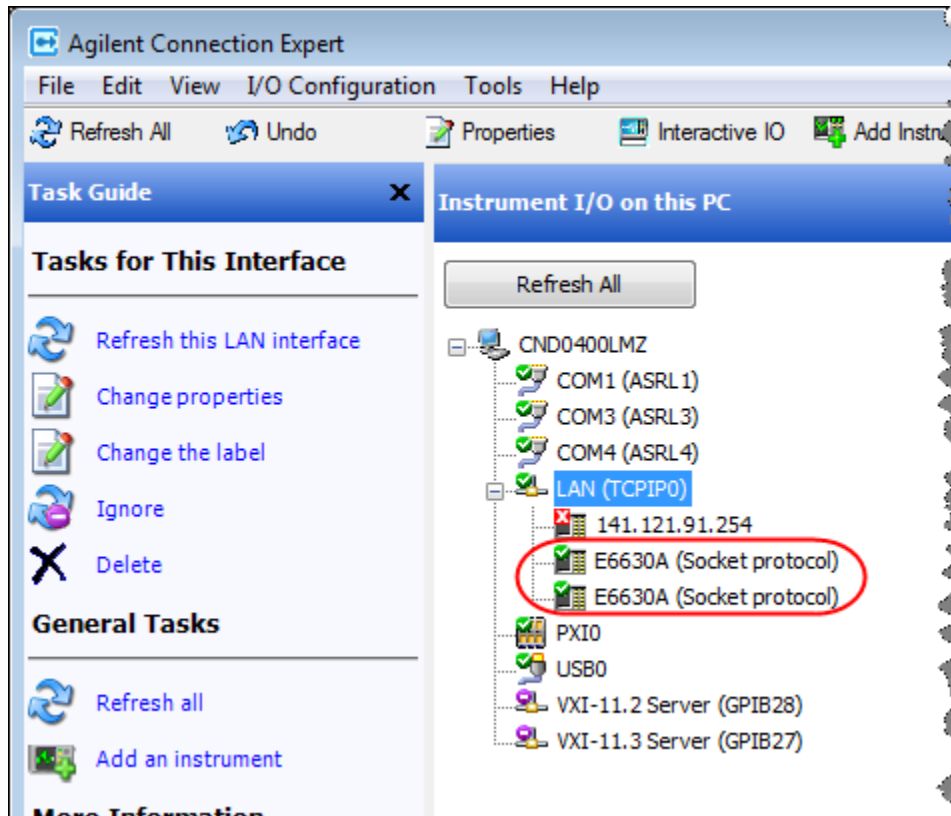
Communicating SCPI using Agilent Connection Expert

You can use Agilent Connection Expert to communicate with devices on any supported network type. Agilent Connection Expert is installed as part of the [Agilent I/O Libraries Suite](#).

SCPI Programming Fundamentals
Simple SCPI Communication Methods

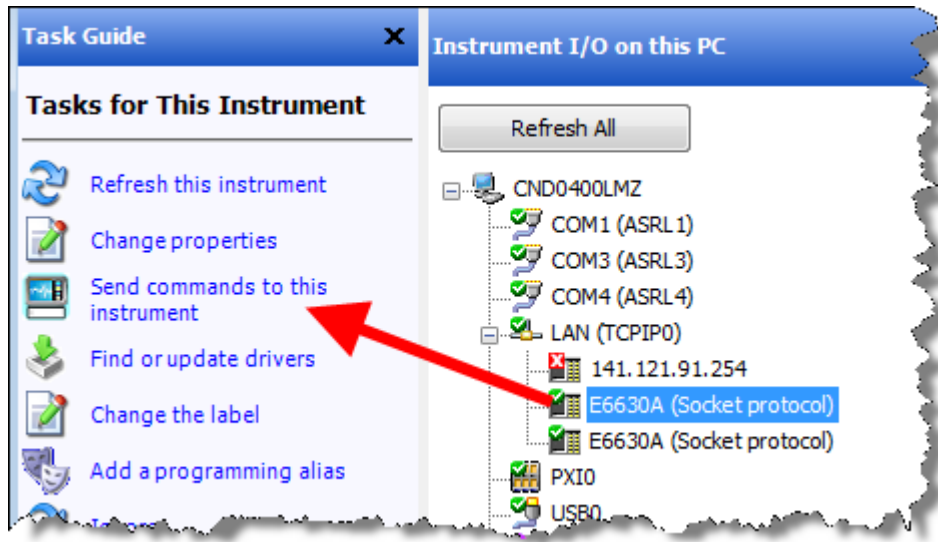
Figure 2-7 below shows part of the Agilent Connection Expert main screen, with two E6630A instruments connected via LAN (actually, they are the left and right sub-instruments of the same E6630A).

Figure 2-7 Agilent Connection Expert Main Screen



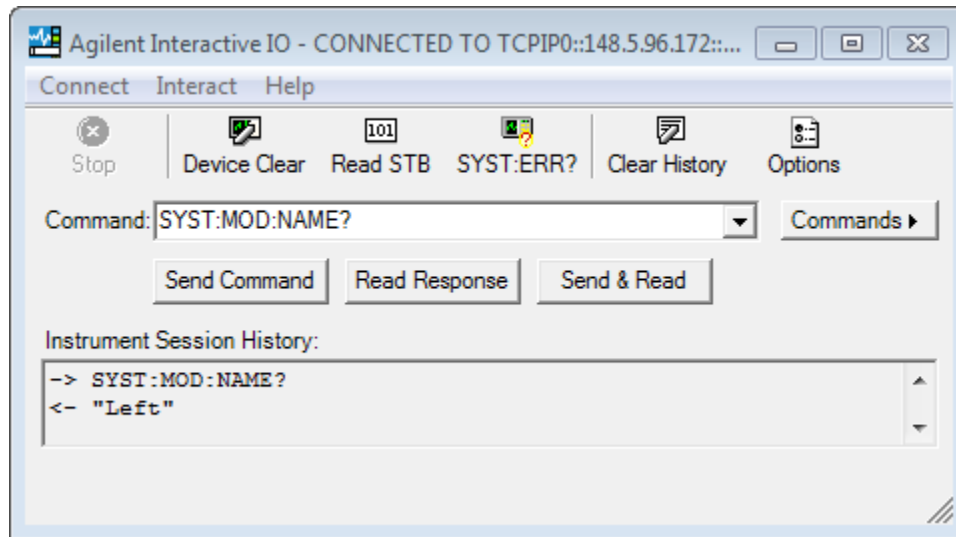
When you click on the on the first of the E6630A instrument icons in this example, the content of the Task Guide panel on the left changes to "Tasks for This Instrument", which includes the selection "Send commands to this instrument", as shown in Figure 2-8 below.

Figure 2-8 Tasks for This Instrument



If you click the item "Send commands to this instrument", the Agilent Interactive IO dialog appears as shown in Figure 2-9 below, which allows you to send SCPI commands to the instrument and read the responses. In the example illustrated, the query SYST:MOD:NAME? is sent, to confirm that the selected instrument is the left sub-instrument of the E6630A.

Figure 2-9 Agilent Interactive IO



For full details of how to use these features, open the Agilent Connection Expert main screen (as shown in Figure 2-7) and select **Help > Connection Expert Help** from the menu.

Techniques for Improving Measurement Performance

This section describes several programming techniques that can improve speed and efficiency. Most, but not all, of these techniques relate to SCPI program design.

- [“Turn off Display Updates” on page 34](#)
- [“Use Binary Data Format instead of ASCII” on page 34](#)
- [“Minimize the Number of Bus Transactions” on page 34](#)
- [“Minimize DUT/instrument Setup changes” on page 35](#)
- [“Use Video or External Triggering for Single Burst Signals” on page 35](#)
- [“To make Power Measurements on Multiple Bursts or Slots use CALCulate:DATA<n>:COMPRESS?” on page 36](#)

Turn off Display Updates

When the instrument is being operated remotely, there is no need to display data on the instrument screen. Display updates slow down the measurement, so measurement speed may be increased by switching off updates.

Send `:DISPlay:ENABle OFF` to turn off the display. In this case, data remains visible, but will no longer be updated.

Use Binary Data Format instead of ASCII

The ASCII data format is the instrument default, since it is easier for humans to read and is required by SCPI for `*RST`. However, data input/output is faster using the binary formats.

`:FORMat:DATA REAL, 64` selects the 64-bit binary data format for all numerical data queries. (The `REAL, 32` format, which is smaller and somewhat faster, should only be used if you do not require full data resolution. Some frequency data may require full 64 bit resolution.)

If you are using a PC rather than UNIX, you may need to change the byte order to little-endian, by sending `:FORMat:BOReR SWAP`. For details, see the "Programming the Analyzer" chapter of any X-Series Help file or Users & Programmers Reference PDF.

When using the binary format, data is sent in a block of bytes prefixed by an ASCII header. For details of the block format, see [“Block Program Data” on page 21](#).

Minimize the Number of Bus Transactions

When you are using the GPIB bus for control of your instrument, each transaction requires driver overhead and bus handshaking, so minimizing these transactions reduces the time used.

- You can reduce bus transactions by sending multiple SCPI commands per transaction. See [“Writing Multiple Commands on the Same Line” on page 21](#).
- When making the same measurement multiple times with small changes in the measurement setup, use the `READ` command, which is faster than using `INITiate` and `FETCh`.

- When changing the frequency and making a measurement repeatedly, you can reduce transactions by sending the optional frequency parameter with the `READ` query (for example, `READ:<meas>? {<freq>}`). These optional parameters are not available in certain modes, such as Spectrum Analyzer or Phase Noise.

The `CONFigure/MEASure/READ` commands for some measurements allow you to send center frequency setup information along with the command (for example, `MEAS:PVT? 935.2MHz`). This sets the Power vs. Time measurement to its defaults, then changes the center frequency to 935.2 MHz, initiates a measurement, waits until it is complete and returns the measurement data.

- When doing bottom/middle/top measurements on Base Stations, you can reduce transactions by making a time slot active at each of the B,M,T frequencies. Then, issue three measurements at once in the programming code and retrieve three data sets with just one bus transaction pair (write, read).

For example, send `READ:PFER? <Freq_bottom>; PFER? <Freq_middle>; PFER? <Freq_top>`. This single transaction initiates three different phase and frequency error measurements at each of the three different frequencies provided and returns three sets of data.

Minimize DUT/instrument Setup changes

- Some instrument setup parameters are common to multiple measurements, making it possible to organize the test process in such a way as to minimize setup changes. If the process involves nested loops, make sure that the innermost loop is the fastest. Also, check whether the loops could be nested in a different order to reduce the number of parameter changes as you step through the test.
- If you must switch between measurements, remember that if you have already set your Meas Setup parameters for a measurement, and you want to make another one of these measurements later, you should use the query `READ:<meas>?`.

The `MEASure:<meas>?` command resets all the settings to the defaults, while `READ` changes back to that measurement without changing the setup parameters from the previous use.

- If you must switch between measurements, remember that **Mode Setup** parameters remain constant across all the measurements in a given mode (for example, Center/Channel Frequency, Amplitude, Radio Standard, Input Selection, Trigger Setup). You do not need to re-initialize these parameters each time you change to a different measurement.

Avoid Unnecessary Use of *RST

Remember that while `*RST` does not change the current Mode, it presets all the measurements and settings to their factory defaults. This forces you to reset the instrument's measurement settings even if they use similar mode setup or measurement settings. See [“Minimize DUT/instrument Setup changes” on page 35](#).

Note also that `*RST` may put the instrument in Single measurement/sweep for some modes.

Use Video or External Triggering for Single Burst Signals

For a non-repetitive or single burst signals, use the IF (video) trigger or external trigger, depending on what you have available.

To achieve consistent triggering and valid data for a single burst measurement, you must synchronize the triggering of the DUT with the instrument. You should use the instrument's internal status system for this.

SCPI Programming Fundamentals

Techniques for Improving Measurement Performance

The first step in this process is to initialize the status register mask to look for the “waiting for trigger” condition (bit 5). Use `:STATus:OPERation:ENABle 32`

Then, in the measurement loop:

1. Send query `:STATus:OPERation:EVENT?` to clear the current contents of the Operation Event Register.
2. Send query `:READ:PVT?` to initiate a measurement (in this example, for GSM Power versus Time) using the previous setup. The measurement then waits for the trigger.

Make sure attenuation is set manually. Do **not** use automatic attenuation, as this requires an additional burst to determine the proper attenuation level before the measurement can be made.

3. Create a small loop that polls the instrument for a status byte value of 128. Then wait 1 msec (or 100 ms if the display is enabled) before checking again, to minimize bus traffic. Repeat these two commands until the condition is set, to ensure that the trigger is armed and ready.
4. Trigger the DUT to send the burst.
5. Retrieve the measurement data.

To make Power Measurements on Multiple Bursts or Slots use `CALCulate:DATA<n>:COMPRESS?`

The Calculate/Compress Trace Data Query is the fastest way to measure power data for multiple bursts/slots. (For details of the command, see the Programming the Analyzer chapter of any X-Series Help file or Users & Programmers Reference PDF.) There are two reasons for this:

1. It can be used to measure data across multiple, consecutive slots/frames with just one measurement, instead of a separate measurement on each slot,
2. It can pre-process and/or decimate the data so that you only return the information that you need, which minimizes data transfer to the computer.

Example: you want to do a power measurement for a GSM base station where you generate a repeating frame with 8 different power levels. Using the Waveform measurement, you can gather all the data with a single `CALC:DATA:COMP?` acquisition.

By sending `:CALC:DATA2:COMP? MEAN,25us,526us,579.6us,8` you can measure the mean power in those bursts. This single command measures the data across all 8 frames, locates the first slot/burst in each of the frames, calculates the mean power of those bursts, then returns the resulting 8 values. The sequence of commands is as follows:

Step	Command	Action
1	<code>:CONF:WAV</code>	Switch to Waveform measurement
2	<code>:WAV:BAND 300khz</code>	Set resolution bandwidth to 300 kHz
3	<code>:WAV:SWE:TIME 5ms</code>	Set sweep time to 5 milliseconds
4	<code>:WAV:BAND:TYPE FLAT</code>	Select flat filter type

Step	Command	Action
5	<code>:WAV:DEC 4;DEC:STAT ON</code>	Select a decimation of 4, and turn on decimation. This reduces the amount of data that must be transferred.
6	<code>:INIT</code>	Initiate measurement and acquire data
7	<code>:CALC:DATA2:COMP? MEAN,25us,526us,579.6us,8</code>	Retrieve the desired data

More Hints & Tips

For more information about optimizing measurement speed using X-Series instruments, see [Agilent Application Note 1583](#).



3 Developing and Deploying VISA Projects

This chapter provides a brief overview of the requirements for development and deployment of Virtual Instrument Software Architecture (VISA) programming projects using various languages. Sections include:

- “Programming in Visual Basic 6 with VISA” on page 39
- “Programming in C or C++ with VISA” on page 39
- “Programming with Microsoft .NET and VISA” on page 40
- “Requirements for Deploying a VISA Project” on page 41

For an overview of the relationship between VISA and other programming tools and drivers, see “X-Series Programming Options” on page 10.

Programming in Visual Basic 6 with VISA

See the VISA online Help section “Using the VISA C API in Microsoft Visual Basic 6”, in the [Agilent VISA Help](#).

Location of Header Files

The required header files `visa32.bas` and `agvisa32.bas` can be found in:

```
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\agvisa\include
```

or

```
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\include
```

Programming in C or C++ with VISA

Full details of X-Series programming in C and C++ are provided in the [Agilent I/O Libraries Suite](#).

Location of Header Files & Libraries

The header file `visa.h` can be found in:

```
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\include
```

Developing and Deploying VISA Projects

Programming with Microsoft .NET and VISA

Programs must link to the VISA libraries `visa32.lib` or `agvisa32.lib`, located in subfolders of:

```
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\lib
```

or

```
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\Lib_x64
```

For more details, see the VISA online Help section "VISA Directories", in the [Agilent VISA Help](#).

Programming with Microsoft .NET and VISA

The [IVI Foundation](#) defines the standard `visa.h` header file for use in C and C++, which provides declarations for the `visa32.dll` C DLL. This header file is distributed by Agilent Technologies, among others. The Foundation also defines header file `visa32.bas` for Microsoft Visual Basic 6. However, there are at present no officially defined header files for programming with the VISA C API in the Microsoft .NET technology languages, such as C# and Visual Basic.NET.

Therefore, Agilent has defined and developed the redistributable .NET header files `visa32.cs` (for C#) and `visa32.vb` (for Visual Basic.NET), to allow programmatic access to the VISA C API from the two most popular .NET languages. To use the VISA C API in a .NET project, include the appropriate file in your project. The compiled .NET assembly will then have all the information it needs to use the VISA C Library (`visa32.dll` or `visa64.dll`).

For programmers accustomed to the VISA-C API, or those not familiar with COM, use of Agilent's .NET header files may offer a preferable approach, because it avoids the overhead of the VISA COM implementation and exposes VISA functionality in a more familiar style.

VISA has specifications for API versions in C and COM, so there are two ways to work with VISA in your .NET applications: via the wrapper already written by Agilent around the C library, or via the Visa COM Interop.

Location of Header Files

The header files `visa32.cs` and `visa32.vb` can be found in:

```
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\agvisa\include
```

or

```
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\include
```

For more details, see the VISA online Help section "Using the VISA C API in Microsoft .NET", in the [Agilent VISA Help](#).

Requirements for Deploying a VISA Project

The only VISA-specific system requirements for deploying your compiled programs on other machines are:

- A valid `visa32.DLL` must be in the system's PATH environment variable.
- The resource address you are trying to open must exist on the system and be configured for the `visa32.DLL` that is found first during the Windows DLL search.

Additionally, you must satisfy the normal .NET requirements, such as having an appropriate version of the .NET framework installed on the deployed systems. Obviously, any other software libraries your program uses at runtime must also be installed.

Multiple VISA DLL Versions

Because each VISA vendor installs its version of the VISA DLL, the VISA DLL on your deployed system may differ from the one with which you developed your application. When multiple vendors' VISA implementations are present, the DLL used is the one that is found first using Microsoft Windows' DLL search rules.

If you developed your program using Agilent VISA, and you wish to ensure that your program uses Agilent VISA even if other VISA implementations are on your deployed systems, you can change the DLL name in all of the method declarations in `visa32.cs` or `visa32.vb` from "`visa32.DLL`" to "`agvisa32.DLL`". This will prevent your program from working with any other vendor's VISA implementation, and will ensure that, if multiple VISA DLLs are installed on the system, your program will use the Agilent DLL implementation.



4 Program Samples

The program samples described here were written for use on a PC running Microsoft Windows.

The description of each sample includes its function, operational details, programming language and driver usage, and the sample file name or root directory.

This chapter is divided into the following sections:

[“Where to find Sample Programs” on page 43](#)

[“Programming Samples” on page 44](#)

Where to find Sample Programs

- Relevant samples are found on the following Agilent Technologies, Inc. web site, in the table of programming examples under "X-Series (PXA/MXA/EXA) Signal Analyzers". Not all samples listed there pertain to the E6630A; those that do are described in this chapter.

http://www.agilent.com/find/sa_programming

- Program samples installed by the [Agilent I/O Libraries Suite](#) may be found (after installation) by opening the Agilent Connection Expert and selecting **Help > Programming Samples** from the menu,

(The Agilent I/O Libraries Suite samples are **not** described in this document, and are in general **not** specific to the E6630A.)

Programing Samples

Samples are available for the following programming languages and development environments:

- [Visual Basic 6](#)
- [C, C++](#)
- [C#.NET & Visual Studio 2010](#)
- [Agilent VEE Pro](#)
- [LabVIEW](#)

Matrix of Program Sample Functionality and Programming Language

In the table below, availability of program samples for each function/language is indicated by page number references. If no page number reference is provided, then there is no available sample for the given functionality in the specified language.

Function	Visual Basic 6	C, C++	C#.NET	VEE	LabVIEW
Transfer Screen Images	45		47	48	48
Serial Poll for Sweep Complete		46	47		
Trace Detector/Couple Markers		46			

Visual Basic 6

NOTE In some cases, Visual Basic 6 files with the extension `.bas` have been renamed with the extension `.bas.txt`, to avoid possible instrument security warnings generated by the `.bas` extension. To use these files in Visual Basic 6, rename them by removing the `.txt` portion of the extension.

The sample uses the VISA driver.

Transfer Screen Images

Function	Transfer Screen Images from the instrument
Description	This example demonstrates how to: <ol style="list-style-type: none">1. Store the current screen image in instrument memory as <code>"D:\PICTURE.PNG"</code>2. Transfer the memory image file via GPIB or LAN3. Store the transferred image in the computer's current directory as <code>"C:\PICTURE.PNG"</code>4. Delete the instrument memory file <code>"D:\PICTURE.PNG"</code>
Language	Visual Basic 6
File name	<code>mxa_screen.bas</code>

C, C++

The samples provided are console applications written in C, but these should also be compilable by most C++ compilers.

1. [Serial Poll for Operation Complete](#)
2. [Trace Detector/Couple Markers](#)

These samples use the VISA driver.

Serial Poll for Operation Complete

Function	Serial Poll for Sweep Complete
Description	This example demonstrates how to: <ol style="list-style-type: none">1. Modify the timeout value2. Initiate a sweep3. Poll the instrument to determine when the operation is complete4. Query and report the sweep result
Language	C
File name	<code>mxs_sweep.c</code>

Trace Detector/Couple Markers

Function	Trace Detector/Couple Markers
Description	This example demonstrates how to: <ol style="list-style-type: none">1. Set various types of trace (Max Hold, Clear Write, Min Hold)2. Relate markers to specified traces3. Couple markers <p>NOTE The instrument supports multiple simultaneous detectors (for example, peak detector for max hold, sample for clear and write, and negative peak for min hold).</p>
Language	C
File name	<code>mxs_tracecouple.c</code>

C#.NET & Visual Studio 2010

The samples provided are written in C# for Visual Studio 2010 (.NET version 4.5),

1. [Transfer Screen Images](#)
2. [Serial Poll for Sweep Complete](#)

These samples use the VISA driver.

Transfer Screen Images

Function	Capture and transfer Screen Images from the instrument
Description	This example demonstrates how to: <ol style="list-style-type: none">1. Store the current screen image as a PNG file on the instrument's D: drive, with a user-specified name2. Retrieve the screen image data from the instrument as a Program Data Block3. Analyze the header of the Program Data Block and extract the PNG bitmap from the block4. Store the extracted bitmap as a PNG file in the computer's current directory, with the same user-specified name5. Delete the PNG file that was stored on the instrument's D: drive
Language	C#
Project Folder	vs2010.net/x_screencapture

Serial Poll for Sweep Complete

Function	Serial Poll for Sweep Complete
Description	This example demonstrates how to: <ol style="list-style-type: none">1. Modify the timeout value2. Initiate a sweep3. Poll the instrument to determine when the operation is complete4. Query and report the sweep result
Language	C#
Project Folder	vs2010.net/x_sweep

Agilent VEE Pro

Transfer Screen Images

Function	Transfer Screen Images from the instrument
Description	This example demonstrates how to: <ol style="list-style-type: none">1. Store the current screen image in instrument memory as "D:\mxascr.PNG"2. Transfer the memory image file via GPIB3. Store the transferred image on the computer, in a user-specified directory, as "capture.gif"4. Delete the instrument memory file "D:\mxascr.PNG"
Language	Agilent VEE Pro
File name	mxa_screenshot.vee

LabVIEW

Transfer Screen Images

Function	Transfer Screen Images from the instrument
Description	The program retrieves screen capture data from the instrument via GPIB, then writes the contents of the binary block to a file, removing the header information before writing it. It uses the VISA protocol to communicate with the instrument.
Language / Driver	LabVIEW/ VISA
File name	MXA Screen Capture via GPIB.llb



A: References

Documents & Web Sites

- 1. IEEE Standard 488.2–1992**
IEEE Standard Codes, Formats, Protocols, and Common Commands for Use With IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation
May be downloaded in Acrobat (PDF) format from:
ieeexplore.ieee.org/iel1/2839/5581/00213762.pdf?arnumber=213762
- 2. IVI Foundation**
(Interchangeable Virtual Instrument Foundation)
<http://www.ivifoundation.org/default.aspx>
- 3. Agilent X-Series Document Library**
Follow the hyperlink:
http://www.agilent.com/find/E6630A_manuals
- 4. Agilent X-Series Signal Analyzer: Getting Started Guide**
Agilent Technologies Inc. 2008-2013. Part Number: subject to change as document is revised.
A printed copy of this document is supplied with each E6630A test set.
It is also available in Acrobat (PDF) form:
 - on the instrument's disk drive at the following location:
C:\Program Files\Agilent\SignalAnalysis\Infrastructure\Help\bookfiles\getstart.pdf
 - via download from:
http://www.agilent.com/find/E6630A_manuals

5. **Agilent I/O Libraries Suite**

Agilent Technologies Inc.

All Agilent VISA, VISA COM, SICL and 488 documentation is included in HTML Help (CHM) format in the Agilent I/O Libraries Suite installer, which may be downloaded from:

www.agilent.com/find/iosuite

After installing the libraries suite, you can access the help by clicking the IO taskbar icon, then selecting Documentation > API Documentation > VISA Documentation from the popup menus.

6. **Agilent VISA Help**

After installing the [Agilent I/O Libraries Suite](#), you can access the VISA Help CHM by clicking the IO taskbar icon, then selecting Documentation > API Documentation > VISA Documentation from the popup menus.

Alternatively, you can find the CHM at the following disk location:

C:\Program Files\Agilent\IO Libraries Suite\Visa.chm

7. **Agilent IVI (Instrument) Drivers**

Installation packages for the Signal Analyzer class driver ("Base Driver"), and instrument-specific drivers, may be downloaded from the "Signal Analyzer Instrument Drivers" page at:

<http://www.agilent.com/find/sa-ivi>

8. **Agilent Application Note 1583**

"Maximizing Measurement Speed with Agilent's X-Series Signal Analyzers"

May be downloaded in Acrobat (PDF) format from:

<http://cp.literature.agilent.com/litweb/pdf/5989-4947EN.pdf>

9. **Agilent Application Note 1595**

"How to Use IVI-COM Drivers in Agilent VEE Pro 8.0"

May be downloaded in Acrobat (PDF) format from:

<http://cp.literature.agilent.com/litweb/pdf/5989-6914EN.pdf>

10. **Agilent VEE Pro**

For links to all available information, see:

www.agilent.com/find/vee

Developer Resources

Agilent Developer Network (ADN)

This website offers a one-stop shop, with links to Instrument Drivers, Example Programs, Product Downloads, Evaluations, Demos, and resources for contacting Agilent regarding development issues:

<http://www.agilent.com/find/adn>

Technical Support

Navigate to the web page below, then select the Technical Support tab for links to all available documentation for the product:

<http://www.agilent.com/find/E6630A>