Keysight M9290A CXA-m PXIe Signal Analyzer

Notice: This document contains references to Agilent. Please note that Agilent's Test and Measurement business has become Keysight Technologies. For more information, go to www.keysight.com

Programmers' Guide

KEYSIGHT TECHNOLOGIES

Notices

Manual Part Number

M9290-90030

Publication Date

October 2014

Published in China

Keysight Technologies Inc.
No 116 Tianfu 4th Street
Chengdu, China 610041

Warranty

Technology Licenses

Restricted Rights Legend

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.

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.keysight.com/find/cxa-m

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

http://www.keysight.com/find/emailupdates

Information on preventing instrument damage can be found at:

http://www.keysight.com/find/PreventingInstrumentRepair

Is your product software up-to-date?

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

http://www.keysight.com/find/techsupport

Table of Contents

1 Introduction to Programming X-Series Applications

How to Use this Manual

This chapter provides overall information regarding programming of CXA-m Signal Analyzer, and how to use the programming documentation provided with the product.

This chapter includes the following topics:

The second chapter, “SCPI Programming Fundamentals” on page 17, 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 37 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 41 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.

TCP/IP Address Configuration

CXA-m signal analyzer support the TCPIP connection standards (represented by the "Network Layer" in Figure 1-1 above) for LAN and HiSLIP (High-Speed LAN Instrument Protocol) 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.

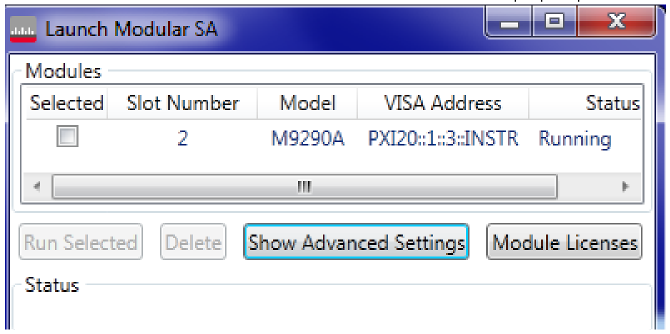
There are two different connection configuration to control a CXA-m signal analyzer in chassis.

- Local Control - The controller is a modular installed in the chassis with CXA-m signal analyzer or The controller is a PC and there is not any controller module installed in chassis. Both of the two connections use the PCI-PCI bridge for data communication.
- Remote Control - The controller is a modular installed in the chassis and an external PC controls the CXA-m signal analyzer via LAN connection as a remote controller.

Address Setting in Local Control

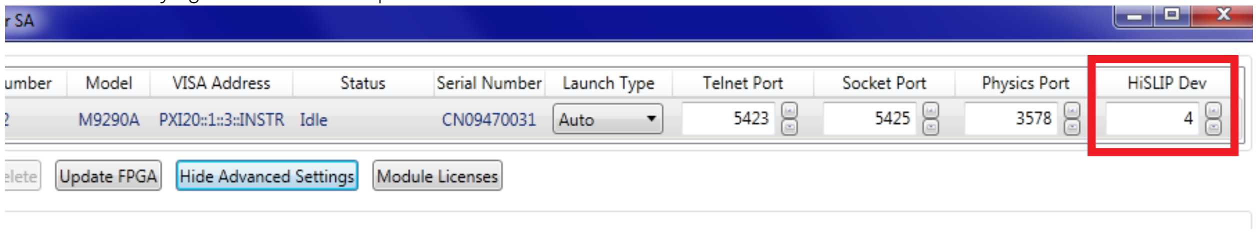
Please refer to the procedures below to set the TCP/IP address in local control.

1. Run **LaunchModularSA.exe** in controller. The pop-up window is shown as below.



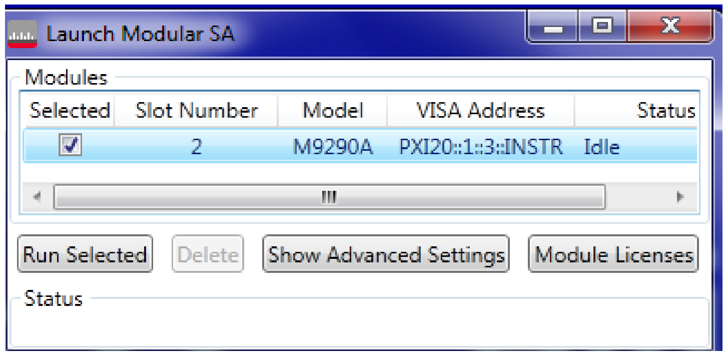
Selected	Slot Number	Model	VISA Address	Status
☐	2	M9290A	PXI20::1::3::INSTR	Running

2. Click **Show Advanced Settings**. For this analyzer, the "HiSLIP Dev" value is 4. This value will be used to set the remote name in Keysight Connection Expert.



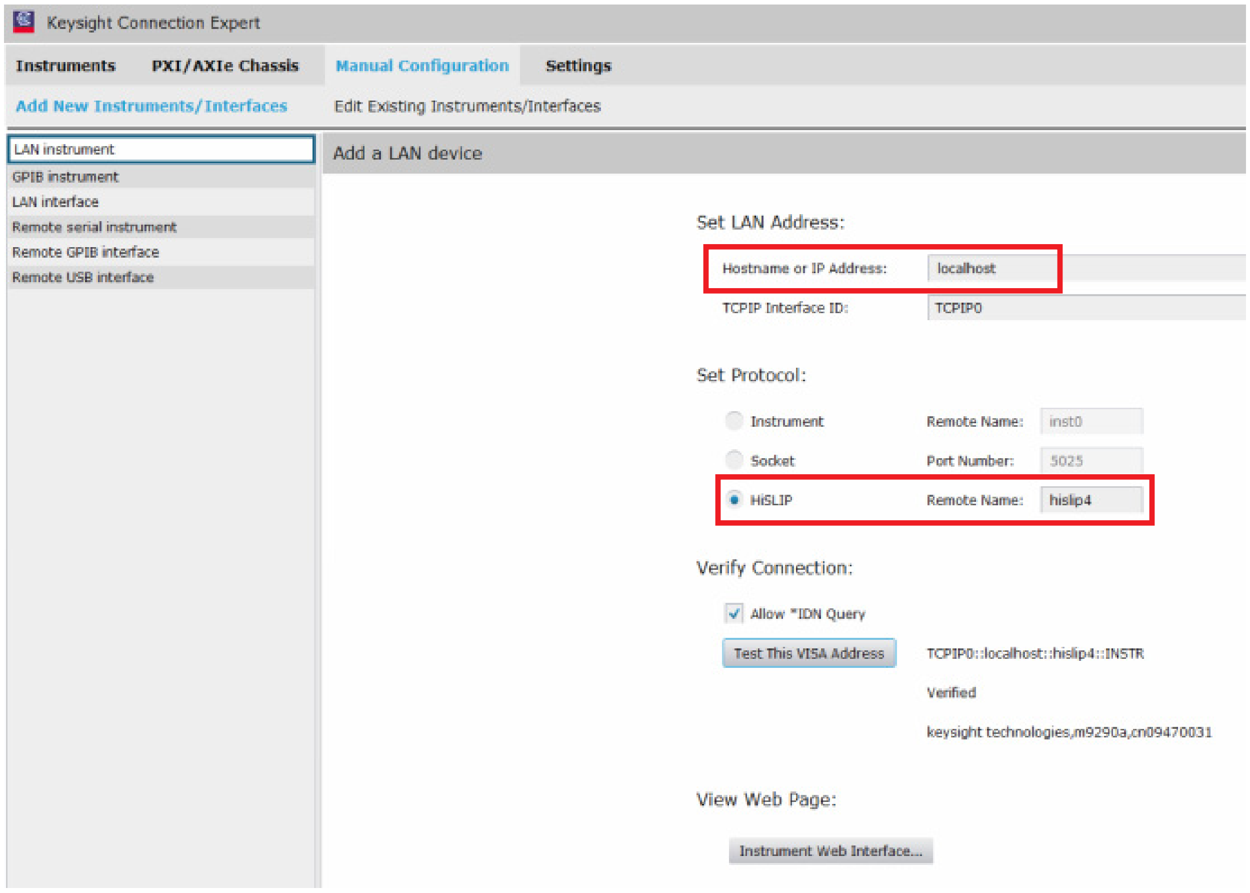
umber	Model	VISA Address	Status	Serial Number	Launch Type	Telnet Port	Socket Port	Physics Port	HiSLIP Dev
2	M9290A	PXI20::1::3::INSTR	Idle	CN09470031	Auto	5423	5425	3578	4

3. Check the box named **Selected** and click **Run Selected** to run the Modular SA application.



Selected	Slot Number	Model	VISA Address	Status
☑	2	M9290A	PXI20::1::3::INSTR	Idle

4. Run Keysight Connection Expert.Set Click **Manual Configuration**.
   - Set the **Hostname or IP Address** as **localhost**.
   - Set Protocol to HiSLIP and set Remote Name to hislip4
   - Check the box named **Allow *IDN Query** and click **Test This VISA Address** to verify the connection.
   - Click Accept as a terminator for the settings



Add a LAN device

Set LAN Address:

Hostname or IP Address: localhost
TCPIP Interface ID: TCPIP0

Set Protocol:

Instrument Remote Name: inst0
Socket Port Number: 5025
● HiSLIP Remote Name: hislip4

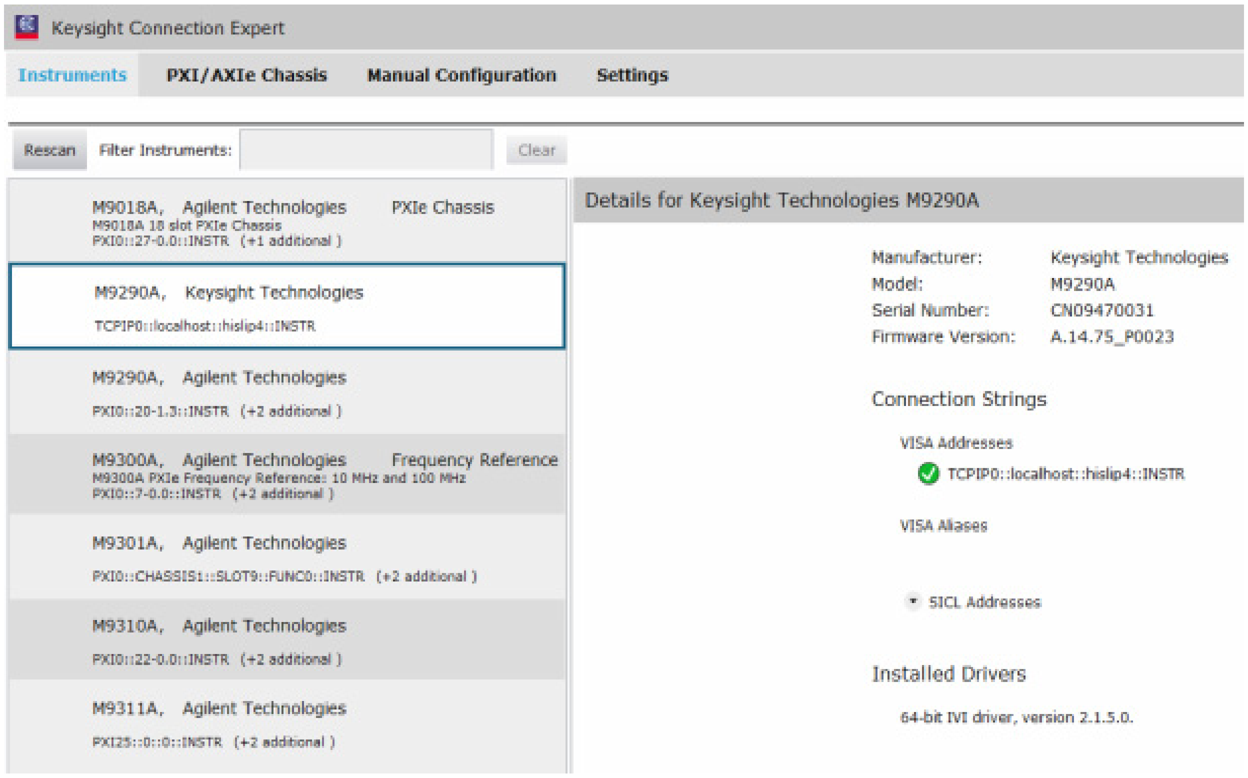
Verify Connection:

✓ Allow *IDN Query

TCPIP0::localhost::hislip4::INSTR
Verified
keysight technologies,m9290a,cn09470031

View Web Page:

5. Click **Instruments** on the tool bar. You will find the M9290A and its TCP/IP address is **TCPIP0::localhost::hislip4::INSTR**. This address is used by the Automatic Test System run in controller or external PC which communicate with M9290A by PCIe-PCIe bridge.



M9018A, Agilent Technologies PXIe Chassis
M9018A 18 slot PXIe Chassis
PXI0::27-0.0::INSTR (+1 additional)

M9290A, Keysight Technologies
TCPIP0::localhost::hislip4::INSTR

M9290A, Agilent Technologies
PXI0::20-1.3::INSTR (+2 additional)

M9300A, Agilent Technologies Frequency Reference
M9300A PXIe Frequency Reference: 10 MHz and 100 MHz
PXI0::7-0.0::INSTR (+2 additional)

M9301A, Agilent Technologies
PXI0::CHASSIS1::SLOT9::FUNC0::INSTR (+2 additional)

M9310A, Agilent Technologies
PXI0::22-0.0::INSTR (+2 additional)

M9311A, Agilent Technologies
PXI25::0::0::INSTR (+2 additional)

Details for Keysight Technologies M9290A

Manufacturer:	Keysight Technologies
Model:	M9290A
Serial Number:	CN09470031
Firmware Version:	A.14.75_P0023

Connection Strings

VISA Addresses

✓ TCPIP0::localhost::hislip4::INSTR

VISA Aliases

SICL Addresses

Installed Drivers

64-bit IVI driver, version 2.1.5.0.

Address Setting in Remote Control

For Remote control, you need know the controller’s IP address. Please press System > Show > System in Modular SA Virtual Front Panel to get the IP address of the controller.

Please follow the step 4 and step 5 of Address Setting in Local Control to set the address. The only difference is to set the Hostname or IP Address as xxx.xxx.xxx.xxx (controller’s IP address) rather than localhost.

X-Series Programming Options

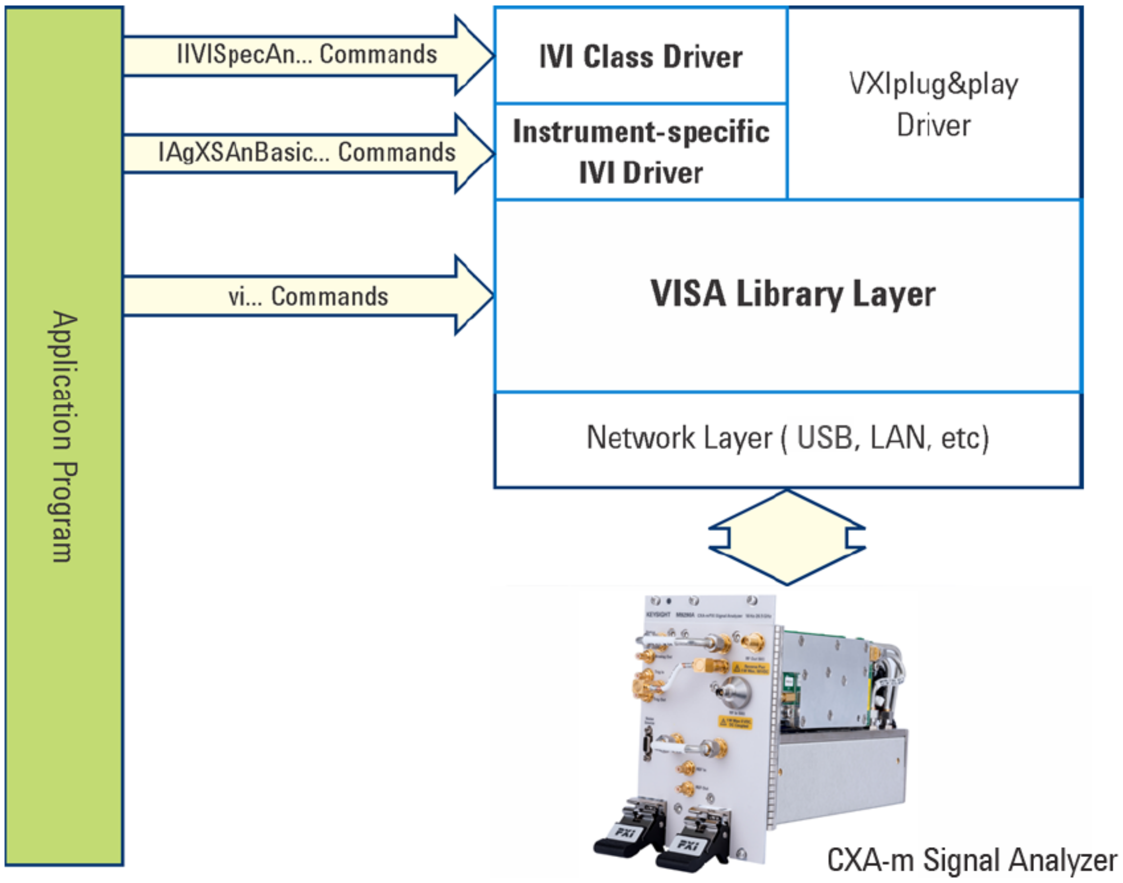
You can program CXA-m signal analyzer using a variety of programming tools, languages and Application Development Environments (ADEs).

There are also several software driver technologies that you can use to program CXA-m signal analyzer, 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 12 shows a conceptual overview of the hierarchy of drivers that are available for X-Series programming.

Table 1-1 Programming Options & Driver Technologies for CXA-m signal analyzer

Method	Description	
Instrument Drivers	Features	As shown in Figure 1-1, Instrument Drivers are built upon, and offer a higher level of abstraction than, VISA Drivers. Instrument drivers offer a shorter learning curve than VISA Drivers, at the expense of reduced operational flexibility. For more details, see "Interchangeable Virtual Instruments (IVI-COM, IVI-C) Drivers" on page 12.
	Acquisition & Licensing	Free download from Keysight and IVI Foundation web sites.
	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	IVI Class Driver and Instrument-Specific IVI Drivers, as shown in Figure 1-1 below. These are enhanced developments of the older **VXIplug&play Drivers**. Also referred to as "IVI-C" and "IVI-COM" Drivers.
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 13.
	Acquisition & Licensing	Free download from Keysight 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, as shown in "VISA Library Layer" in Figure 1-1 below.
Keysight VEE	Features	An integrated, graphically-oriented, standalone ADE, which supports instruments from Keysight and other manufacturers. For more details, see "Keysight VEE Pro" on page 13.
	Acquisition & Licensing	License must be purchased from Keysight.
	Requires separate ADE?	No
	Driver Support	Supports both Instrument Drivers and VISA.

Figure 1-1 M9290A Software Driver Hierarchy



Interchangeable Virtual Instruments (IVI-COM, IVI-C) Drivers

IVI Drivers are defined by the IVI Foundation, as an enhanced development of the earlier VXIplug&play Instrument Drivers.

With IVI drivers you do not need to have in-depth test instrument knowledge to develop sophisticated measurement software.

Keysight supports IVI Drivers for the following architectures:

- IVI-COM Drivers are based on the Microsoft Component Object Model (COM) technology, offering the seamless integration in all environments that is generally associated with COM.
- IVI-C Drivers are based on C-language shared libraries, and are intended to cater to National Instruments LabWindows/CVI.

IVI driver download packages for X-Series instruments can be found at the URL:

http://www.keysight.com/find/sa-ivi

Note that the Keysight I/O Libraries Suite must be installed and the hardware interface must be configured, before installing the IVI Drivers.

IVI Shared Components are required by all IVI-COM and IVI-C drivers. IVI Shared Components are automatically installed when you install the Keysight IO Libraries Suite.

VISA Driver Options

Keysight 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 (Keysight 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 controller.

The Keysight IO Libraries Suite includes the following libraries:

Item	Library Name	Documentation Location & Notes
1	Keysight Virtual Instrument Software Architecture (VISA)[1]	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 `visa.h` header file for use with C and C++, which provides declarations for the `visa32.dll` library. Additionally, Keysight has developed the header files `visa32.cs` and `visa32.bas`, to permit the VISA DLL to be used with C#.NET and Visual Basic.NET respectively. For more information, see the VISA Documentation Help in the Keysight I/O Libraries Suite.
2	VISA for the Common Object Model (VISA COM)[1]	The VISA COM I/O API is a programming interface standardized by the IVI Foundation for communicating with instruments over various hardware interfaces. Keysight Technologies offers an implementation of the VISA COM I/O standard that is compatible with Keysight 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. For more information, see the VISA COM Help in the Keysight I/O Libraries Suite.
3	Keysight Standard Instrument Control Library (SICL)[1]	SICL is compatible only with Keysight interfaces, whereas VISA is an industry-wide standard. This library is not described further in this document. For more information, see the SICL Documentation Help in the Keysight I/O Libraries Suite. Note that SICL is supported only for the C and VB6 languages; there is no SICL support for .NET languages.
4	Keysight 488	Compatible with the NI-488.2 Application Programming Interface (API) from National Instruments, and used for GPIB programming only. This library is not described further in this document. For more information, see the Keysight 488 Help in the Keysight I/O Libraries Suite.

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

Keysight VEE Pro

Keysight 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.

Keysight 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 Keysight VEE Pro.

For information about using IVI Instrument Drivers with Keysight VEE, see Keysight Application Note 1595.

Programming Documentation Roadmap

Most CXA-m 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 Keysight CXA-m Document Library.

Table 1-2 X-Series Documentation Resources

Resource	Description
CXA-m Programmer's Guide (This document)	Provides general programming information on the following topics: • 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: • 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. 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.
Keysight M9290A CXA-m PXIe Signal Analyzer: Getting Started Guide	Provides valuable sections related to programming including: • Licensing New Measurement Application Software - After Initial Purchase • Using the CXA-m Application Launcher to connect to the instrument This manual is shipped with the instrument as a printed document.
Keysight Application Notes	Printable PDF versions of pertinent application notes.
Keysight I/O Libraries Suite	The download package includes documentation describing the Keysight Virtual Instrument Software Architecture (VISA) library, and showing how to use it to develop I/O applications and instrument drivers on Windows PCs. **Not** included on CXA-m Product Information DVD.

Keysight IVI (Instrument) Drivers	The driver download packages include documentation (in Help CHM format) describing the IVI Class and Instrument-Specific Drivers. If the drivers are installed in the default location on your computer drive, the CHM files may be found in the folders:
	`C:\Program Files\IVI Foundation\IVI\Components`
	and:
	`C:\Program Files\IVI Foundation\IVI\Drivers`
	Not included on CXA-m Product Information DVD.

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

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 18
- "Creating Valid Commands" on page 19
- "Special Characters in Commands" on page 19
- "Parameters in Commands" on page 21
- "Writing Multiple Commands on the Same Line" on page 23

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 23.

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 23.

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
`:[SENSe:]BANDwidth[:RESolution] <freq>`	The following sample commands are all identical. They all cause the same result. • `:Sense:Band:Res 1700` • `:BANDWIDTH:RESOLUTION 1.7e3` • `:sens:band 1.7KHZ` • `:SENS:band 1.7E3Hz` • `:band 1.7kHz` • `:bandwidth:RES 1.7e3Hz`
`:MEASure:SPECtrum[n]?`	• `:MEAS:SPEC?` • `:Meas:spec?` • `:meas:spec3?` 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.
`[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle`	• `:DET:FUNC neg` • `:Detector:Func Pos`
`:INITiate:CONTinuous ON|OFF|1|0`	The sample commands below are identical. • `:INIT:CONT ON` • `:init:continuous 1`

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.	Command: `TRIGger:SOURce EXTernal|INTernal|LINE` The choices are external, internal, and line. Ex: `TRIG:SOURCE INT` is one possible command choice.
|	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]:AUTO` The following commands are all valid and have identical effects: Ex1: `bandwidth:auto` Ex2: `band:resolution:auto` Ex3: `sense:bandwidth:auto`

Special Character	Meaning	Example
`< >`	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	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. The following keywords may also be used in commands, but not all commands allow keyword variables. • 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. 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. Example query: `SENSE:FREQ:CENTER? MAX`

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.
<time> <seconds>	A rational number followed by optional units. The default units are seconds. Acceptable units include: ks, s, ms, µs, ns.
<voltage>	A rational number followed by optional units. The default units are Volts. Acceptable units include: V, mV, µV, nV
<current>	A rational number followed by optional units. The default units are Amperes. Acceptable units include: A, mA, µA, nA.
<power>	A rational number followed by optional units. The default units are W. Acceptable units include: kW, W, mW, µW, nW, pW.
<ampl>	A rational number followed by optional units. The default units are dBm. Acceptable units include: dBm, dBmV, dBµV.
<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.

Type	Description
<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: ▪ 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 #odddddd where ‘d’ represents an octal digit 0 to 7. So #o24 can be used instead of the decimal number 20. ▪ Binary, #Bdddddddddddddddd or #bdddddddddddddddd 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>	An array of rational numbers corresponding to displayed trace data. See the description of the `FORMat:DATA` 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. A SCPI command often refers to a block of current trace data with a variable name such as: Trace1, Trace2, or trace3, depending on which trace is being accessed.
<arbitrary block data>	A block of data bytes. The first information sent in the block is an ASCII header beginning with #. 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. A data query returns each block of data in the following format: `#DNNN<nnn binary data bytes>`; where `#DNNN` is the header. To parse this data: 1. Read two characters (`#D`), where D tells you how many N characters follow the D character, 2. Read D characters, and convert to an integer that specifies the number of data bytes in the block, 3. Read NNN bytes into a real array. **Example**: Header value = `#512320` • 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 `REAL,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** ✓ **Valid Command**	**Problem**
✗ `PWR:ATT 40dB` ✓ `POW:ATT 40dB`	The short form of `POWER` is `POW`, not `PWR`.
✗ `FREQ:STAR 30MHz;MIX:RANG -20dBm` ✓ `FREQ:STAR 30MHz;POW: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.
✗ `:POW:ATT 40dB;TRIG:FREQ:STAR 2.3GHz` ✓ `:POW:ATT 40dB;:FREQ:STAR 2.3GHz`	`:FREQ:STAR` is in the `:SENSE` subsystem, not the `:TRIGGER` subsystem.
✗ `:POW:ATT?:FREQ:STAR?` ✓ `: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 -5dB;:FREQ:STAR 10MHz` ✓ `:POW:ATT 5dB;:FREQ:STAR 10MHz`	Attenuation cannot be a negative value.

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).

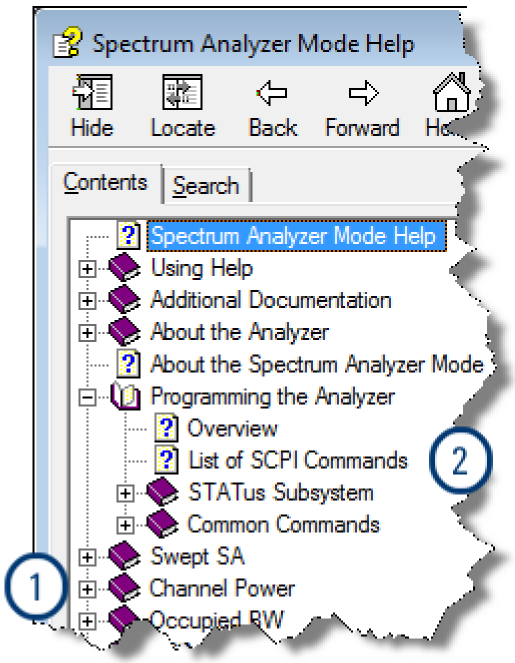
Reference Type	Usage & More Information
Users & Programmers Reference Manuals	All available PDF manuals are included on the X-Series Spectrum Analyzer Documentation DVD, in the folder `\files`. You can also download all Users & Programmers Reference manuals from the Keysight web site, by using the hyperlinks in the Additional Documentation section of the instrument's Embedded Help. 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.
Embedded Help	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 25. 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. All available Compiled Help Metafiles (CHMs) are also included on the X-Series Spectrum Analyzer Documentation DVD, in the `\help` subfolder. The CHM Help file for each measurement application has a name of the form `<mode_name>.en-us.chm`.

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



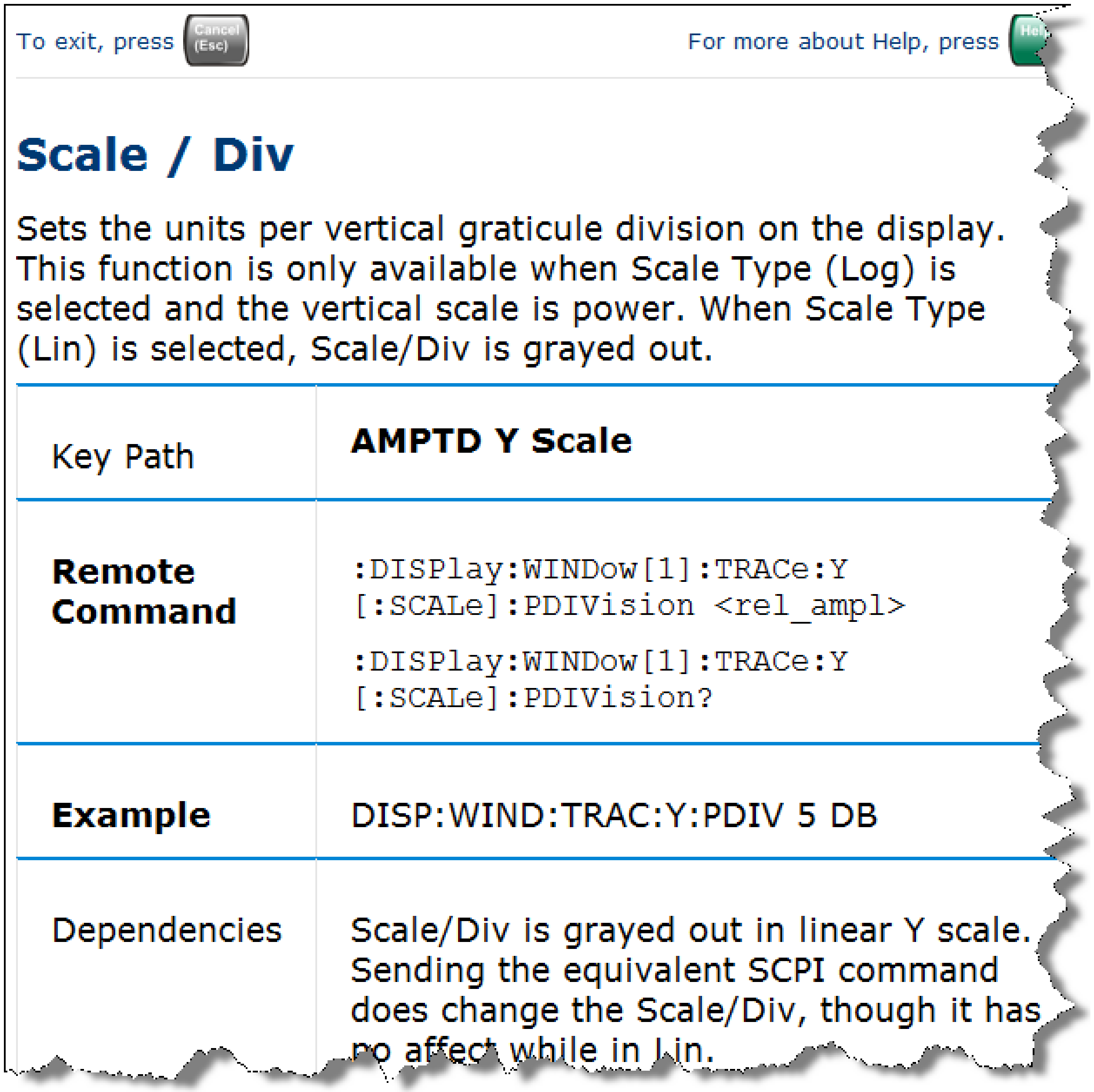
1. "Help Topic Content" on page 26
2. "List of SCPI Commands" on page 27

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



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**

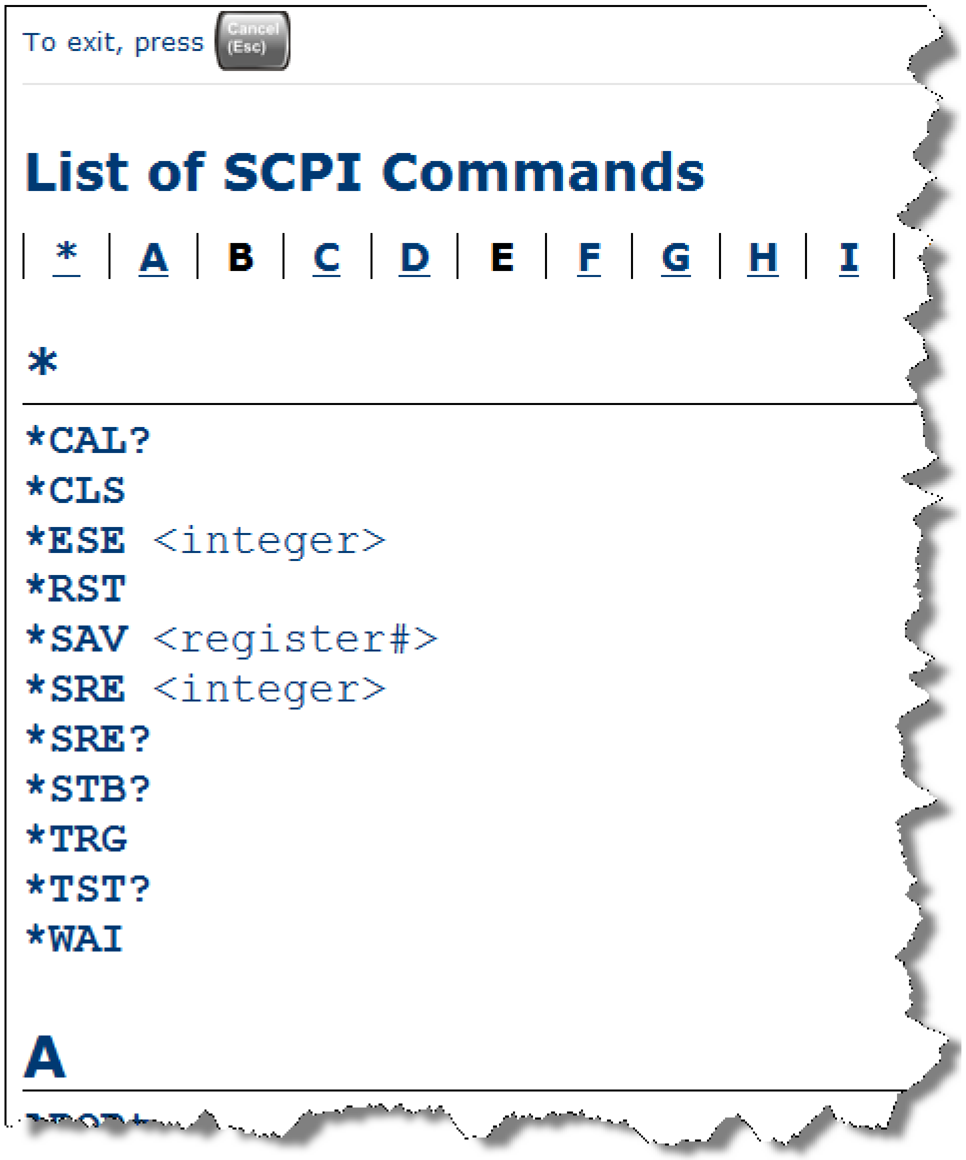
Remote Command	`:DISPlay:WINDow[1]:TRACe:Y[:SCALe]:PDIVision <rel_ampl>`
	`:DISPlay:WINDow[1]:TRACe:Y[:SCALe]:PDIVision?`

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


To exit, press

List of SCPI Commands

| * | A | B | C | D | E | F | G | H | I |

*

```
*CAL?
*CLS
*ESE <integer>
*RST
*SAV <register#>
*SRE <integer>
*SRE?
*STB?
*TRG
*TST?
*WAI
```

A

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:

- “Communicating SCPI Using Telnet” on page 28
- “Communicating SCPI using Keysight Connection Expert” on page 30

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 29.

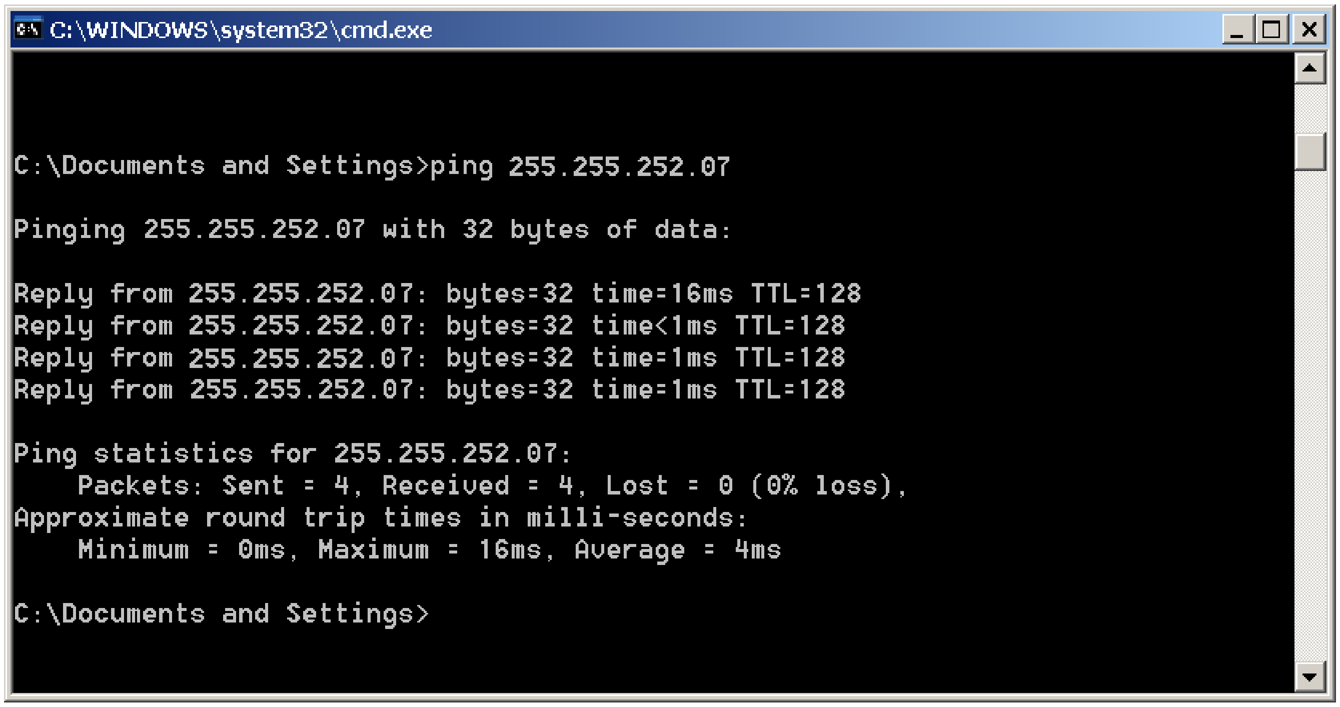
TIP In newer versions of Microsoft Windows (Windows Vista and Windows 7), you may first need to enable the Telnet client. See “Enabling Telnet in Windows” on page 30.

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 Keysight M9290A CXA-m PXIe 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 29.
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**.
3	Enable computer’s Telnet client if required	See “Enabling Telnet in Windows” on page 30.
4	Test LAN connection	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.

Figure 2-4 Command Window and ping Command results

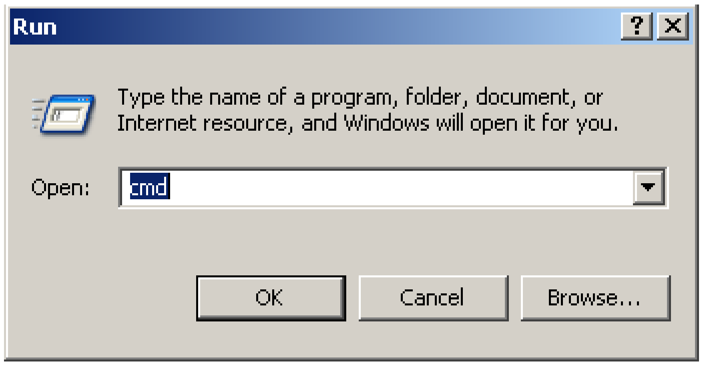


```
C:\Documents and Settings>ping 255.255.252.07

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 Keysight 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

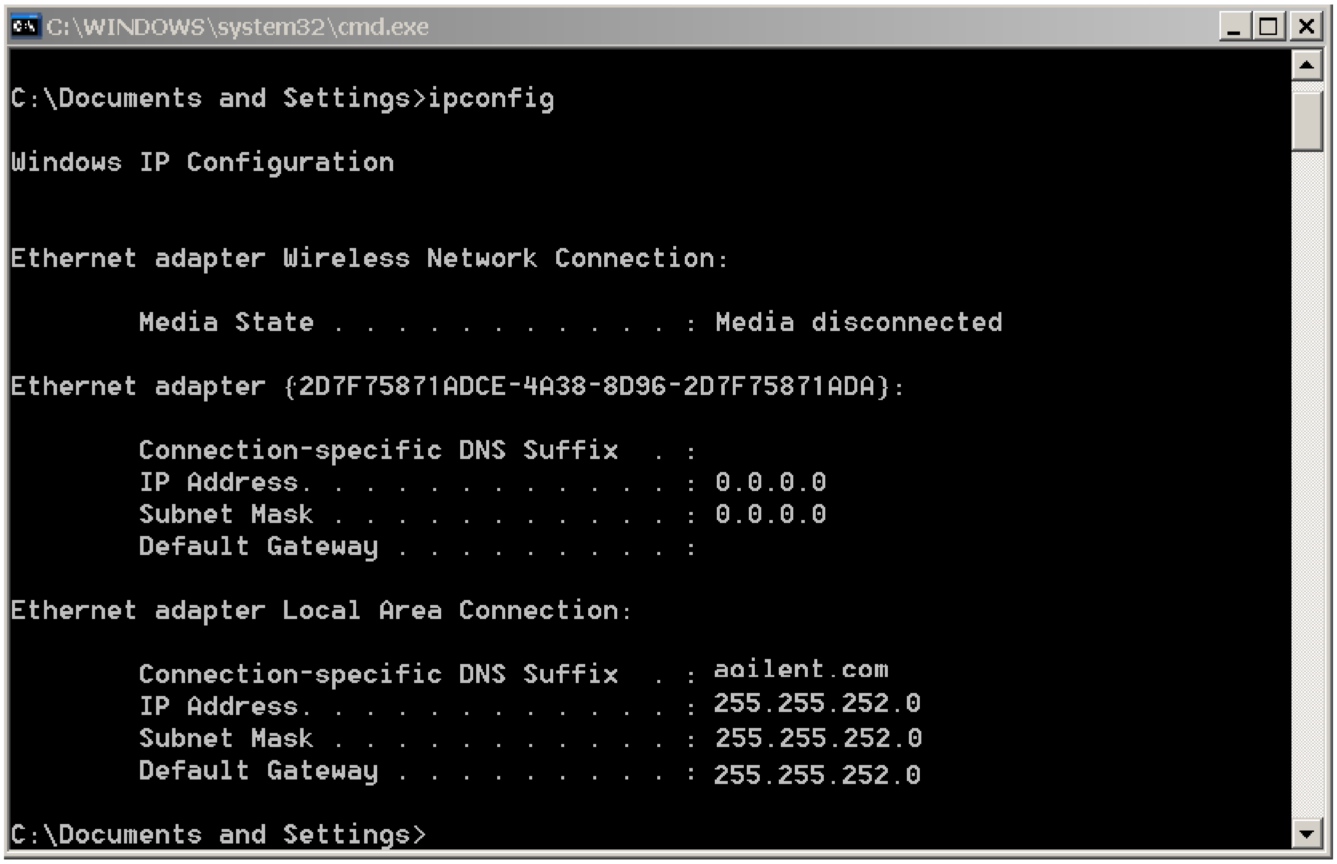


Run
?
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
cmd
OK
Cancel
Browse...

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:\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  . : agilent.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:

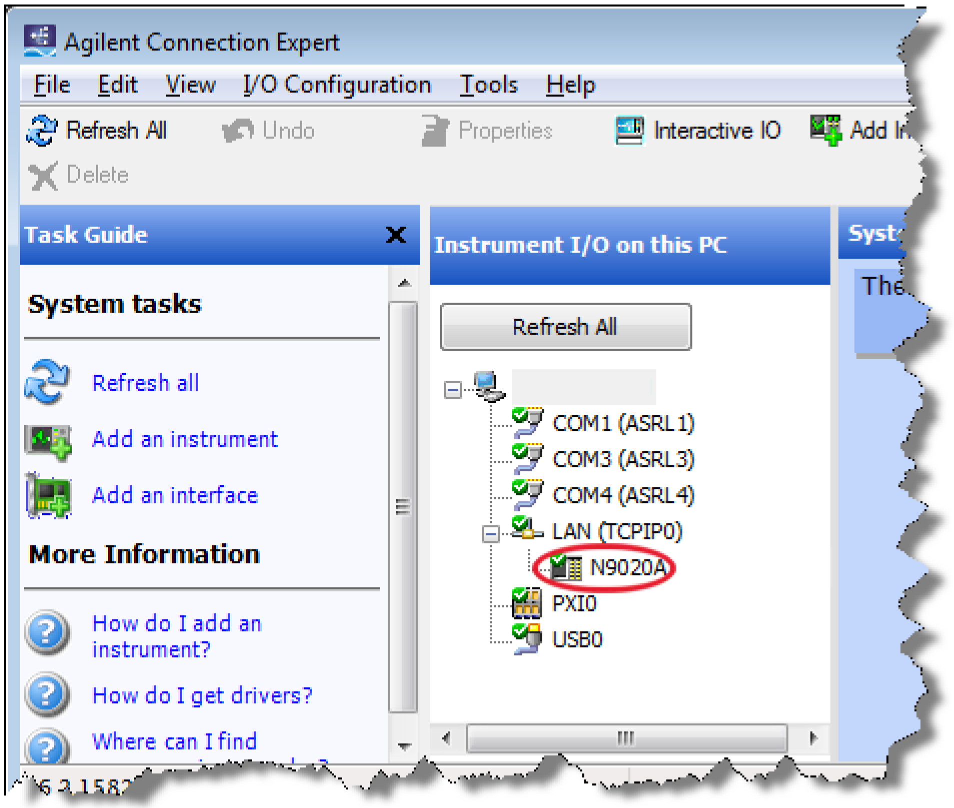
Step	Actions
1. Open Windows Control Panel	From the Windows Start menu, select **Control Panel**.
2. Select **Programs**	
3. Display Windows Features dialog	In the **Programs and Features** group, click **Turn Windows features on or off**. The Windows Features dialog appears.
4. Enable Telnet client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

Communicating SCPI using Keysight Connection Expert

You can use Keysight Connection Expert to communicate with devices on any supported network type. Keysight Connection Expert is installed as part of the Keysight I/O Libraries Suite.

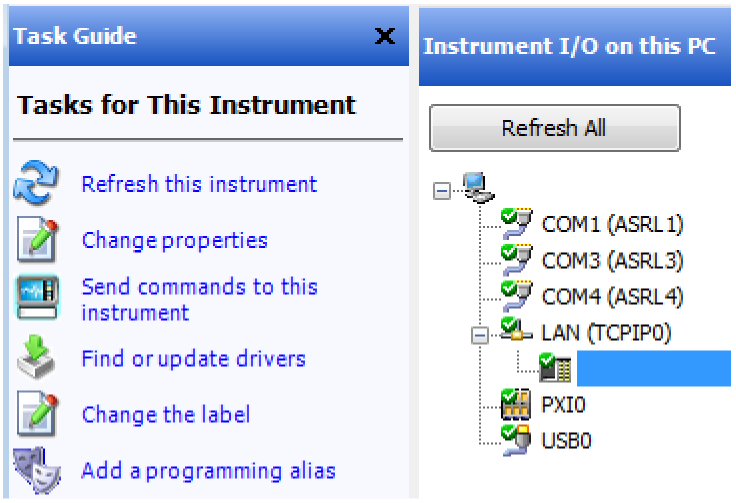
Figure 2-7 below shows part of the Keysight Connection Expert main screen, with one N9020A instrument connected via LAN.

Figure 2-7 Keysight Connection Expert Main Screen



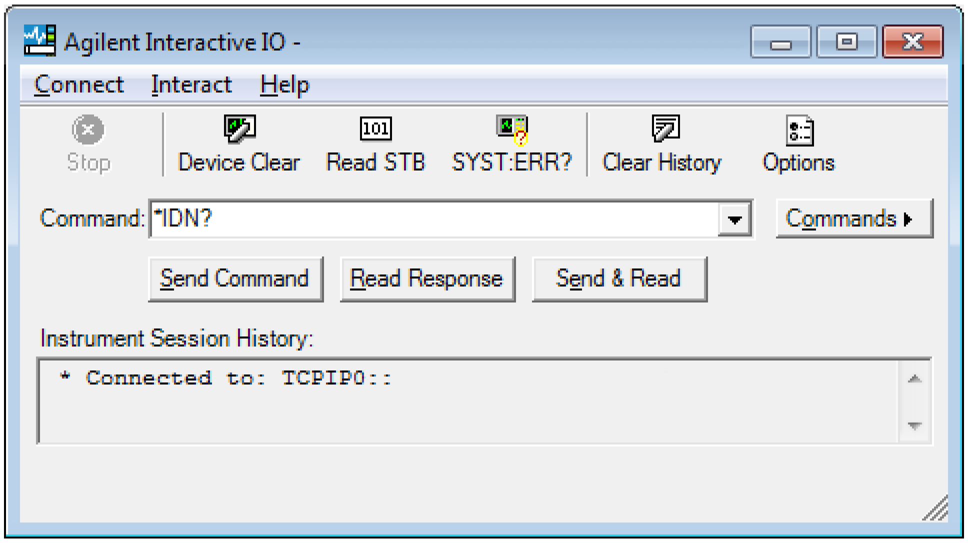
When you click on the N9020A instrument icon in this example, the content of the Task Guide panel on the left changes to "Tasks for This Instrument", which includes the selection "Send commands to this instrument", as shown in Figure 2-8 below.

Figure 2-8 Tasks for This Instrument



If you click the item "Send commands to this instrument", the Keysight 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.

Figure 2-9 Keysight Interactive IO



Command: *IDN?

Instrument Session History:

```
* Connected to: TCPIP0::
```

For full details of how to use these features, open the Keysight 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

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:BORDer 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 23.

Use Private LAN Connection

Note that LAN transfer speeds are affected by the volume of LAN traffic, and may be degraded if, for example, the instrument is connected to a busy enterprise LAN. Thus you may want to use a private LAN that is dedicated for the test system.

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 33.

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

Avoid Automatic Attenuator Setting

Many of the one-button measurements use an internal process for automatically setting the value of the attenuator. It requires measuring an initial burst to identify the proper attenuator setting before the next burst can be measured properly. If you know the amount of attenuation or the signal level needed for your measurement, just set it.

Note that spurious types of measurements must be done with the attenuator set to automatic (for measurements such as: Output RF Spectrum, Transmit Spurs, Adjacent Channel Power, Spectrum Emission Mask). These types of measurements start by tuning to the signal, then they tune away from it and must be able to reset the attenuation value as needed.

Avoid using RFBurst trigger for Single Burst Signals

RFBurst triggering works best when measuring signals with repetitive bursts. For a non-repetitive or single burst signals, use the IF (video) trigger or external trigger, depending on what you have available.

RFBurst triggering depends on its establishment of a valid triggering reference level, based on previous bursts. If you only have a single burst, the peak detection nature of this triggering function, may result in the trigger being done at the wrong level/point generating incorrect data, or it may not trigger at all.

Making a Single Burst Measurement

To achieve consistent triggering and valid data for this type of measurement application, you must synchronize the triggering of the DUT with the instrument. You should use the instrument’s internal status system for this.

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	`:CONF:WAV`	Switch to Waveform measurement
2	`:WAV:BAND 300khz`	Set resolution bandwidth to 300 kHz
3	`:WAV:SWE:TIME 5ms`	Set sweep time to 5 milliseconds
4	`:WAV:BAND:TYPE FLAT`	Select flat filter type
5	`:WAV:DEC 4;DEC:STAT ON`	Select a decimation of 4, and turn on decimation. This reduces the amount of data that must be transferred.
6	`:INIT`	Initiate measurement and acquire data
7	`:CALC:DATA2:COMP? MEAN,25us,526us,579.6us,8`	Retrieve the desired data

More Hints & Tips

For more information about optimizing measurement speed using X-Series instruments, see Keysight 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 37
- "Programming in C or C++ with VISA" on page 37
- "Programming with Microsoft .NET and VISA" on page 38
- "Requirements for Deploying a VISA Project" on page 39

For an overview of the relationship between VISA and other programming tools and drivers, see "X-Series Programming Options" on page 11.

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 Keysight 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 Keysight 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
```

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 Keysight 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 Keysight 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, Keysight 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 Keysight'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 Keysight 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 Keysight 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 Keysight VISA, and you wish to ensure that your program uses Keysight 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 Keysight DLL implementation.

4 Program Samples

The program samples described here were written for use on a controller 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

- Unless otherwise stated, all the sample programs described in this chapter are available in the `\progexamples` directory on the M9290A CXA-m PXIe Signal Analyzer Software & Product Information DVD.
- After the Keysight XSA software installation is completed, all the sample programs described in this chapter are available in the directory of `C:\Program Files\IVI Foundation\IVI\Drivers\M9290A\Examples` on the controller’s hard disk.
- Most of the M9290A samples can also be found on the Keysight Technologies, Inc. web site at URL:

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

Spectrum Analyzer Mode Programing Samples

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

- Visual Basic 6
- C, C++
- C#.NET & Visual Studio 2010
- Keysight VEE Pro
- MATLAB

NOTE These samples have all been tested and validated as functional in the Spectrum Analyzer mode. They have not necessarily been tested in other modes. However, they should work in all other modes, except where exceptions are noted.

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	MATLAB
Retrieve Screen Image	43		46	47	
Read Binary Trace Data	43		47[a]	47	
Poll Method for Operation Complete		44	46		
SRQ Interrupt Method for Operation Complete (Multi-threaded)		44	46		
Set and Query Relative Band Power Markers		44			
Set Traces and Couple Markers		45			
Phase Noise Trace Math Calculation		45	47		
Upload a State File					48
Switch Instrument Mode	43[b]		46[c]		48

a. This functionality is included in the C# sample for Phase Noise Trace Math.
b. This functionality is included in the Visual Basic 6 sample for Reading Binary Trace Data.
c. This functionality is included in the C# sample for the SRQ Interrupt Method.

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.

1. Retrieve Screen Images
2. Read Binary Trace Data

All the samples use the VISA driver.

Retrieve Screen Images

Function	Transfer Screen Images from the instrument
Description	This example demonstrates how to: 1. Store the current screen image in instrument memory as "`D:\PICTURE.PNG`" 2. Transfer the memory image file via LAN 3. Store the transferred image in the computer's current directory as "`C:\PICTURE.PNG`" 4. Delete the instrument memory file "`D:\PICTURE.PNG`"
Language	Visual Basic 6
File name	`mxa_screen.bas`

Read Binary Trace Data

Function	Read Binary Block Trace data from the instrument
Description	This example demonstrates how to: 1. Open a VISA session via LAN 2. Modify the timeout value 3. Send the `*IDN?` query to the instrument, then display the result 4. Change the instrument mode to Spectrum Analyzer 5. Set the Trace data format to `REAL,32` or `REAL,64` 6. Set the instrument to Single Sweep 7. Initiate a sweep 8. Read the trace data and display it 9. Store the trace data to the file "`bintrace.txt`" The binary data transfer method is faster than the default ASCII transfer mode, because less data is sent over the bus. For more information about data formats, see the section "Remote Measurement Functions" in any X-Series Help file or Users & Programmers Reference PDF.
Language	Visual Basic 6
File name	`bintrace.bas`

C, C++

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

1. Poll Method for Operation Complete
2. SRQ Method for Operation Complete
3. Set and Query Relative Band Power Markers
4. Set Traces and Couple Markers
5. Phase Noise Trace Math

All the samples use the VISA driver.

Poll Method for Operation Complete

Function	Serial Poll for Sweep Complete
Description	This example demonstrates how to: 1. Modify the timeout value 2. Initiate a sweep 3. Poll the instrument to determine when the operation is complete 4. Query and report the sweep result
Language	C
File name	`mxa_sweep.c`

SRQ Method for Operation Complete

Function	**Service Request Method (SRQ)** determines when a measurement is done by waiting for SRQ, then reading the Status Register.
Description	This example demonstrates how to: 1. Define an SRQ interrupt handler 2. Set up mode and measurement parameters 3. Set the service request mask to assert SRQ when either a measurement is uncalibrated or an error message has occurred 4. Install the interrupt handler 5. Initiate a sweep 6. Wait for an SRQ interrupt 7. When an SRQ interrupt occurs, examine its source and type and report the result 8. Uninstall the interrupt handler The STATus subsystem of commands is used to monitor and query hardware status. For details of these commands and registers, see the section "Measurement Group of Commands" in any X-Series Help file or Users & Programmers Reference PDF.
Language	C
File name	`mxa_srq.c`

Set and Query Relative Band Power Markers

Function	Relative Band Power Markers
Description	This example demonstrates how to: 1. Set up a calibration signal 2. Set Markers 1 through 5 as Band Power Markers 3. Obtain the band power of Markers 2 through 5, relative to Marker 1
Language	C
File name	`mxa_bpm.c`

Set Traces and Couple Markers

Function	Trace Detector/Couple Markers
Description	This example demonstrates how to:

1. Set various types of trace (Max Hold, Clear Write, Min Hold)
2. Relate markers to specified traces
3. Couple markers

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).

Language	C
File name	`mxa_tracecouple.c`

Phase Noise Trace Math

Function	Phase Noise Trace Math Calculation
Description	This example demonstrates how to remove instrument noise from phase noise, by:

1. Setting up a calibration signal
2. Setting local oscillator phase noise behavior
3. Setting Trace 1 type to average and initiate a sweep
4. Turning off calibration signal
5. Setting Trace 2 type to average and initiate a sweep
6. Calculating the power difference between Trace 1 and Trace 2, saving the result as Trace 3

Language	C
File name	`mxa_phasenoise.c`

C#.NET & Visual Studio 2010

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

1. Retrieve Screen Images
2. Poll Method for Operation Complete
3. SRQ Method for Operation Complete
4. Phase Noise Trace Math

All the samples use the VISA driver.

Retrieve Screen Images

Function	Capture and transfer Screen Images from the instrument
Description	This example demonstrates how to: 1. Store the current screen image as a PNG file on the instrument's `D:` drive, with a user-specified name 2. Retrieve the screen image data from the instrument as a Program Data Block 3. Analyze the header of the Program Data Block and extract the PNG bitmap from the block 4. Store the extracted bitmap as a PNG file in the computer's current directory, with the same user-specified name 5. Delete the PNG file that was stored on the instrument's `D:` drive
Language	C#
Project Folder	`vs2010.net/x_screencapture`

Poll Method for Operation Complete

Function	Serial Poll for Sweep Complete
Description	This example demonstrates how to: 1. Modify the timeout value 2. Initiate a sweep 3. Poll the instrument to determine when the operation is complete 4. Query and report the sweep result
Language	C#
Project Folder	`vs2010.net/x_sweep`

SRQ Method for Operation Complete

Function	**Service Request Method (SRQ)** determines when a measurement is done by waiting for SRQ, then reading the Status Register.
Header/ Library	`visa32.cs`
Description	This multi-threaded example demonstrates how to: 1. Define an SRQ interrupt handler 2. Set up mode and measurement parameters 3. Set the service request mask to assert SRQ when either a measurement is uncalibrated or an error message has occurred 4. Install the interrupt handler 5. Initiate a sweep 6. Set up a wait for multiple events 7. When an SRQ interrupt occurs, examine its source and type and report the result 8. Uninstall the interrupt handler The STATus subsystem of commands is used to monitor and query hardware status. For details of these commands and registers, see the section "Measurement Group of Commands" in any X-Series Help file or Users & Programmers Reference PDF.
Language	C#
Project Folder	`vs2010.net/x_srq`

Phase Noise Trace Math

Function	Phase Noise Trace Math Calculation
Description	This example demonstrates how to remove instrument noise from phase noise. The program does the following: 1. Set up a calibration signal 2. Set local oscillator phase noise behavior 3. Set Trace 1 type to average and initiate a sweep 4. Turn off calibration signal 5. Set Trace 2 type to average and initiate a sweep 6. Calculate the power difference between Trace 1 and Trace 2, then save the result as Trace 3 7. Retrieve Trace 3 data from the instrument as a binary data block 8. Optionally, save the retrieved trace data in an on-disk text file
Language	C#
Project Folder	`vs2010.net/x_phasenoise`

Keysight VEE Pro

1. Retrieve Screen Images
2. Retrieve Trace Data

Retrieve Screen Images

Function	Transfer Screen Images from the instrument
Description	This example demonstrates how to: 1. Store the current screen image in instrument memory as “`D:\mxascr.PNG`” 2. Transfer the memory image file 3. 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	Keysight VEE Pro
File name	`mxa_screencapture.vee`

Retrieve Trace Data

Function	Transfer trace data from the instrument.
Description	For each available data format (`INTeger,32`, `REAL,32`, `REAL,64`, and `ASCii`), the program does the following: 1. Sets the Trace data format 2. Sets the instrument to Single Sweep 3. Initiates a sweep 4. Reads the trace data and plots it graphically (using the default value of 1001 trace points) For more information about data formats, see `:FORMat:DATA` in the "Programming the Analyzer" chapter of any X-Series Help file or Users & Programmers Reference PDF.
Language	Keysight VEE Pro
File name	`transfertrace.vee`

MATLAB

These samples are not available on the M9290A Software & Product Information DVD. You can download them from http://www.keysight.com/find/sa_programming

Upload a File

Function	Upload a state file to the instrument
Description	The program opens a state file on the computer’s hard disk, transfers it to the instrument via LAN, then stores the file on the instrument’s `D:` drive.
Language / Driver	MATLAB / IVI Instrument Drivers
File name	`Upload_File_to_SA.m`

IVI-COM Personality (Mode) Select

Function	Check instrument identification fields and change mode to SA.

Description	This example does the following: 1. Checks Instrument Model, Firmware Revision and Serial Number 2. Selects SA Mode 3. Sets a Center Frequency 4. Reads the Instrument Error Queue
Language / Driver	MATLAB / IVI-COM Instrument Drivers
File name	`IVI_Personality_Select.m`

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. **Keysight CXA-m Document Library**

   Select one of the following hyperlinks, depending on the product name of your instrument:

   http://www.keysight.com/find/cxa-m_manuals

4. **Keysight M9290A CXA-m PXIe Signal Analyzer: Getting Started Guide**

   Keysight Technologies 2014. Part Number: subject to change as document is revised.

   A printed copy of this document is supplied with each Keysight M9290A CXA-m PXIe Singal Analyzer.

   It is also available in Acrobat (PDF) form:

   - on the Software & Product Information DVD supplied with each instrument,
   - on the instrument’s disk drive at the following location:

     C:\Program Files\Keysight\SignalAnalysis\Infrastructure\Help\bookfiles\getstart.pdf

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

   Keysight Technologies Inc.

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

   www.keysight.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. **Keysight VISA Help**

   After installing the Keysight 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\Keysight\IO Libraries Suite\Visa.chm`

7. **Keysight 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.keysight.com/find/sa-ivi

8. **Keysight Application Note 1583**

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

   May be downloaded in Acrobat (PDF) format from:

   http://literature.cdn.keysight.com/litweb/pdf/5989-4947EN.pdf

9. **Keysight Application Note 1595**

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

   May be downloaded in Acrobat (PDF) format from:

   http://literature.cdn.keysight.com/litweb/pdf/5989-6914EN.pdf

10. **Keysight VEE Pro**

    For links to all available information, see:

    www.keysight.com/find/vee

Developer Resources

Developer Network

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

http://www.keysight.com/find/adn

Technical Support

Navigate to one of the web pages below, according to the name of your product, then select the Technical Support tab for links to all available documentation for the product:

http://www.keysight.com/find/cxa-m