



```
147 SP_API SeStatus seSetIQCaptureType(int device, SeIQCaptureType captureType);
148 SP_API SeStatus seSetIQCenterFreq(int device, double centerFreqHz);
149 SP_API SeStatus seSetIQCenterFreq(int device, double "centerFreqHz");
150 SP_API SeStatus seSetIQSampleRate(int device, int decimation);
151 SP_API SeStatus seSetIQBandwidth(int device, SeBool enableSoftwareFilter, double bandwidth);
152 SP_API SeStatus seSetIQExtTriggerEdge(int device, SeTriggerEdge edge);
153 SP_API SeStatus seSetIQExtTriggerEdge(int device, SeTriggerEdge "edge");
154
155 SP_API SeStatus seSetAudioCenterFreq(int device, double centerFreqHz);
156 SP_API SeStatus seSetAudioType(int device, SeAudioType audioType);
157 SP_API SeStatus seSetAudioFilters(int device, double ifBandwidth, double audioBpf, double audioHpf);
158 SP_API SeStatus seSetAudioDeemphasis(int device, double deemphasis);
159
160 SP_API SeStatus seConfigure(int device, SeMode mode);
161 SP_API SeStatus seGetCurrentNode(int device, SeMode "mode");
162 SP_API SeStatus seAbort(int device);
163
164 SP_API SeStatus seSetSweepParameters(int device, double "actualBW", double "actualVBW",
165                                     double "actualStartFreq", double "binSize", int "sweepSize");
166 SP_API SeStatus seSetRealTimeParameters(int device, double "actualBW", int "sweepSize", double "actual
167                                     double "binSize", int "frameWidth", int "frameHeight", double "p
168 SP_API SeStatus seSetIQParameters(int device, double "sampleRate", double "bandwidth");
169
170 // Performs a single sweep, blocking function
171 SP_API SeStatus seGetSweep(int device, float "sweepMin", float "sweepMax", int64_t "nsSinceEpoch");
172
173 // Queue sweep mechanisms
174 SP_API SeStatus seStartSweep(int device, int pos);
175 SP_API SeStatus seFinishSweep(int device, int pos, float "sweepMin", float "sweepMax", int64_t "nsSince
176
177 SP_API SeStatus seGetRealTimeFrame(int device, float "frame", float "alphaFrame", float "sweepMin",
178                                     float "sweepMax", int "frameCount", int64_t "nsSinceEpoch");
179
180 //SP_API SeStatus seSetIQImpulse(int device, float "iqBuf", int "iqBufSize", SeBool purge);
181 SP_API SeStatus seSetIQ(int device, float "iqBuf", int "iqBufSize", double "triggers", int triggerBufSize
182                                     int64_t "nsSinceEpoch", SeBool purge, int "sampleLoss", int "samplesRemaining");
183
184 SP_API SeStatus seGetAudio(int device, float "audio");
185
186 SP_API SeStatus seGetGPSInfo(int device, SeBool refresh, SeBool "updated", int64_t "secSinceEpoch",
187
188     VTUserDataTrailerField isReferenceLock;
189     VTUserDataTrailerField isOverRange;
190     VTUserDataTrailerField isSampleLoss;
191     uint32_t associatedContextPktCount;
192 } VTUserDataTrailer;
193
194 #typedef struct VTUserDataPkt {
195     VTUserDataPkt& operator= (const VTUserDataPkt &pkt) {
196         prologue = pkt.prologue;
197         trailer = pkt.trailer;
198         data.resize(pkt.data.size());
199         memcpy(&data[0], &pkt.data[0], data.size());
200         return *this;
201     }
202     VTUserPktPrologue prologue;
203     std::vector<float> data;
204     VTUserDataTrailer trailer;
205 } VTUserDataPkt;
206
207 #typedef struct VTUserContextIndicators {
208     bool isContextFieldChanged;
209     bool isBandwidth;
210     bool isRffreq;
211     bool isRefLevel;
212     bool isAtten;
213     bool isSampleRate;
214     bool isTemperature;
215     bool isDevId;
216     bool isDevModel;
217     bool isGPS;
218 } VTUserContextIndicators;
219
220 #typedef struct VTUserGPS {
221     double latitude;
222     double longitude;
223     double altitude;
224     uint32_t seconds;
225     uint32_t picosec;
226 }
```

VSG60 SCPI Programming Manual User Guide

VSG60 SCPI Programming Manual

Published 10/21/2025

©2025, Signal Hound

1502 SE Commerce Ave, Suite 101

Battle Ground, WA

Phone 360-313-7997

Contents

1 Version Notes	4
2 Introduction / About SCPI	4
3 SCPI command basics.....	4
3.1 Commands	4
3.2 Multiple commands	5
3.3 Parameters	5
3.4 Return Values.....	6
3.5 Special Characters	6
4 Getting Started	7
5 Functionality provided through SCPI	8
6 Examples	8
7 Functions	9
7.1 Common Commands	9
7.2 Display.....	9
7.3 System Functions.....	10
7.3.1 <i>Device Management</i>	11
7.3.2 <i>Errors</i>	12
7.4 Reference	13
7.5 Output.....	13
7.6 Frequency	13
7.7 Power	14
7.8 Impairments.....	14
7.9 Amplitude Modulation.....	16
7.10 Frequency Modulation.....	16
7.11 Pulse Modulation.....	17
7.12 Multitone.....	17
7.13 Step Sweep	18
7.14 Ramp Sweep.....	19
7.15 AWGN	19
7.16 Custom Digital Modulation	20
7.17 802.11a	22
7.18 802.11n	23
7.19 802.11ac.....	24
7.20 802.11ax.....	25
7.21 Arb.....	27
7.22 Streaming	29

1 Version Notes

SCPI commands can and will change as the VSG60 software evolves. It is recommended that when you update the VSG60 software in an installation that is controlled via SCPI, to review the version notes and determine if any functionality needs to be updated. See the separate document title *scpi_version_notes.txt* for a full list of changes for each version of the VSG60 software.

2 Introduction / About SCPI

SCPI (Standard Commands for Programmable Instruments) is a standard which covers the set of commands used to program various instruments. The standard covers the syntax, form, behavior, etc. of these commands in attempt to reduce development time for the user.

For the purposes of Signal Hound and the VSG60 software, a user can send SCPI commands to control the VSG60 software in an automatic fashion. SCPI commands are sent to instruments over many interfaces, commonly GPIB, VXI, USB, Ethernet, etc. The VSG60 software accepts commands over a network socket. The VSG60 software will accept a single network connection in which it can receive SCPI commands and send responses.

This document will cover the basics of SCPI commands, how to get started programming the VSG60 software, and will cover the full SCPI command set implemented by the VSG60 software.

3 SCPI command basics

This section contains a quick overview of the SCPI command syntax and usage to the extent that is relevant to the VSG60 software. The VSG60 does not utilize all functionality in the SCPI standard and as such said functionality will not be covered here.

3.1 Commands

A SCPI command is comprised of a series of keywords separated by colons. A command may be followed by a '?' to represent a query, a series of parameters separated by spaces, or both.

`:SENSE:FREQUENCY:CENTER 1GHz` (Example command for setting the center frequency to 1GHz)

`:sense:frequency:center?` (Example command for querying the current center frequency)

Commands are case insensitive. Each keyword in a command can have a short and long form. Both can be used interchangeably.

`:SENSe:FREQuency:CENTer` is a command with three keywords. Each keyword has a short and long form. The short form is denoted by the uppercase characters and the long form is the full keyword including the upper and lower-case characters. For example, `FREQ` is the short form of `FREQUENCY`. When constructing a command, the short and long form can be interchanged. For example, you could construct the command as such, `:SENS:FREQUENCY:CENT` where `SENSE` and `CENTER` are sent as short form and `FREQUENCY` as longform.

Some commands are options and are denoted as such by the '[' characters.

`[:SENSe]:FREQuency:CENTer` is a command where the first keyword is optional. This command can be sent as `FREQ:CENT` and still be interpreted correctly.

Commands are terminated with a newline character. For example

```
:SENS:FREQ:CENT 1GHZ\n
```

Commands will be processed once a newline is reached. Additionally, a newline will reset the current keyword path.

3.2 Multiple commands

Multiple commands can be sent to the device at once using the semi colon character separating each command.

```
:SENS:FREQ:CENT 1GHz; :SENS:FREQ:SPAN 10MHz\n
```

This is an example of sending two commands at once. Additionally, when sending multiple commands, you don't need to repeat all keywords leading up to the final keyword for commands after the first.

```
:SENS:FREQ:CENT 1GHz; SPAN 10MHz\n
```

Here `SPAN` retains the `:SENS:FREQ:` keywords from the previous command. To prevent this from happening use the colon character leading the second command. For example

```
:SENS:FREQ:CENT 1GHz; :SPAN 10MHz\n
```

This is an invalid series of commands, since `span` is prefixed with a colon command which reset the previous keywords.

3.3 Parameters

There are several types of parameters that can be sent in commands.

Boolean	ON OFF 0 1
Keyword <bool>	Character specific strings for a given command. These keywords can also have short and long form.

Numeric <integer> <double>	Numeric parameters take either the form of integer or decimal values. Examples include 1 1.23 9 3.14
Frequency <freq>	These are numeric parameters with a frequency suffix. Possible frequency suffixes include HZ KHZ MHZ GHZ The suffixes are case insensitive. If a suffix is not present, Hz is the default unit. Examples include 1kHz 20MHz 12GHz Any function that returns a frequency will return the frequency in Hz with no suffix present.
Amplitude <amplitude>	These are numeric parameters with an amplitude suffix. Possible amplitude suffixes include DBM DBMV DBUV MV The suffixes are case insensitive. A suffix must be present unless indicated otherwise. Examples include -20DBM 60dbuv If a function returns an amplitude, it will return the amplitude in the current software units without a suffix.

3.4 Return Values

Values returned from the VSG60 software (as a result of sending a query command) are separated by a semi-colon if multiple query commands are sent in one string and are terminated by a newline. For example, sending

```
"CALC:MARK:MAX; X?; Y?\n"
```

results in a return string of

```
"1000000;-20\n"
```

The command sent performs a peak search and queries the X and Y positions of the marker. The return is the X and Y positions separated by a semicolon and terminated with a newline.

3.5 Special Characters

This section describes the numerous special characters that are present in the commands in this document.

Character	Description	Example
-----------	-------------	---------

	Vertical stroke between parameters indicates multiple choices	FLATtop GAUSSian The choices are between FLATTOP or GAUSSIAN. Provide one or the other.
[]	Square brackets indicate an optional keyword	:SYSTem:ERRor[:NEXT]? Next is an optional keyword and the command could also be composed as :SYSTem:ERRor?
<>	Angle brackets around a parameter indicate a type and angle brackets should not be included in the user command.	*RCL <int> <int> is the type of parameter and an example of using this command would be *RCL 1 Notice the angle brackets are not included.

4 Getting Started

See the SCPI examples found in the SDK download on any of the Signal Hound product download pages. The examples use the C programming language and a common VISA library implementation.

Instrument control is performed by connecting to the VSG60 software on TCP/IP port 5024. On this port, a user can send and receive raw SCPI commands. It is not necessary to use a I/O library like VISA to communicate with the VSG60 software but it can simplify several operations. It is possible to communicate directly over the socket with socket programming. The computer that is communicating with the VSG60 software does not have to be the same computer running the VSG60 software and does not have to be a Windows platform.

It is recommended to use a VISA library if available. Several implementations of VISA exist. Commonly used ones include Keysight's I/O libraries, and NI's VISA libraries. You can also use VISA implementations that exist in other languages/environments such as MATLAB, LabVIEW, and Python.

Connecting to the socket interface using VISA looks like this

```
viOpen(rm, "TCPIP::localhost::5025:SOCKET", VI_NULL, VI_NULL, &inst);
```

Additionally, when using a VISA library, it is necessary to set the VI_ATTR_TERMCHAR_EN attribute to true. This will terminate the read operation when the termination character is received. The termination character should be set to the newline ('\n') character if it is not set by default. The code for this is below.

```
viSetAttribute(inst, VI_ATTR_TERMCHAR_EN, VI_TRUE);  
  
viSetAttribute(inst, VI_ATTR_TERMCHAR, '\n');
```

Only one connection to the VSG60 software can be active at a time. The connection can be terminated by either closing the socket connection, either through the socket library you are using, the viClose function if you are using a VISA library, or by closing your application. The VSG60 will immediately begin waiting for another socket connection when the previous one is ended.

5 Functionality provided through SCPI

The table below details what functionality is covered under the current SCPI command set. Functionality will be added over time. If functionality you need it not available, please contact us at support@signalhound.com to make requests.

Functionality	Implemented
AM	Yes
FM	Yes
Multitone	Yes
Step Sweep	Yes
Ramp Sweep	Yes
AWGN	Yes
Digital Mod	Yes
OFDM	No
Bluetooth LE	No
IEEE 802.11 a/n/ac/ax	Yes
LTE	No
Arb	Yes
Streaming	Yes

6 Examples

SCPI examples are found in the SDK which can be downloaded from the Signal Hound website.

7 Functions

7.1 Common Commands

The software supports the following common commands.

Command	<div><div>*IDN?</div><div>*RCL <int></div><div>*SAV <int></div><div>*RST</div><div>*TRG</div><div>*OPC</div><div>*ESR?</div></div>
Description	<div><div>*IDN?, Query the serial number and name of the device.</div><div>*RCL, Load preset [1-9].</div><div>*SAV, Save preset [1-9].</div><div>*RST, Same as PRESet, see below.</div><div>*TRG, Triggers the device.</div><div>*OPC, Tells the instrument that after all the commands are executed and finished to set the ESR bit 0 (OPC bit) to 1. This command in combination with the *ESR? command can be used for synchronization through polling. See the C++ SCPI examples in the SDK for an example of polling using these commands.</div><div>*ESR?, Returns the Event Status Register (ESR). Only bit 0 is used at this time. Bit 0 represents Operation Complete (OPC). Returns 0 if *OPC has been seen but there are still commands to be executed and finished. Sends a 1 when all commands have been finished and executed. This command in combination with the *ESR? command can be used for synchronization through polling. See the C++ SCPI examples in the SDK for an example of polling using these commands.</div></div>
Examples	<div><div>*IDN?</div><div>*RCL 1</div><div>*SAV 1</div><div>*TRG</div><div>*RST</div><div>*OPC</div><div>*ESR?</div></div>
Software Controls	<div><div>Status Bar</div><div>File Menu -> Presets -> Load</div><div>File Menu -> Presets -> Save</div><div>Preset Key</div><div>Trigger Key</div></div>
Couplings	<div><div>None</div></div>
Preset	<div><div>N/A</div></div>
Notes	

7.2 Display

Command	<div><div>:DISPlay:HIDE <bool></div></div>
---------	--

	:DISPlay:HIDE?
Description	HIDE, When set to true, hides the VSG60 application. The application will be hidden in the taskbar but will continue to be visible in the task manager. The SCPI lockout dialog will continue to be visible but can be disabled in the preferences menu, prior to setting the application hidden. HIDE?, Returns true when the application is not visible.
Examples	DISP:HIDE 1 DISP:HIDE?
Software Controls	
Couplings	None
Preset	
Notes	

7.3 System Functions

The following commands are used to perform system level software actions and query information about the system.

Command	:SYSTem:COMMunicate:GTLocal :SYSTem:CLOSe :SYSTem:PRESet :SYSTem:PRESet? :SYSTem:PRESet[:USER]:SAVE <filename> :SYSTem:PRESet[:USER]:LOAD <filename> :SYSTem:VERsion?
Description	COMMunicate:GTLocal, Puts the software in local mode. CLOSe, Disconnect any active device and closes the software. There is not a way to reopen the software using SCPI commands. This will also terminate the socket connection. PRESet, Presets the active device. This will power cycled the active device and return the software to the initial power on state. This process can take between 6-20 seconds depending on the device type. PRESet?, Presets the active device. This will close and reopen the active device. This process can take between 6-20 seconds depending on the device type. Returns 0 or 1 depending on success. (1 for success) PRESet[:USER]:SAVE, Save a preset with the given file name. The file name should have extension ".ini". PRESet[:USER]:LOAD, Load the preset given by the file name. If the preset does not exist, nothing occurs. The file name should have extension ".ini". VERsion?, Returns the software version number.
Examples	SYST:CLOS SYST:PRESET? SYST:PRESET:USER:SAVE "C:/Users/Me/Documents/SignalHound/customPreset2.ini" SYST:PRESET:USER:LOAD

"C:/Users/Me/Documents/SignalHound/customPreset2.ini"	
SYSTEM:VERSION?	
SYST:COMM:GTL	
Software	Status Bar
Controls	File -> Exit
	Preset
	Presets -> Save User Preset
	Presets -> Load User Preset
	Help -> About
	Remote Mode Dialog -> Return to Local
Couplings	None
Preset	N/A
Notes	

7.3.1 Device Management

The functions below allow you to remotely manage the active device in the software. This is useful for error recovery in the event a device disconnect occurs due, or if one is managing multiple Signal Hound devices on one PC.

Connecting Signal Hound devices can take between 3-20 seconds depending on the type of device and the state of the device prior to interfacing it. If the VISA timeout is shorter than the time it takes to connect the device in the software, you will need to loop on timeout until you receive the connect status return.

Command	:SYSTem:DEvice:ACTive? :SYSTem:DEvice:COUNt? :SYSTem:DEvice:LIST? :SYSTem:DEvice:CONnect? <int> :SYSTem:DEvice:DISConnect?
Description	ACTive?, Returns whether or not a device is currently connected and active in the software. Look at the *IDN? function to request information about the device. COUNt?, Returns the number of devices connected to the PC. No device may be active when this function is called. IE, you must call DISConnect? before calling this function. LIST?, Returns all serial numbers available. The serial numbers are returned as ascii integers and are comma separated. To determine how many serial numbers are present, use the COUNt? function. CONnect?, Connect a device. You need to provide the serial number of the device to connect. Returns 0 or 1 depending on if the device successfully opened. DISConnect?, Disconnects the active device. Returns 1 when finished.
Examples	SYST:DEV:ACT? SYST:DEV:COUNT? SYSTEM:DEVICE:LIST? SYSTEM:DEVICE:CONNECT? 30700189

	SYSTEM:DEV:CONNECT?
	SYST:DEV:DISC?
Software	File Menu -> File -> Connect
Controls	File Menu -> File -> Disconnect
Couplings	Only one device can be active at a time.
Preset	N/A
Notes	

7.3.2 Errors

The VSG60 software maintains a list of system errors available to the user. Errors are stored with a unique ID, name, and description. The types of issues represented in the error list are settings conflicts, SCPI issues such as invalid parameter types or instructions, file I/O errors, etc.

It is recommended to frequently check for errors when utilizing SCPI in the software. Check the SCPI examples to see how to quickly poll for any present errors.

The errors are returned in the form

```
"ID,description;error information"
```

ID is a unique integer for the error. The description is an ascii text description for the error, and error information is any additional context information for the error generated. An example error message is below.

```
"-2,Invalid Parameter;Expected frequency parameter"
```

This error indicates the SCPI parser was expecting a frequency parameter and was either unable to find it or was unable to parse it as a frequency.

Once the error queue is empty, the software will return the 'no error' error when the next system error is requested. 'No error' has an ID of 0.

Command	:SYSTem:ERRor:COUNT? :SYSTem:ERRor[:NEXT]? :SYSTem:ERRor:CLEAr
Description	COUNT?, Returns the number of errors in the error queue. NEXT?, Returns the next error in the queue, and removing it from the queue. CLEAR, Remove all errors from the queue, returns nothing.
Examples	SYST:ERR:COUN? SYSTEM:ERROR:NEXT? SYST:ERR? SYST:ERR:CLEAR
Software	Utilities -> Show Error Log
Controls	Error Info -> Clear Button

Couplings	None
Preset	N/A
Notes	None

7.4 Reference

These commands control the reference oscillator settings the of the spectrum analyzer.

Command	<code>[:SOURce]:ROSCillator:SOURce INTernal EXTernal</code> <code>[:SOURce]:ROSCillator:SOURce?</code>
Description	Specify whether the generator should use the internal reference or use an external reference.
Examples	<code>:SENSE:ROSCILLATOR:SOURCE INTERNAL</code> <code>ROSC:SOUR EXT</code> <code>rosc:source?</code>
Software Controls	Ext Ref
Couplings	None
Preset	On program startup, internal reference is selected.
Notes	None

7.5 Output

Command	<code>:OUTPut[:STATe] ON OFF 0 1</code> <code>:OUTPut[:STATe]?</code> <code>:OUTPut:MODulation[:STATe] ON OFF 0 1</code> <code>:OUTPut:MODulation[:STATe]?</code> <code>:OUTPut:CORRection:IMBalance FULL PARTial</code> <code>:OUTPut:CORRection:IMBalance?</code>
Description	
Examples	<code>:OUTPUT 1</code> <code>:OUTPUT:MOD ON</code> <code>:OUTP:CORR:IMB PARTIAL</code>
Software Controls	RF On/Off Mod On/Off Edit -> Preferences -> Image Imbalance Correction
Couplings	None
Preset	On program startup, both RF and Mod off Imbalance correction is set to full by default
Notes	None

7.6 Frequency

Command	<code>[:SOURce]:FREQuency <freq></code> <code>[:SOURce]:FREQuency?</code> <code>[:SOURce]:FREQuency:STEP[:INCRement] <freq></code> <code>[:SOURce]:FREQuency:STEP[:INCRement]?</code>
---------	--

Description	
Examples	FREQ 2.45GHz FREQ? FREQ:STEP 20MHz FREQ:STEP?
Software Controls	Freq Step
Couplings	None
Preset	
Notes	None

7.7 Power

Command	[:SOURce]:POWer <double> [:SOURce]:POWer? [:SOURce]:POWer:STEP[:INCRement] <double> [:SOURce]:POWer:STEP[:INCRement]?
Description	
Examples	POW -20 POW? POW:STEP 1 POW:STEP?
Software Controls	Level (dBm) Step (dB)
Couplings	None
Preset	
Notes	None

7.8 Impairments

Command	:OUTPut:IMPairments:LEVel:OFFSet <double> :OUTPut:IMPairments:LEVel:OFFSet? :OUTPut:IMPairments:FLATness[:STATe] <bool> :OUTPut:IMPairments:FLATness[:STATe]? :OUTPut:IMPairments:FREQuency:OFFSet <freq> :OUTPut:IMPairments:FREQuency:OFFSet? :OUTPut:IMPairments:FREQuency:INVert <bool> :OUTPut:IMPairments:FREQuency:INVert? :OUTPut:IMPairments:LSPur[:STATe] <bool> :OUTPut:IMPairments:LSPur[:STATe]? :OUTPut:IMPairments:IOFFset <int> :OUTPut:IMPairments:IOFFset? :OUTPut:IMPairments:QOFFset <int> :OUTPut:IMPairments:QOFFset? :OUTPut:IMPairments:IMBalance:AMPL <double> :OUTPut:IMPairments:IMBalance:AMPL? :OUTPut:IMPairments:IMBalance:PHASe <double>
---------	--

	<pre> :OUTPut:IMPairments:IMBalance:PHASE? :OUTPut:IMPairments:SRATE:MULtiplier <double> :OUTPut:IMPairments:SRATE:MULtiplier? :OUTPut:IMPairments:AWGN[:STATE] <bool> :OUTPut:IMPairments:AQGN[:STATE]? :OUTPut:IMPairments:AWGN:SNR <double> :OUTPut:IMPairments:AWGN:SNR? :OUTPut:IMPairments:AWGN:IBWidth <freq> :OUTPut:IMPairments:AWGN:IBWidth? :OUTPut:IMPairments:CHANnel[:STATE] <bool> :OUTPut:IMPairments:CHANnel[:STATE]? :OUTPut:IMPairments:CHANnel:LENGth? :OUTPut:IMPairments:CHANnel:DATA <I1>, <Q1>, <I2>, <Q2>, ..., <In>, <Qn> :OUTPut:IMPairments:CHANnel:DATA? :OUTPut:IMPairments:PNOIse[:STATE] <bool> :OUTPut:IMPairments:PNOIse[:STATE]? </pre>
Description	<p>CHANnel:LENGth?, Returns the number of complex taps in the channel impairment filter.</p> <p>CHANnel:DATA, Set the channel impairment taps. The I/Q values should be provided as alternating I/Q complex values, each I and Q value sent as a separate SCPI parameter, as ascii. A comma should separate all I/Q values. A comma should not be placed after the last Q value. An error will be thrown if an odd number of parameters is provided. See example below and programming example for usage.</p> <p>CHANnel:DATA?, Returns the channel impairment taps as a list of real imaginary values, each value separated by a comma. For example, a 3 tap filter will be returned as 6 comma separated ascii values, real imaginary alternating.</p>
Examples	<pre> OUTP:IMP:LEV:OFFS 10 OUTP:IMP:FREQ:OFFS 1MHz OUTP:IMP:FREQ:INV ON OUTP:IMP:LSP ON OUTP:IMP:IOFF 10 OUTP:IMP:QOFF -22 OUTP:IMP:IMBA:AMPL 2 OUTP:IMP:IMBA:PHASE 3.14159 OUTP:IMP:SRAT:MULT 1 OUTP:IMP:AWGN ON OUTP:IMP:AWGN:SNR 30 OUTP:IMP:AWGN:IBW 10MHz OUTP:IMP:CHAN 1 OUTP:IMP:CHAN:LEN? OUTP:IMP:CHAN:DATA 0.0, 0.0, 1.0, 0.0, 0.0, 0.0 This line loaded a 3 tap all pass channel filter. OUTP:IMP:CHAN:DATA? OUTP:IMP:PNOISE:STATE ON </pre>
Software Controls	<p>Impairment Controls -> Level Offset</p> <p>Impairment Controls -> User Flatness</p>

	Impairment Controls -> Frequency Offset Impairment Controls -> Low Spur Mode Impairment Controls -> I Offset Impairment Controls -> Q Offset Impairment Controls -> I/Q Phase Imbalance (deg) Impairment Controls -> I/Q Ampl Imbalance (dB) Impairment Controls -> Sample Rate Error (ppm) Impairment Controls -> AWGN Enabled Impairment Controls -> AWGN SNR (dB) Impairment Controls -> AWGN Bandwidth Impairment Controls -> Channel Filter Enabled Impairment Controls -> Phase Noise Enabled
Couplings	None
Preset	
Notes	None

7.9 Amplitude Modulation

Command	[:SOURce]:AM[:STATe] <bool> [:SOURce]:AM[:STATe]? [:SOURce]:AM:FREQuency <freq> [:SOURce]:AM:FREQuency? [:SOURce]:AM:SHAPE SINE TRIangle SQUare RAMP [:SOURce]:AM:SHAPE? [:SOURce]:AM:DEPTh[:LINear] <double> [:SOURce]:AM:DEPTh[:LINear]?
Description	
Examples	AM ON AM:FREQ 10kHz AM:SHAPE SINE AM:DEPTH 50
Software Controls	AM Controls -> Enabled AM Controls -> Rate AM Controls -> Depth(%) AM Controls -> Shape
Couplings	None
Preset	
Notes	None

7.10 Frequency Modulation

Command	[:SOURce]:FM[:STATe] <bool> [:SOURce]:FM[:STATe]? [:SOURce]:FM:FREQuency <freq> [:SOURce]:FM:FREQuency? [:SOURce]:FM:SHAPE SINE TRIangle SQUare RAMP [:SOURce]:FM:SHAPE?
---------	---

	[:SOURce]:FM:DEViation <double> [:SOURce]:FM:DEViation?
Description	
Examples	FM ON FM:FREQ 20kHz FM:SHAPE RAMP FM:DEV 100kHz
Software Controls	FM Controls -> Enabled FM Controls -> Rate FM Controls -> Deviation FM Controls -> Shape
Couplings	None
Preset	
Notes	None

7.11 Pulse Modulation

Command	[:SOURce]:PULM[:STATe] <bool> [:SOURce]:PULM[:STATe]? [:SOURce]:PULM:TRIGger:TYPE SINGLE CONTinuous [:SOURce]:PULM:TRIGger:TYPE? [:SOURce]:PULM:INTernal:PWIDth <time> [:SOURce]:PULM:INTernal:PWIDth? [:SOURce]:PULM:INTernal:PERiod <time> [:SOURce]:PULM:INTernal:PERiod?
Description	
Examples	PULM ON PULM:TRIG:TYPE CONT PULM:INT:PWID 10us PULM:INT:PER 1ms
Software Controls	Pulse Controls -> Enabled Pulse Controls -> Trigger Mode Pulse Controls -> Width Pulse Controls -> Period
Couplings	None
Preset	
Notes	None

7.12 Multitone

Command	[:SOURce]:MTONE[:STATe] <bool> [:SOURce]:MTONE[:STATe]? [:SOURce]:MTONE:PHASe FIXed RANDom PARAbolic [:SOURce]:MTONE:PHASe? [:SOURce]:MTONE:PHASe:SEED <int> [:SOURce]:MTONE:PHASe:SEED? [:SOURce]:MTONE:NTONes <int>
----------------	--

	[:SOURce]:MTONE:NTONes? [:SOURce]:MTONE:FSPacing <freq> [:SOURce]:MTONE:FSPacing? [:SOURce]:MTONE:FNOTch <freq> [:SOURce]:MTONE:FNOTch?
Description	
Examples	MTON ON MTON:PHAS PARA MTON:PHAS:SEED 1234 MTON:NTON 1001 MTON:FSP 10kHz MTON:FNOT 1MHz
Software Controls	Multitone Controls -> Enabled Multitone Controls -> Tone Phase Multitone Controls -> Seed Multitone Controls -> Tone Count Multitone Controls -> Freq Spacing Multitone Controls -> Notch Width
Couplings	None
Preset	
Notes	None

7.13 Step Sweep

Command	[:SOURce]:STEP[:STATE] <bool> [:SOURce]:STEP[:STATE]? [:SOURce]:STEP:TRIGger:TYPE SINGLE CONTinuous [:SOURce]:STEP:TRIGger:TYPE? [:SOURce]:STEP:TYPE FREQ FREQAMPL [:SOURce]:STEP:TYPE? [:SOURce]:STEP:FREQuency:STARt <freq> [:SOURce]:STEP:FREQuency:STARt? [:SOURce]:STEP:FREQuency:STOP <freq> [:SOURce]:STEP:FREQuency:STOP? [:SOURce]:STEP:POINts <int> [:SOURce]:STEP:POINts? [:SOURce]:STEP:AMPLitude:STARt <double> [:SOURce]:STEP:AMPLitude:STARt? [:SOURce]:STEP:AMPLitude:STOP <double> [:SOURce]:STEP:AMPLitude:STOP? [:SOURce]:STEP:DWELL <time> [:SOURce]:STEP:DWELL?
Description	
Examples	STEP ON STEP:TRIG:TYPE SING STEP:TYPE FREQ STEP:FREQ:STAR 1GHz STEP:FREQ:STOP 2GHz

	STEP:POIN 1000 STEP:AMPL:START -20 STEP:AMPL:STOP -100 STEP:DWEL 100ms
Software Controls	Step Sweep Controls -> Enabled Step Sweep Controls -> Trigger Mode Step Sweep Controls -> Sweep Type Step Sweep Controls -> Start Freq Step Sweep Controls -> Stop Freq Step Sweep Controls -> Points Step Sweep Controls -> Start Level Step Sweep Controls -> Stop Level Step Sweep Controls -> Dwell Time
Couplings	None
Preset	
Notes	None

7.14 Ramp Sweep

Command	[:SOURce]:RAMP[:STATe] <bool> [:SOURce]:RAMP[:STATe]? [:SOURce]:RAMP:TRIGger:TYPE SINGLE CONTinuous [:SOURce]:RAMP:TRIGger:TYPE? [:SOURce]:RAMP:FREQuency:SPAN <freq> [:SOURce]:RAMP:FREQuency:SPAN? [:SOURce]:RAMP:SWEep:TIME <time> [:SOURce]:RAMP:SWEep:TIME? [:SOURce]:RAMP:SWEep:PERiod <time> [:SOURce]:RAMP:SWEep:PERiod?
Description	
Examples	RAMP ON RAMP:TRIG:TYPE SING RAMP:FREQ:SPAN 20MHz RAMP:SWE:TIME 1ms RAMP:SWE:PER 1s
Software Controls	Ramp Sweep Controls -> Enabled Ramp Sweep Controls -> Trigger Mode Ramp Sweep Controls -> Span Ramp Sweep Controls -> Sweep Time Ramp Sweep Controls -> Period
Couplings	None
Preset	
Notes	None

7.15 AWGN

Command	[:SOURce]:RADio:AWGN[:STATe] <bool>
---------	-------------------------------------

	[:SOURce]:RADio:AWGN[:STATe]? [:SOURce]:RADio:AWGN:BWIDth <freq> [:SOURce]:RADio:AWGN:BWIDth? [:SOURce]:RADio:AWGN:LENGth <time> [:SOURce]:RADio:AWGN:LENGth? [:SOURce]:RADio:AWGN:SEED <int> [:SOURce]:RADio:AWGN:SEED?
Description	
Examples	RAD:AWGN ON RAD:AWGN:BWID 20M RAD:AWGN:LEN 100ms RAD:AWGN:SEED 23
Software Controls	AWGN Controls -> Enabled AWGN Controls -> Bandwidth AWGN Controls -> Length AWGN Controls -> Seed
Couplings	None
Preset	
Notes	None

7.16 Custom Digital Modulation

Command	[:SOURce]:RADio:CUSTom[:STATe] <bool> [:SOURce]:RADio:CUSTom[:STATe]? [:SOURce]:RADio:CUSTom:TRIGger:TYPE SINGLE CONTinuous [:SOURce]:RADio:CUSTom:TRIGger:TYPE? [:SOURce]:RADio:CUSTom:IDLE <time> [:SOURce]:RADio:CUSTom:IDLE? [:SOURce]:RADio:CUSTom:SRATe <freq> [:SOURce]:RADio:CUSTom:SRATe? [:SOURce]:RADio:CUSTom:MODulation[:TYPE] BPSK DBPSK QPSK DQPSK OQPSK P4DQPSK PSK8 D8PSK PSK16 QAM16 QAM64 QAM256 QAM1024 ASK FSK2 FSK4 FSK8 FSK16 CUSTom [:SOURce]:RADio:CUSTom:MODulation[:TYPE]? [:SOURce]:RADio:CUSTom:MODulation:CUSTom <float>,<float>,...,<float> [:SOURce]:RADio:CUSTom:MODulation:CUSTom? [:SOURce]:RADio:CUSTom:MODulation:CUSTom:LENGth? [:SOURce]:RADio:CUSTom:MODulation:CUSTom:VALid? [:SOURce]:RADio:CUSTom:FILTer RNYQuist NYQuist GAUSSian RECTangle HALFsine CUSTom [:SOURce]:RADio:CUSTom:FILTer? [:SOURce]:RADio:CUSTom:FILTer:ALPHa <double> [:SOURce]:RADio:CUSTom:FILTer:ALPHa? [:SOURce]:RADio:CUSTom:FILTer:LENGth <int> [:SOURce]:RADio:CUSTom:FILTer:LENGth? [:SOURce]:RADio:CUSTom:DATA PN7 PN9 PN15 PN21 CUSTom [:SOURce]:RADio:CUSTom:DATA? [:SOURce]:RADio:CUSTom:DATA:SEED <int>
----------------	--

	[:SOURce]:RADio:CUSTom:DATA:SEED? [:SOURce]:RADio:CUSTom:DATA:SEquence <string> [:SOURce]:RADio:CUSTom:DATA:SEquence? [:SOURce]:RADio:CUSTom:MODulation:FSK[:DEViation] <freq> [:SOURce]:RADio:CUSTom:MODulation:FSK[:DEViation]? [:SOURce]:RADio:CUSTom:OVERsample <int> [:SOURce]:RADio:CUSTom:OVERsample?
Description	The string for the data sequence must contain only ascii '0's and '1's. If any other character is present, including whitespace, a system error will be thrown and the custom bit sequence will not be set.
Examples	RAD:CUST ON RAD:CUST:TRIG:TYPE SING RAD:CUST:TIME 1ms RAD:CUST:SRAT 1MHz RAD:CUST:MOD QAM16 RAD:CUST:MOD:CUST 1,1,-1,1,-1,-1,1,-1 RAD:CUST:MOD:CUST? RAD:CUST:MOD:CUST:LENG? RAD:CUST:MOD:CUST:VAL? RAD:CUST:FILT RNYQ RAD:CUST:FILT:ALPH 0.2 RAD:CUST:FILT:LEN 16 RAD:CUST:DATA PN15 RAD:CUST:DATA:SEED 11 RAD:CUST:DATA:SEQ 0011001101010101 RAD:CUST:DATA:SEQ? RAD:CUST:MOD:FSK:DEV 250kHz RAD:CUST:OVER 4
Software Controls	Digital Mod Controls -> Enabled Digital Mod Controls -> Trigger Mode Digital Mod Controls -> Idle Interval Digital Mod Controls -> Symbol Rate Digital Mod Controls -> Modulation Type Digital Mod Controls -> Define Modulation Digital Mod Controls -> Filter Type Digital Mod Controls -> Filter Alpha Digital Mod Controls -> Filter Length (symbols) Digital Mod Controls -> Sequence Digital Mod Controls -> Sequence Seed Digital Mod Controls -> Define Sequence Digital Mod Controls -> FSK Deviation Digital Mod Controls -> Oversample
Couplings	None
Preset	
Notes	

7.17 802.11a

Command	<pre>[:SOURCE] :RADio:WLAN:NONHT: [:STATE] <bool> [:SOURCE] :RADio:WLAN:NONHT: [:STATE]> [:SOURCE] :RADio:WLAN:NONHT:TRIGger:TYPE SINGLE CONTinuous [:SOURCE] :RADio:WLAN:NONHT:TRIGger:TYPE? [:SOURCE] :RADio:WLAN:NONHT:IDLE <time> [:SOURCE] :RADio:WLAN:NONHT:IDLE? [:SOURCE] :RADio:WLAN:NONHT:MCS <int> [:SOURCE] :RADio:WLAN:NONHT:MCS? [:SOURCE] :RADio:WLAN:NONHT:SRATe <freq> [:SOURCE] :RADio:WLAN:NONHT:SRATe? [:SOURCE] :RADio:WLAN:NONHT:INTERleave [:STATE] <bool> [:SOURCE] :RADio:WLAN:NONHT:INTERleave [:STATE]? [:SOURCE] :RADio:WLAN:NONHT:SCRAMble [:STATE] <bool> [:SOURCE] :RADio:WLAN:NONHT:SCRAMble [:STATE]? [:SOURCE] :RADio:WLAN:NONHT:SCRAMble:INIT <int> [:SOURCE] :RADio:WLAN:NONHT:SCRAMble:INIT? [:SOURCE] :RADio:WLAN:NONHT:WINDow:LENgth <int> [:SOURCE] :RADio:WLAN:NONHT:WINDow:LENgth? [:SOURCE] :RADio:WLAN:NONHT:DATA PN7 PN9 PN15 PN21 CUSTom [:SOURCE] :RADio:WLAN:NONHT:DATA? [:SOURCE] :RADio:WLAN:NONHT:DATA:SEED <int> [:SOURCE] :RADio:WLAN:NONHT:DATA:SEED? [:SOURCE] :RADio:WLAN:NONHT:DATA:LENgth <int> [:SOURCE] :RADio:WLAN:NONHT:DATA:LENgth? [:SOURCE] :RADio:WLAN:NONHT:SEQuence <string> [:SOURCE] :RADio:WLAN:NONHT:SEQuence?</pre>
Description	<p>MCS values should be between [0,7]</p> <p>Only sample rate can be specified, subcarrier spacing is controlled through the UI only. These settings are linked and can be controlled through just the sample rate control.</p> <p>Data length controls the number of bytes to use from the source. If the source is shorter than the length, the data is repeated until the length is met.</p> <p>The string for the data sequence must contain only ascii '0's and '1's. If any other character is present, including whitespace, a system error will be thrown and the custom bit sequence will not be set.</p>
Examples	<pre>RAD:WLAN:NONHT:TRIG:TYPE CONT RAD:WLAN:NONHT:IDLE 10ms RAD:WLAN:NONHT:MCS 7 RAD:WLAN:NONHT:SRATE 20MHz RAD:WLAN:NONHT:INTER true RAD:WLAN:NONHT:SCRAM 93 RAD:WLAN:NONHT:SCRAM:INIT 13 RAD:WLAN:NONHT:WIN:LEN 4 RAD:WLAN:NONHT:DATA PN15</pre>

	RAD:WLAN:NONHT:SEED 1
	RAD:WLAN:NONHT:DATA:LEN 1500
	RAD:WLAN:NONHT:DATA:SEQ 0011001101010101
Software Controls	See IEEE 802.11a controls
Couplings	None
Preset	
Notes	None

7.18 802.11n

Command	<pre>[:SOURce]:RADio:WLAN:HT[:STATe] <bool> [:SOURce]:RADio:WLAN:HT[:STATe]> [:SOURce]:RADio:WLAN:HT:TRIGger:TYPE SINGLE CONTInuous [:SOURce]:RADio:WLAN:HT:TRIGger:TYPE? [:SOURce]:RADio:WLAN:HT:IDLE <time> [:SOURce]:RADio:WLAN:HT:IDLE? [:SOURce]:RADio:WLAN:HT:MCS <int> [:SOURce]:RADio:WLAN:HT:MCS? [:SOURce]:RADio:WLAN:HT:GI SHORT LONG [:SOURce]:RADio:WLAN:HT:GI? [:SOURce]:RADio:WLAN:HT:SRATe <freq> [:SOURce]:RADio:WLAN:HT:SRATe? [:SOURce]:RADio:WLAN:HT:INTERleave[:STATe] <bool> [:SOURce]:RADio:WLAN:HT:INTERleave[:STATe]? [:SOURce]:RADio:WLAN:HT:SCRAMble[:STATe] <bool> [:SOURce]:RADio:WLAN:HT:SCRAMble[:STATe]? [:SOURce]:RADio:WLAN:HT:SCRAMble:INIT <int> [:SOURce]:RADio:WLAN:HT:SCRAMble:INIT? [:SOURce]:RADio:WLAN:HT:WINDow:LENgth <int> [:SOURce]:RADio:WLAN:HT:WINDow:LENgth? [:SOURce]:RADio:WLAN:HT:DATA PN7 PN9 PN15 PN21 CUSTom [:SOURce]:RADio:WLAN:HT:DATA? [:SOURce]:RADio:WLAN:HT:DATA:SEED <int> [:SOURce]:RADio:WLAN:HT:DATA:SEED? [:SOURce]:RADio:WLAN:HT:DATA:LENgth <int> [:SOURce]:RADio:WLAN:HT:DATA:LENgth? [:SOURce]:RADio:WLAN:HT:SEQuence <string> [:SOURce]:RADio:WLAN:HT:SEQuence?</pre>
Description	<p>MCS values should be between [0,7]</p> <p>Only sample rate can be specified, subcarrier spacing is controlled through the UI only. These settings are linked and can be controlled through just the sample rate control.</p> <p>Data length controls the number of bytes to use from the source. If the source is shorter than the length, the data is repeated until the length is met.</p>

	The string for the data sequence must contain only ascii '0's and '1's. If any other character is present, including whitespace, a system error will be thrown and the custom bit sequence will not be set.
Examples	RAD:WLAN:HT:TRIG:TYPE CONT RAD:WLAN:HT:IDLE 10ms RAD:WLAN:HT:MCS 7 RAD:WLAN:HT:GI LONG RAD:WLAN:HT:SRATE 20MHz RAD:WLAN:HT:INTER true RAD:WLAN:HT:SCRAM 93 RAD:WLAN:HT:SCRAM:INIT 13 RAD:WLAN:HT:WIN:LEN 4 RAD:WLAN:HT:DATA PN15 RAD:WLAN:HT:SEED 1 RAD:WLAN:HT:DATA:LEN 1500 RAD:WLAN:HT:DATA:SEQ 0011001101010101
Software Controls	See IEEE 802.11n controls
Couplings	None
Preset	
Notes	None

7.19 802.11ac

Command	[:SOURce]:RADio:WLAN:VHT[:STATe] <bool> [:SOURce]:RADio:WLAN:VHT[:STATe]> [:SOURce]:RADio:WLAN:VHT:TRIGger:TYPE SINGLE CONTinuous [:SOURce]:RADio:WLAN:VHT:TRIGger:TYPE? [:SOURce]:RADio:WLAN:VHT:IDLE <time> [:SOURce]:RADio:WLAN:VHT:IDLE? [:SOURce]:RADio:WLAN:VHT:MCS <int> [:SOURce]:RADio:WLAN:VHT:MCS? [:SOURce]:RADio:WLAN:VHT:GI SHORT LONG [:SOURce]:RADio:WLAN:VHT:GI? [:SOURce]:RADio:WLAN:VHT:SRATe <freq> [:SOURce]:RADio:WLAN:VHT:SRATe? [:SOURce]:RADio:WLAN:VHT:INTERleave[:STATe] <bool> [:SOURce]:RADio:WLAN:VHT:INTERleave[:STATe]? [:SOURce]:RADio:WLAN:VHT:SCRAMble[:STATe] <bool> [:SOURce]:RADio:WLAN:VHT:SCRAMble[:STATe]? [:SOURce]:RADio:WLAN:VHT:SCRAMble:INIT <int> [:SOURce]:RADio:WLAN:VHT:SCRAMble:INIT? [:SOURce]:RADio:WLAN:VHT:GID <int> [:SOURce]:RADio:WLAN:VHT:GID? [:SOURce]:RADio:WLAN:VHT:PAID <int> [:SOURce]:RADio:WLAN:VHT:PAID? [:SOURce]:RADio:WLAN:VHT:WINDow:LENgth <int> [:SOURce]:RADio:WLAN:VHT:WINDow:LENgth?
---------	--

	<pre>[:SOURCE]:RADio:WLAN:VHT:DATA PN7 PN9 PN15 PN21 CUSTom [:SOURCE]:RADio:WLAN:VHT:DATA? [:SOURCE]:RADio:WLAN:VHT:DATA:SEED <int> [:SOURCE]:RADio:WLAN:VHT:DATA:SEED? [:SOURCE]:RADio:WLAN:VHT:DATA:LENgth <int> [:SOURCE]:RADio:WLAN:VHT:DATA:LENgth? [:SOURCE]:RADio:WLAN:VHT:SEQuence <string> [:SOURCE]:RADio:WLAN:VHT:SEQuence?</pre>
Description	<p>MCS values should be between [0,8]</p> <p>Only sample rate can be specified, subcarrier spacing is controlled through the UI only. These settings are linked and can be controlled through just the sample rate control.</p> <p>Data length controls the number of bytes to use from the source. If the source is shorter than the length, the data is repeated until the length is met.</p> <p>The string for the data sequence must contain only ascii '0's and '1's. If any other character is present, including whitespace, a system error will be thrown and the custom bit sequence will not be set.</p>
Examples	<pre>RAD:WLAN:VHT:TRIG:TYPE CONT RAD:WLAN:VHT:IDLE 10ms RAD:WLAN:VHT:MCS 7 RAD:WLAN:VHT:GI LONG RAD:WLAN:VHT:SRATE 20MHz RAD:WLAN:VHT:INTER true RAD:WLAN:VHT:SCRAM 93 RAD:WLAN:VHT:SCRAM:INIT 13 RAD:WLAN:VHT:GID 63 RAD:WLAN:VHT:PAID 275 RAD:WLAN:VHT:WIN:LEN 4 RAD:WLAN:VHT:DATA PN15 RAD:WLAN:VHT:SEED 1 RAD:WLAN:VHT:DATA:LEN 1500 RAD:WLAN:VHT:DATA:SEQ 0011001101010101</pre>
Software Controls	See IEEE 802.11ac controls
Couplings	None
Preset	
Notes	None

7.20 802.11ax

Command	<pre>[:SOURCE]:RADio:WLAN:HE[:STATe] <bool> [:SOURCE]:RADio:WLAN:HE[:STATe]> [:SOURCE]:RADio:WLAN:HE:TRIGger:TYPE SINGLE CONTInuous [:SOURCE]:RADio:WLAN:HE:TRIGger:TYPE? [:SOURCE]:RADio:WLAN:HE:IDLE <time></pre>
---------	--

	<pre> [:SOURCE]:RADio:WLAN:HE:IDLE? [:SOURCE]:RADio:WLAN:HE:BWIDth 20M 40M [:SOURCE]:RADio:WLAN:HE:BWIDth? [:SOURCE]:RADio:WLAN:HE:CODing BCC LDPC [:SOURCE]:RADio:WLAN:HE:CODing? [:SOURCE]:RADio:WLAN:HE:MCS <int> [:SOURCE]:RADio:WLAN:HE:MCS? [:SOURCE]:RADio:WLAN:HE:GI <int> [:SOURCE]:RADio:WLAN:HE:GI? [:SOURCE]:RADio:WLAN:HE:SRATe <freq> [:SOURCE]:RADio:WLAN:HE:SRATe? [:SOURCE]:RADio:WLAN:HE:SCRAMble:INIT <int> [:SOURCE]:RADio:WLAN:HE:SCRAMble:INIT? [:SOURCE]:RADio:WLAN:HE:WINdow:LENgth <int> [:SOURCE]:RADio:WLAN:HE:WINdow:LENgth? [:SOURCE]:RADio:WLAN:HE:DATA PN7 PN9 PN15 PN21 CUSTom [:SOURCE]:RADio:WLAN:HE:DATA? [:SOURCE]:RADio:WLAN:HE:DATA:SEED <int> [:SOURCE]:RADio:WLAN:HE:DATA:SEED? [:SOURCE]:RADio:WLAN:HE:DATA:LENgth <int> [:SOURCE]:RADio:WLAN:HE:DATA:LENgth? [:SOURCE]:RADio:WLAN:HE:SEQuence <string> [:SOURCE]:RADio:WLAN:HE:SEQuence? </pre>
Description	<p>MCS values should be between [0,11]</p> <p>GI values should be between [0,3] and represent the 4 choices in the guard interval combo box.</p> <p>Only sample rate can be specified, subcarrier spacing is controlled through the UI only. These settings are linked and can be controlled through just the sample rate control.</p> <p>Data length controls the number of bytes to use from the source. If the source is shorter than the length, the data is repeated until the length is met.</p> <p>The string for the data sequence must contain only ascii '0's and '1's. If any other character is present, including whitespace, a system error will be thrown and the custom bit sequence will not be set.</p>
Examples	<pre> RAD:WLAN:HE:TRIG:TYPE CONT RAD:WLAN:HE:IDLE 10ms RAD:WLAN:HE:BWIDTH 20M RAD:WLAN:HE:CODING BCC RAD:WLAN:HE:MCS 7 RAD:WLAN:HE:GI 1 RAD:WLAN:HE:SRATE 20MHz RAD:WLAN:HE:SCRAM 93 RAD:WLAN:HE:SCRAM:INIT 13 RAD:WLAN:HE:WIN:LEN 4 RAD:WLAN:HE:DATA PN15 </pre>

	RAD:WLAN:HE:SEED 1
	RAD:WLAN:HE:DATA:LEN 1500
	RAD:WLAN:HE:DATA:SEQ 0011001101010101
Software Controls	See IEEE 802.11ax controls
Couplings	None
Preset	
Notes	None

7.21 Arb

Command	<pre>[:SOURce]:RADio:ARB[:STATe] <bool> [:SOURce]:RADio:ARB[:STATe]? [:SOURce]:RADio:ARB:TRIGger:TYPE SINGLE CONTInuous [:SOURce]:RADio:ARB:TRIGger:TYPE? [:SOURce]:RADio:ARB:SRATe <freq> [:SOURce]:RADio:ARB:SRATe? [:SOURce]:RADio:ARB:IQ:SCALE:AUTO[:STATe] <bool> [:SOURce]:RADio:ARB:IQ:SCALE:AUTO[:STATe]? [:SOURce]:RADio:ARB:IQ:SCALE <double> [:SOURce]:RADio:ARB:IQ:SCALE? [:SOURce]:RADio:ARB:IQ:SCALE:AVERAge[:STATe] <bool> [:SOURce]:RADio:ARB:IQ:SCALE:AVERAge[:STATe]? [:SOURce]:RADio:ARB:SAMPle:PERiod <int> [:SOURce]:RADio:ARB:SAMPle:PERiod? [:SOURce]:RADio:ARB:SAMPle:OFFSet <int> [:SOURce]:RADio:ARB:SAMPle:OFFSet? [:SOURce]:RADio:ARB:SAMPle:COUNt <int> [:SOURce]:RADio:ARB:SAMPle:COUNt? [:SOURce]:RADio:ARB:WAVeform? [:SOURce]:RADio:ARB:WAVeform:LENgth? [:SOURce]:RADio:ARB:WAVeform:LOAD:CSV <filename> [:SOURce]:RADio:ARB:WAVeform:LOAD:BINSC <filename> [:SOURce]:RADio:ARB:WAVeform:LOAD:BINFC <filename> [:SOURce]:RADio:ARB:WAVeform:LOAD:MIDAS <filename> [:SOURce]:RADio:ARB:WAVeform:LOAD:WAV <filename> [:SOURce]:RADio:ARB:WAVeform:LOAD:SEQuence <filename> [:SOURce]:RADio:ARB:WAVeform:LOAD:IQ:ASCII <I1>, <Q1>, <I2>, <Q2>, ..., <In>, <Qn> [:SOURce]:RADio:ARB:WAVeform:LOAD? [:SOURce]:RADio:ARB:WAVeform:UNLOAD</pre>
Description	<p>STATe, Enable/disable the Arb output mode.</p> <p>TRIGger:TYPE, Set the trigger mode for Arb output.</p> <p>SRATe, Set the Arb output sample rate.</p> <p>IQ:SCALE:AUTO:STATe, Enable/disable auto I/Q scaling.</p>

IQ:SCALE, Set the I/Q scale to be used when auto scaling is disabled.

IQ:SCALE:AVERAge:STATe, Enable/disable how to calculate the output power of the signal.

SAMPlE:PERiod, Set the waveform period in samples. Period is calculated after accounting for the offset and count.

SAMPlE:OFFSet, Set the waveform offset in samples. Specifies how many samples into the loaded waveform to start playback. Between offset and count, this allows users to only play a portion of the loaded waveform.

SAMPlE:COUNt, Specify the number of samples after the offset to output. Between offset and count, this allows users to only play a portion of the loaded waveform.

WAVeform?, Queries the name of the loaded waveform. Returns an empty string if no file is loaded.

WAVeform:LENgth?, Returns the total number of samples in the loaded waveform. The number returned does not include the offset and count values specified above. If no file is loaded, this returns 0.

LOAD, Loads various file types. The file name provided must specify a file that matches the file type specified by the load SCPI function used. See the software UI manual for more information.

LOAD:BINSC, Loads 16-bit complex integer binary file with provided filename.

LOAD:BINFC, Loads 32-bit complex float binary file with provided filename.

LOAD:SEQuence, Loads a custom sequence file (created with the sequence editor) with the provided filename.

LOAD:IQ:ASCIi, Load an I/Q waveform sent over SCPI. The I/Q values should be provided as alternating I/Q complex values, each I and Q value sent as a separate SCPI parameter, as ascii. A comma should separate all I/Q values. A comma should not be placed after the last Q value. An error will be thrown if an odd number of parameters is provided. See example below and programming example for usage.

LOAD?, Returns 1 if a waveform is loaded.

UNLOAD, Unloads any loaded waveform.

Examples

```
RAD:ARB ON
RAD:ARB:TRIG:TYPE SING
RAD:ARB:SRAT 10MHz
RAD:ARB:IQ:SCALE:AUTO ON
RAD:ARB:IQ:SCALE 50
RAD:ARB:IQ:SCALE:AVERAGE OFF
RAD:ARB:SAMPLE:PERIOD 10000
RADIO:ARB:SAMPLE:OFFSET 1024
RAD:ARB:SAMP:OFFS?
RAD:ARB:SAMP:COUNT 5000
RAD:ARB:WAV?
RAD:ARB:WAVEFORM:LENGTH?
```

```
RAD:ARB:WAV:LOAD:CSV "file.csv"
```

Please note, that the quotations must appear in the command. If using a programming language like C/C++, you must escape sequence the quote in the string, for example

```
"RAD:ARB:WAV:LOAD:CSV \"file.csv\""
```

	RAD:ARB:WAV:LOAD:BINFC "file.bin" RAD:ARB:WAV:LOAD:IQ:ASCII 1.0, 0.0, 1.0, 0.0, -1.0, 0.0, -1.0, 0.0 <This line loaded an I/Q waveform with 4 I/Q samples where the first two samples were (1.0,0.0) and the last two samples were (-1.0, 0.0)> See the programming examples for another example of using this function. RAD:ARB:WAV:LOAD? RAD:ARB:WAV:UNLOAD
Software Controls	Arb Controls -> Enabled Arb Controls -> Trigger Mode Arb Controls -> Sample Rate Arb Controls -> Auto Scale Arb Controls -> I/Q Scale (%) Arb Controls -> Output Signal Average Arb Controls -> Period Arb Controls -> Sample Offset Arb Controls -> Samples to Use Arb Controls -> Samples in File Arb Controls -> Load Arb Controls -> Unload File
Couplings	None
Preset	No file loaded.
Notes	None

7.22 Streaming

Command	[:SOURce]:STREAMing[:STATe] <bool> [:SOURce]:STREAMing[:STATe]? [:SOURce]:STREAMing:SRATe <freq> [:SOURce]:STREAMing:SRATe? [:SOURce]:STREAMing:IQ:SCALE <double> [:SOURce]:STREAMing:IQ:SCALE? [:SOURce]:STREAMing:WAVeform:LOAD:BINSC <filename> [:SOURce]:STREAMing:WAVeform:LOAD:BINFC <filename> [:SOURce]:STREAMing:WAVeform:LOAD:WAV <filename> [:SOURce]:STREAMing:WAVeform:LOAD? [:SOURce]:STREAMing:WAVeform:UNLOAD
Description	STATe, Enable/disable the streaming output mode. SRATe, Set the output sample rate. IQ:SCALE, Set the I/Q scale as a percentage. LOAD:BINSC, Loads 16-bit complex integer binary file with provided filename. LOAD:BINFC, Loads 32-bit complex float binary file with provided filename. LOAD:WAV, Loads .wav file with provided filename. UNLOAD, Unloads all waveform files.
Examples	STREAMING ON STREAM:SRAT 10MHz STREAMING:IQ:SCALE 50

```
STREAM:WAV:LOAD:BINSC "file.bin"
```

Please note, that the quotations must appear in the command. If using a programming language like C/C++, you must escape sequence the quote in the string, for example

```
"STREAM:WAV:LOAD:BINSC \"file.bin\""
```

```
STREAM:WAV:LOAD:BINFC "file.bin"
```

```
STREAM:WAV:LOAD:WAV "file.wav"
```

```
STREAM:WAV:LOAD?
```

```
STREAM:WAV:UNLOAD
```

Software Controls	Streaming Controls -> Enabled
	Streaming Controls -> Sample Rate
	Streaming Controls -> I/Q Scale (%)
	Streaming Controls -> Load Files
	Streaming Controls -> Unload Files
Couplings	None
Preset	No files loaded.
Notes	None
