Series
MS278XB
High Performance Signal Analyzer

Programming Manual

Software Version 4.xx

490 JARVIS DRIVE
MORGAN HILL, CA 95037-2809

P/N: 10410-00274
REVISION: D
PRINTED: JULY 2007

WARRANTY

The Anritsu product(s) listed on the title page is (are) warranted against defects in materials and workmanship for three years from the date of shipment.

Anritsu's obligation covers repairing or replacing products which prove to be defective during the warranty period. Buyers shall prepay transportation charges for equipment returned to Anritsu for warranty repairs. Obligation is limited to the original purchaser. Anritsu is not liable for consequential damages.

LIMITATION OF WARRANTY

The foregoing warranty does not apply to Anritsu connectors that have failed due to normal wear. Also, the warranty does not apply to defects resulting from improper or inadequate maintenance by the Buyer, unauthorized modification or misuse, or operation outside of the environmental specifications of the product. No other warranty is expressed or implied, and the remedies provided herein are the Buyer's sole and exclusive remedies.

END-USER LICENSE AGREEMENT FOR ANRITSU SIGNATURE SOFTWARE

IMPORTANT-READ CAREFULLY: This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and Anritsu for the Signature software product identified above, which includes computer software and associated media and printed materials, and may include "online" or electronic documentation ("SOFTWARE PRODUCT" or "SOFTWARE"). By receiving or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA.

SOFTWARE PRODUCT LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

1. GRANT OF LICENSE. This EULA grants you the following rights:

a. You may use ONE copy of the Software Product identified above only on the hardware product (Signature Signal Analyzer and its internal computer) which it was originally installed. The SOFTWARE is in "use" on a computer when it is loaded into temporary memory (for example, RAM) or installed into permanent memory (for example, hard disk, CD-ROM, or other storage device) of that computer. However, installation on a network server for the sole purpose of internal distribution to one or more other computer(s) shall not constitute "use."

b. Solely with respect to electronic documents included with the SOFTWARE, you may make an unlimited number of copies (either in hardcopy or electronic form), provided that such copies shall be used only for internal purposes and are not republished or distributed to any third party.

2. OWNERSHIP. Except as expressly licensed to you in this Agreement, Anritsu retains all right, title, and interest in and to the SOFTWARE PRODUCT; provided, however, that, subject to the license grant in Section 1.a and Anritsu's ownership of the underlying SOFTWARE PRODUCT, you shall own all right, title and interest in and to any Derivative Technology of the Product created by or for you.

3. COPYRIGHT. All title and copyrights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the SOFTWARE PRODUCT), the accompanying printed materials, and any copies of the SOFTWARE PRODUCT are owned by Anritsu or its suppliers. The SOFTWARE PRODUCT is protected by copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE PRODUCT like any other copyrighted material except that you may make one copy of the SOFTWARE PRODUCT solely for backup or archival purposes. You may not copy any printed materials accompanying the SOFTWARE PRODUCT.

4. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.

a. Limitations on Reverse Engineering, Decompilation, and Disassembly. You may not reverse engineer, decompile, or disassemble the SOFTWARE, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.

b. Rental. You may not rent or lease the SOFTWARE PRODUCT.

c. Software Transfer. You may permanently transfer all of your rights under this EULA, provided that you retain no copies, you transfer all of the SOFTWARE PRODUCT (including the Signature Signal Analyzer, all component parts, the media and printed materials, any upgrades, this EULA, and, if applicable, the Certificate of Authenticity), and the recipient agrees to the terms of this EULA.

d. Termination. Without prejudice to any other rights, Anritsu may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the SOFTWARE PRODUCT.

DISCLAIMER OF WARRANTY

TRADEMARK ACKNOWLEDGMENTS

NOTICE

UPDATES

Updates, if any, can be downloaded from the Documents area of the Anritsu web site at:
http://www.us.anritsu.com

Table of Contents

Chapter 1— General Information

Chapter 2— GPIB Programming Commands

Table of Contents (Continued)

Chapter 1
General Information

1-1 Scope of This Manual

This manual provides programming information and data for all models of the Series MS278XB signal analyzer. The Command Dictionary provides the entire command set for programming all features available with the MS278XB. Consequently, not all of the codes documented in this manual apply to all models within the series. It is expected that the user is aware of the feature set available within the model for which programming is being written. Feature set information is documented in the operation manual.

1-2 Related Manuals

All models in the MS278XB series are covered within the same set of manuals. The manual set consists of an Operation Manual (OM), Part Number: 10410-00273; a Maintenance Manual (MM), Part Number: 10410-0275; and this Programming Manual.

1-3 Introduction

This chapter contains information on how to setup the MS278XB remote interface and programming environment. Equipment requirements and interface/configuration instructions are provided for all remote interface options. The following sections provide basic information and a brief description of the MS278XB facilities for remote operation.

1-4 GPIB Interface Port Selection

The MS278XB fully supports the GPIB IEEE Std. 488.2–1992. The IEEE-488 General Purpose Interface Bus (GPIB) is an instrumentation interface for integrating additional instruments, computers, and other measurement devices into remote controllable systems. The IEEE Std. 488.2 specifies the use of protocols, formats, and certain common commands for use with the GPIB. All MS278XB front panel functions (except power on/off and GPIB test) can be controlled remotely using the GPIB commands listed in this manual and an external computer equipped with an IEEE 488.2 GPIB controller.

Required Equipment

The following list represents the minimum equipment requirements for a GPIB controllable MS278XB system:

- A MS278XB Spectrum/ Vector Signal Analyzer
- A computer/controller that supports GPIB IEEE 488.2 (The examples in this chapter address IBM compatible computers.)
- An IEEE 488 GPIB interface (built in, or add in peripheral card) with appropriate driver software (The National Instruments GPIB IEEE-488.2 interface is assumed for all examples in this chapter.)
- Appropriate software (any of the following):
    - Microsoft® Visual C#®, Visual Basic®, Visual C++®, or Java®
    - Any other programming language or application software that supports the IEEE 488 GPIB interface
- A GPIB cable (preferably two meters long).

> **Note:** The IBM PC and National Instruments GPIB interface were chosen for demonstrating the MS278XB GPIB operation in this manual. Any other GPIB controller that conforms to the IEEE 488 standard can be used to interface with the MS278XB.

IEEE 488 Bus Functional Elements

Effective communications between devices on the GPIB requires three functional elements:

- Talker
- Listener
- Controller

Each device on the GPIB is categorized as one of these elements depending on its current interface function and capabilities.

Talker

A talker is a device capable of sending device-dependent data to another device on the bus when addressed to talk. Only one GPIB device at a time can be an active talker.

Listener

A listener is a device capable of receiving device-dependent data from another device on the bus when addressed to listen. Any number of GPIB devices can be listeners simultaneously.

Controller

A controller is a device, usually a computer, capable of managing the operation of the GPIB. Only one GPIB device at a time can be an active controller. The active controller manages the transfer of device-dependent data between GPIB devices by designating who will talk and who will listen.

System Controller

The system controller is the device that always retains ultimate control of the GPIB. When the system is first powered-up, the system controller is the active controller and manages the GPIB. The system controller can pass control to another device, making it the new active controller. The new active controller, in turn, may pass control on to yet another device. Even if it is not the active controller, the system controller maintains control of the Interface Clear (IFC) and Remote Enable (REN) interface management lines and can thus take control of the GPIB at anytime.

When in the GPIB operating mode, the MS278XB can function as a listener, talker, controller, or system controller.

IEEE 488 Bus Structure

The GPIB uses 16 signal lines to carry data and commands between the devices connected to the bus. The interface signal lines are organized into three functional groups:

- Data Bus (8 lines)
- Data Byte Transfer Control Bus (3 lines)
- General Interface Management Bus (5 lines)

The signal lines in each of the three groups are designated according to function. Figure 1-1 and Table 1-1 on page 5 illustrate these designations.

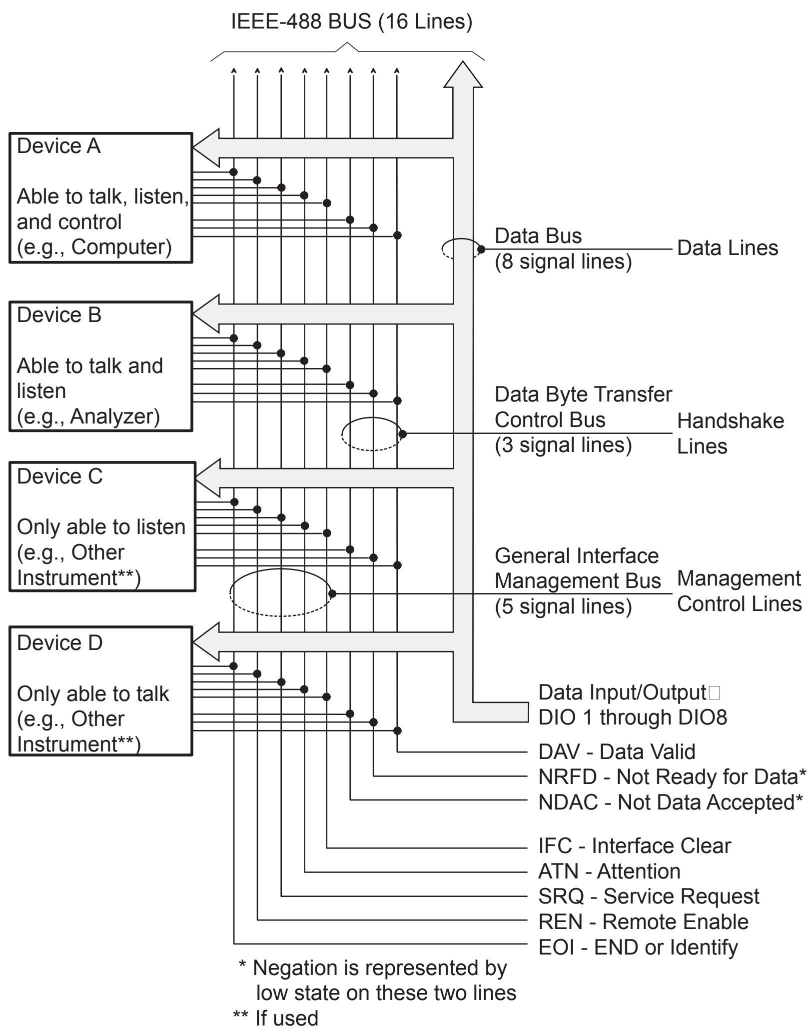


Figure 1-1. IEEE 488.2 Bus Interface Connections and Structure

Table 1-1. Interface Bus Signal Line Designations

Bus Type	Signal Line Name	Function
Function	DIO1–DIO8	Data Input/Output, 1 thru 8
Data Byte Transfer and Control	DAV NRFD NDAC	Data Available Not Ready For Data Not Data Accepted
General Interface Control	ATN IFC SRQ REN EOI	Attention Interface Clear Service Request Remote Enable End Or Identify

Data Bus Description

The data bus is the conduit for the transfer of data and commands between the devices on the GPIB. It contains eight bi-directional, active-low signal lines—DIO 1 through DIO 8. Data and commands are transferred over the data bus in byte-serial, bit-parallel form. This means that one byte of data (eight bits) is transferred over the bus at a time. DIO 1 represents the least-significant bit (LSB) in this byte and DIO 8 represents the most-significant bit (MSB). Bytes of data are normally formatted in seven-bit ASCII (American Standard Code for Information Interchange) code. The eighth (parity) bit is not used.

Each byte placed on the data bus represents either a command or a data byte. If the Attention (ATN) interface management line is TRUE while the data is transferred, then the data bus is carrying a bus command which is to be received by every GPIB device. If ATN is FALSE, then a data byte is being transferred and only the active listeners will receive that byte.

Data Byte Transfer Control Bus Description

Control of the transfer of each byte of data on the data bus is accomplished by a technique called the "three-wire handshake," which involves the three signal lines of the Data Byte Transfer Control Bus. This technique forces data transfers at the speed of the slowest listener, which ensures data integrity in multiple listener transfers. One line (DAV) is controlled by the talker, while the other two (NRFD and NDAC) are wired-OR lines shared by all active listeners. The handshake lines, like the other GPIB lines, are active low. The technique is described briefly in the following paragraphs and is illustrated in Figure 1-2. For further information, refer to ANSI/IEEE Standard 488.1.

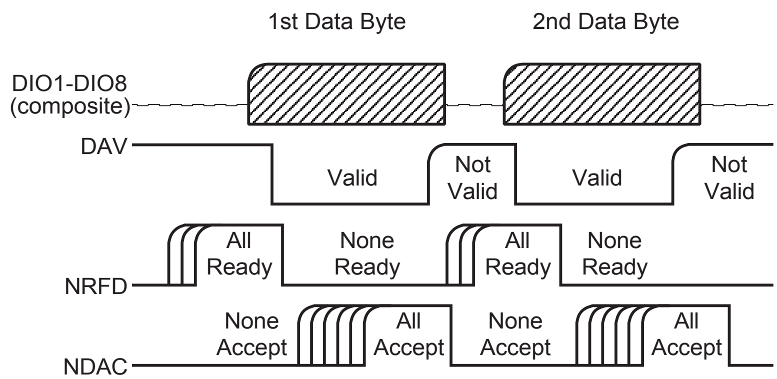


Figure 1-2. Typical GPIB Handshake Operation

DAV (Data Valid)

This line is controlled by the active talker. Before sending any data, the talker verifies that NDAC is TRUE (active low) which indicates that all listeners have accepted the previous data byte. The talker then places a byte on the data lines and waits until NRFD is FALSE (high) which indicates that all addressed listeners are ready to accept the information. When both NRFD and NDAC are in the proper state, the talker sets the DAV line TRUE (active low) to indicate that the data on the bus is valid (stable).

NRFD (Not Ready For Data)

This line is used by the listeners to inform the talker when they are ready to accept new data. The talker must wait for each listener to set the NRFD line FALSE (high) which they will do at their own rate. This assures that all devices that are to accept the data are ready to receive it.

NDAC (Not Data Accepted)

This line is also controlled by the listeners and is used to inform the talker that each device addressed to listen has accepted the data. Each device releases NDAC at its own rate, but NDAC will not go FALSE (high) until the slowest listener has accepted the data byte.

General Interface Management Bus Description

The general interface management bus is a group of five signal lines used to manage the flow of information across the GPIB. A description of the function of each of the individual control lines is provided below.

ATN (Attention)

The active controller uses the ATN line to define whether the information on the data bus is a command or is data. When ATN is TRUE (low), the bus is in the command mode and the data lines carry bus commands. When ATN is FALSE (high), the bus is in the data mode and the data lines carry device-dependent instructions or data.

EOI (End or Identify)

The EOI line is used to indicate the last byte of a multibyte data transfer. The talker sets the EOI line TRUE during the last data byte.

The active controller also uses the EOI line in conjunction with the ATN line to initiate a parallel poll sequence.

IFC (Interface Clear)

Only the system controller uses this line. When IFC is TRUE (low), all devices on the bus are placed in a known, quiescent state (unaddressed to talk, unaddressed to listen, and service request idle).

REN (Remote Enable)

Only the system controller uses this line. When REN is set TRUE (low), the bus is in the remote mode and devices are addressed either to listen or to talk. When the bus is in remote and a device is addressed, it receives instructions from the GPIB rather than from its front panel. When REN is set FALSE (high), the bus and all devices return to local operation.

SRQ (Service Request)

The SRQ line is set TRUE (low) by any device requesting service by the active controller.

Device Interface Function Capability

An interface function is the GPIB system element which provides the basic operational facility through which a device can receive, process, and send messages. Each specific interface function may only send or receive a limited set of messages within particular classes of messages. As a result, a set of interface functions is necessary to achieve complete communications among devices on the GPIB. ANSI/IEEE Std. 488.1 defines each of the interface functions along with its specific protocol.

ANSI/IEEE Std. 488.2 specifies the minimum set of IEEE 488.1 interface capabilities that each GPIB device must have. This minimum set of interface functions assures that the device is able to send and receive data, request service, and respond to a device clear message. Table 1-2 lists the interface function capability of the MS278XB analyzer.

Table 1-2. MS278XB Interface Function Capability

Function Identifier	Function	MS278XB Capability
AH1	Acceptor Handshake	Complete Compatibility
SH1	Source Handshake	Complete Compatibility
T6	Talker	No Talk Only (TON)
L4	Listener	No Listen Only (LON)
SR1	Service Request	Complete Compatibility
RL1	Remote/Local	Complete Compatibility
PP1	Parallel Poll	Complete Compatibility
DC1	Device Clear	Complete Compatibility
DT1	Device Trigger	Complete Compatibility
C0, C1, C2, C3, C28	Controller Capability Options	C0, No Compatibility C1, System Controller C2, Send IFC and Take Charge C3, Send REN C28, Send IF Messages
E2	Tri-state Drivers	Three-state Bus Drivers

Message Types

There are three types of information ted over the GPIB—interface function messages, device-specific commands, and data and instrument status messages.

Interface Function Messages

The controller manages the flow of information on the GPIB using interface function messages, usually called commands or command messages. Interface function messages perform such functions as initializing the bus, addressing and unaddressing devices, and setting device modes for remote or local operation.

There are two types of commands—multiline and uniline. Multiline commands are bytes sent by the active controller over the data bus (DIO1-DIO8) with ATN set TRUE. Uniline commands are signals carried by the individual interface management lines.

The user generally has control over these commands; however, the extent of user control depends on the implementation and varies with the specific GPIB interface hardware and software used with the external controller.

Device-Specific Commands

These commands are keywords or mnemonic codes sent by the external controller to control the setup and operation of the addressed device or instrument. The commands are normally unique to a particular instrument or class of instruments and are described in its documentation.

Device-specific commands are transmitted over the data bus of the GPIB to the device in the form of ASCII strings containing one or more keywords or codes. They are decoded by the device’s internal controller and cause the various instrument functions to be performed.

Data and Instrument Status Messages

These messages are sent by the device to the external controller via the GPIB. They contain measurement results, instrument status, or data files that the device transmits over the data bus in response to specific requests from the external controller. The contents of these messages are instrument specific and may be in the form of ASCII strings or binary data.

In some cases data messages will be transmitted from the external controller to the device. For example, messages to load calibration data.

An SRQ (service request) is an interface function message sent from the device to the external controller to request service from the controller, usually due to some predetermined status condition or error. To send this message, the device sets the SRQ line of the General Interface Management Bus true, then sends a status byte on the data bus lines.

An SRQ interface function message is also sent by the device in response to a serial poll message from the controller, or upon receiving an Output Status Byte(s) command from the controller. The protocols associated with the SRQ functions are defined in the ANSI/IEEE Std. 488.2-1992 document.

The manner in which interface function messages and device-specific commands are invoked in programs is implementation specific for the GPIB interface used with the external controller. Even though both message types are represented by mnemonics, they are implemented and used in different ways.

Normally, the interface function messages are sent automatically by the GPIB driver software in response to invocation of a software function. For example, to send the IFC (Interface Clear) interface function message, one would call the ibsic function of the National Instruments software driver. On the other hand, the command *RST (Reset) is sent in a command string to the addressed device. In the case of the National Instruments example, this would be done by using the ibwrt function call.

GPIB Interface Connection

Connect your external controller to the IEEE 488.2 GPIB interface connector on the rear panel. A pinout listing of this connector is contained in Table 1-3 on page 12.

The GPIB system can accommodate up to 15 devices at any one time. To achieve maximum performance on the bus, proper timing and voltage level relationships must be maintained. If either the cable length between separate instruments or the accumulated cable length between all instruments is too long, the data and control lines cannot be driven properly and the system may fail to perform. The following guidelines should be observed:

- No more than 15 instruments may be installed on the bus (including the controller).
- Total accumulative cable length (in meters) may not exceed two times the number of bus instruments or 20 meters—whichever is less.
- Individual cable length should not exceed 4 meters.
- 2/3 of the devices must be powered on.
- Devices should not be powered on while the bus is in operation (that is; actively sending or receiving messages, data, etc.).
- Minimize cable lengths to achieve maximum data transfer rates.

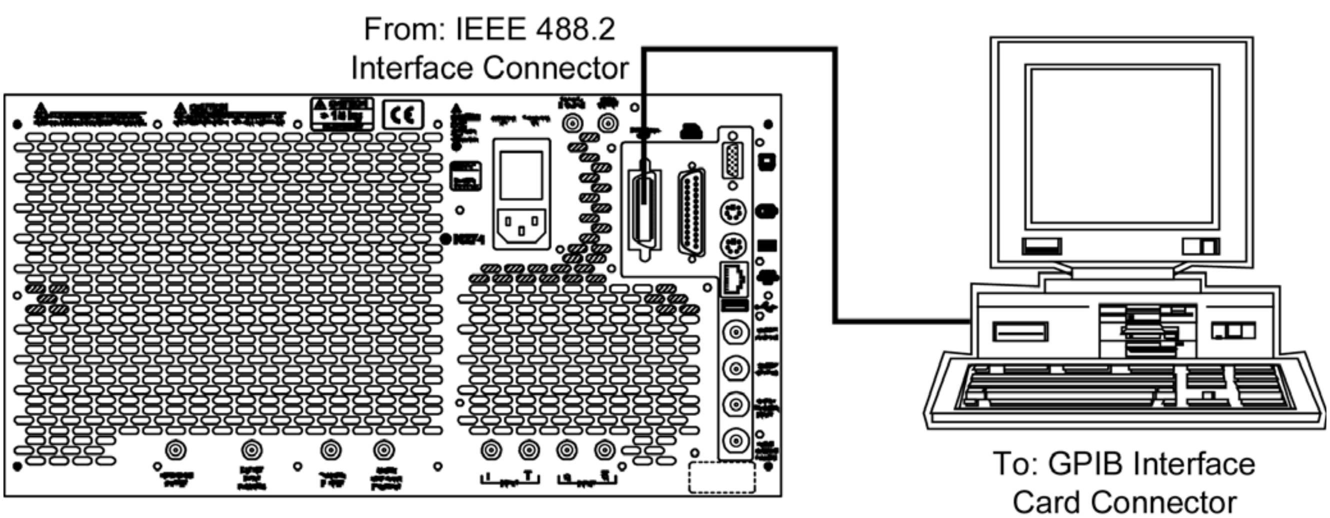
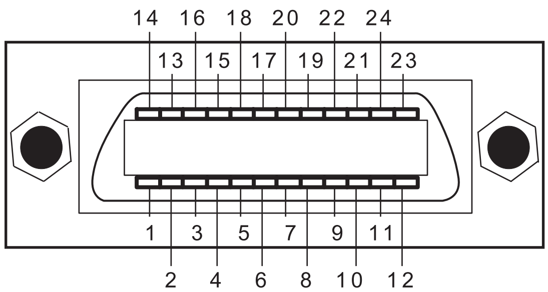


Figure 1-3. GPIB Setup

Table 1-3. IEEE 488.2 GPIB Connector Pinout Diagram



Pin	Name	Description
1-4	DIO 1 through DIO 4	Data Input/Output. Bits are HIGH when the data is logical 0 and LOW when the data is logical 1.
5	EOI	End or Identify. A low-true state indicates that the last byte of a multi byte message has been placed on the line.
6	DAV	Data Valid. A low-true state indicates that the talker has (1) sensed that NRFD is LOW, (2) placed a byte of data on the bus, and (3) waited an appropriate length of time for the data to settle.
7	NRFD	Not Ready For Data. A high-true state indicates that valid data has not yet been accepted by a listener.
8	NDAC	Not Data Accepted. A low-true state indicates that the current data byte has been accepted for internal processing by a listener.
9	IFC	Interface Clear. A low-true state places all bus instruments in a known state—such as, unaddressed to talk, unaddressed to listen, and service request idle.
10	SRQ	Service Request. A low-true state indicates that a bus instrument needs service from the controller.
11	ATN	Attention. A low-true state enables the controller to respond to both its own listen/talk address and to appropriate interface messages—such as, device clear and serial poll.
12	Shield	Ground Point.
13-16	DIO 5 through DIO 8	Data Input/Output. Bits are high with the data is logical 0 and LOW when the data is logical 1.
17	REN	Remote Enable. A low-true state enables bus instruments to be operated remotely, when addressed.
18-24	GND	Logic ground.

Configuration

Configure the MS278XB as shown in Figure 1-3. Apply power to the MS278XB and allow the system to power up. Once the software has finished loading and start-up testing is complete, the MS278XB is ready to be remotely controlled via the GPIB. It is important to note that the MS278XB will not respond to GPIB commands until the system's software has been loaded.

Apply power to the computer/controller and load the appropriate programming language.

The default GPIB address for the MS278XB is one (1) and is assumed for all examples in this chapter. To change the default GPIB address, select the following:

Step 1. Access the System main menu, expand the Configuration sub-menu, press the IO Config button, and then select GPIB.

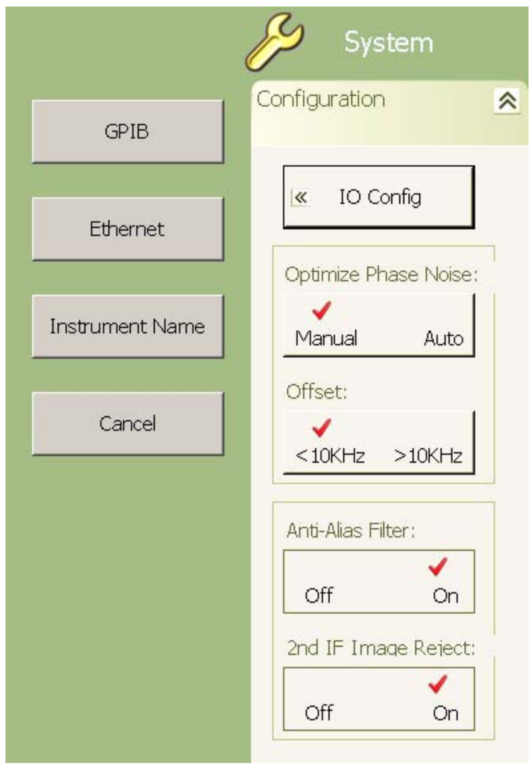


Figure 1-4. MS278XB Configuration Sub-menu

This brings up the Measurement and Automation Explorer window, below.

> **Note:** When changing the GPIB configuration, it is important to launch NI_MAX from the System | Configuration | GPIB button rather than launching from the All Programs shortcut in Windows. This ensures that the System software maintains access to the GPIB hardware.

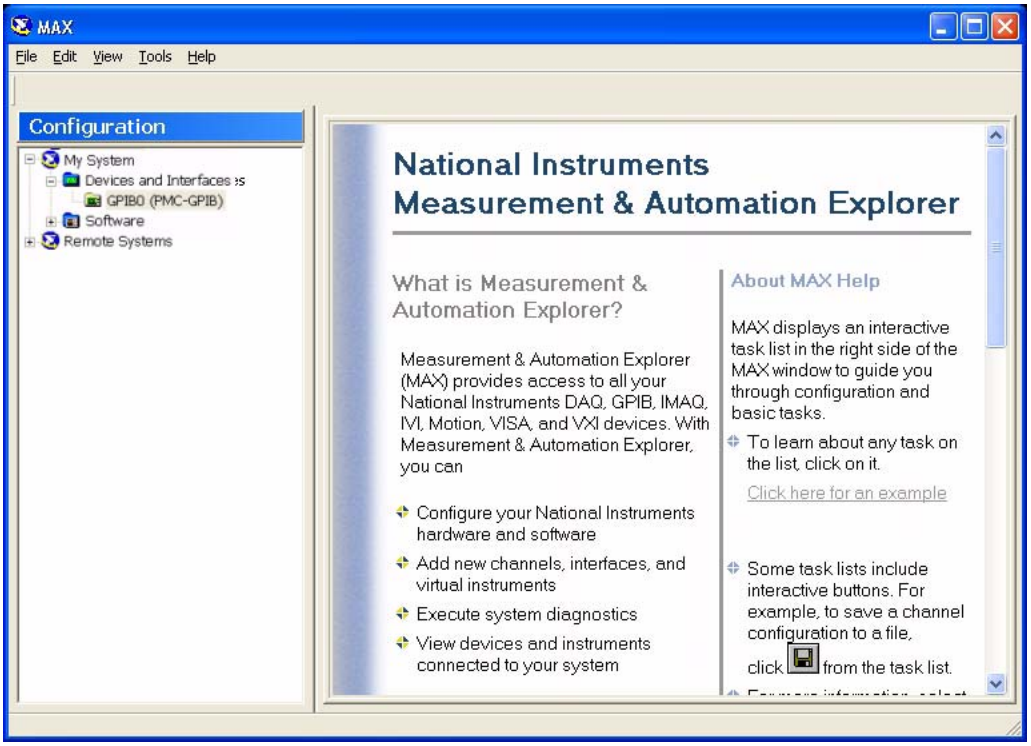


Figure 1-5. National Instruments Measurement and Automation Explorer

Step 2. On the left hand panel, go to My Systems | Devices and Interfaces | GPIB0 (PMC-GPIB), right click on GPIB0 select properties from the pop-up menu.

Step 3. In the GPIB Configuration dialog, change the Primary GPIB Address to the desired value.

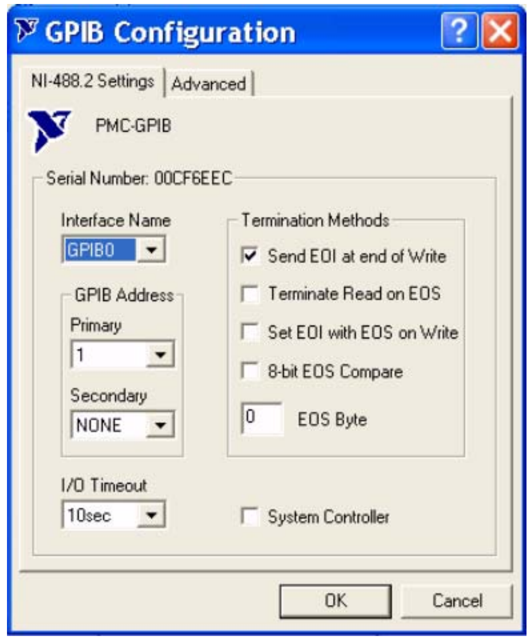


Figure 1-6. GPIB Configuration Dialog

Step 4. Make similar changes on the Remote PC side by selecting the System Controller choice and changing the GPIB address as required.

When Signature is selected as the system controller, the message “System Controller” is displayed in Signature’s status bar.

Note: Signature cannot be remotely controlled through GPIB by another remote PC when it is selected as the system controller.

Local Operation

To return to local front panel operation, press the Local/ Remote button located on the Remote Control menu.

Local operation will be restored unless the MS278XB is programmed for local lockout.

1-5 Ethernet Interface Port Selection

The MS278XB fully supports the IEEE-802.3 standard. Most MS278XB front panel functions (except power on/off and GPIB test) can be remotely controlled via a network server and an Ethernet connection. The MS278XB software supports the TCP/IP network protocol.

Ethernet Bus Structure

Ethernet uses a bus or star topology where all the interfacing devices are connected to a central cable called the bus or are connected to a hub. Ethernet uses the CSMA/CD access method to handle simultaneous transmissions over the bus. CSMA/CD stands for *Carrier Sense Multiple Access/Collision Detection*. This standard enables network devices to detect simultaneous data channel usage, called a *collision*, and provides for a contention protocol. When a network device detects a collision, the CSMA/CD standard dictates that the data will be retransmitted after waiting a random amount of time. If a second collision is detected, the data is again retransmitted after waiting twice as long. This is known as exponential back off.

The TCP/IP protocol setup requires the following:

- IP Address: Every computer/electronic device in a TCP/IP network requires an IP address. An IP address has four numbers (each between 0 and 255) separated by periods. For example: 128.111.122.42 is a valid IP address.
- Subnet Mask: The subnet mask distinguishes the portion of the IP address that is the network ID from the portion that is the station ID. The subnet mask 255.255.0.0, when applied to the IP address given above, would identify the network ID as 128.111 and the station ID as 122.42. All stations in the same Local Area Network should have the same network ID, but different station IDs.
- Default Gateway: A TCP/IP network can have a gateway to communicate beyond the LAN identified by the network ID. A gateway is a computer or electronic device that is connected to two different networks and can move TCP/IP data from one network to the other. A single LAN that is not connected to other LANs requires a default gateway setting of 0.0.0.0. If you have a gateway, then the default gateway would be set to the appropriate value of your gateway
- Ethernet Address: An Ethernet address is a unique 48-bit value that identifies a network interface card to the rest of the network. Every network card has a unique ethernet address permanently stored into its memory.

Because the MS278XB runs under the Windows XP platform, this setup process can be automated or manually configured using easy to follow setup wizards and dialog boxes as shown in "Network Interface Setup" on page 19 after your Network Connections are made.

Network Connections

The MS278XB supports 10/100BASE-T. You can connect the analyzer directly to your LAN via the RJ45 connector on the rear panel. Refer to Figure 1-7, below, for an illustration. A pinout and crossover cable assembly diagrams are shown in Table 1-4 and Table 1-5.

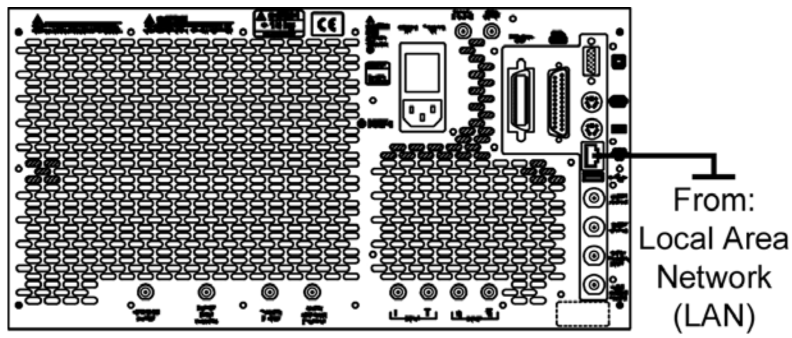
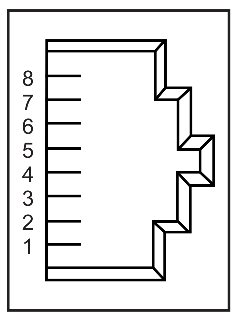


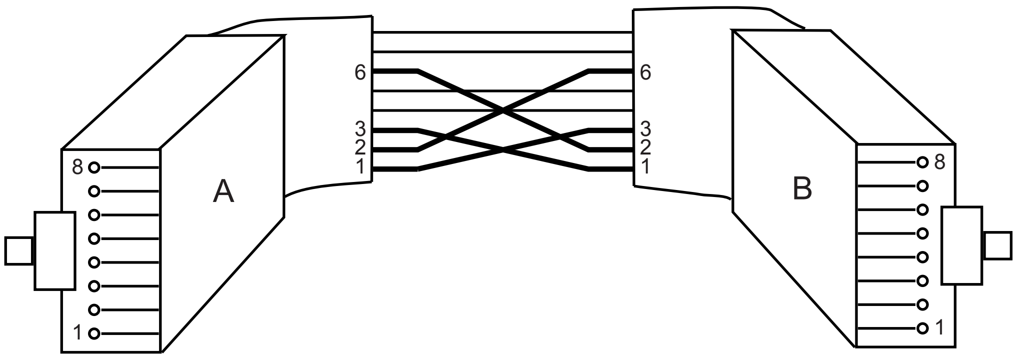
Figure 1-7. Ethernet Connection

Table 1-4. 8-pin Ethernet RJ45 Connector Pinout Diagram



Pin	Name	Description	Wire Color
1	TX+	Transmit data (> +3 volts)	White/Orange
2	TX–	Transmit data (< –3 volts)	Orange
3	RX+	Receive data (< –3 volts)	White/Green
4	–	Not used (common mode termination)	Blue
5	–	Not used (common mode termination)	White/Blue
6	RX–	Receive data (< –3 volts)	Green
7	–	Not used (common mode termination)	White/Brown
8	–	Not used (common mode termination)	Brown

Table 1-5. 8-pin Ethernet EIA/TIA 568B Cable Cross-over Wiring Diagram



Connector A		Connects To	Connector B	
Pin	Name		Pin	Name
1	TX+		3	RX+
2	TX–		6	RX–
3	RX+		1	TX+
6	RX–		2	TX–
4	–		4	–
5	–		5	–
7	–		7	–
8	–		8	–

Network Interface Setup

TCP/IP connectivity requires setting up the parameters described at the beginning of this section. You may need to contact your network administrator or refer to your network documentation for further assistance. The following is brief overview of how to set up a general LAN connection on both the MS278XB and the remote machine:

> **Note:** You may need to consult your network documentation or network administrator for assistance in manually configuring your network setup.

Step 1. From the Start menu, select Control Panel | Network Connections.

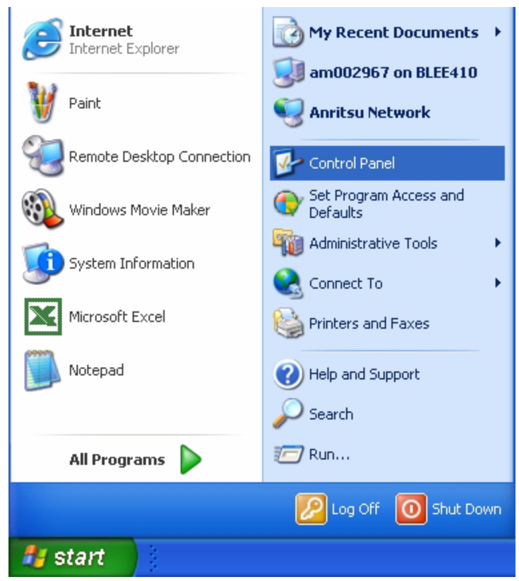


Figure 1-8. Start Menu

Step 2. From the Control Panel, select Network Connections.

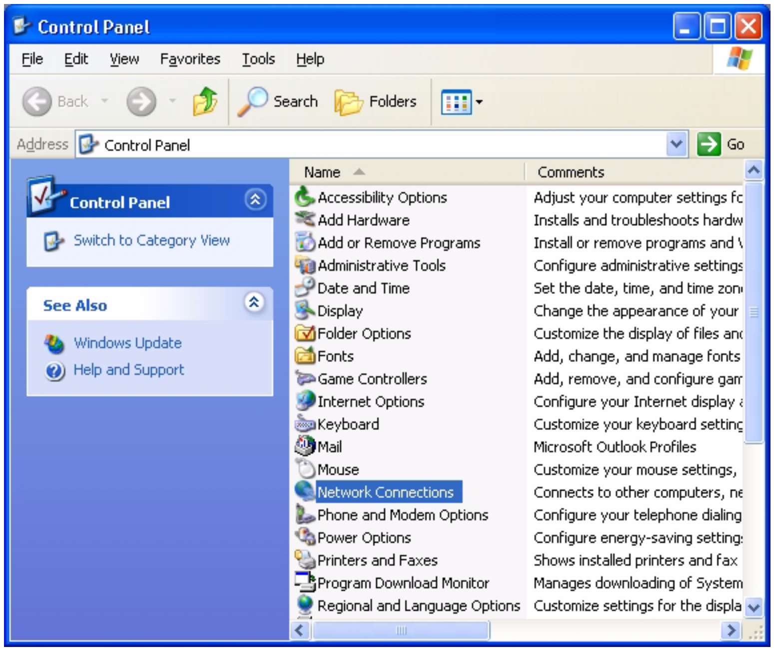


Figure 1-9. Control Panel

Step 3. In the Network Connections window, under Network Tasks on the left, select *Create a new connection*.

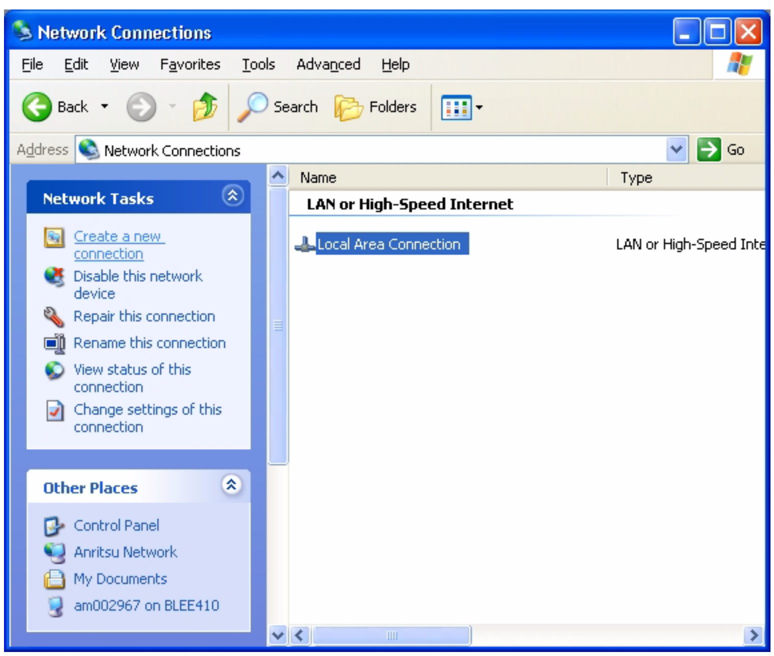


Figure 1-10. Network Connections

Step 4. Follow the directions in the New Connection Wizard to setup the new connection.

Figure 1-11. New Connection Wizard

Step 5. If a connection needs to be manually set up or modified, you can right click on the connection name in the Network Connections window (Figure 1-10) and select Properties from the pop-up dialog box.

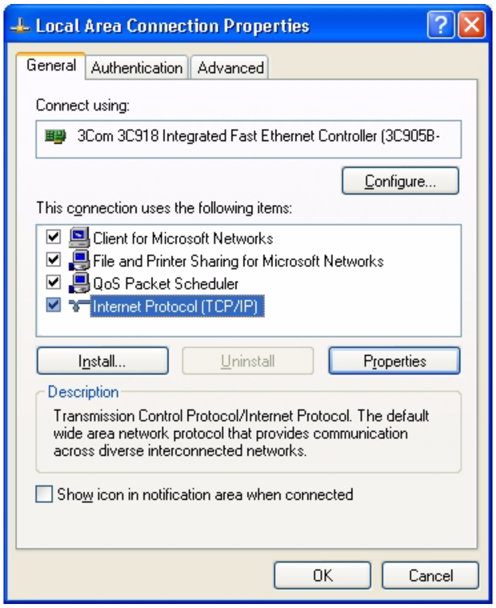


Figure 1-12. Local Area Connection Properties

Step 6. From the properties dialog above, select Internet Protocol (TCP/IP) and click on the Properties button.

Step 7. In the Internet Protocol (TCP/IP) Properties dialog, you can select to obtain an IP address automatically or manually configure your network connection.

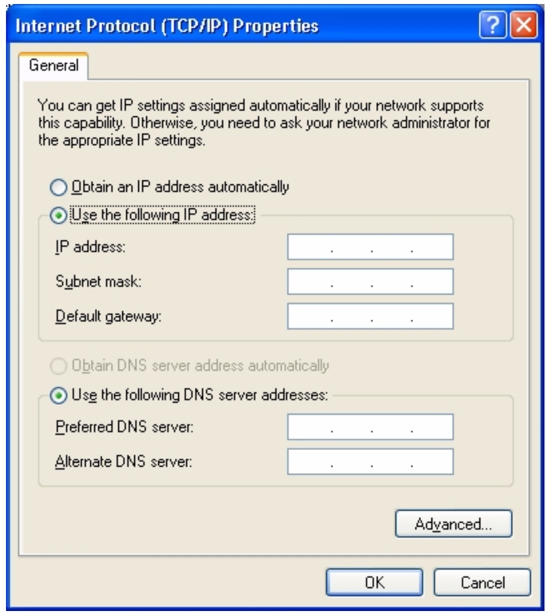


Figure 1-13. General Internet Protocol (TCP/IP) Properties

Step 8. For additional setup configurations, select Obtain an IP address automatically, then select the Alternate Configuration tab.

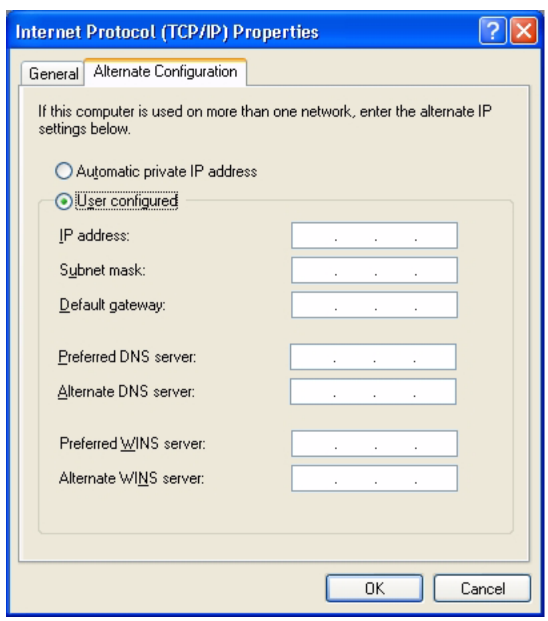


Figure 1-14. Alternate Internet Protocol (TCP/IP) Properties

1-6 SCPI Programming Over Ethernet

With Signature software version 4.0, the instrument can be remotely programmed using SCPI commands VIA LAN using the VXI-11 protocol. The VISA I/O library is used on the client side to facilitate the communications. The SCPI command set listed in the GPIB Programming Commands chapter applies to LAN programming as well.

The following VB6 example shows how to perform LAN programming using the NIVISA interface. To run the example, you must have NIVISA 2.5 or later installed and you must select the VISA library (visa32.dll) as a reference in your Visual Basic Project.

```
Public Sub Lan_Example()

   Const MAX_CHR_COUNT = 100
   Dim status As ViStatus
   Dim defaultRM As ViSession
   Dim session As ViSession
   Dim retCount As Long
   Dim buffer As String * MAX_CHR_COUNT

   ' Open Session to the Default Resource Manager resource
   status = viOpenDefaultRM(defaultRM)

   If (status < VI_SUCCESS) Then

      MsgBox "Error Initializing "
      Exit Sub

   End If

   status = viOpen(defaultRM, "TCPIP0::" + frmScript.txtHostName + "::INSTR",
   VI_NULL, 20000, session)

   If status <> 0 Then MsgBox "viOpen() failed.": Exit Sub

   ' Set the timeout
   status = viSetAttribute(session, VI_ATTR_TMO_VALUE, 5000)

   ' Write your command
   status = viWrite(session, "*IDN?", 5, retCount)

   ' Read back the result
   status = viRead(session, buffer, MAX_CHR_COUNT, retCount)
   MsgBox "*IDN?:  " & Left(Trim(buffer), retCount)

   ' Close down the system
   status = viClose(session)
   status = viClose(defaultRM)

End Sub
```

Figure 1-15. Programming SCPI Support through LAN in VB6

1-7 Programming with Web Services

This Section describes the MS278XB Ethernet programming interface using TCP/IP and Web Services. One of the primary advantages of Web Services architecture is that it allows programs written in different languages on different platforms to communicate with each other in a standards-based way. XML Web Services work with standard Web protocols—XML, HTTP and TCP/IP.

Web Services Protocol

Web Services expose useful functionality to Web users through a standard Web protocol. In most cases, the protocol used is Simple Object Access Protocol (SOAP).

Description of SOAP

SOAP is the communications protocol for XML Web Services. SOAP is a specification that defines the XML format for messages—a well-formed XML fragment enclosed in a couple of SOAP elements is a SOAP message.

There are other parts of the SOAP specification that describe how to represent program data as XML and how to use SOAP to do Remote Procedure Calls. These optional parts of the specification are used to implement RPC-style applications where a SOAP message containing a callable function, and the parameters to pass to the function, is sent from the client and the server returns a message with the results of the executed function. Signature's implementation of SOAP supports RPC applications.

SOAP also supports document style applications where the SOAP message is just a wrapper around an XML document.

The last optional part of the SOAP specification defines what an HTTP message that contains a SOAP message looks like. The HTTP binding is optional, but almost all SOAP implementations support it because it's the only standardized protocol for SOAP. For this reason, there's a common misconception that SOAP requires HTTP. Some implementations support MSMQ, MQ Series, SMTP, or TCP/IP transports, but almost all current XML Web Services use HTTP because it is ubiquitous.

You must use a SOAP toolkit to create and parse the SOAP messages. These toolkits generally translate function calls from some kind of language to a SOAP message. For example, the Microsoft SOAP Toolkit 3.0 translates COM function calls to SOAP. The types of function calls and the data types of the parameters supported vary with each SOAP implementation, so a function that works with one toolkit may not work with another. This is not a limitation of SOAP, but rather of the particular implementation you are using.

Web Services Description Language

XML Web Services provide a way to describe their own interfaces in enough detail to allow a user to build a client application to talk to them. This description is usually provided in an XML Web Services Description Language (WSDL) document.

WSDL

A WSDL file is an XML document that describes a set of SOAP messages and how the messages are exchanged. Since WSDL is XML, it is readable and editable, but in most cases it is generated and consumed by software.

The notation that a WSDL file uses to describe message formats is based on the XML Schema standard, which means it is both programming language neutral and standards based. This makes it suitable for describing XML Web Services interfaces that are accessible from a wide variety of platforms and programming languages.

In addition to describing message contents, WSDL defines where the service is available and what communications protocol is used to talk to the service. This means that the WSDL file defines everything required to write a program to work with an XML Web Service. There are several tools available to read a WSDL file and generate the code required to communicate with an XML Web Service. Some of the most capable of these tools are in Microsoft Visual Studio® .NET.

Many current SOAP toolkits include tools to generate WSDL files from existing program interfaces, but there are few tools for writing WSDL directly. The WSDL specification and a more detailed description of WSDL is available at: http://www.w3.org/TR/wsdl.

Universal Discovery Description and Integration

XML Web Services are registered so that potential users can find them easily. This is done with Universal Discovery Description and Integration (UDDI).

UDDI

UDDI is the yellow pages of Web Services. A UDDI directory entry is an XML file that describes the services it offers. There are three parts to an entry in the UDDI directory:

- The "white pages" describe the company offering the service: name, address, contacts, etc.
- The "yellow pages" include industrial categories based on standard taxonomies such as the North American Industry Classification System and the Standard Industrial Classification.
- The "green pages" describe the interface to the service in enough detail for someone to write an application to use the Web Service.

The way services are defined is through a UDDI document called a Type Model or tModel. In many cases, the tModel contains a WSDL file that describes a SOAP interface to an XML Web Service, but the tModel is flexible enough to describe almost any kind of service. The UDDI directory also includes several ways to search for the services you need to build your applications.

The WS-Inspection specification allows you to browse through a collection of XML Web Services offered on a specific server to find which ones might meet your needs. More information about UDDI is available at: http://www.uddi.org/about.html.

WS-Security is one of the specifications in the Global Web Services Architecture. Operational management needs, such as routing messages among many servers and configuring those servers dynamically for processing, are also part of the Global Web Services Architecture, and are met by the WS-Routing specification and the WS-Referral specification.

Programming Interface

Signature supports four Web Services:

- Signature System Control Web Service:

  This Web Service supports many Application Program Interfaces (APIs) that are applicable at the system level, as opposed to being measurement specific.

- Signature Spectrum Web Service:

  This Web Service supports many APIs to be able to control the instrument when in the spectrum measurement mode.

- Signature Modulation Web Service:

  This Web Service supports many APIs to be able to control the instrument when in the digital demodulation measurement mode.

- Signature WCDMA Web Service:

  This Web Service supports many APIs to be able to control the instrument when in the Wideband Code Division Multiple Access measurement mode.

The controller machine (usually a PC/Laptop) and Signature should be able to access each other over http via a network connection. The controller machine runs the client application. Figure 1-16, below, shows a typical control flow in a call to set the center frequency and read back the center frequency from the instrument using the Spectrum Measurement Web Service.

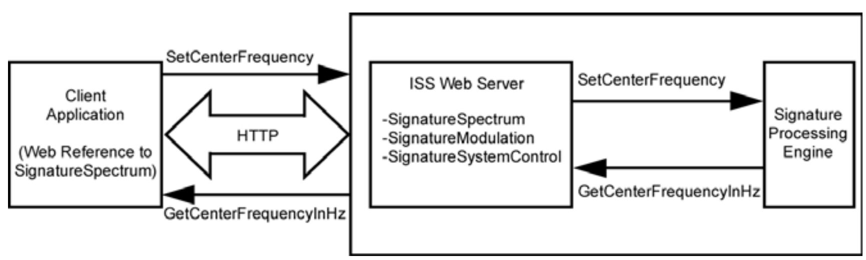


Figure 1-16. Web Service Control Flow Chart

Service Help Pages

Web Services also supply online programming help in the form of Service Help pages. When called from a Web browser without supplying a recognized query string, an .asmx file returns an automatically generated service help page for the XML Web service.

Signature provides access to the following service help pages using the following URLs:

http://localhost/SignatureSystemControl/SignatureSystemControl.asmx

http://localhost/SignatureSpectrum/SignatureSpectrum.asmx

http://localhost/SignatureModulation/SignatureModulation.asmx

http://localhost/SignatureWCDMA/SignatureWCDMA.asmx

Note: The URL may also point to a remote machine by using Signature's "computer name" instead of localhost.

These service help pages provide a list of the methods that the Web Service provides, which you can access programmatically. Each method has a link that takes you to additional information about that method.

The figure below illustrates the SignatureSpectrum class asmx help page.

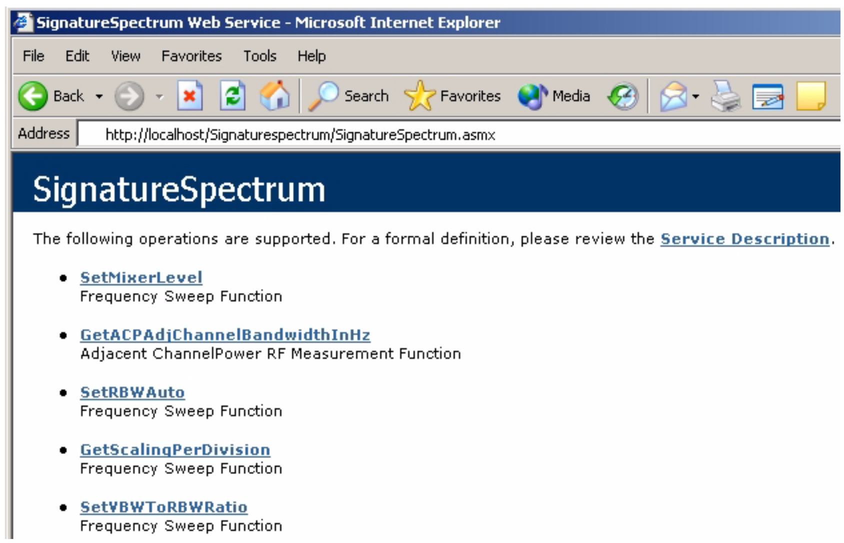


Figure 1-17. http://localhost/SignatureSpectrum/SignatureSpectrum.asmx Help Page

In addition, these pages contain links to the Web Service description document (WSDL) as shown in the figure below:.

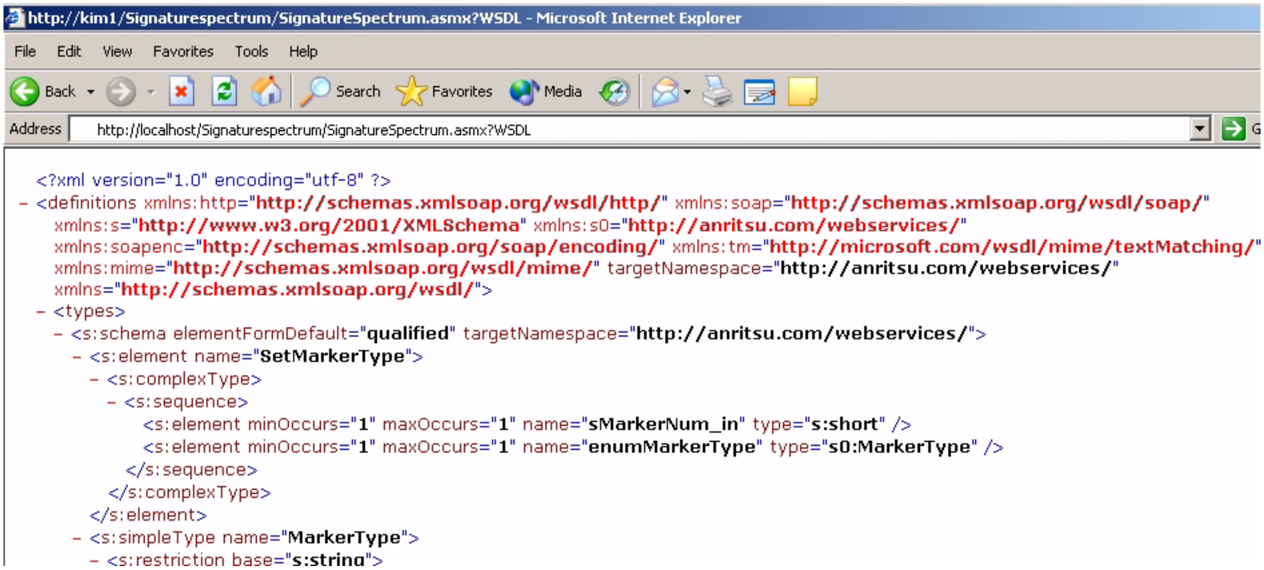


Figure 1-18. http://localhost/SignatureSpectrum/SignatureSpectrum.asmx?WSDL Help Page

Service Method Help Page

The service method help page provides additional information that relates to a particular Web Service method. The page provides the ability to invoke the method using the HTTP-POST protocol. At the bottom of the page, the service method help page provides sample request and response messages for the protocols that the Web Service method supports.

SignatureSpectrum

Click here for a complete list of operations.

GetCenterFrequency

Frequency Sweep Function

Test

The test form is only available for requests from the local machine.

SOAP

The following is a sample SOAP request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /spectrummeasurement/spectrummeasurement.asmx HTTP/1.1
Host: kim1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetCenterFrequency"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://w
  <soap:Body>
    <GetCenterFrequency xmlns="http://tempuri.org/" />
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://w
  <soap:Body>
    <GetCenterFrequencyResponse xmlns="http://tempuri.org/">
      <enumFreqUnits_out>Hz or KHz or MHz or GHz</enumFreqUnits_out>
      <dnewValue_out>double</dnewValue_out>
    </GetCenterFrequencyResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 1-19. http://localhost/SignatureSpectrum/SignatureSpectrum.asmx? Method Help Page

Initialization in the .NET Environment

The following procedure describes how to initialize the SignatureSpectrum web service. The same process is used to initialize the SignatureSystemControl and Signature-Modulation Web Services.

Step 1. In the Solution Explorer pane on Visual Studio .NET 2003, right click on *Web References* under the project listing. Refer to SampleWSClient in Figure 1-20 below:

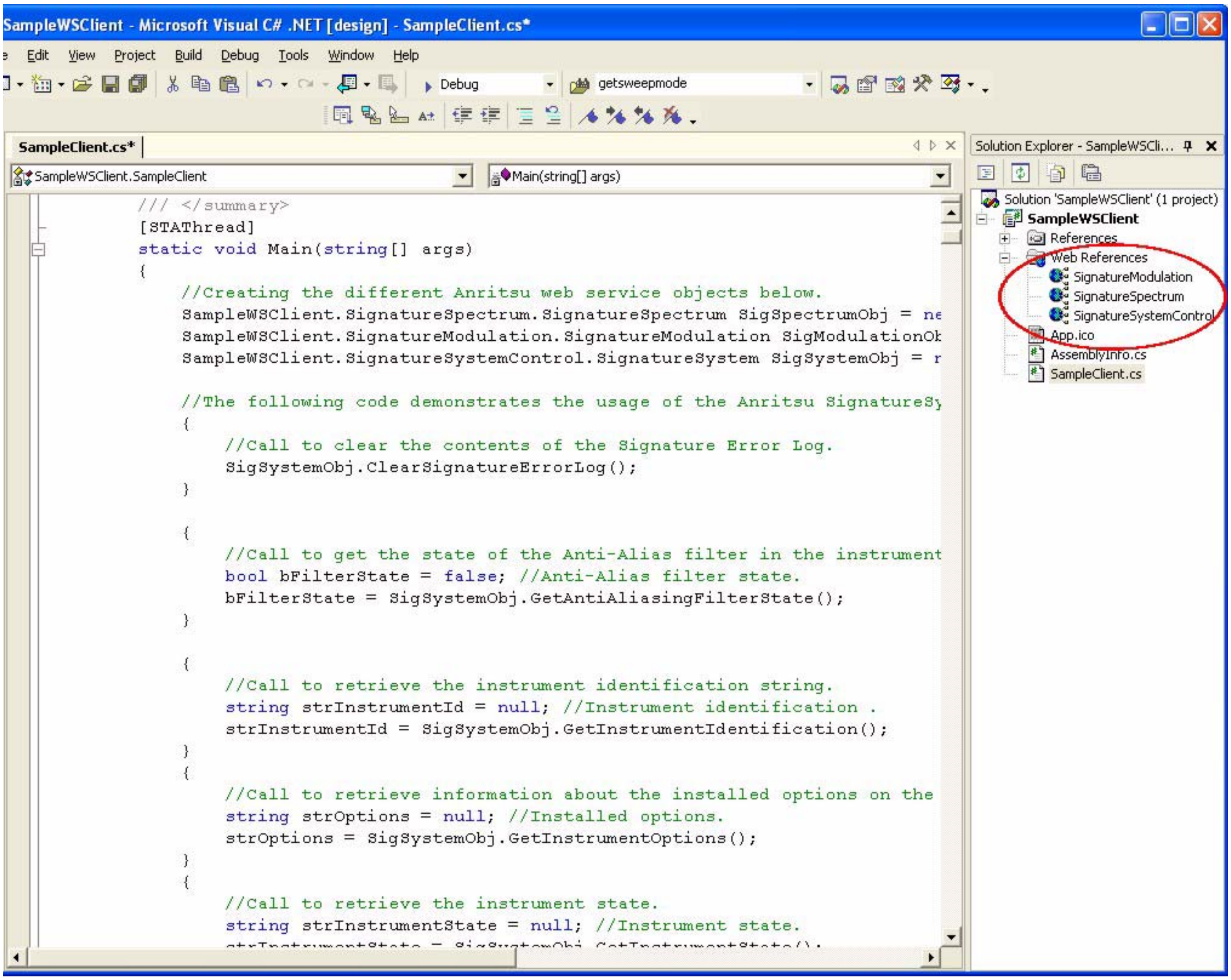


Figure 1-20. Visual Studio.NET SampleWSClient

Step 2. On the resulting pop-up dialog, click on *Add Web Reference* to open the Add Web Reference dialog shown in Figure 1-21 below:

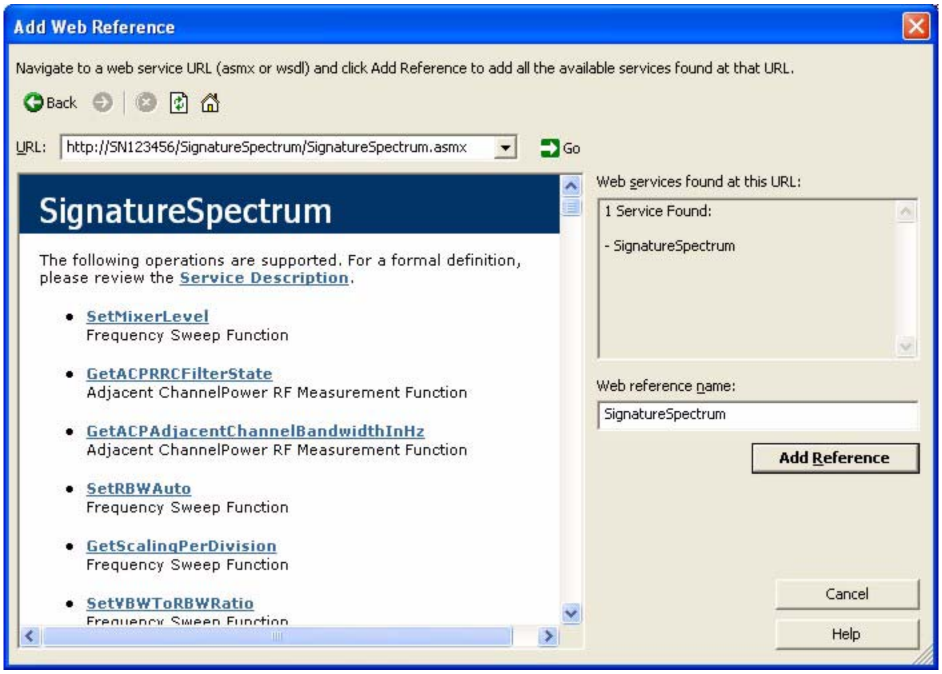


Figure 1-21. Add Web Reference Dialog

Step 3. In the URL field, type in the URL of the Web Service of interest. In this example, http://SN123456/SignatureSpectrum/SignatureSpectrum.asmx.

Step 4. Change the *Web reference name* to be the same as the Web Services name that you are initializing. In this example, SignatureSpectrum.

Step 5. Click on the *Add Reference* button. The object corresponding to this Web Service is then added to your application.

Step 6. Initialize the object in your application as follows:

```
SignatureSpectrum.SignatureSpectrum SigSpectrumObj = 
null;
```

Step 7. Instantiate the object of the above type as follows:

```
SigSpectrumObj = new 
SignatureSpectrum.SignatureSpectrum();
```

Step 8. Call a method on the above created object as follows: (Once the reference is added, it is a matter of calling methods on the object to communicate with the instrument.)

The following example sets the center frequency to 50 MHz in the SPA mode:

```
//Call to set the center frequency to 50 MHz.
double dValue_in = 0.0;
dValue_in = 50.0;
SigSpectrumObj.SetCenterFrequency(dValue_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

The following example reads the marker frequency value in the SPA mode:

```
//Call to read Marker3's frequency from the system
when in the SPA mode.
short sMarkerNum_in = 3;
double dValue_out = 0.0;
dValue_out =
SigSpectrumObj.getFrequencyMarkerPositionInHz(sMarkerN
um_in);
```

Initialization in the Visual Basic 6 Environment

Step 1. Download and install the Microsoft SOAP Toolkit from Microsoft's web page:

http://msdn.microsoft.com/webservices/building/soaptk/

Step 2. Run VB6 and start a new project.

Step 3. Add the SOAP reference to the project:

Project | References menu

Step 4. Select the "Microsoft SOAP Type Library V3.0."

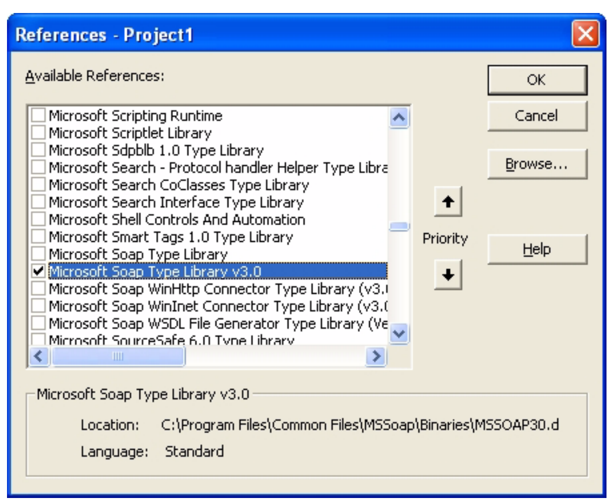


Figure 1-22. Adding a Reference in VB6

Now, you are ready to begin coding.

Step 1. Define a variable for each of the interfaces you wish to access:

- SignatureSystemControl
- SignatureSpectrum
- SignatureModulation
- SignatureWCDMA

Step 2. Assign the variable type as a new MSSOAPLib30.SoapClient30, for example:

```
Dim SignatureSystem As New MSSOAPLib30.SoapClient30
```

Step 3. Connect the new variable to the Web Services address, for example:

```
SignatureSystemControl.MSSoapInit &_
"http://SN123456/SignatureSystemControl/” &_
“SignatureSystemControl.asmx?wsdl"
```

Step 4. Issue any desired Web Services calls, for example:

```
Call SignatureSystemControl.Preset
```

> **Note:** The intellisense functionality of the Microsoft SOAP object variable does not include the Web Services commands. These commands must be typed directly into the editor without using the intellisense functionality.

Additionally, the properties of the SOAP object are available to use, such as:

```
SignatureSpectrum.ConnectorProperty("Timeout") = 1000
```

The default SOAP values will work, but you can adjust these values as necessary. See the SOAP documentation for full details on these properties and their use.

Chapter 2
GPIB Programming Commands

2-1 Introduction

This chapter contains all of the GPIB commands, Standard Commands for Programmable Instrumentation (SCPI), that are implemented in the instrument. The SCPI commands are grouped by their respective subsystems, first listed in tables, then described in detail. The notation corresponds to one of the SCPI standards to a large extent. The SCPI conformity information can be taken from the individual description of the commands.

> **Note:** When changing the GPIB configuration, it is important to launch NI_MAX from the System | Configuration | GPIB button rather than launching from the All Programs shortcut in Windows. This ensures that the System software maintains access to the GPIB hardware. Refer to “Configuration” on page 1-13 for more information on configuring the GPIB.

GPIB Commands

Each command subsystem first lists all of the commands of that subsystem in a Table of Commands. These tables are useful for quickly looking up commands and their general implementation.

The Command column provides an overview of the commands and their hierarchical arrangement.

The Parameter Form column indicates the requested parameters together with their specified range.

The Character Data or Units column indicates the basic unit of the physical parameters or the parameter syntax.

The Notes column indicates:

- If the command has a query form
- If the command is a query only
- If the command is implemented only with a certain instrument option

The different levels of the SCPI command hierarchy are represented in a table by means of indentations to the right. Lower command levels are indented farther to the right. Observe that the complete notation of the command always includes the higher levels as well. For example, **[:SENSe<1|2>]:FREQuency:CENTer** is represented in the table as follows:

```
[:SENSe<1|2>] (first level)

   :FREQuency (second level)

      :CENTer (third level)
```

A GPIB command should be entirely in the LONG or SHORT format, but never a mix of the two.

Command Descriptions

The complete details of the commands are given following the table of commands for each SCPI subsystem. If applicable, an example for each command, the default value, and the SCPI information is written out at the end of the individual description. The modes for which a command can be used are indicated by the following abbreviations:

- SYS: SignatureSystemControl Class
- SPA: Spectrum Analysis and SignatureSpectrum Class
- VSA: Vector Signal Analysis and SignatureModulation Class
- WCDMA: WCDMA Signal Analysis and SignatureWCDMA Class

Note: The spectrum analysis mode is implemented in the basic unit. For the VSA mode, the corresponding options are required.

When Web Services methods can be associated with SCPI commands, those methods are listed as Associated Web Services Methods at the the end of the GPIB command descriptions. If there are no associated Web Services methods, then an association is not mentioned. All Web Services methods are detailed in Chapter 3.

GPIB Common Commands

Common commands are used to control instrument status registers, status reporting, synchronization, data storage, and other common functions. All common commands are identified by the leading asterisk in the command word. The common commands are fully defined in IEEE 488.2.

*IDN? (Identification Query)

This query returns an instrument identification string in IEEE- 488.2 specified <NR1> format (four fields separated by commas). The fields are: <Manufacturer>, <Model Number>, <Serial Number>, <Software Version>.

*OPT? (Option Identification Query)

This command returns a string identifying any device options.

*RST (Reset Command)

Resets the MS2781B to a pre-defined condition with all user programmable parameters set to their default values. These default parameter values are listed under each SCPI command in this manual. This command does not affect the Output Queue, Status Byte Register, Standard Event Register, or calibration data.

Note: This command clears the current front panel setup. If this setup is needed for future testing, save it as a stored setup before issuing the *RST command.

2-2 :CALCulate Subsystem

The **:CALCulate** subsystem contains commands for converting instrument data, transforming and carrying out corrections. These functions are carried out after data acquisition.

Table 2-1. :CALCulate Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:CALCulate<1|2>	-		
:ACP	-		
:ADJacent	-		
:RESult?	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	query only
:ALT<1|2>	-		
:RESult?	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	
:MAIN	-		
:RESult?	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	query only
:CHP	-		
:RESult?	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	query only
:MARKer<1 to 5>	-		
:ACTive?	<boolean>	TRUE|FALSe	
:AOFF	N/A		event only
:FUNCtion	-		
:CENTer	N/A		event only
:TYPE	<char>	NOISe|Off	
:TYPE?	<char>	NOISe|Off	
:MAXimum	-		
:CENTer	N/A		event only
:NEXT	N/A		event only
[:PEAK]	N/A		event only
:MODE	<char>	REL|ABS	
:MODE?	<char>	REL|ABS	
:TRACe	<nv>	1|2|3|4|5	
:TRACe?	<nv>	1|2|3|4|5	
:X	<nv>	HZ|S	
:X?	<nv>	HZ|S	

Table 2-1. :CALCulate Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:Y?	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	query only
[:STATe]	<boolean>	ON|OFF	
:OBW	-		
:POWer	-		
:RESult?	<nv>	HZ|KHZ|MHZ|GHZ	
:XDBS	-		
:RESult?	<nv>	unitless	
:TOI	-		
:RESult?	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	
:UNIT	-		
:POWer	<nv>	DBM|DBMV|DBUV|W|V	
:POWer?	<nv>	DBM|DBMV|DBUV|W|V	
:WCDMa	-		
:MARKer<1|2>	-		
:STATe	<boolean>	ON|OFF	
:STATe?	<boolean>	ON|OFF	
:TYPE?	<char>	NORMAL	

:CALCulate<1|2>:ACP:ADJacent:RESult?

Return Parameters: DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW

Parameter Form: <nv>

Default: N/A

Description: This command returns the power of the adjacent channel. The return power unit is in the currently active amplitude unit except when the ACP measurement is in the Relative mode, then the return unit is in "DB." The upper return value follows the lower return value.
For example: –35.2DBM –47.1DBM

Precondition: The adjacent channel power measurement mode must be active.

Query Form: N/A

Example: `:CALC1:ACP:ADJ:RES?`

Modes: SPA

Associated Web Services Methods: `GetACPRAdjacentChannelResults (SPA)`

:CALCulate<1|2>:ACP:ALT<1|2>:RESult?

Return Parameters: DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW

Parameter Form: <nv>

Default: N/A

Description: This command returns the power value of the specified alternate channel. The return power unit is in the currently active amplitude unit except when the ACP measurement is in the Relative mode, then the return unit is in "DB." The lower return value follows the upper return value. For example: –47.1DBM –35.2DBM

Precondition: The adjacent channel power measurement mode must be active.

Query Form: N/A

Example: `:CALC1:ACP:ALT1:RES?`

Modes: SPA

Associated Web Services Methods: `GetACPRAlternateChannel1Results (SPA)`
`GetACPRAlternateChannel2Results (SPA)`

:CALCulate<1|2>:ACP:MAIN:RESult?

Return Parameters: DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW

Parameter Form: <nv>

Default: N/A

Description: This command returns the main channel power measurement result. The return power unit is in the currently active amplitude unit. For example: –14.0DBM|–10.0MV

Precondition: The adjacent channel power measurement mode must be active.

Query Form: N/A

Example: `:CALC1:ACP:MAIN:RES?`

Modes: SPA

Associated Web Services Methods: `GetACPRMainChannelResults (SPA)`

:CALCulate<1|2>:CHP:RESult?

Return Parameters: DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW

Parameter Form: <nv>

Default: N/A

Description: This command returns the channel power measurement result.

If the measurement is valid, the return value will be string data that represents the powe. For example: **60.8**

If the value is invalid, due to span too small or some other condition, the return value will be four dashes. For example: **----**

Precondition: The channel power measurement mode must be active.

Query Form: N/A

Example: `:CALC1:CHP:RES?`

Modes: SPA

Associated Web Services Methods: `GetChannelPowerResults (SPA)`

:CALCulate<1|2>:MARKer:ACTive?

Return Parameters:	–1|1|2|3|4|5
Parameter Form:	<nv>
Default:	–1
Description:	This command returns the current active marker number. **-1** indicates that no markers are active.
Query Form:	N/A
Example:	`:CALC1:MARK:ACT?`
Modes:	SPA
Associated Web Services Methods:	`GetCurrentMarker (SPA)`

:CALCulate<1|2>:MARKer:AOFF

Parameters:	None. This command is an event, which is why it has no parameters.
Parameter Form:	N/A
Default:	.N/A
Description:	This command switches off the active markers.
Query Form:	None. This command is an event, which is why it has no query.
Example:	`:CALC1:MARK:AOFF`
Modes:	SPA
Associated Web Services Methods:	`SwitchOffAllMarkers (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:FUNCtion:CENTer

Parameters:	None. This command is an event, which is why it has no parameters.
Parameter Form:	N/A
Default:	N/A
Description:	This command sets the center frequency to that of the current marker.
Precondition:	The specified marker must be on.
Query Form:	None. This command is an event, which is why it has no query.
Example:	`:CALC1:MARK:FUNC:CENT`
Modes:	SPA
Associated Web Services Methods:	`SetCenterToMarkerFreq (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:FUNCtion:TYPE

Parameters:	NOISe|OFF
Parameter Form:	<char>
Default:	OFF
Description:	This command turns the indicated noise marker on or off as follows: NOISe: noise marker on Off: noise marker off
Query Form:	`:CALCulate<1|2>:MARKer<1 to 5>:FUNCtion:TYPE?`
Example:	`:CALC1:MARK1:FUNC:TYPE NOIS`
Modes:	SPA
Associated Web Services Methods:	`SetAsNoiseMarker (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:FUNCtion:TYPE?

Return Parameters: NOISe|OFF

Parameter Form: <char>

Default: OFF

Description: This command queries the indicated noise marker status with the following results:
NOISe: noise marker on
Off: noise marker off

Query Form: N/A

Example: `:CALC1:MARK1:FUNC:TYPE?`

Modes: SPA

Associated Web Services Methods: `IsNoiseMarker (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:MAXimum:CENTer

Parameters: None. This command is an event, which is why it has no parameters.

Parameter Form: N/A

Default: N/A

Description: This command sets the indicated marker's peak frequency as the center frequency.

Precondition: The specified marker must be on.

Query Form: None. This command is an event, which is why it has no query.

Example: `:CALC1:MARK1:MAX:CENT`

Modes: SPA

Associated Web Services Methods: `SetPeakToCenter (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:MAXimum:NEXT

Parameters:	None. This command is an event, which is why it has no parameters.
Parameter Form:	N/A
Default:	N/A
Description:	This command positions the marker to the next lower maximum value in the trace memory.
Precondition:	The specified marker must be on.
Query Form:	None. This command is an event, which is why it is has no query.
Example:	`:CALC1:MARK1:MAX:NEXT`
Modes:	SPA
Associated Web Services Methods:	`SetMarkerToNextPeak (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:MAXimum[:PEAK]

Parameters:	None. This command is an event, which is why it has no parameters.
Parameter Form:	N/A
Default:	N/A
Description:	This command positions the marker to the current maximum value in the trace memory.
Precondition:	The specified marker must be on.
Query Form:	None. This command is an event, which is why it has no query.
Example:	`:CALC1:MARK1:MAX`
Modes:	SPA
Associated Web Services Methods:	`SetMarkerToPeak (SPA)`

:CALCulate<1|2>:MARKer<2 to 5>:MODE

Parameters:	REL|ABS
Parameter Form:	<char>
Default:	ABS
Description:	This command sets the indicated marker's type to the following: REL: delta marker ABS: normal marker
Precondition:	Marker 1 must be active and cannot be set as a relative marker.
Query Form:	N/A
Example:	`:CALC1:MARK2:MODE REL`
Modes:	SPA
Associated Web Services Methods:	`SetMarkerMode (SPA)`

:CALCulate<1|2>:MARKer<2 to 5>:MODE?

Return Parameters:	REL|ABS
Parameter Form:	<char>
Default:	ABS
Description:	This command queries the indicated marker's type with the following results: REL: delta marker ABS: normal marker
Query Form:	N/A
Example:	`:CALC1:MARK1:MODE?`
Modes:	SPA
Associated Web Services Methods:	`GetMarkerMode (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:TRACe

Parameters:	1|2|3|4|5
Parameter Form:	<nv>
Default:	1
Description:	This command assigns the indicated marker to the indicated trace.
Precondition:	The specified marker and trace must be on.
Query Form:	`:CALCulate<1|2>:MARKer<1 to 5>:TRACe?`
Example:	`:CALC1:MARK1:TRAC 1`
Modes:	SPA
Associated Web Services Methods:	`SetMarkerToTrace (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:TRACe?

Return Parameters:	1|2|3|4|5
Parameter Form:	<nv>
Default:	1
Description:	This command returns the trace number that is associated with the indicated marker.
Precondition:	The specified marker and trace must be on.
Query Form:	N/A
Example:	`:CALC1:MARK1:TRAC?`
Modes:	SPA
Associated Web Services Methods:	`GetTraceAttachedToMarker (SPA)`

`:CALCulate<1|2>:MARKer<1 to 5>:X`

Parameters:	0 to MAX (frequency or sweep time)
Parameter Form:	<nv>
Default:	None. This command is an event, which is why it does not have a default value.
Description:	This command positions the selected marker to the indicated frequency (span > 0) or time (span = 0). The sweep time unit is used when in zero span; otherwise, the frequency unit is used. GPIB uses the same command for zero span and frequency domain. The input value is time when in zero span; otherwise, a frequency value is used.
Precondition:	The specified marker must be on.
Query Form:	`:CALCulate<1|2>:MARKer<1 to 5>:X?`
Example:	`:CALC1:MARK1:X 10.7MHZ`
Modes:	SPA
Associated Web Services Methods:	`SetFrequencyMarkerPosition (SPA)`

`:CALCulate<1|2>:MARKer<1 to 5>:X?`

Return Parameters:	0 to MAX (frequency or sweep time)
Parameter Form:	<nv>
Default:	None. This command is a query, which is why it does not have a default value.
Description:	This command returns the X value of the marker. If the sweep mode is zero span, the unit is time; otherwise, the unit is frequency.
Precondition:	The specified marker must be on.
Query Form:	N/A
Example:	`:CALC1:MARK1:X?`
Modes:	SPA
Associated Web Services Methods:	`GetFrequencyMarkerPositionInHz (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>:Y?

Return Parameters: +300DBM to –300DBM
The return units are based on the the instrument's amplitude units setting and the following units are supported:
DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW

Parameter Form: <nv>

Default: None. This command is a query, which is why it does not have a default value.

Description: This command queries the selected marker amplitude value.

Precondition: The specified marker must be on.

Query Form: N/A

Example: `:CALC1:MARK1:Y?`

Modes: SPA

Associated Web Services Methods: `GetMarkerAmplitude (SPA)`

:CALCulate<1|2>:MARKer<1 to 5>[:STATe]

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command switches the indicated marker on or off. If no marker indication is made, marker 1 is selected automatically.

Query Form: None

Example: `:CALC1:MARK1 ON`

Modes: SPA

Associated Web Services Methods: `None`

:CALCulate<1|2>:OBW:POWer:RESult?

Parameters:	None. This command is a query, which is why it has no input parameters.
Parameter Form:	<nv>
Default:	None. This command is a query, which is why it has no default value.
Description:	This command returns the occupied bandwidth measurement results with the frequency units based on the OBW percentage setting.
Precondition:	The occupied bandwidth measurement mode must be active.
Query Form:	N/A
Example:	`:CALC1:OBW:POW:RES?`
Modes:	SPA
Associated Web Services Methods:	`GetOccupiedBWResultsInHz (SPA)`

:CALCulate<1|2>:OBW:XDBS:RESult?

Parameters:	None. This command is a query, which is why it has no input parameters.
Parameter Form:	<nv>
Default:	None. This command is a query, which is why it has no default value.
Description:	This command returns the occupied bandwidth measurement results with the frequency units based on the XdB setting.
Precondition:	The occupied bandwidth measurement mode must be active.
Query Form:	N/A
Example:	`:CALC1:OBW:XDBS:RES?`
Modes:	SPA
Associated Web Services Methods:	`GetOccupiedBWResultsInHz (SPA)`

:CALCulate<1|2>:TOI:RESult?

Parameters:	None. This command is a query, which is why it has no input parameters.
Parameter Form:	N/A
Default:	N/A
Description:	This command returns the third order intercept measurement result.
Precondition:	The third order intercept measurement mode must be active.
Query Form:	N/A
Example:	`:CALC1:TOI:RES?`
Modes:	SPA
Associated Web Services Methods:	`GetTOIResults (SPA)`

:CALCulate<1|2>:UNIT:POWer

Parameters:	DBM|DBMV|DBUV|W|V
Parameter Form:	<char>
Default:	DBM
Description:	This command selects the unit for input level units. Possible values are dBm, dBmV, dBµV, Watt, or Volt.
Query Form:	`:CALCulate<1|2>:UNIT:POWer?`
Example:	`:CALC1:UNIT:POW DBM`
Modes:	SPA, VSA
Associated Web Services Methods:	`SetAmplitudeUnits (SPA)` `SetAmplitudeUnits (VSA)`

:CALCulate<1|2>:UNIT:POWer?

Return Parameters: DBM|DBMV|DBUV|W|V

Parameter Form: <char>

Default: DBM

Description: This command queries the input power units setting. Possible values are dBm, dBmV, dBµV, Watt, or Volt.

Query Form: N/A

Example: `:CALC1:UNIT:POW?`

Modes: SPA, VSA

Associated Web Services Methods: `GetAmplitudeUnits (SPA)` `GetAmplitudeUnits (VSA)`

:CALCulate<1|2>:WCDMa:MARKer<1|2>:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command switched the specified marker on or off.

Query Form: `:CALCulate<1|2>:WCDMa:MARKer<1|2>:STATe?`

Example: `:CALC1:WCDM:MARK1:STAT ON`

Modes: WCDMA

Associated Web Services Methods: `SetMarkerMode (WCDMA)`

:CALCulate<1|2>:WCDMa:MARKer<1|2>:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the specified marker state.

Query Form: N/A

Example: `:CALC1:WCDM:MARK1:STAT?`

Modes: WCDMA

Associated Web Services Methods: `GetMarkerMode (WCDMA)`

:CALCulate<1|2>:WCDMa:MARKer<1|2>:TYPE?

Return Parameters: NORMAL

Parameter Form: <char>

Default: NORMAL

Description: This command queries the specified marker type. Delta markers are not currently supported.

Query Form: N/A

Example: `:CALC1:WCDM:MARK1:TYPE?`

Modes: WCDMA

Associated Web Services Methods: `None`

2-3 :DIAGnostic Subsystem

The **:DIAGnostic** subsystem contains commands that support instrument diagnostics for maintenance, service, and repair. In accordance with the SCPI standard, all of these commands are device specific.

Table 2-2. :DIAGnostic Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:DIAGnostic	-		
:SERVice	-		
:INPut	-		
[:SELect]	<char>	CALibration|RF	
:NSOurce	<boolean>	ON|OFF	

:DIAGnostic:SERVice:INPut[:SELect]

Parameters: CALibration|RF

Parameter Form: <char>

Default: RF

Description: This command toggles between the RF input on the front panel and the internal 50 MHz reference signal.

Query Form: None

Example: `:DIAG:SERV:INP CAL`

Modes: SYS

Associated Web Services Methods: `SwitchOnCalibratorSignal (SYS)`

`:DIAGnostic:SERVice:NSOurce`

Parameters: ON|OFF

Parameter Form: <boolean>

Default: On

Description: This command switches the 28V noise source supply at the rear connector on and off.

Query Form: None

Example: `:DIAG:SERV:NSO ON`

Modes: SYS

Associated Web Services Methods: None

2-4 :DISPlay Subsystem

The **:DISPlay** subsystem contains commands for controlling the display of textual and graphical information, as well as trace data, on the screen.

Table 2-3. :DISPlay Subsystem Commands

<table>
<tr><th>Command</th><th>Parameter Form</th><th>Character Data or Units</th><th>Notes</th></tr>
<tr><td>:DISPlay</td><td>-</td><td></td><td></td></tr>
<tr><td>[:WINDow<1|2>]</td><td>-</td><td></td><td></td></tr>
<tr><td>:TRACe<1 to 5></td><td>-</td><td></td><td></td></tr>
<tr><td>:MODE</td><td><char></td><td>CWRite|AVERage|MAXHold|MINHold|VIEW|OFF</td><td></td></tr>
<tr><td>:MODE?</td><td><char></td><td>CWRITE|AVERAGE|MAXHOLD|MINHOLD|VIEW| OFF</td><td></td></tr>
<tr><td>:Y</td><td>-</td><td></td><td></td></tr>
<tr><td>:SPACing</td><td><char></td><td>LINear|LOGarithmic</td><td></td></tr>
<tr><td>:SPACing?</td><td><char></td><td>LINear|LOGarithmic</td><td></td></tr>
<tr><td>[:SCALe]</td><td>-</td><td></td><td></td></tr>
<tr><td>:PDIVision</td><td><nv></td><td>DB</td><td></td></tr>
<tr><td>:PDIVision?</td><td><nv></td><td>DB</td><td></td></tr>
<tr><td>:RLEVel</td><td><nv></td><td>DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW</td><td></td></tr>
<tr><td>:OFFSet</td><td><nv></td><td>DB</td><td></td></tr>
<tr><td>:OFFSet?</td><td><nv></td><td>DB</td><td></td></tr>
<tr><td>:RLEVel?</td><td><nv></td><td>DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW</td><td>query only</td></tr>
</table>

:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:MODE

Parameters: CWRite|AVERage|MAXHold|MINHold|VIEW|OFF

Parameter Form: <char>

Default: CWRite for TRACe1, OFF for TRACe2 to TRACe5

Description: This command defines the type of display and the evaluation of the traces. There is no “ON” parameter.

Query Form: `:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:MODE?`

Example: `:DISP:TRAC1:MODE MAXH`

Modes: SPA

Associated Web Services Methods: `SetTraceMode (SPA)`

:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:MODE?

Return Parameters: CWRITE|AVERAGE|MAXHOLD|MINHOLD|VIEW|OFF

Parameter Form: <char>

Default: CWRite for TRACe1, OFF for TRACe2 to TRACe5

Description: This command returns the type of display.

Query Form: N/A

Example: `:DISP:TRAC1:MODE?`

Modes: SPA

Associated Web Services Methods: `GetTraceMode (SPA)`

:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y:SPACing

Parameters:	LINear|LOGarithmic
Parameter Form:	<char>
Default:	LOGarithmic
Description:	This command sets the vertical spacing mode to linear or logarithmic.
Query Form:	`:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y:SPACing?`
Example:	`:DISP:TRAC1:Y:SPAC LIN`
Modes:	SPA
Associated Web Services Methods:	`SetScaleTypeLinear (SPA)`

:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y:SPACing?

Return Parameters:	LINear|LOGarithmic
Parameter Form:	<char>
Default:	LOGarithmic
Description:	This command queries the vertical scale setting.
Query Form:	N/A
Example:	`:DISP:TRAC1:Y:SPAC?`
Modes:	SPA
Associated Web Services Methods:	`GetScaleTypeLinear (SPA)`

:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:PDIVision

Parameters: 0.1 to 20

Parameter Form: <nv>

Default: 10

Description: This command defines the scaling of the Y-axis in the current unit.

Resolution:
0.1 dB for 0.1 dB to 1 dB range
1 dB for 1 dB to 20 dB range

The numeric suffix in TRACe<1 to 5> is not significant.

Query Form: `:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:PDIVision?`

Example: `:DISP:TRAC1:Y:PDIV 1`

Modes: SPA

Associated Web Services Methods: `SetScalingPerDivision (SPA)`

:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:PDIVision?

Return Parameters: 0.1 to 20

Parameter Form: <nv>

Default: 10

Description: This command defines the scaling of the Y-axis in the dB unit.

Resolution:
0.1 dB for 0.1 dB to 1 dB range
1 dB for 1 dB to 20 dB range

The numeric suffix in TRACe<1 to 5> is not significant.

Query Form: N/A

Example: `:DISP:TRAC1:Y:PDIV?`

Modes: SPA

Associated Web Services Methods: `GetScalingPerDivision (SPA)`

:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel

Parameters: –150DBM to 30DBM
Additionally supported units:
DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW

Parameter Form: <nv>

Default: -0DBM

Description: This command sets the reference level with a resolution of 0.01 dB.
The numeric suffix in TRACe<1 to 5> is not significant.

Query Form: `:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel?`

Example: `:DISP:TRAC1:Y:RLEV 0DBM`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`SetReferenceLevel (SPA)`
`SetReferenceLevel (VSA)`
`SetReferenceLevel (WCDMA)`

:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel?

Return Parameters: –150DBM to 30DBM
The return units are based on the the instrument’s amplitude units setting and the following units are supported:
DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW

Parameter Form: <nv>

Default: 0DBM

Description: This command returns the reference level value.
The numeric suffix in TRACe<1 to 5> is not significant.

Query Form: N/A

Example: `:DISP:TRAC1:Y:RLEV?`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`GetReferenceLevel (SPA)`
`GetReferenceLevel (VSA)`
`GetReferenceLevel (WCDMA)`

`:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel:OFFSet`

Parameters: –300DB to 300DB

Parameter Form: <nv>

Default: 0DB

Description: This command sets the reference level offset with a resolution of 0.01 dB. The numeric suffix in TRACe<1 to 5> is not significant.

Query Form: `:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel:OFFSet?`

Example: `:DISP:TRAC1:Y:RLEV:OFFS -10DB`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`SetReferenceLevelOffset (SPA)`
`SetReferenceLevelOffset (VSA)`
`SetReferenceLevelOffset (WCDMA)`

`:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel:OFFSet?`

Return Parameters: –300DB to 300DB

Parameter Form: <nv>

Default: 0DB

Description: This command returns the reference level offset value. The numeric suffix in TRACe<1 to 5> is not significant.

Query Form: N/A

Example: `:DISP:TRAC1:Y:RLEV:OFFS?`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`GetReferenceLevelOffsetIndB (SPA)`
`GetReferenceLevelOffsetIndB (VSA)`
`GetReferenceLevelOffsetIndB (WCDMA)`

2-5 :HCOPy Subsystem

The **:HCOPy** subsystem contains commands for exporting display data to the printer.

Table 2-4. :HCOPy Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:HCOPy	-		
[:IMMediate]	N/A		event only

:HCOPy[:IMMediate]

Parameters: None. This command is an event, which is why it has no parameters.

Parameter Form: N/A

Default: N/A

Description: This command sends the current display data to the printer.

Query Form: N/A

Example: `:HCOP`

Modes: SYS

Associated Web Services Methods: None

2-6 :INITiate<1|2> Subsystem

The **`:INITiate<1|2>`** subsystem contains commands for the initialization of the trigger subsystem.

Table 2-5. :INITiate<1|2> Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
`:INITiate<1|2>`	-		
`:CONTinuous`	<boolean>	ON|OFF	
`:CONTinuous?`	<boolean>	ON|OFF	
`:SWEep`			
`AVERage?`	<boolean>	1|0	
`:SWEep?`	<boolean>	1|0	query only
`[:IMMediate]`	N/A		event only

`:INITiate<1|2>:CONTinuous`

Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command sets the trigger system. Setting **`INITiate:CONTinuous ON`** corresponds to a continuous sweep (Free Run). For example, the sweep of the analyzer is cyclically repeated. Setting **`INITiate:CONTinuous OFF`** corresponds to a single sweep.

Query Form: `:INITiate<1|2>:CONTinuous?`

Example: `:INIT1:CONT OFF`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`SetSweepMode (SPA)`
`SetSweepMode (VSA)`
`SetSweepMode (WCDMA)`

:INITiate<1|2>:CONTinuous?

Return Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	ON
Description:	This command determines the trigger system setting of continuous (Free Run, ON) or single sweep (OFF).
Query Form:	N/A
Example:	`:INIT1:CONT?`
Modes:	SPA, VSA, WCDMA

Associated Web Services Methods:	`GetSweepMode (SPA)`
	`GetSweepMode (VSA)`
	`GetSweepMode (WCDMA)`

:INITiate<1|2>:SWEep:AVERage?

Return Parameters:	1|0
Parameter Form:	<boolean>
Default:	0
Description:	Outputs the trace averaging status. 1 indicates that the trace averaging is complete; 0 indicates that the trace averaging is not complete. This command should be issued after the sweep start.
Query Form:	N/A
Example:	`:INIT1:SWE:AVER?`
Modes:	SPA
Associated Web Services Methods:	`IsTraceAveragingComplete (SPA)`

:INITiate<1|2>:SWEep?

Return Parameters: 1|0

Parameter Form: <boolean>

Default: 0

Description: Outputs the sweep status. 1 indicates that the sweep is complete; 0 indicates that the sweep is not complete.

Query Form: N/A

Example: `:INIT1:SWE?`

Modes: SYS, VSA

Associated Web Services Methods: `IsSweepComplete (VSA)`

:INITiate<1|2>[:IMMediate]

Parameters: None. This command is an event, which is why it has no parameters.

Parameter Form: N/A

Default: N/A

Description: This command initiates a sweep when in the single sweep mode.

Query Form: None

Example: `:INIT1`

Modes: SYS, SPA, VSA, WCDMA

Associated Web Services Methods:
```
StartSweep (SYS)
StartSweep (SPA)
StartSweep (VSA)
StartSweep (WCDMA)
```

2-7 :INPut<1|2> Subsystem

The **`:INPut<1|2>`** subsystem contains commands for setting the input port parameters.

Table 2-6. :INPut<1|2> Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
`:INPut<1|2>`	-		
`:ATTenuation`	<nv>	DB	
`:AUTO`	<boolean>	ON|OFF	no query
`:ATTenuation?`	<nv>	DB	
`:MIXer`	-		
`[:POWer]`	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	
`[:POWer]?`	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	
`:MODE`	<char>	IQDL|IQDH|IQSL|IQSH|RFWIDE|RFNARROW	VSA only
`:MODE?`	<char>	IQDL|IQDH|IQSL|IQSH|RFWIDE|RFNARROW	VSA only

`:INPut<1|2>:ATTenuation`

Parameters: 0DB to 62DB

Parameter Form: <nv>

Default: AUTO is set to ON.

Description: This command sets the input attenuator. The attenuation of the input calibration line can be programmed in steps of 2 dB. If the attenuation is programmed directly, the coupling to the reference level is switched off.

Query Form: `:INPut<1|2>:ATTenuation?`

Example: `:INP1:ATT 40DB`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`SetAttenuation (SPA)`
`SetAttenuation (VSA)`
`SetAttenuation (WCDMA)`

:INPut<1|2>:ATTenuation:AUTO

Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command automatically couples the input attenuation to the reference level.

Query Form: N/A

Example: `:INP1:ATT:AUTO ON`

Modes: SPA, WCDMA

Associated Web Services Methods:
`SetAttenuationModeAuto (SPA)`
`SetAttenuationModeAuto (WCDMA)`

:INPut<1|2>:ATTenuation?

Return Parameters: 0DB to 62DB

Parameter Form: <nv>

Default: AUTO is set to ON.

Description: This command queries the input attenuation.

Query Form: N/A

Example: `:INP1:ATT?`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`GetAttenuationIndB (SPA)`
`GetAttenuationModeAuto (SPA)`
`GetAttenuationIndB (VSA)`
`GetAttenuationIndB (WCDMA)`
`GetAttenuationModeAuto (WCDMA)`

:INPut<1|2>:MIXer[:POWer]

Parameters:	0DBM to –50DBM The following units are also supported with the appropriate conversion: DBMV|DBUV|MV|UV|NV|PV|W|MW|UW|NW|PW|FW|AW|ZW|YW
Parameter Form:	<nv>
Default:	–10DBM
Description:	This command sets the internal mixer level.
Precondition:	The parameter terminator must use the current active amplitude unit.
Query Form:	`:INPut<1|2>:MIXer[:POWer]?`
Example:	`:INP1:MIX -30DBM`
Modes:	SPA
Associated Web Services Methods:	`SetMixerLevel (SPA)`

:INPut<1|2>:MIXer[:POWer]?

Return Parameters:	0DBM to –50DBM The following units are also supported with the appropriate conversion: DBMV|DBUV|MV|UV|NV|PV|W|MW|UW|NW|PW|FW|AW|ZW|YW
Parameter Form:	<nv>
Default:	–10DBM
Description:	This command queries the internal mixer level.
Query Form:	N/A
Example:	`:INP1:MIX?`
Modes:	SPA, VSA
Associated Web Services Methods:	`GetMixerLevel (SPA)` `GetMixerLevel (VSA)`

`:INPut<1|2>:MODE`

Parameters: IQDL|IQDH|IQSL|IQSH|RFWIDE|RFNARROW

Parameter Form: <char>

Default: RFWIDE

Description: This command sets the input signal type as follows:
IQDL: IQ Differential Low
IQDH: IQ Differential High
IQSL: IQ Single Low
IQSH: IQ Single High
RFWIDE: Wideband RF
RFNARROW: Narrowband RF

Precondition: The modulation measurement mode must be activated via GPIB.

Query Form: `:INPut<1|2>:MODE?`

Example: `:INP1:MODE RFWIDE`

Modes: VSA

Associated Web Services Methods: `SetInputSignal (VSA)`

:INPut<1|2>:MODE?

Return Parameters: IQDL|IQDH|IQSL|IQSH|RFWIDE|RFNARROW

Parameter Form: <char>

Default: RFWIDE

Description: This command queries the input signal types with the following return values:
IQDL: IQ Differential Low
IQDH: IQ Differential High
IQSL: IQ Single Low
IQSH: IQ Single High
RFWIDE: Wideband RF
RFNARROW: Narrowband RF

Precondition: The modulation measurement mode must be activated via GPIB.

Query Form: N/A

Example: `:INP1:MODE?`

Modes: VSA

Associated Web Services Methods: `GetInputSignal (VSA)`

2-8 :INSTrument<1|2> Subsystem

The **:INSTrument<1|2>** subsystem contains commands for selecting the instrument operating mode.

Table 2-7. :INSTrument<1|2> Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:INSTrument<1|2>			
:NSELect	<nv>	1|2|3|4|5|6|7|9	
:NSELect?	<nv>	1|2|3|4|5|6|7|9	
:SELect	<char>	SANalyzer|DDEMod|OBW|ACP|CHP|TOI|MCCP|WCDMa	
:SELect?	<char>	SAN|DDEM|OBW|ACP|CHP|TOI|MCCP|WCDMA	

:INSTrument<1|2>:NSELect

Parameters: 1|2|3|4|5|6|7|9

Parameter Form: <nv>

Default: 1

Description: This command switches between modes by means of numbers as follows:

1: spectrum analysis
2: vector signal analysis, digital demodulation
3: occupied bandwidth
4: adjacent channel power
5: channel power
6: third order intercept
7: multicarrier channel power
9: wideband code division multiple access

Switching to 2 or 9 is only possible in conjunction with the Vector Signal Analysis or WCDMA options, respectively.

Query Form: `:INST1:NSEL?`

Example: `:INST1:NSEL 2`

Modes: SYS

Associated Web Services Methods: `SetActiveMeasurement (SYS)`

:INSTrument<1|2>:NSELect?

Return Parameters: 1|2|3|4|5|6|7|9

Parameter Form: <nv>

Default: 1

Description: This command queries the currently set operating mode with the following numerical return values:

1: spectrum analysis
2: vector signal analysis, digital demodulation
3: occupied bandwidth
4: adjacent channel power
5: channel power
6: third order intercept
7: multicarrier channel power
9: wideband code division multiple access

A return of 2 or 9 is only possible in conjunction with the Vector Signal Analysis or WCDMA options, respectively.

Query Form: N/A

Example: `:INST1:NSEL?`

Modes: SYS

Associated Web Services Methods: None

:INSTrument<1|2>:SELect

Parameters:	SANalyzer|DDEMod|OBW|ACP| CHP|TOI|MCCP|WCDMa
Parameter Form:	<char>
Default:	SANalyzer
Description:	This command switches between operating modes by means of text parameters as follows: SANalyzer: spectrum analysis DDEMod: vector signal analysis, digital demodulation OBW: occupied bandwidth ACP: adjacent channel power CHP: channel power TOI: third order intercept MCCP: multicarrier channel power WCDMa: wideband code division multiple access Switching to DDEMod or WCDMa is only possible in conjunction with the Vector Signal Analysis or WCDMA options, respectively.
Query Form:	`:INST1:SEL?`
Example:	`:INST1:SEL DDEM`
Modes:	SYS
Associated Web Services Methods:	`SetActiveMeasurement (SYS)`

:INSTrument<1|2>:SELect?

Return Parameters: SAN|DDEM|OBW|ACP|CHP|TOI|MCCP|WCDMA

Parameter Form: <char>

Default: SAN

Description: This command queries the currently set operating mode with the following return strings:

SAN: spectrum analysis
DDEM: vector signal analysis, digital demodulation
OBW: occupied bandwidth
ACP: adjacent channel power
CHP: channel power
TOI: third order intercept
MCCP: Multicarrier Channel Power
WCDMA: Wideband Code Division Multiple Access

A return of DDEM or WCDMA is only possible in conjunction with the Vector Signal Analysis or WCDMA options, respectively.

Query Form: N/A

Example: `:INST1:SEL?`

Modes: SYS

Associated Web Services Methods: None

2-9 :STATus Subsystem

The :STATus subsystem contains commands for general system control and queries.

Table 2-8. :STATus Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:STATus	-		
:QUEStionable	-		
:POWer	-		
:CONDition?	<char>	comma delimited character string	query only

:STATus:QUEStionable:POWer:CONDition?

Return Parameters: Character string output.

Parameter Form: <char>

Default: None. This command is a query, which is why it does not have a default value.

Description: Returns the system status in a comma delimited string as follows: <NarrowBandIfOverLoaded = YES|NO, WideBandIFOverloaded = YES|NO, LoUnlocked = YES|NO, RFOverloaded = YES|NO>

Query Form: N/A

Example: `:STAT:QUES:POW:COND?`

Modes: SYS

Associated Web Services Methods: `GetInstrumentState (SYS)`

2-10 :SYSTem Subsystem

The **:SYSTem** subsystem contains commands for general system control and queries.

Table 2-9. :SYSTem Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:SYSTem	-		
:ERRor	-		
:CLEar	-		
:ALL	N/A		event only
:LIST?	<char>	character string output	query only
:ERRor?	<char>	character string output	query only
:FILTer	-		
:AALias	<boolean>	ON|OFF	VSA only
:AALias?	<boolean>	ON|OFF	VSA only
:NOISeoptimize	-		
:AUTO	<boolean>	ON|OFF	
:AUTO?	<boolean>	ON|OFF	
:OFFSet	<char>	CLOSe|FAR	
:OFFSet?	<char>	CLOSe|FAR	
:PRESet	N/A		event only
:RTL	N/A		event only
:STANdard	<char>	WCDMA|GENeric	VSA only
:STANdard?	<char>	WCDMA|GENeric	VSA only
:VERSion?	<char>	character string output	query only

:SYSTem:ERRor:CLEar:ALL

Parameters: None. This command is an event, which is why it has no parameters.

Parameter Form: N/A

Default: N/A

Description: This command deletes all entries in the table SYSTEM MESSAGES.

Query Form: `:SYSTem:ERRor:LIST?`
`:SYSTem:ERRor?`

Example: `:SYST:ERR:CLE:ALL`

Modes: SYS

Associated Web Services Methods: `ClearSignatureErrorLog (SYS)`

:SYSTem:ERRor:LIST?

Return Parameters: Character string output.

Parameter Form: <char>

Default: N/A

Description: This command reads all system messages and returns a list of comma separated strings. Each string corresponds to an entry in the table SYSTEM MESSAGES. If the error list is empty, an empty string "" is returned. This command is a query; therefore, it has no *RST value. Example return string:
<UI -Error while bringing GPIB service: The server threw an exception, 3/22/2005 10:54:03 AM>

Query Form: N/A

Example: `:SYST:ERR:LIST?`

Modes: SYS

Associated Web Services Methods: `GetSignatureErrorLog (SYS)`

:SYSTem:ERRor?

Return Parameters: Character string output.

Parameter Form: <char>

Default: N/A

Description: This command queries the most recent entry in the error queue. The command returns the error as follows:
<UI -Error while bringing GPIB service: The server threw an exception, 3/22/2005 10:54:03 AM>

Query Form: N/A

Example: `:SYST:ERR?`

Modes: SYS

Associated Web Services Methods: `GetLastError (SYS)`

`:SYSTem:FILTer:AALias`

Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	This command sets the anti-aliasing filter on or off.
Precondition:	The modulation measurement mode must be activated via GPIB.
Query Form:	`:SYSTem:FILTer:AALias?`
Example:	`:SYST:FILT:AAL ON`
Modes:	SYS
Associated Web Services Methods:	`ToggleAntiAliasingFilterState (SYS)`

`:SYSTem:FILTer:AALias?`

Return Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	This command queries the anti-aliasing filter status.
Precondition:	The modulation measurement mode must be activated via GPIB.
Query Form:	N/A
Example:	`:SYST:FILT:AAL?`
Modes:	SYS
Associated Web Services Methods:	`GetAntiAliasingFilterState (SYS)`

:SYSTem:NOISeoptimize:AUTO

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command sets the Auto Optomize Phase Noise mode setting On or Off.

Query Form: `:SYSTem:NOISeoptimize:AUTO?`

Example: `SYST:NOIS:AUTO ON`

Modes: SYS

Associated Web Services Methods: `SetPhaseNoiseOptimizationMode (SYS)`

:SYSTem:NOISeoptimize:AUTO?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command queries the Auto Optomize Phase Noise mode setting of On or Off.

Query Form: N/A

Example: `SYST:NOIS:AUTO?`

Modes: SYS

Associated Web Services Methods: `GetPhaseNoiseOptimizationMode (SYS)`

:SYSTem:NOISeoptimize:OFFSet

Return Parameters: CLOSe | FAR

Parameter Form: <char>

Default: FAR

Description: This command sets the phase noise measurement optimization offset from the carrier to either CLOSe (less than 85 kHz) or FAR (greater than 85 kHz).

Precondition: If the Phase Noise Optimization is set to AUTO, then FAR is returned.

Query Form: `:SYSTem:NOISeoptimize:OFFSet?`

Example: `SYST:NOIS:OFFS CLOS`

Modes: SYS

Associated Web Services Methods: None

:SYSTem:NOISeoptimize:OFFSet?

Return Parameters: CLOSe | FAR

Parameter Form: <char>

Default: FAR

Description: This command queries the phase noise measurement optimization offset from the carrier as either CLOSe (less than 85 kHz) or FAR (greater than 85 kHz).

Precondition: If the Phase Noise Optimization is set to AUTO, then FAR is returned.

Query Form: :N/A

Example: `SYST:NOIS:OFFS?`

Modes: SYS

Associated Web Services Methods: `GetPhaseNoiseOptimizationOffset (SYS)`

:SYSTem:PRESet

Parameters: None. This command is an event, which is why it has no parameters.

Parameter Form: N/A

Default: N/A

Description: This command triggers an instrument reset. The affect of this command corresponds to that of the PRESET key with manual control or to the *RST command.

Query Form: None

Example: `:SYST:PRES`

Modes: SYS

Associated Web Services Methods: `Preset (SYS)`

:SYSTem:RTL

Parameters: None. This command is an event, which is why it has no parameters.

Parameter Form: N/A

Default: N/A

Description: This command brings the instrument out of Remote mode. The front panel and touch screen operations are active after sending this command. This command has the same effect as pressing the System menu's Local button.

Query Form: None

Example: `:SYST:RTL`

Modes: SYS

Associated Web Services Methods: `SetInstrumentMode (SYS)`

:SYSTem:STANdard

Parameters:	WCDMa|GENEric
Parameter Form:	<char>
Default:	GENEric
Description:	This command sets the measurement standard.
Precondition:	The modulation measurement mode must be activated via GPIB.
Query Form:	N/A
Example:	`:SYST:STAN WCDM`
Modes:	SYS
Associated Web Services Methods:	`SetStandardType (SYS)`

:SYSTem:STANdard?

Return Parameters:	WCDMA|GENERIC
Parameter Form:	<char>
Default:	GENERIC
Description:	This command queries the measurement standard.
Precondition:	The modulation measurement mode must be activated via GPIB.
Query Form:	N/A
Example:	`:SYST:STAN?`
Modes:	SYS
Associated Web Services Methods:	`GetStandardType (SYS)`

:SYSTem:VERSion?

Return Parameters: Character string output.

Parameter Form: <char>

Default: None. This command is a query, which is why it does not have a default value.

Description: This command queries the number of the SCPI version; such as, V1.999.

Query Form: N/A

Example: `:SYST:VERS?`

Modes: SYS

Associated Web Services Methods: `GetSoftwareVersionNumber (SYS)`

2-11 :TRACe Subsystem

The **:TRACe** subsystem contains commands for accessing the instruments trace data.

Table 2-10. :TRACe Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:TRACe	-		
:DDEMod	-		
:DATA			
:BITStream	<nv> <nv>	<start symbol> <number of symbols>	VSA only
:EVMTime	<nv> <nv>	<start symbol> <number of symbols>	VSA only
:IQVector	<nv> <nv>	<start symbol> <number of symbols>	VSA only
:POWertime	<nv> <nv>	<start symbol> <number of symbols>	VSA only
:TRACe<1 to 5>?	<nr2>		query only

:TRACe:DDEMod:DATA:BITStream?

Return Parameters: <start symbol> <number of symbols>

Parameter Form: <nv> <nv>

Default: None. This command is a query, which is why it does not have a default value.

Description: This command queries the digital demodulation bitstream data of the indicated trace. The data is output in IEEE 488.2 arbitrary block format with the first character in the header as a "#" followed by a nonzero digit indicating the number of digits to follow. The digits that follow represent the amount of data bytes in the arbitrary block data.
For example, one data block would be as follows:
#216<b><b><b><b><b><b><b><b><b><b><b><b><b><b><b><b>
where
"#" is the start of the header.
"2" specifies that the next two digits form the block length.
"16" specifies that the block length is 16 bytes and is also the end of the header.
"<b>" represents a single byte of data in the data block and all 16 bytes represent the entire block of data.

The start symbol and number of symbols range from 1 to the total number of symbols where the total number of symbols is constrained to: 625 ≥ (Symbol Rate x Capture Time ≥ 1.

Precondition: The modulation measurement mode must be activated via GPIB.

Query Form: N/A

Example:

```
:INST:SEL DDEM
:TRAC:DDEM:DATA:BITS? 1 256
```

Modes: VSA

Associated Web Services Methods:

```
GetModulationBitStreamSize (VSA)
GetModulationBitStream (VSA)
```

:TRACe:DDEMod:DATA:EVMT?

Return Parameters: <start symbol> <number of symbols>

Parameter Form: <nv> <nv>

Default: None. This command is a query, which is why it does not have a default value.

Description: This command queries the digital demodulation error vector magnitude time data of the indicated trace. The data is output in IEEE 488.2 arbitrary block format with the first character in the header as a "#" followed by a non-zero digit indicating the number of digits to follow. The digits that follow represent the amount of data bytes in the arbitrary block data.
For example, one data block would be as follows:
#216<b><b><b><b><b><b><b><b><b><b><b><b><b><b><b><b>
where
"#" is the start of the header.
"2" specifies that the next two digits form the block length.
"16" specifies that the block length is 16 bytes and is also the end of the header.
"<b>" represents a single byte of data in the data block and all 16 bytes represent the entire block of data.

The start symbol and number of symbols range from 1 to the total number of symbols where the total number of symbols is constrained to: 625 ≥ (Symbol Rate x Capture Time ≥ 1.

Precondition: The modulation measurement mode must be activated via GPIB.

Query Form: N/A

Example:

```
:INST:SEL DDEM
:TRAC:DDEM:DATA:EVMT? 1 256
```

Modes: VSA

Associated Web Services Methods:

```
GetModEvmTimeDataSize (VSA)
GetModEvmTimeData (VSA)
```

:TRACe:DDEMod:DATA:IQV?

Return Parameters: <start symbol> <number of symbols>

Parameter Form: <nv> <nv>

Default: None. This command is a query, which is why it does not have a default value.

Description: This command queries the digital demodulation IQ vector data of the indicated trace. The data is output in IEEE 488.2 arbitrary block format with the first character in the header as a "#" followed by a nonzero digit indicating the number of digits to follow. The digits that follow represent the amount of data bytes in the arbitrary block data. For example, one data block would be as follows:
#216<b><b><b><b><b><b><b><b><b><b><b><b><b><b><b><b>
where
"#" is the start of the header.
"2" specifies that the next two digits form the block length.
"16" specifies that the block length is 16 bytes and is also the end of the header.
"<b>" represents a single byte of data in the data block and all 16 bytes represent the entire block of data.

The data block is I/Q interleaved with four data bytes representing each I or Q point. In this example:

$$\underbrace{\text{<b><b><b><b>}}_{I}\ \underbrace{\text{<b><b><b><b>}}_{Q}\ \underbrace{\text{<b><b><b><b>}}_{I}\ \underbrace{\text{<b><b><b><b>}}_{Q}$$

The I and Q values are Floating point values in constellation units, which is the same as the marker readout. The output data for each symbol consists of 16 points of data, 8 "I" points and 8 "Q" points. The first pair of data points represent the symbol and the other 14 data points (7 data point pairs) are interpolated data points.

The start symbol and number of symbols range from 1 to the total number of symbols where the total number of symbols is constrained to: 625 ≥ (Symbol Rate x Capture Time ≥ 1.

Precondition: The modulation measurement mode must be activated via GPIB.

Query Form: N/A

Example:

```
:INST:SEL DDEM
:TRAC:DDEM:DATA:IQV? 1 256
```

Modes: VSA

Associated Web Services Methods:

```
GetIQVectorDataSize (VSA)
GetIQVectorData (VSA)
```

:TRACe:DDEMod:DATA:POWertime?

Return Parameters: <start symbol> <number of symbols>

Parameter Form: <nv> <nv>

Default: None. This command is a query, which is why it does not have a default value.

Description: This command queries the digital demodulation power time data of the indicated trace. The data is output in IEEE 488.2 arbitrary block format with the first character in the header as a "#" followed by a non-zero digit indicating the number of digits to follow. The digits that follow represent the amount of data bytes in the arbitrary block data. For example, one data block would be as follows:
#216<b><b><b><b><b><b><b><b><b><b><b><b><b><b><b><b>
where
"#" is the start of the header.
"2" specifies that the next two digits form the block length.
"16" specifies that the block length is 16 bytes and is also the end of the header.
"<b>" represents a single byte of data in the data block and all 16 bytes represent the entire block of data.

The start symbol and number of symbols range from 1 to the total number of symbols where the total number of symbols is constrained to: 625 ≥ (Symbol Rate x Capture Time ≥ 1.

Precondition: The modulation measurement mode must be activated via GPIB.

Query Form: N/A

Example:
```
:INST:SEL DDEM
:TRAC:DDEM:DATA:POW? 1 256
```

Modes: VSA

Associated Web Services Methods:
```
GetModPowerWaveformDataSize (VSA)
GetModPowerWaveformData (VSA)
```

:TRACe<1 to 5>?

Parameters:	None. This command is a query, which is why it does not have any input parameters.
Parameter Form:	<nr2>
Default:	None. This command is a query, which is why it does not have a default value.
Description:	This command queries the indicated trace data. A set of 501 comma separated values are returned.
Precondition:	The specified trace must be active.
Query Form:	N/A
Example:	`:TRAC1?`
Modes:	SPA
Associated Web Services Methods:	`GetTraceData (SPA)`

2-12 :TRIGger<1|2> Subsystem

The **:TRIGger<1|2>** subsystem contains commands for synchronizing instrument actions with events. This makes it possible to control and synchronize the start of a sweep with some other event.

Table 2-11. :TRIGger<1|2> Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:TRIGger<1|2>	-		
[:SEQuence]	-		
:HOLDoff	<nv>	S|MS|US	no query
:HOLDoff?	<nv>	S|MS|US	
:LEVel	-		
:EXTernal	<nv>	V|MV|UV	
:EXTernal?	<nv>	V|MV|UV	
:VIDeo	<nv>	DBM|DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW	no query
:SLOPe	<char>	POS|NEG	
:SLOPe?	<char>	POS|NEG	query only
:SOURce	<char>	EXTernal|INTernal|LINe|VIDeo|WIDeif|TTL	
:SOURce?	<char>	EXTernal|INTernal|LINe|VIDeo|WIDeif|TTL	query only

:TRIGger<1|2>[:SEQuence]:HOLDoff

Parameters: 0MS to 65MS

Parameter Form: <nv>

Default: 0MS

Description: This command sets the length of the trigger delay:

maximum range = current sweep time
maximum resolution = +65 ms

Query Form: `:TRIGger<1|2>[:SEQuence]:HOLDoff?`

Example: `:TRIG1:HOLD 5MS`

Modes: SPA, WCDMA

Associated Web Services Methods: `SetTriggerDelay (SPA)`
`SetTriggerDelay (WCDMA)`

:TRIGger<1|2>[:SEQuence]:HOLDoff?

Return Parameters: 0MS to 65MS

Parameter Form: <nv>

Default: 0MS

Description: This command queries the length of the trigger delay:

maximum range = current sweep time
maximum resolution = +65 ms

Query Form: N/A

Example: `:TRIG1:HOLD?`

Modes: SPA, WCDMA

Associated Web Services Methods:
`GetTriggerDelayInSecs (SPA)`
`GetTriggerDelayInSecs (WCDMA)`

:TRIGger<1|2>[:SEQuence]:LEVel:EXTernal

Parameters: –10V to +10V

Parameter Form: <nv>

Default: 1.4V (TTL)

Description: This command sets the level of the external trigger source with a resolution of 0.1V.

Query Form: None

Example: `:TRIG1:LEV:EXT 2V`

Modes: SPA, WCDMA

Associated Web Services Methods:
`SetExternalTriggerLevel (SPA)`
`SetExternalTriggerLevel (WCDMA)`

:TRIGger<1|2>[:SEQuence]:LEVel:EXTernal?

Return Parameters:	–10V to +10V
Parameter Form:	<nv>
Default:	1.4V (TTL)
Description:	This command queries the level of the external trigger source.
Query Form:	N/A
Example:	`:TRIG1:LEV:EXT?`
Modes:	SPA, WCDMA
Associated Web Services Methods:	`GetExternalTriggerLevelInVolts (SPA)` `GetExternalTriggerLevelInVolts (WCDMA)`

:TRIGger<1|2>[:SEQuence]:LEVel:VIDeo

Parameters:	0DBM to –150DBM Additionally supported units: DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW
Parameter Form:	<nv>
Default:	0DBM
Description:	This command sets the level of the video trigger source with a resolution of 1 dB.
Query Form:	`:TRIGger<1|2>[:SEQuence]:LEVel:VIDeo?`
Example:	`:TRIG1:LEV:VID -50DBM`
Modes:	SPA, WCDMA
Associated Web Services Methods:	`SetVideoTriggerLevel (SPA)` `SetVideoTriggerLevel (WCDMA)`

:TRIGger<1|2>[:SEQuence]:LEVel:VIDeo?

Parameters: 0DBM to –150DBM
Additionally supported units:
DBMV|DBUV|W|V|MW|UW|NW|MV|UV|NV|PW|PV|FW|AW|ZW|YW

Parameter Form: <nv>

Default: 0DBM

Description: This command queries the level of the video trigger source.

Query Form: N/A

Example: `:TRIG1:LEV:VID?`

Modes: SPA, WCDMA

Associated Web Services Methods:
`GetVideoTriggerLevel (SPA)`
`GetVideoTriggerLevel (WCDMA)`

:TRIGger<1|2>[:SEQuence]:SLOPe

Parameters: POSitive|NEGative

Parameter Form: <char>

Default: POSitive

Description: This command sets the slope of the trigger signal. The selected trigger slope applies to all trigger signal sources.

Query Form: `:TRIGger<1|2>[:SEQuence]:SLOPe?`

Example: `:TRIG1:SLOP NEG`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`SetTriggerEdgeRising (SPA)`
`SetTriggerEdgeRising (VSA)`
`SetTriggerEdgeRising (WCDMA)`

:TRIGger<1|2>[:SEQuence]:SLOPe?

Return Parameters: POSitive|NEGative

Parameter Form: <char>

Default: POSitive

Description: This command queries the slope setting of the trigger signal. The selected trigger slope applies to all trigger signal sources.

Query Form: N/A

Example: `:TRIG1:SLOP?`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`IsTriggerEdgeRising (SPA)`
`IsTriggerEdgeRising (VSA)`
`IsTriggerEdgeRising (WCDMA)`

:TRIGger<1|2>[:SEQuence]:SOURce

Parameters: EXTernal|INTernal|LINe|VIDeo|WIDeif|TTL

Parameter Form: <char>

Default: INTernal

Description: This command selects the trigger source for the start of a sweep.

Query Form: `:TRIGger<1|2>[:SEQuence]:SOURce?`

Example: `:TRIG1:SOUR EXT`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`SetTriggerSource (SPA)`
`SetTriggerSource (VSA)`
`SetTriggerSource (WCDMA)`

:TRIGger<1|2>[:SEQuence]:SOURce?

Return Parameters:	EXTernal|INTernal|LINe|VIDeo|WIDeif|TTL
Parameter Form:	<char>
Default:	INTernal
Description:	This command returns the trigger source for the start of a sweep.
Query Form:	N/A
Example:	`:TRIG1:SOUR?`
Modes:	SPA, VSA, WCDMA

Associated Web Services Methods:	`GetTriggerSource (SPA)`
	`GetTriggerSource (VSA)`
	`GetTriggerSource (WCDMA)`

2-13 [:SENSe]:ACP Subsystem

The **[:SENSe]:ACP** subsystem contains commands for setting the adjacent channel power measurement parameters and modes.

Table 2-12. [:SENSe]:ACP Subsystem Commands

<table>
<tr><th>Command</th><th>Parameter Form</th><th>Character Data or Units</th><th>Notes</th></tr>
<tr><td>[:SENSe<1|2>]</td><td>-</td><td></td><td></td></tr>
<tr><td>:ACP</td><td>-</td><td></td><td></td></tr>
<tr><td>:ADJacent</td><td>-</td><td></td><td></td></tr>
<tr><td>:CHBandwidth</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHBandwidth?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHSPacing</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHSPacing?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:ALT<1|2></td><td>-</td><td></td><td></td></tr>
<tr><td>:CHBandwidth</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHBandwidth?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHSPacing</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHSPacing?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:CHBandwidth</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHBandwidth?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:FACTor</td><td>-</td><td></td><td></td></tr>
<tr><td>:ROLLoff</td><td><nv></td><td>unitless</td><td></td></tr>
<tr><td>:ROLLoff?</td><td><nv></td><td>unitless</td><td></td></tr>
<tr><td>:FFT</td><td>-</td><td></td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:FILTer</td><td>-</td><td></td><td></td></tr>
<tr><td>:RRC</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:RRC?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:HZ</td><td>-</td><td></td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:NOISecomp</td><td>-</td><td></td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:SRATe</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
</table>

Table 2-12. [:SENSe]:ACP Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:SRATe?	<nv>	HZ|KHZ|MHZ|GHZ	
:TYPE	<char>	REL|ABS	
:TYPE?	<char>	REL|ABS	

[:SENSe<1|2>]:ACP:ADJacent:CHBandwidth

Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command sets the adjacent channel bandwidth parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:ADJacent:CHBandwidth?`

Example: `:ACP:ADJ:CHB 1MHZ`

Modes: SPA

Associated Web Services Methods: `SetACPAdjChannelBandwidth (SPA)`

[:SENSe<1|2>]:ACP:ADJacent:CHBandwidth?

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command queries the adjacent channel bandwidth parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:ADJ:CHB?`

Modes: SPA

Associated Web Services Methods: `GetACPAdjacentChannelBandwidthInHz (SPA)`

[:SENSe<1|2>]:ACP:ADJacent:CHSPacing

Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command sets the channel spacing parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:ADJacent:CHSPacing?`

Example: `:ACP:ADJ:CHSP 2MHZ`

Modes: SPA

Associated Web Services Methods: `SetACPAdjacentChannelSpacing (SPA)`

[:SENSe<1|2>]:ACP:ADJacent:CHSPacing?

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command queries the channel spacing parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:ADJ:CHSP?`

Modes: SPA

Associated Web Services Methods: `GetACPAdjacentChannelSpacingInHz (SPA)`

[:SENSe<1|2>]:ACP:ADJacent:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the adjacent channel power measurement on or off.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:ADJacent:STATe?`

Example: `:ACP:ADJ:STAT ON`

Modes: SPA

Associated Web Services Methods: `ToggleACPAdjacentChannelState (SPA)`

[:SENSe<1|2>]:ACP:ADJacent:STATe?

Return Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	This command queries the adjacent channel power measurement status.
Precondition:	The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:ACP:ADJ:STAT?`
Modes:	SPA
Associated Web Services Methods:	`GetACPAdjacentChannelState (SPA)`

[:SENSe<1|2>]:ACP:ALT<1|2>:CHBandwidth

Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	5MHZ
Description:	This command sets the specified alternate channel bandwidth parameter for the adjacent channel power measurement.
Precondition:	The adjacent channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:ACP:ALT<1|2>:CHBandwidth?`
Example:	`:ACP:ALT1:CHB 1MHZ`
Modes:	SPA
Associated Web Services Methods:	`SetACPAlternateChannel1Bandwidth (SPA)` `SetACPAlternateChannel2Bandwidth (SPA)`

[:SENSe<1|2>]:ACP:ALT<1|2>:CHBandwidth?

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command queries the specified alternate channel bandwidth parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:ALT1:CHB?`

Modes: SPA

Associated Web Services Methods: `GetACPAlternateChannel1BandwidthInHz (SPA)`
`GetACPAlternateChannel2BandwidthInHz (SPA)`

[:SENSe<1|2>]:ACP:ALT<1|2>:CHSPacing

Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 10MHZ (ALT1) or 15MHZ (ALT2)

Description: This command sets the specified alternate channel spacing parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:ALT<1|2>:CHSPacing?`

Example: `:ACP:ALT1:CHSP 2MHZ`

Modes: SPA

Associated Web Services Methods: `SetACPAlternateChannel1Spacing (SPA)`
`SetACPAlternateChannel2Spacing (SPA)`

[:SENSe<1|2>]:ACP:ALT<1|2>:CHSPacing?

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 10MHZ (ALT1) or 15MHZ (ALT2)

Description: This command queries the specified alternate channel spacing parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:ALT1:CHSP?`

Modes: SPA

Associated Web Services Methods:
`GetACPAlternateChannel1SpacingInHz (SPA)`
`GetACPAlternateChannel2SpacingInHz (SPA)`

[:SENSe<1|2>]:ACP:ALT<1|2>:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the specified alternate channel measurement on or off for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:ALT<1|2>:STATe?`

Example: `:ACP:ALT1:STAT ON`

Modes: SPA

Associated Web Services Methods:
`ToggleACPAlternateChannel1State (SPA)`
`ToggleACPAlternateChannel2State (SPA)`

[:SENSe<1|2>]:ACP:ALT<1|2>:STATe?

Return Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	This command queries the specified alternate channel measurement status for the adjacent channel power measurement.
Precondition:	The adjacent channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:ACP:ALT1:STAT?`
Modes:	SPA
Associated Web Services Methods:	`GetACPAlternateChannel1State (SPA)` `GetACPAlternateChannel2State (SPA)`

[:SENSe<1|2>]:ACP:CHBandwidth

Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	5MHZ
Description:	This command sets the channel bandwidth parameter for the adjacent channel power measurement.
Precondition:	The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:ACP:CHBandwidth?`
Example:	`:ACP:CHB 1MHZ`
Modes:	SPA
Associated Web Services Methods:	`SetACPChannelBandwidth (SPA)`

[:SENSe<1|2>]:ACP:CHBandwidth?

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command queries the channel bandwidth parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:CHB?`

Modes: SPA

Associated Web Services Methods: `GetACPChannelBandwidthInHz (SPA)`

[:SENSe<1|2>]:ACP:FACTor:ROLLoff

Parameters: 0.1 to 1.0

Parameter Form: <nv>

Default: 0.5

Description: This command sets the roll-off factor (a) parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:FACTor:ROLLoff?`

Example: `:ACP:FACT:ROLL 0.8`

Modes: SPA

Associated Web Services Methods: `SetACPRollOffFactor (SPA)`

[:SENSe<1|2>]:ACP:FACTor:ROLLoff?

Return Parameters:	0.1 to 1.0
Parameter Form:	<nv>
Default:	0.5
Description:	This command queries the roll-off factor (a) parameter for the adjacent channel power measurement.
Precondition:	The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:ACP:FACT:ROLL?`
Modes:	SPA
Associated Web Services Methods:	`GetACPRollOffFactor (SPA)`

[:SENSe<1|2>]:ACP:FFT:STATe

Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	This command sets the FFT state on or off for the adjacent channel power measurement.
Precondition:	The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:ACP:FFT:STATe?`
Example:	`:ACP:FFT:STAT ON`
Modes:	SPA
Associated Web Services Methods:	`ToggleACPFFTState (SPA)`

[:SENSe<1|2>]:ACP:FFT:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the FFT state for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:FFT:STAT?`

Modes: SPA

Associated Web Services Methods: `GetACPFFTState (SPA)`

[:SENSe<1|2>]:ACP:FILTer:RRC

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the RRC filter on or off for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:FILTer:RRC?`

Example: `:ACP:FILT:RRC ON`

Modes: SPA

Associated Web Services Methods: `ToggleACPRRCFilterState (SPA)`

[:SENSe<1|2>]:ACP:FILTer:RRC?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the RRC filter status for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:FILT:RRC?`

Modes: SPA

Associated Web Services Methods: `GetACPRRCFilterState (SPA)`

[:SENSe<1|2>]:ACP:HZ:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the Hertz state on or off for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:HZ:STATe?`

Example: `:ACP:HZ:STAT ON`

Modes: SPA

Associated Web Services Methods: `SetACPDivisionPerHzState (SPA)`

[:SENSe<1|2>]:ACP:HZ:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the Hertz state for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:HZ:STAT?`

Modes: SPA

Associated Web Services Methods: `GetACPDivisionPerHzState (SPA)`

[:SENSe<1|2>]:ACP:NOISecomp:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the noise compensation state on or off for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:NOISecomp:STATe?`

Example: `:ACP:NOIS:STAT ON`

Modes: SPA

Associated Web Services Methods: `ToggleACPNoiseCompensationState (SPA)`

[:SENSe<1|2>]:ACP:NOISecomp:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the noise compensation state for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:NOIS:STAT?`

Modes: SPA

Associated Web Services Methods: `GetACPNoiseCompensationState (SPA)`

[:SENSe<1|2>]:ACP:SRATe

Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 3.84MHZ

Description: This command sets the symbol rate parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:ACP:SRATe?`

Example: `:ACP:SRAT 1MHZ`

Modes: SPA

Associated Web Services Methods: `SetACPSymbolRate (SPA)`

[:SENSe<1|2>]:ACP:SRATe?

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 3.84MHZ

Description: This command queries the symbol rate parameter for the adjacent channel power measurement.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:SRAT?`

Modes: SPA

Associated Web Services Methods: `GetACPSymbolRateInHz (SPA)`

[:SENSe<1|2>]:ACP:TYPE

Parameters: REL|ABS

Parameter Form: <char>

Default: REL

Description: This command sets the adjacent channel power measurement type of relative or absolute.

Query Form: `[:SENSe<1|2>]:ACP:TYPE?`

Example: `:ACP:TYPE ABS`

Modes: SPA

Associated Web Services Methods: `SetACPOBMMode (SPA)`

[:SENSe<1|2>]:ACP:TYPE?

Return Parameters: REL|ABS

Parameter Form: <char>

Default: REL

Description: This command queries the adjacent channel power measurement type.

Precondition: The adjacent channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:ACP:TYPE?`

Modes: SPA

Associated Web Services Methods: `GetACPOBMMode (SPA)`

2-14 [:SENSe]:BANDwidth Subsystem

The **[:SENSe]:BANDwidth** subsystem contains commands for setting the spectrum measurement bandwidth parameters and modes.

Table 2-13. [:SENSe]:BANDwidth Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
[:SENSe<1|2>]	-		
:BANDwidth	-		
:VIDeo	<nv>	HZ|KHZ|MHZ	
:AUTO	<boolean>	ON|OFF	
:AUTO?	<boolean>	ON|OFF	
:RATio	<nv>	unitless	
:RATio?	<nv>	unitless	
:VIDeo?	<nv>	HZ|KHZ|MHZ	
[:RESolution]	<nv>	HZ|KHZ|MHZ	
:AUTO	<boolean>	ON|OFF	
:AUTO?	<boolean>	1|0	
:RATio	<nv>	unitless	
:RATio?	<nv>	unitless	
:TYPE	<char>	NORMal|FFT|WFFT|NFFT| ZSPA	
:TYPE?	<char>	NORMal|FFT|WFFT|NFFT| ZSPA	
[:RESolution]?	<nv>	HZ|KHZ|MHZ	

[:SENSe<1|2>]:BANDwidth:VIDeo

Parameters: 1HZ to 10MHZ

Parameter Form: <nv>

Default: 10MHZ

Description: This command sets the video bandwidth parameter.

Query Form: `[:SENSe<1|2>]:BANDwidth:VIDeo?`

Example: `:BAND:VID 10KHZ`

Modes: SPA

Associated Web Services Methods: `SetVBW (SPA)`

[:SENSe<1|2>]:BANDwidth:VIDeo:AUTO

Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	ON
Description:	This command sets the video bandwidth mode.
Query Form:	`[:SENSe<1|2>]:BANDwidth:VIDeo:AUTO?`
Example:	`:BAND:VID:AUTO OFF`
Modes:	SPA
Associated Web Services Methods:	`SetVBWAuto (SPA)`

[:SENSe<1|2>]:BANDwidth:VIDeo:AUTO?

Return Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	ON
Description:	This command queries the video bandwidth mode.
Query Form:	N/A
Example:	`:BAND:VID:AUTO?`
Modes:	SPA
Associated Web Services Methods:	`GetVBWAuto (SPA)`

`[:SENSe<1|2>]:BANDwidth:VIDeo:RATio`

Parameters: 0.001 to 1000

Parameter Form: <nv>

Default: 5

Description: This command sets the VBW/RBW ratio.

Query Form: `[:SENSe<1|2>]:BANDwidth:VIDeo:RATio?`

Example: `:BAND:VID:RAT 1`

Modes: SPA

Associated Web Services Methods: `SetVBWToRBWRatio (SPA)`

`[:SENSe<1|2>]:BANDwidth:VIDeo:RATio?`

Return Parameters: 0.001 to 1000

Parameter Form: <nv>

Default: 5

Description: This command queries the VBW/RBW ratio.

Query Form: N/A

Example: `:BAND:VID:RAT?`

Modes: SPA

Associated Web Services Methods: `GetVBWToRBWRatio (SPA)`

`[:SENSe<1|2>]:BANDwidth:VIDeo?`

Parameters:	1HZ to 10MHZ
Parameter Form:	<nv>
Default:	10MHZ
Description:	This command queries the video bandwidth parameter setting.
Query Form:	N/A
Example:	`:BAND:VID?`
Modes:	SPA
Associated Web Services Methods:	`SetVBW (SPA)`

`[:SENSe<1|2>]:BANDwidth[:RESolution]`

Parameters:	10HZ to 8MHZ
Parameter Form:	<nv>
Default:	Span / 50
Description:	This command sets the resolution bandwidth parameter.
Query Form:	`[:SENSe<1|2>]:BANDwidth:RESolution?`
Example:	`:BAND 1KHZ`
Modes:	SPA
Associated Web Services Methods:	`SetRBW (SPA)`

[:SENSe<1|2>]:BANDwidth[:RESolution]:AUTO

Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command sets the resolution bandwidth mode.

Query Form: [:SENSe<1|2>]:BANDwidth[:RESolution]:AUTO?

Example: :BAND:AUTO OFF

Modes: SPA

Associated Web Services Methods: SetRBWAuto (SPA)

[:SENSe<1|2>]:BANDwidth[:RESolution]:AUTO?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command queries the resolution bandwidth mode.

Query Form: N/A

Example: :BAND:AUTO?

Modes: SPA

Associated Web Services Methods: GetRBWAuto (SPA)

[:SENSe<1|2>]:BANDwidth[:RESolution]:RATio

Parameters: 2 to 10000

Parameter Form: <nv>

Default: 50

Description: This command sets the span/RBW ratio.

Query Form: `[:SENSe<1|2>]:BANDwidth[:RESolution]:RATio?`

Example: `:BAND:RAT 100`

Modes: SPA

Associated Web Services Methods: `SetSpanToRBWRatio (SPA)`

[:SENSe<1|2>]:BANDwidth[:RESolution]:RATio?

Return Parameters: 2 to 10000

Parameter Form: <nv>

Default: 50

Description: This command queries the span/RBW ratio.

Query Form: N/A

Example: `:BAND:RAT?`

Modes: SPA

Associated Web Services Methods: `GetSpanToRBWRatio (SPA)`

[:SENSe<1|2>]:BANDwidth[:RESolution]:TYPE

Parameters: NORMal|FFT|WFFT|NFFT|ZSPA

Parameter Form: <char>

Default: NORMal

Description: This command sets the sweep type to normal, fast fourier transform, wideband fast fourier transform, or zero span time domain.

Query Form: `[:SENSe<1|2>]:BANDwidth[:RESolution]:TYPE?`

Example: `:BAND:TYPE FFT`

Modes: SPA

Associated Web Services Methods: `SetSweepType (SPA)`

[:SENSe<1|2>]:BANDwidth[:RESolution]:TYPE?

Return Parameters: NORMal|FFT|WFFT|NFFT|ZSPA

Parameter Form: <char>

Default: NORMal

Description: This command queries the sweep type. Possible return values are normal, fast fourier transform, wideband fast fourier transform, or zero span time domain.

Query Form: N/A

Example: `:BAND:TYPE?`

Modes: SPA

Associated Web Services Methods: `GetSweepType (SPA)`

`[:SENSe<1|2>]:BANDwidth[:RESolution]?`

Return Parameters:	10HZ to 8MHZ
Parameter Form:	<nv>
Default:	Span / 50
Description:	This command queries the resolution bandwidth parameter.
Query Form:	N/A
Example:	`:BAND?`
Modes:	SPA
Associated Web Services Methods:	`GetRBWInHz (SPA)`

2-15 [:SENSe]:CCDF Subsystem

The `[:SENSe]:CCDF` subsystem contains commands for setting up the CCDF (Complementary Cumulative Distribution Function) parameters and modes.

Table 2-14. [:SENSe]:CCDF Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
`[:SENSe<1|2>]`	-		
`:CCDF`	-		
`:GAUSsianreferenceline`	-		
`:STATe`	<boolean>	ON|OFF	
`:STATe?`	<boolean>	ON|OFF	
`:RBW`	<char>	<nv>HZ|KHZ|MHZ|GHZ	
`:RBW?`	<char>	<nv>HZ|KHZ|MHZ|GHZ	
`:RESults?`	<char> input <nv> output	Mean|Peak|Crest|TargetSamples|AcquiredSamples|p1_prob|p01_prob|p001_prob|p0001_prob|p00001_prob|p000001_prob|p0000001_prob|Valid	query only
`:TARget`			
`:SAMPles`	<nv>	unitless	
`:SAMPles?`	<nv>	unitless	
`:XAXis`			
`:MAXimum`	<nv>	unitless	
`:MAXimum?`	<nv>	unitless	

`[:SENSe<1|2>]:CCDF:GAUSsianreferenceline:STATe`

Parameters: ON | OFF

Parameter Form: <boolean>

Default: OFF

Description: Sets CCDF Gaussian reference line status for the spectrum measurement.

Query Form: `[:SENSe<1|2>]:CCDF:GAUSsianreferenceline:STATe?`

Example: `:CCDF:GAUS:STAT ON`

Modes: SPA

Associated Web Service: `SetGaussianReferenceLineStatus (SPA)`

[:SENSe<1|2>]:CCDF:GAUSsianreferenceline:STATe?

Return Parameters:	ON | OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	gets CCDF Gaussian reference line status for the spectrum measurement.
Query Form:	N/A
Example:	`:CCDF:GAUS:STAT?`
Modes:	SPA
Associated Web Service:	`GetGaussianReferenceLineStatus (SPA)`

[:SENSe<1|2>]:CCDF:RBW

Parameters:	1HZ to 8GHZ
Parameter Form:	<char>
Default:	10MHZ
Description:	Sets CCDF RBW for the spectrum measurement.
Query Form:	`[:SENSe<1|2>]:CCDF:RBW?`
Example:	`:CCDF:RBW 10MHZ`
Modes:	SPA
Associated Web Service:	`SetCCDFRBW (SPA)`

`[:SENSe<1|2>]:CCDF:RBW?`

Return Parameters:	1HZ to 8GHZ
Parameter Form:	<char>
Default:	10MHZ
Description:	gets CCDF RBW for the spectrum measurement.
Query Form:	N/A
Example:	`:CCDF:RBW?`
Modes:	SPA
Associated Web Service:	`GetCCDFRBW (SPA)`

`[:SENSe<1|2>]:CCDF:RESults?`

Return Parameters:	**Input:** Mean | Peak | Crest | TargetSamples | AcquiredSamples | p1_prob | p01_prob | p001_prob | p0001_prob | p00001_prob | p000001_prob | p0000001_prob | Valid **Output:** <nv>
Parameter Form:	<char>
Default:	N/A
Description:	gets CCDF results for the spectrum measurement.
Query Form:	N/A
Example:	`:CCDF:RES?`
Modes:	SPA
Associated Web Service:	`GetCCDFResults (SPA)`

`[:SENSe<1|2>]:CCDF:TARget:SAMPles`

Parameters:	100 to 1000000000
Parameter Form:	<nv>
Default:	4000
Description:	Sets CCDF target number of samples for the spectrum measurement.
Query Form:	`[:SENSe<1|2>]:CCDF:TARget:SAMPles?`
Example:	`:CCDF:TARG:SAMP 4000`
Modes:	SPA
Associated Web Service:	`SetTargetNumberOfSamples (SPA)`

`[:SENSe<1|2>]:CCDF:TARget:SAMPles?`

Return Parameters:	100 to 1000000000
Parameter Form:	<nv>
Default:	4000
Description:	gets CCDF target number of samples for the spectrum measurement.
Query Form:	N/A
Example:	`:CCDF:TARG:SAMP?`
Modes:	SPA
Associated Web Service:	`GetTargetNumberOfSamples (SPA)`

[:SENSe<1|2>]:CCDF:XAXis:MAXimum

Parameters: 1 to 200

Parameter Form: <nv>

Default: 6

Description: Sets CCDF X axis maximum for the spectrum measurement.

Query Form: `[:SENSe<1|2>]:CCDF:XAXis:MAXimum?`

Example: `:CCDF:XAX:MAX 6`

Modes: SPA

Associated Web Service: `SetXAxisMaximum (SPA)`

[:SENSe<1|2>]:CCDF:XAXis:MAXimum?

Return Parameters: 1 to 200

Parameter Form: <nv>

Default: 6

Description: gets CCDF X axis maximum for the spectrum measurement.

Query Form: N/A

Example: `:CCDF:XAX:MAX?`

Modes: SPA

Associated Web Service: `GetXAxisMaximum (SPA)`

2-16 [:SENSe]:CHP Subsystem

The **[:SENSe]:CHP** subsystem contains commands for setting up the channel power measurement parameters and modes.

Table 2-15. [:SENSe]:CHP Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
[:SENSe<1|2>]	-		
:CHP	-		
:BANDwidth	<nv>	HZ|KHZ|MHZ|GHZ	
:BANDwidth?	<nv>	HZ|KHZ|MHZ|GHZ	
:FACTor	-		
:ROLLoff	<nv>	unitless	
:ROLLoff?	<nv>	unitless	
:FFT	-		
:STATe	<boolean>	ON|OFF	
:STATe?	<boolean>	ON|OFF	
:FILTer	-		
:RRC	<boolean>	ON|OFF	
:RRC?	<boolean>	ON|OFF	
:HZ	-		
:STATe	<boolean>	ON|OFF	
:STATe?	<boolean>	ON|OFF	
:NOISecomp	-		
:STATe	<boolean>	ON|OFF	
:STATe?	<boolean>	ON|OFF	
:SRATe	<nv>	HZ|KHZ|MHZ|GHZ	
:SRATe?	<nv>	HZ|KHZ|MHZ|GHZ	
:TYPE	<char>	REL|ABS	
:TYPE?	<char>	REL|ABS	

[:SENSe<1|2>]:CHP:BANDwidth

Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	5MHZ
Description:	This command sets the channel power bandwidth parameter.
Precondition:	The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:CHP:BANDwidth?`
Example:	`:CHP:BAND 2MHZ`
Modes:	SPA
Associated Web Services Methods:	`SetChannelPowerBandwidth (SPA)`

[:SENSe<1|2>]:CHP:BANDwidth?

Return Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	5MHZ
Description:	This command queries the channel power bandwidth parameter.
Precondition:	The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:CHP:BAND?`
Modes:	SPA
Associated Web Services Methods:	`GetChannelPowerBandwidthInHz (SPA)`

[:SENSe<1|2>]:CHP:FACTor:ROLLoff

Parameters:	0.1 to 1.0
Parameter Form:	<nv>
Default:	0.22
Description:	This command sets the roll-off factor parameter for the channel power measurement.
Precondition:	The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:CHP:FACTor:ROLLoff?`
Example:	`:CHP:FACT:ROLL 0.8`
Modes:	SPA
Associated Web Services Methods:	`SetChannelPowerRollOffFactor (SPA)`

[:SENSe<1|2>]:CHP:FACTor:ROLLoff?

Return Parameters:	0.1 to 1.0
Parameter Form:	<nv>
Default:	0.22
Description:	This command queries the roll-off factor parameter for the channel power measurement.
Precondition:	The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:CHP:FACT:ROLL?`
Modes:	SPA
Associated Web Services Methods:	`GetChannelPowerRollOffFactor (SPA)`

[:SENSe<1|2>]:CHP:FFT:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the fast fourier transform state on or off for the channel power measurement.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:CHP:FFT:STATe?`

Example: `:CHP:FFT:STAT ON`

Modes: SPA

Associated Web Services Methods: `ToggleChannelPowerFFTState (SPA)`

[:SENSe<1|2>]:CHP:FFT:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the fast fourier transform state for the channel power measurement.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:CHP:FFT:STAT?`

Modes: SPA

Associated Web Services Methods: `GetChannelPowerFFTState (SPA)`

[:SENSe<1|2>]:CHP:FILTer:RRC

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the RRC filter on or off for the channel power measurement.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:CHP:FILTer:RRC?`

Example: `:CHP:FILT:RRC ON`

Modes: SPA

Associated Web Services Methods: `ToggleChannelPowerRRCFilterState (SPA)`

[:SENSe<1|2>]:CHP:FILTer:RRC?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the RRC filter status for the channel power measurement.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:CHP:FILT:RRC?`

Modes: SPA

Associated Web Services Methods: `GetChannelPowerRRCFilterState (SPA)`

[:SENSe<1|2>]:CHP:HZ:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the Hertz state on or off for the channel power measurement.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:CHP:HZ:STATe?`

Example: `:CHP:HZ:STAT ON`

Modes: SPA

Associated Web Services Methods: `SetChannelPowerDivisionPerHzState (SPA)`

[:SENSe<1|2>]:CHP:HZ:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the Hertz state for the channel power measurement.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:CHP:HZ:STAT?`

Modes: SPA

Associated Web Services Methods: `GetChannelPowerDivisionPerHzState (SPA)`

[:SENSe<1|2>]:CHP:NOISecomp:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the noise compensation on or off for the channel power measurement.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: This command has no query.

Example: `:CHP:NOIS:STAT ON`

Modes: SPA

Associated Web Services Methods: `ToggleCPNoiseCompensationState (SPA)`

[:SENSe<1|2>]:CHP:NOISecomp:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the noise compensation state for the channel power measurement.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:CHP:NOIS:STAT?`

Modes: SPA

Associated Web Services Methods: `GetCPNoiseCompensationState (SPA)`

`[:SENSe<1|2>]:CHP:SRATe`

Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	3.84MHZ
Description:	This command sets the symbol rate parameter for the channel power measurement.
Precondition:	The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:CHP:SRATe?`
Example:	`:CHP:SRAT 2.54MHZ`
Modes:	SPA
Associated Web Services Methods:	`SetChannelPowerSymbolRate (SPA)`

`[:SENSe<1|2>]:CHP:SRATe?`

Return Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	3.84MHZ
Description:	This command queries the symbol rate parameter for the channel power measurement.
Precondition:	The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:CHP:SRAT?`
Modes:	SPA
Associated Web Services Methods:	`GetChannelPowerSymbolRateInHz (SPA)`

[:SENSe<1|2>]:CHP:TYPE

Parameters: REL|ABS

Parameter Form: <char>

Default: REL

Description: This command sets the channel power measurement type of relative or absolute.

Query Form: `[:SENSe<1|2>]:CHP:TYPE?`

Example: `:CHP:TYPE REL`

Modes: SPA

Associated Web Services Methods: `SetChannelPowerOBMMode (SPA)`

[:SENSe<1|2>]:CHP:TYPE?

Return Parameters: REL|ABS

Parameter Form: <char>

Default: REL

Description: This command queries the channel power measurement type.

Precondition: The channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:CHP:TYPE?`

Modes: SPA

Associated Web Services Methods: `GetChannelPowerOBMMode (SPA)`

2-17 [:SENSe]:DDEMod Subsystem

The **[:SENSe]:DDEMod** subsystem contains commands for setting up the digital demodulation parameters and modes. These commands function for the VSA measurement mode only.

Table 2-16. [:SENSe]:DDEMod Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
[:SENSe<1|2>]	-		
:DDEMod	-		
:DIFFcode	-		
[:STATe]	<boolean>	ON|OFF	VSA only
[:STATe]?	<boolean>	ON|OFF	VSA only
:DISPlay	-		
:FORMat	<char>	VECT|EVMTime|POWERTime|SUMMary|IEYE|QEYE|CONStellation	VSA only
:FORMat?	<char>	VECT|EVMTime|POWERTime|SUMMary|IEYE|QEYE|CONStellation	VSA only
:FILTer	-		
:ALPHa	<nv>	unitless	VSA only
:ALPHa?	<nv>	unitless	VSA only
:MEASurement	<char>	LPF|NQF|RNF	VSA only
:MEASurement?	<char>	LPF|NQF|RNF	VSA only
:FORMat	<char>	BPSK|QPSK|P4QPSK|8PSK|3P8PSK|16QAM|32QAM|64QAM|128QAM|256QAM	VSA only
:FORMat?	<char>	BPSK|QPSK|P4QPSK|8PSK|3P8PSK|16QAM|32QAM|64QAM|128QAM|256QAM	VSA only
:MARKer<1|2>	-		
:MAXimum	-		
:NEXT	N/A		event only VSA only
[:PEAK]	N/A		event only VSA only
:STATe	<boolean>	ON|OFF	VSA only
:STATe?	<boolean>	ON|OFF	VSA only
:X	<nv>	unitless	VSA only
:X?	<nv>	unitless	VSA only
:Y?	<nv><nv>	unitless	VSA only
:NUMTap	<nv>	unitless	VSA only
:NUMTap?	<nv>	unitless	VSA only

Table 2-16. [:SENSe]:DDEMod Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
`:RANGe`	-		
`:TRACking`	<nv>	ON|OFF	VSA only
`:TRACking?`	<nv>	ON|OFF	VSA only
`:RESult?`	<char>	EVM|EVMPeak|EVMPeakPos|EVM95|PhaseErr| PhaseErrPeak|PhaseErrPeakPos|AmpErr|AmpErrPeak|AmpErrPeakPos|Power|Offset|CarrierFreqErr|SymbolClockErr	query only VSA only
`:SRATe`	<nv>	HZ|KHZ|MHZ	VSA only
`:SRATe?`	<nv>	HZ|KHZ|MHZ	VSA only

`[:SENSe<1|2>]:DDEMod:DIFFcode[:STATe]`

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the differential encoding state on or off.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:DIFFcode[:STATe]?`

Example: `:DDEM:DIFF ON`

Modes: VSA

Associated Web Services Methods: `SetDifferentialEncodingOn (VSA)`

[:SENSe<1|2>]:DDEMod:DIFFcode[:STATe]?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the differential encoding state.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:DIFF?`

Modes: VSA

Associated Web Services Methods: `IsDifferentialEncodingOn (VSA)`

[:SENSe<1|2>]:DDEMod:DISPlay:FORMat

Parameters: VECT|EVMTime|POWERTime|SUMMary|IEYE|QEYE|CONStellation

Parameter Form: <char>

Default: VECT

Description: This command defines the trace display as follows:
VECT = Vector
EVMTime = EVM/Time
POWERTime = Power/Time
SUMMary = Summary
IEYE = EYE Diagram from I Value
QEYE = EYE Diagram from Q Value
CONStellation = Constellation

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:DISPlay:FORMat?`

Example: `:DDEM:DISP:FORM SUMM`

Modes: VSA

Associated Web Services Methods: `SetGraphType (VSA)`

[:SENSe<1|2>]:DDEMod:DISPlay:FORMat?

Return Parameters: VECT|EVMTime|POWERTime|SUMMary|IEYE|QEYE|CONStellation

Parameter Form: <char>

Default: VECT

Description: This command queries the trace display with the following results:
VECT = Vector
EVMTime = EVM/Time
POWERTime = Power/Time
SUMMary = Summary
IEYE = EYE Diagram from I Value
QEYE = EYE Diagram from Q Value
CONStellation = Constellation

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:DISP:FORM?`

Modes: VSA

Associated Web Services Methods: `GetGraphType (VSA)`

[:SENSe<1|2>]:DDEMod:FILTer:ALPHa

Parameters: 0.1 to 1.0

Parameter Form: <nv>

Default: 0.5

Description: This command sets the roll-off factor (alpha value).

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:FILTer:ALPHa?`

Example: `:DDEM:FILT:ALPH 0.5`

Modes: VSA

Associated Web Services Methods: `SetFilterRollOffFactor (VSA)`

[:SENSe<1|2>]:DDEMod:FILTer:ALPHa?

Return Parameters: 0.1 to 1.0

Parameter Form: <nv>

Default: 0.5

Description: This command queries the modulation roll-off factor (alpha value).

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:FILT:ALPH?`

Modes: VSA

Associated Web Services Methods: `GetFilterRollOffFactor (VSA)`

[:SENSe<1|2>]:DDEMod:FILTer:MEASurement

Parameters: LPF|NQF|RNF

Parameter Form: <char>

Default: RNF

Description: This command sets the modulation filter type as follows:

LPF: Low Pass Filter
NQF: Nyquist Filter
RNF: Root Nyquist Filter

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:FILTer:MEASurement?`

Example: `:DDEM:FILT:MEAS LPF`

Modes: VSA

Associated Web Services Methods: `SetFilterType (VSA)`

[:SENSe<1|2>]:DDEMod:FILTer:MEASurement?

Return Parameters: LPF|NQF|RNF

Parameter Form: <char>

Default: RNF

Description: This command queries the modulation filter type with the following return strings:

LPF: Low Pass Filter
NQF: Nyquist Filter
RNF: Root Nyquist Filter

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:FILT:MEAS?`

Modes: VSA

Associated Web Services Methods: `GetFilterType (VSA)`

[:SENSe<1|2>]:DDEMod:FORMat

Parameters: BPSK|QPSK|P4QPSK|8PSK|3P8PSK|16QAM|32QAM|64QAM|128QAM|256QAM

Parameter Form: <char>

Default: QPSK

Description: This command sets the modulation format.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:FORMat?`

Example: `:DDEM:FORM BPSK`

Modes: VSA

Associated Web Services Methods: `SetModulationType (VSA)`

[:SENSe<1|2>]:DDEMod:FORMat?

Return Parameters: BPSK|QPSK|P4QPSK|8PSK|3P8PSK|16QAM|32QAM|64QAM|128QAM|256QAM

Parameter Form: <char>

Default: QPSK

Description: This command queries the modulation format.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:FORM?`

Modes: VSA

Associated Web Services Methods: `GetModulationType (VSA)`

[:SENSe<1|2>]:DDEMod:MARKer<1|2>:MAXimum:NEXT

Parameters: None. This command is an event, which is why it has no parameters.

Parameter Form: N/A

Default: N/A

Description: This command sets the indicated marker to the next peak.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: None

Example: `:DDEM:MARK1:MAX:NEXT`

Modes: VSA

Associated Web Services Methods: `SetMarkerToNextPeak (VSA)`

[:SENSe<1|2>]:DDEMod:MARKer<1|2>:MAXimum[:PEAK]

Parameters: None. This command is an event, which is why it has no parameters.

Parameter Form: N/A

Default: N/A

Description: This command sets the indicated marker to the trace peak.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: None

Example: `:DDEM:MARK1:MAX`

Modes: VSA

Associated Web Services Methods: `SetMarkerToPeak (VSA)`

[:SENSe<1|2>]:DDEMod:MARKer<1|2>:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the indicated marker on or off.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:MARKer<1|2>:STATe?`

Example: `:DDEM:MARK1:STAT ON`

Modes: VSA

Associated Web Services Methods: `SetMarkerMode (VSA)`

[:SENSe<1|2>]:DDEMod:MARKer<1|2>:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the indicated marker's mode.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:MARK1:STAT?`

Modes: VSA

Associated Web Services Methods: None

[:SENSe<1|2>]:DDEMod:MARKer<1|2>:X

Parameters: 0 to number of symbols

Parameter Form: <nv>

Default: None. This command is an event, which is why it does not have a default value.

Description: This command sets the indicated marker's position in symbol number.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:MARKer<1|2>:X?`

Example: `:DDEM:MARK1:X 100`

Modes: VSA

Associated Web Services Methods: `SetMarkerPosition (VSA)`

[:SENSe<1|2>]:DDEMod:MARKer<1|2>:X?

Return Parameters:	0 to number of symbols
Parameter Form:	<nv>
Default:	None. This command is a query, which is why it does not have a default value.
Description:	This command queries the indicated marker's position in symbol number.
Precondition:	The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:DDEM:MARK1:X?`
Modes:	VSA
Associated Web Services Methods:	`GetMarkerPosition (VSA)`

[:SENSe<1|2>]:DDEMod:MARKer<1|2>:Y?

Return Parameters: Character data output.

Parameter Form: <char>

Default: None. This command is a query, which is why it does not have a default value.

Description: This command queries the marker position and value. The character data output is comma delimited and contains the graph type, symbol number, marker value, or I and Q values as follows:

<Power Time, Symbol Number, Power Value>
<EVM Time, Symbol Number, EVM Value>
<EYE I, Symbol Number, I Value, Q Value>
<EYE Q, Symbol Number, I Value, Q Value>
<EYE, Symbol Number, I Value, Q Value>
<Vector, Symbol Number, I Value, Q Value>
<Constellation, Symbol Number, I Value, Q Value>

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:MARK1:Y?`

Modes: VSA

Associated Web Services Methods:
`GetVectorDiagramIQMarkerPosition (VSA)`
`GetConstellationDiagramIQMarkerPosition (VSA)`
`GetEye_IDiagramIMarkerPosition (VSA)`
`GetEye_QDiagramQMarkerPosition (VSA)`
`GetEVMDiagramMarkerPosition (VSA)`
`GetPowerDiagramMarkerPosition (VSA)`

[:SENSe<1|2>]:DDEMod:NUMTap

Parameters:	1 to 2048
Parameter Form:	<nv>
Default:	256
Description:	This command sets the number of taps parameter.
Precondition:	The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:DDEMod:NUMTap?`
Example:	`:DDEM:NUMT 320`
Modes:	VSA
Associated Web Services Methods:	`SetNumOfTaps (VSA)`

[:SENSe<1|2>]:DDEMod:NUMTap?

Return Parameters:	1 to 2048
Parameter Form:	<nv>
Default:	256
Description:	This command queries the number of taps parameter.
Precondition:	The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:DDEM:NUMT?`
Modes:	VSA
Associated Web Services Methods:	`GetNumOfTaps (VSA)`

[:SENSe<1|2>]:DDEMod:RANGe:TRACking

Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command sets the tracking range mode.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:RANGe:TRACking?`

Example: `:DDEM:RANG:TRAC OFF`

Modes: VSA

Associated Web Services Methods: `SetTrackingFlag (VSA)`

[:SENSe<1|2>]:DDEMod:RANGe:TRACking?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command queries the tracking range mode.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:RANG:TRAC?`

Modes: VSA

Associated Web Services Methods: `GetTrackingFlag (VSA)`

[:SENSe<1|2>]:DDEMod:RESult?

Return Parameters: Character data output.

Parameter Form: <char>

Default: None. This command is a query, which is why it does not have a default value.

Description: This command returns comma delimited values for the following items depending on the instrument's current measurement mode:

EVM<sp><nv>,
AmpErr<sp><nv>,
QuadErr<sp><nv>,
EVMPeak<sp><nv>,
AmpErrPeak<sp><nv>,
IQImbalance<sp><nv>,
EVMPeakPos<sp><nv>,
AmpErrPeakPos<sp><nv>,
IQOffset<sp><nv>,
EVM95<sp><nv>,
Power<sp><nv>,
CarrierFreqErr<sp><nv>,
PhaseErr<sp><nv>,
Rho<sp><nv>,
SymbolClockErr<sp><nv>,
PhaseErrPeak<sp><nv>,
MER<sp><nv>,
AmpDroop<sp><nv>,
PhaseErrPeakPo<sp><nv>

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:RES?`

Modes: VSA

Associated Web Services Methods:
```
GetSymbolRateError (VSA)
GetCarrierFrequencyError (VSA)
GetModulationSummaryData (VSA)
GetEVM (VSA)
```

[:SENSe<1|2>]:DDEMod:SRATe

Parameters: 10KHZ to 4MHZ

Parameter Form: <nv>

Default: 12.5MHZ

Description: This command sets the modulation symbol rate.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:DDEMod:SRATe?`

Example: `:DDEM:SRAT 1MHZ`

Modes: VSA

Associated Web Services Methods: `SetSymbolRate (VSA)`

[:SENSe<1|2>]:DDEMod:SRATe?

Return Parameters: 10KHZ to 4MHZ

Parameter Form: <nv>

Default: 1MHZ

Description: This command queries the modulation symbol rate.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:DDEM:SRAT?`

Modes: VSA

Associated Web Services Methods: `GetSymbolRateInHz (VSA)`

2-18 [:SENSe]:DETector Subsystem

The **[:SENSe]:DETector** subsystem contains a command for setting the trace detector modes.

Table 2-17. [:SENSe]:DETector Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
`[:SENSe<1|2>]`	-		
`:DETector<1 to 5>`	<char>	MAXIMUM|NORMAL|MINIMUM|SAMPLE|AVERAGE|RMS	
`:DETector<1 to 5>?`	<char>	MAXIMUM|NORMAL|MINIMUM|SAMPLE|AVERAGE|RMS	

[:SENSe<1|2>]:DETector<1 to 5>

Parameters: MAXIMUM | NORMAL | MINIMUM | SAMPLE | AVERAGE | RMS

Parameter Form: <char>

Default: NORMAL

Description: This command sets the detector type of a trace.

Precondition: The specified trace must be active.

Query Form: `[:SENSe<1|2>]:DETector<1 to 5>?`

Example: `:DET1 MAX`

Modes: SPA

Associated Web Services Methods: `SetTraceDetectionType (SPA)`

[:SENSe<1|2>]:DETector<1 to 5>?

Return Parameters:	MAXIMUM|NORMAL|MINIMUM|SAMPLE|AVERAGE|RMS
Parameter Form:	<char>
Default:	NORMAL
Description:	This command queries the detector type of a trace.
Precondition:	The specified trace must be active.
Query Form:	N/A
Example:	`:DET1?`
Modes:	SPA
Associated Web Services Methods:	`GetTraceDetectionType (SPA)`

2-19 [:SENSe]:FREQuency Subsystem

The **[:SENSe]:FREQuency** subsystem contains commands for setting the frequency parameters.

Table 2-18. [:SENSe]:FREQuency Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
`[:SENSe<1|2>]`	-		
`:FREQuency`	-		
`:CENTer`	<nv>	HZ|KHZ|MHZ|GHZ	
`:CENTer?`	<nv>	HZ|KHZ|MHZ|GHZ	
`:OFFSet`	<nv>	HZ|KHZ|MHZ|GHZ	
`:OFFSet?`	<nv>	HZ|KHZ|MHZ|GHZ	
`:SPAN`	<nv>	HZ|KHZ|MHZ|GHZ	
`:SPAN?`	<nv>	HZ|KHZ|MHZ|GHZ	
`:STARt`	<nv>	HZ|KHZ|MHZ|GHZ	
`:STARt?`	<nv>	HZ|KHZ|MHZ|GHZ	
`:STOP`	<nv>	HZ|KHZ|MHZ|GHZ	
`:STOP?`	<nv>	HZ|KHZ|MHZ|GHZ	

[:SENSe<1|2>]:FREQuency:CENTer

Parameters: 5HZ to 8.079999995GHZ

Parameter Form: <nv>

Default: 4GHZ

Description: This command sets the center frequency parameter.

Query Form: `[:SENSe<1|2>]:FREQuency:CENTer?`

Example: `:FREQ:CENT 1GHZ`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`SetCenterFrequency (SPA)`
`SetCenterFrequency (VSA)`
`SetCenterFrequency (WCDMA)`

[:SENSe<1|2>]:FREQuency:CENTer?

Return Parameters: 5HZ to 8.079999995GHZ

Parameter Form: <nv>

Default: 4GHZ

Description: This command queries the center frequency parameter.

Query Form: N/A

Example: `:FREQ:CENT?`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`GetCenterFrequencyInHz (SPA)`
`GetCenterFrequencyInHz (VSA)`
`GetCenterFrequencyInHz (WCDMA)`

[:SENSe<1|2>]:FREQuency:OFFSet

Parameters: –100GHZ to +100GHZ

Parameter Form: <nv>

Default: 0HZ

Description: This command sets the center frequency offset parameter.

Query Form: `[:SENSe<1|2>]:FREQuency:OFFSet?`

Example: `:FREQ:OFFS 100HZ`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods:
`SetFrequencyOffset (SPA)`
`SetFrequencyOffset (VSA)`
`SetFrequencyOffset (WCDMA)`

[:SENSe<1|2>]:FREQuency:OFFSet?

Return Parameters: –100GHZ to +100GHZ

Parameter Form: <nv>

Default: 0HZ

Description: This command queries the center frequency offset parameter.

Query Form: N/A

Example: `:FREQ:OFFS?`

Modes: SPA, VSA, WCDMA

Associated Web Services Methods: `GetFrequencyOffsetInHz (SPA)`
`GetFrequencyOffsetInHz (VSA)`
`GetFrequencyOffsetInHz (WCDMA)`

[:SENSe<1|2>]:FREQuency:SPAN

Parameters: 10HZ to 8GHZ

Parameter Form: <nv>

Default: 8GHZ

Description: This command sets the span frequency parameter.

Query Form: `[:SENSe<1|2>]:FREQuency:SPAN?`

Example: `:FREQ:SPAN 1MHZ`

Modes: SPA

Associated Web Services Methods: `SetFrequencySpan (SPA)`

[:SENSe<1|2>]:FREQuency:SPAN?

Return Parameters: 10HZ to 8GHZ

Parameter Form: <nv>

Default: 8GHZ

Description: This command queries the span frequency parameter.

Query Form: N/A

Example: `:FREQ:SPAN?`

Modes: SPA

Associated Web Services Methods: `GetFrequencySpanInHz (SPA)`

[:SENSe<1|2>]:FREQuency:STARt

Parameters: 0HZ to 8.079999990GHZ

Parameter Form: <nv>

Default: 0HZ

Description: This command sets the start frequency parameter.

Query Form: `[:SENSe<1|2>]:FREQuency:STARt?`

Example: `:FREQ:STAR 100KHZ`

Modes: SPA

Associated Web Services Methods: `SetStartFrequency (SPA)`

`[:SENSe<1|2>]:FREQuency:STARt?`

Return Parameters: 0HZ to 8.079999990GHZ

Parameter Form: <nv>

Default: 0HZ

Description: This command queries the start frequency parameter.

Query Form: N/A

Example: `:FREQ:STAR?`

Modes: SPA

Associated Web Services Methods: `GetStartFrequencyInHz (SPA)`

`[:SENSe<1|2>]:FREQuency:STOP`

Parameters: 10HZ to 8.08GHZ

Parameter Form: <nv>

Default: 8GHZ

Description: This command sets the start frequency parameter.

Query Form: `[:SENSe<1|2>]:FREQuency:STOP?`

Example: `:FREQ:STOP 10MHZ`

Modes: SPA

Associated Web Services Methods: `SetStopFrequency (SPA)`

[:SENSe<1|2>]:FREQuency:STOP?

Return Parameters:	10HZ to 8.08GHZ
Parameter Form:	<nv>
Default:	8GHZ
Description:	This command queries the stop frequency parameter.
Query Form:	N/A
Example:	`:FREQ:STOP?`
Modes:	SPA
Associated Web Services Methods:	`GetStopFrequencyInHz (SPA)`

2-20 [:SENSe]:MCP Subsystem

The **`[:SENSe]:MCP`** subsystem contains commands for setting the Mulitcarrier Channel Power measurement parameters.

Table 2-19. [:SENSe]:MCP Subsystem Commands

<table>
<tr><th>Command</th><th>Parameter Form</th><th>Character Data or Units</th><th>Notes</th></tr>
<tr><td>[:SENSe<1|2>]</td><td>-</td><td></td><td></td></tr>
<tr><td>:MCP</td><td>-</td><td></td><td></td></tr>
<tr><td>:ALT<1|2></td><td>-</td><td></td><td></td></tr>
<tr><td>:CHBandwidth</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHBandwidth?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHSPacing</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHSPacing?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:CHCount</td><td><nv></td><td>unitless</td><td></td></tr>
<tr><td>:CHCount?</td><td><nv></td><td>unitless</td><td></td></tr>
<tr><td>:FACTor</td><td>-</td><td></td><td></td></tr>
<tr><td>:ROLLoff</td><td><nv></td><td>unitless</td><td></td></tr>
<tr><td>:ROLLoff?</td><td><nv></td><td>unitless</td><td></td></tr>
<tr><td>:FFT</td><td>-</td><td></td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:FILTer</td><td>-</td><td></td><td></td></tr>
<tr><td>:RRC</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:RRC?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:HZ</td><td>-</td><td></td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:NOISecomp</td><td>-</td><td></td><td></td></tr>
<tr><td>:STATe</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:STATe?</td><td><boolean></td><td>ON|OFF</td><td></td></tr>
<tr><td>:REFChannel</td><td><char></td><td>1 to 12 + HighestPower|LowestPower</td><td></td></tr>
<tr><td>:REFChannel?</td><td><char></td><td>1 to 12 + HighestPower|LowestPower</td><td></td></tr>
<tr><td>:SRATe</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:SRATe?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:TX<1 to 12></td><td>-</td><td></td><td></td></tr>
<tr><td>:CHBandwidth</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
<tr><td>:CHBandwidth?</td><td><nv></td><td>HZ|KHZ|MHZ|GHZ</td><td></td></tr>
</table>

Table 2-19. [:SENSe]:MCP Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:CHInfo?			
:CHSPacing	<nv>	HZ|KHZ|MHZ|GHZ	
:CHSPacing?	<nv>	HZ|KHZ|MHZ|GHZ	
:TYPE	<char>	REL|ABS	
:TYPE?	<char>	REL|ABS	

[:SENSe<1|2>]:MCP:ALT<1|2>:CHBandwidth

Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command sets the specified alternate channel bandwidth parameter for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:ALT<1|2>:CHBandwidth?`

Example: `:MCP:ALT1:CHB 1MHZ`

Modes: SPA

Associated Web Services Methods: None

`[:SENSe<1|2>]:MCP:ALT<1|2>:CHBandwidth?`

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command queries the specified alternate channel bandwidth parameter for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:ALT1:CHB?`

Modes: SPA

Associated Web Services Methods: None

`[:SENSe<1|2>]:MCP:ALT<1|2>:CHSPacing`

Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 10MHZ (ALT1) or 15MHZ (ALT2)

Description: This command sets the specified alternate channel spacing parameter for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:ALT<1|2>:CHSPacing?`

Example: `:MCP:ALT1:CHSP 2MHZ`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:ALT<1|2>:CHSPacing?

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 10MHZ (ALT1) or 15MHZ (ALT2)

Description: This command queries the specified alternate channel spacing parameter for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:ALT1:CHSP?`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:ALT<1|2>:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the specified alternate channel measurement on or off for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:ALT<1|2>:STATe?`

Example: `:MCP:ALT1:STAT ON`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:ALT<1|2>:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the specified alternate channel measurement status for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode and specified alternate channel must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:ALT1:STAT?`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:CHCount

Parameters: 1 to 12

Parameter Form: <nv>

Default: 4

Description: This command sets the number of channels for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:CHCount?`

Example: `:MCP:CHC 8`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:CHCount?

Return Parameters: 1 to 12

Parameter Form: <nv>

Default: 4

Description: This command queries the number of channels for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:CHC?`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:FACTor:ROLLoff

Parameters: 0.1 to 1.0

Parameter Form: <nv>

Default: 0.5

Description: This command sets the roll-off factor (a) parameter for the mulicarrier channel power measurement.

Precondition: The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:FACTor:ROLLoff?`

Example: `:MCP:FACT:ROLL 0.8`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:FACTor:ROLLoff?

Return Parameters:	0.1 to 1.0
Parameter Form:	<nv>
Default:	0.5
Description:	This command queries the roll-off factor (a) parameter for the mulicarrier channel power measurement.
Precondition:	The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:MCP:FACT:ROLL?`
Modes:	SPA
Associated Web Services Methods:	None

[:SENSe<1|2>]:MCP:FFT:STATe

Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	This command sets the FFT state on or off for the mulicarrier channel power measurement.
Precondition:	The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:MCP:FFT:STATe?`
Example:	`:MCP:FFT:STAT ON`
Modes:	SPA
Associated Web Services Methods:	None

[:SENSe<1|2>]:MCP:FFT:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the FFT state for the mulicarrier channel power measurement.

Precondition: The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:FFT:STAT?`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:FILTer:RRC

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the RRC filter on or off for the mulicarrier channel power measurement.

Precondition: The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:FILTer:RRC?`

Example: `:MCP:FILT:RRC ON`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:FILTer:RRC?

Return Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	This command queries the RRC filter status for the mulicarrier channel power measurement.
Precondition:	The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:MCP:FILT:RRC?`
Modes:	SPA
Associated Web Services Methods:	None

[:SENSe<1|2>]:MCP:HZ:STATe

Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	This command sets the Hertz state on or off for the mulicarrier channel power measurement.
Precondition:	The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:MCP:HZ:STATe?`
Example:	`:MCP:HZ:STAT ON`
Modes:	SPA
Associated Web Services Methods:	None

[:SENSe<1|2>]:MCP:HZ:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the Hertz state for the mulicarrier channel power measurement.

Precondition: The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:HZ:STAT?`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:NOISecomp:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the noise compensation state on or off for the mulicarrier channel power measurement.

Precondition: The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:NOISecomp:STATe?`

Example: `:MCP:NOIS:STAT ON`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:NOISecomp:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the noise compensation state for the mulicarrier channel power measurement.

Precondition: The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:NOIS:STAT?`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:REFChannel

Parameters: 1 to 12 + HighestPower|LowestPower

Parameter Form: <char>

Default: 1HighestPower

Description: This command sets the reference channel number for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:REFChannel?`

Example: `:MCP:REFC 2LowestPower`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:REFChannel?

Return Parameters:	1 to 12 + HighestPower|LowestPower
Parameter Form:	<char>
Default:	1HighestPower
Description:	This command queries the reference channel number for the multicarrier channel power measurement.
Precondition:	The multicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:MCP:REFC?`
Modes:	SPA
Associated Web Services Methods:	None

[:SENSe<1|2>]:MCP:SRATe

Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	3.84MHZ
Description:	This command sets the symbol rate parameter for the mulicarrier channel power measurement.
Precondition:	The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:MCP:SRATe?`
Example:	`:MCP:SRAT 1MHZ`
Modes:	SPA
Associated Web Services Methods:	None

[:SENSe<1|2>]:MCP:SRATe?

Return Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	3.84MHZ
Description:	This command queries the symbol rate parameter for the mulicarrier channel power measurement.
Precondition:	The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:MCP:SRAT?`
Modes:	SPA
Associated Web Services Methods:	None

[:SENSe<1|2>]:MCP:TX<1 to 12>:CHBandwidth

Parameters:	1HZ to 8GHZ
Parameter Form:	<nv>
Default:	5MHZ
Description:	This command sets the specified transmit channel bandwidth parameter for the multicarrier channel power measurement.
Precondition:	The multicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:MCP:TX<1 to 12>:CHBandwidth?`
Example:	`:MCP:TX2:CHB 2MHZ`
Modes:	SPA
Associated Web Services Methods:	None

`[:SENSe<1|2>]:MCP:TX<1 to 12>:CHBandwidth?`

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 5MHZ

Description: This command queries the specified transmit channel bandwidth parameter for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:TX2:CHB?`

Modes: SPA

Associated Web Services Methods: None

`[:SENSe<1|2>]:MCP:TX<1 to 12>:CHSPacing`

Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 10MHZ

Description: This command sets the specified transmit channel spacing parameter for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:MCP:TX<1 to 12>:CHSPacing?`

Example: `:MCP:TX2:CHSP 5MHZ`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:TX<1 to 12>:CHSPacing?

Return Parameters: 1HZ to 8GHZ

Parameter Form: <nv>

Default: 10MHZ

Description: This command queries the specified transmit channel spacing parameter for the multicarrier channel power measurement.

Precondition: The multicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:TX2:CHSP?`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:TYPE

Parameters: REL|ABS

Parameter Form: <char>

Default: REL

Description: This command sets the mulicarrier channel power measurement type of relative or absolute.

Query Form: `[:SENSe<1|2>]:MCP:TYPE?`

Example: `:MCP:TYPE ABS`

Modes: SPA

Associated Web Services Methods: None

[:SENSe<1|2>]:MCP:TYPE?

Return Parameters: REL|ABS

Parameter Form: <char>

Default: REL

Description: This command queries the mulicarrier channel power measurement type.

Precondition: The mulicarrier channel power measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:MCP:TYPE?`

Modes: SPA

Associated Web Services Methods: None

2-21 [:SENSe]:OBW Subsystem

The **[:SENSe]:OBW** subsystem contains commands for setting the occupied bandwidth parameters.

Table 2-20. [:SENSe]:OBW Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
[:SENSe<1|2>]	-		
:OBW	-		
:POWer	-		
:PERCent	<nv>	unitless	
:PERCent?	<nv>	unitless	
:XDBS	<nv>	DB	
:XDBS?	<nv>	DB	

[:SENSe<1|2>]:OBW:POWer:PERCent

Parameters: 10 to 99.9

Parameter Form: <nv>

Default: 99

Description: This command sets the percentage power limits for the occupied bandwidth measurement.

Precondition: The occupied bandwidth measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[SENSe<1|2>]:OBW:POWer:PERCent?`

Example: `:OBW:POW:PERC 50`

Modes: SPA

Associated Web Services Methods: `SetOBWPercentagePower (SPA)`

[:SENSe<1|2>]:OBW:POWer:PERCent?

Return Parameters: 10 to 99.9

Parameter Form: <nv>

Default: 99

Description: This command queries the percentage power limits for the occupied bandwidth measurement.

Precondition: The occupied bandwidth measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:OBW:POW:PERC?`

Modes: SPA

Associated Web Services Methods: `GetOBWPercentagePower (SPA)`

[:SENSe<1|2>]:OBW:XDBS

Parameters: 0.1 to 100

Parameter Form: <nv>

Default: 26

Description: This command sets the X dB limits for the occupied bandwidth measurement.

Precondition: The occupied bandwidth measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[SENSe<1|2>]:OBW:XDBS?`

Example: `:OBW:XDBS 30DB`

Modes: SPA

Associated Web Services Methods: `SetOBWBandwidth (SPA)`

[:SENSe<1|2>]:OBW:XDBS?

Return Parameters: 0.1 to 100

Parameter Form: <nv>

Default: 26

Description: This command queries the X dB limits for the occupied bandwidth measurement.

Precondition: The occupied bandwidth measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:OBW:XDBS?`

Modes: SPA

Associated Web Services Methods: `GetOBWBandwidthIndB (SPA)`

2-22 [:SENSe]:ROSCillator Subsystem

The **[:SENSe]:ROSCillator** subsystem contains a command for controlling the reference signal source.

Table 2-21. [:SENSe]:ROSCillator Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
[:SENSe<1|2>]	-		
:ROSCillator	-		
:EXT	-		
:STATe?	<char>	LOCK|UNLOCK	query only
:SOURce	<char>	INTernal|EXTernal	
:SOURce?	<char>	INTernal|EXTernal	

[:SENSe<1|2>]:ROSCillator:EXT:STATe?

Return Parameters: LOCK|UNLOCK

Parameter Form: <char>

Default: None. This command is a query, which is why it has no default parameters.

Description: This command queries the external reference source state.

Query Form: N/A

Example: `:ROSC:EXT:STAT?`

Modes: VSA

Associated Web Services Methods: `GetLockStatus (VSA)`

[:SENSe<1|2>]:ROSCillator:SOURce

Parameters:	INTernal|EXTernal
Parameter Form:	<char>
Default:	INTernal
Description:	This command sets the reference source to either internal or external.
Query Form:	`[:SENSe<1|2>]:ROSCillator:SOURce?`
Example:	`:ROSC:SOUR EXT`
Modes:	SYS
Associated Web Services Methods:	`SetReferenceType (SYS)`

[:SENSe<1|2>]:ROSCillator:SOURce?

Return Parameters:	INTernal|EXTernal
Parameter Form:	<char>
Default:	INTernal
Description:	This command queries the reference source setting of either internal or external.
Query Form:	`[:SENSe<1|2>]:ROSCillator:SOURce?`
Example:	`:ROSC:SOUR?`
Modes:	SYS
Associated Web Services Methods:	`GetReferenceType (SYS)`

2-23 [:SENSe]:SWEep Subsystem

The **[:SENSe]:SWEep** subsystem contains a command for setting the sweep time parameter and the sweep mode.

Table 2-22. [:SENSe]:SWEep Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
[:SENSe<1|2>]	-		
:SWEep	-		
:AVERage	<nv>	unitless	
:AVERage?	<nv>	unitless	
:COUNt?	<nv>	unitless	query only
:TIME	<nv>	MS|S|KS	
:AUTO	<boolean>	ON|OFF	
:COUPling	<char>	SPEed|ACCuracy	
:COUPling?	<char>	SPEed|ACCuracy	
:AUTO?	<boolean>	ON|OFF	
:TIME?	<nv>	MS|S|KS	

[:SENSe<1|2>]:SWEep:AVERage

Parameters: 1 to 10000

Parameter Form: <nv>

Default: 5

Description: This command sets the sweep averaging parameter.

Query Form: `[:SENSe<1|2>]:SWEep:AVERage?`

Example: `:SWE:AVER 5`

Modes: SPA

Associated Web Services Methods: `SetNumberOfAverages (SPA)`

[:SENSe<1|2>]:SWEep:AVERage?

Return Parameters:	1 to 10000
Parameter Form:	<nv>
Default:	5
Description:	This command queries the sweep averaging parameter.
Query Form:	N/A
Example:	`:SWE:AVER?`
Modes:	SPA
Associated Web Services Methods:	`GetNumberOfAverages (SPA)`

[:SENSe<1|2>]:SWEep:COUNt?

Return Parameters:	1 to 10000
Parameter Form:	<nv>
Default:	None. This command is a query, which is why it does not have a default value.
Description:	This command queries the current sweep count when in averaging mode.
Query Form:	N/A
Example:	`:SWE:COUN?`
Modes:	SPA
Associated Web Services Methods:	`GetSweepCount (SPA)`

[:SENSe<1|2>]:SWEep:TIME

Parameters: 5MS to 10KS (0.1MS to 10KS in zero span mode)

Parameter Form: <nv>

Default: 16MS

Description: This command sets the sweep time parameter. This command is not valid for FFT sweep modes.

Query Form: `[:SENSe<1|2>]:SWEep:TIME?`

Example: `:SWE:TIME 1S`

Modes: SPA

Associated Web Services Methods: `SetSweepTime (SPA)`

[:SENSe<1|2>]:SWEep:TIME:AUTO

Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command sets the sweep time mode.
Auto = k*Span/(RBW)^2 - if VBW is >= RBW or
Auto = k*Span/(VBW*RBW) if VBW < RBW where:
k (k-factor) = 1.8 for auto-speed mode
k (k-factor) = 5 for auto-accuracy mode

Query Form: `[:SENSe<1|2>]:SWEep:TIME:AUTO?`

Example: `:SWE:TIME:AUTO OFF`

Modes: SPA

Associated Web Services Methods: `SetSweepTimeAuto (SPA)`

`[:SENSe<1|2>]:SWEep:TIME:AUTO:COUPling`

Parameters:	SPEed|ACCuracy
Parameter Form:	<char>
Default:	SPEed
Description:	This command toggles the sweep time mode between speed or accuracy.
Query Form:	`[:SENSe<1|2>]:SWEep:TIME:AUTO:COUPling?`
Example:	`:SWE:TIME:AUTO:COUP SPE`
Modes:	SPA
Associated Web Services Methods:	`SetSweepTimeMode (SPA)`

`[:SENSe<1|2>]:SWEep:TIME:AUTO:COUPling?`

Return Parameters:	SPEed|ACCuracy
Parameter Form:	<char>
Default:	SPEed
Description:	This command queries the sweep time mode.
Query Form:	N/A
Example:	`:SWE:TIME:AUTO:COUP?`
Modes:	SPA
Associated Web Services Methods:	`GetSweepTimeMode (SPA)`

[:SENSe<1|2>]:SWEep:TIME:AUTO?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command queries the sweep time mode.
Auto = k*Span/(RBW)^2 - if VBW is >= RBW or
Auto = k*Span/(VBW*RBW) if VBW < RBW where:
k (k-factor) = 1.8 for auto-speed mode
k (k-factor) = 5 for auto-accuracy mode

Query Form: N/A

Example: `:SWE:TIME:AUTO?`

Modes: SPA

Associated Web Services Methods: `GetSweepTimeAuto (SPA)`

[:SENSe<1|2>]:SWEep:TIME?

Return Parameters: 5MS to 10KS (0.1MS to 10KS in zero span mode)

Parameter Form: <nv>

Default: 16MS

Description: This command queries the sweep time parameter. This command is not valid for FFT sweep modes.

Query Form: N/A

Example: `:SWE:TIME?`

Modes: SPA

Associated Web Services Methods: `GetSweepTimeInSecs (SPA)`

2-24 [:SENSe]:TCAPture Subsystem

The **[:SENSe]:TCAPture** subsystem contains commands for setting the modulation capture time.

Table 2-23. [:SENSe]:TCAPture Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
[:SENSe<1|2>]	-		
:TCAPture	-		
:LENGth	<nv>	unitless	VSA only
:LENGth?	<nv>	unitless	VSA only

[:SENSe<1|2>]:TCAPture:LENGth

Parameters: 100 to 10000
Constrained by the following:
10,000 ≥ (Symbol Rate x Capture Time) ≥ 100

Parameter Form: <nv>

Default: 1000

Description: This command sets the symbol capture number.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:TCAPture:LENGth?`

Example: `:TCAP:LENG 200`

Modes: VSA

Associated Web Services Methods: `SetCaptureTime (VSA)`
`SetSymbolRate (VSA)`

[:SENSe<1|2>]:TCAPture:LENGth?

Return Parameters: 10,000 to 100
Constrained by the following:
10,000 ≥ (Symbol Rate x Capture Time) ≥ 100

Parameter Form: <nv>

Default: 1000

Description: This command queries the symbol capture number.

Precondition: The modulation measurement mode must be activated via GPIB. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:TCAP:LENG?`

Modes: VSA

Associated Web Services Methods: `GetCaptureTimeInSecs (VSA)`
`GetSymbolRateInHz (VSA)`

2-25 [:SENSe]:WCDMa Subsystem

The **[:SENSe]:WCDMa** subsystem contains commands for setting the wideband code division multiple access measurement parameters and modes.

Table 2-24. [:SENSe]:WCDMa Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
[:SENSe<1|2>]	-		
:WCDMa	-		
:ACQUisition	-		
:ANAlysis	-		
:LENgth	<char>	<nv>FRAMes|SLOTs|CHIPs|SECs|MSECs|USECs|NSECs	
:LENgth?	<char>	<nv>FRAMes|SLOTs|CHIPs|SECs|MSECs|USECs|NSECs	
:STARt	<char>	<nv>FRAMes|SLOTs|CHIPs|SECs|MSECs|USECs|NSECs	
:STARt?	<char>	<nv>FRAMes|SLOTs|CHIPs|SECs|MSECs|USECs|NSECs	
:CAPTure	-		
:LENgth	<char>	<1 to 8>FRAMes	
:LENgth?	<char>	<1 to 8>FRAMes	
:INVersion	<char>	NORMal|INVert	
:INVersion?	<char>	NORMal|INVert	
:RRCFilter	-		
:STATe	<boolean>	ON|OFF	
:STATe?	<boolean>	ON|OFF	
:DISPlay	-		

Table 2-24. [:SENSe]:WCDMa Subsystem Commands

<table>
<tr><th>Command</th><th>Parameter Form</th><th>Character Data or Units</th><th>Notes</th></tr>
<tr><td>:FORMat</td><td><char></td><td>QPVSummary|QPVector|
QPCSummary|QPConstellation|
QPPTSummary|QPPTime|
QPETSummary|QPETime|
QPMETSummary|QPMETime|
QPPETSummary|QPPETime|
QPAETSummary|QPAETime|
QPISummary|QPIeye|
QPQSummary|QPQeye|
QPIQSummary|QPIQeye|
CPVSummary|CPVector|
CPConstellation|CPPTSummary|
CPPTime|CPETS|CPETime|
CPMETSummary|CPMETime|
CPPETSummary|CPPETime|
CPAETSummary|CPAETime|
CPISummary|CPI|
CPQSummary|CPQeye|
CPIQSummary|CPIQeye|
CDPOwer|CDPSUmmary|
CDPZOom|CDPZSummary|
CDERror|CDESummary|
CDEZOom|CDEZSummary|
CDVSingle|CDVEctor|CDCSingle|
CDCOnstellation|CDSSingle|
CDSUmmary|PVSSingle|PVSLot|
EVSSingle|EVSLot|EVTSingle|
EVTIme|MEVTSingle|MEVTIme|
PEVTSingle|PEVTIme|
AEVTSingle|CDISingle|CDIEye|
CDQSingle|CDQEye|CDIQSingle|
CDQEye|CDIQSingle|CDIQEye</td><td></td></tr>
<tr><td>:FORMat?</td><td><char></td><td>same as above</td><td></td></tr>
<tr><td>:DMODulation</td><td>-</td><td></td><td></td></tr>
<tr><td>:ACTivechannels</td><td>-</td><td></td><td></td></tr>
<tr><td>:THReshold</td><td><char></td><td><nv>DB</td><td></td></tr>
<tr><td>:THReshold?</td><td><char></td><td><nv>DB</td><td></td></tr>
<tr><td>[:CODE]</td><td><char></td><td>AUTO|MANUal| TESTModel</td><td></td></tr>
<tr><td>[:CODE]?</td><td><char></td><td>AUTO|MANUAL| TESTMODEL</td><td></td></tr>
<tr><td>:COMPressedmode</td><td>-</td><td></td><td></td></tr>
<tr><td>:CODEchannel</td><td><nv></td><td>1 to 12</td><td></td></tr>
<tr><td>:CODEchannel?</td><td><nv></td><td>1 to 12</td><td></td></tr>
<tr><td>:SPFactor</td><td><char></td><td>1|2|4|8|16|32|64|128|256|512</td><td></td></tr>
<tr><td>:SPFactor?</td><td><char></td><td>1|2|4|8|16|32|64|128|256|512</td><td></td></tr>
<tr><td>[:MODE]</td><td><char></td><td>OFF|AUTO|MANual</td><td></td></tr>
<tr><td>[:MODE]?</td><td><char></td><td>OFF|AUTO|MANual</td><td></td></tr>
</table>

Table 2-24. [:SENSe]:WCDMa Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:MAXSpreadfactor	<nv>	256|512	
:MAXSpreadfactor?	<nv>	256|512	
:ROTAtion	<nv>	0|45	
:ROTAtion?	<nv>	0|45	
:SCRAmblingcode	<nv>	unitless	
:AUTO	<boolean>	ON|OFF	
:AUTO?	<boolean>	ON|OFF	
:COMPressedchannel	<char>	ORDinary|LEFTalternate|RIGHtalternate	
:COMPressedchannel?	<char>	ORD|LEFT|RIGHT	
:TYPE	<char>	LONG|SHORt	
:TYPE?	<char>	LONG|SHORT	
:SCRAmblingcode?	<nv>		
:SYNCreference	-		
:CODEchannel	<nv>	unitless	
:CODEchannel?	<nv>	unitless	
:SPFactor	<char>	1,2,4,8,…,512|MAX	
:SPFactor?	<char>	1,2,4,8,…,512	
[:REFerence]	<char>	PCPIch|MANual	
[:REFerence]?	<char>	PCPICH|MANUAL	
:TRANsmitdiversity	-		
[:ANTenna]	<char>	OFF|ANT1|ANT2	
[:ANTenna]?	<char>	OFF|ANT1|ANT2	
:TYPE	<boolean>	ON|OFF	
:TYPE?	<boolean>	ON|OFF	
:LINK	-		
:STATe	<boolean>	ON|OFF	
:STATe?	<boolean>	ON|OFF	
:MARKer<1|2>	-		
:CDPError?	<nv>	unitless	query only
:CDPLevel?	<nv>	unitless	query only
:POSition	<nv>	unitless	
:POSition?	<nv>	unitless	
:MODanalysis	-		
:CDPower	-		
:REFerence	<char>	ABSolute|RELative	
:REFerence?	<char>	ABSolute|RELative	
:RACH	-		
:ANAlysis	-		
[:LENgth]	<char>	<nv>USEC	

Table 2-24. [:SENSe]:WCDMa Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:LEN?	<char>	<nv>USEC	
:CAPTure	-		
[:LENgth]	<char>	<nv>MSEC	
:LEN?	<char>	<nv>MSEC	
:SUMMary?	<char> input <nv> output	PreamblesDisplayed| PreambleSignature|TimeOffset| OnOffRatio| FilteredPowerOn| FilteredPowerOff_Leading| FilteredPowerOffTrailing| FilteredPowerOffAverage	query only
:SCH	-		
:DISPlay	-		
:FORMat	<char>	BINARY|HEX	
:FORMat?	<char>	BINARY|HEX	
:SLOTnumber	-		
:DATAbits	<nv>	unitless	
:DATAbits?	<nv>	unitless	
:SUMMary	<char> input <nv> output	SCHState| SecondarySynchronizationCode| ScramblingCodeGroup| ScramblingCodeNumber| TotalSCHPowerAbsolute| PSCHPowerAbsolute| SSCHPowerAbsolute| TotalSCHPowerRelative| PSCHPowerRelative| SSCHPowerRelative| TransmitPower	query only
:SUMMary?	<char> input <nv> output	FERR|RHO|EVM|MCEVM|PEVM| PHERR|AERR|IQOFF|SCODE| SPWR|PSPWR|SSPWR| PCDE	query only
:TPC	-		
:PICH	-		
:AVERage	-		
:TIMeperslot	<char>	<nv>USEC	
:TIMeperslot?	<char>	<nv>USEC	
:COMand	-		
:PATTern	<nv>	unitless	
:PATTern?	<nv>	unitless	
:LIMitcheck	-		
:STATe	<boolean>	ON | OFF	
:STAT?	<boolean>	ON | OFF	

Table 2-24. [:SENSe]:WCDMa Subsystem Commands

Command	Parameter Form	Character Data or Units	Notes
:PICH or :PICHcompensation	-		
:CHANnelnumber	<nv>	unitless	
:CHANnelnumber?	<nv>	unitless	
:TIMIngOffset	<nv>	unitless	
:TIMIngOffset?	<nv>	unitless	
:STATe	<boolean>	ON | OFF	
:STAT?	<boolean>	ON | OFF	
:SCHCompensation	-		
:STATe	<boolean>	ON | OFF	
:STAT?	<boolean>	ON | OFF	
:STEP	-		
:SIZe	<char>	<nv>DB	
:SIZe?	<char>	<nv>DB	
:TIMe	-		
[:RANge]	<char>	<nv>FRAME | <nv>SLOT	
:RAN?	<char>	<nv>FRAME | <nv>SLOT	

[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:LENgth

Parameters: <nv>FRAMes | SLOTs | CHIPs | SECs | MSECs | USECs | NSECs

Parameter Form: <char>

Default: 2304CHIP

Description: This command sets the WCDMA analysis length.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:LENgth?`

Example: `:WCDM:ACQU:ANA:LEN 1280CHIPs`

Modes: WCDMA

Associated Web Services Methods: `SetAnalysisLength (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:LENgth?

Return Parameters: <nv>FRAMes|SLOTs|CHIPs|SECs|MSECs|USECs|NSECs

Parameter Form: <char>

Default: 2304CHIP

Description: This command queries the WCDMA analysis length.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:ACQU:ANA:LEN?`

Modes: WCDMA

Associated Web Services Methods: `GetAnalysisLength (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:STARt

Parameters: <nv>FRAMes|SLOTs|CHIPs|SECs|MSECs|USECs|NSECs

Parameter Form: <char>

Default: 256CHIP

Description: This command sets the WCDMA analysis start time.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:STARt?`

Example: `:WCDM:ACQU:ANA:STAR 256CHIP`

Modes: WCDMA

Associated Web Services Methods: `SetAnalysisStart (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:STARt?

Return Parameters:	<nv>FRAMes|SLOTs|CHIPs|SECs|MSECs|USECs|NSECs
Parameter Form:	<char>
Default:	256CHIP
Description:	This command queries the WCDMA analysis start time.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:ACQU:ANA:STAR?`
Modes:	WCDMA
Associated Web Services Methods:	`GetAnalysisStart (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:CAPTure:LENgth

Parameters:	<1 to 8>FRAM
Parameter Form:	<char>
Default:	1FRAM
Description:	This command sets the WCDMA analysis capture length.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:WCDMa:ACQUisition:CAPTure:LENgth?`
Example:	`:WCDM:ACQU:CAPT:LEN 8FRAM`
Modes:	WCDMA
Associated Web Services Methods:	`SetCaptureLength (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:CAPTure:LENgth?

Return Parameters: <1 to 8>FRAM

Parameter Form: <char>

Default: 1FRAM

Description: This command queries the WCDMA analysis capture length.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:ACQU:CAPT:LEN?`

Modes: WCDMA

Associated Web Services Methods: `GetCaptureLength (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:INVersion

Parameters: NORMal|INVert

Parameter Form: <char>

Default: NORM

Description: This command sets the WCDMA acquisition setting.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:ACQUisition:INVersion?`

Example: `:WCDM:ACQU:INV INV`

Modes: WCDMA

Associated Web Services Methods: `SetSpectrumInversionMode (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:INVersion?

Return Parameters: NORMal|INVert

Parameter Form: <char>

Default: NORM

Description: This command queries the WCDMA acquisition setting.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:ACQU:INV?`

Modes: WCDMA

Associated Web Services Methods: `GetSpectrumInversionMode (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:RRCFilter:STATe

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the root raised cosine filter state.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:ACQUisition:RRCFilter:STATe?`

Example: `:WCDM:ACQU:RRCF:STAT ON`

Modes: WCDMA

Associated Web Services Methods: `SetRRCFilter (WCDMA)`

[:SENSe<1|2>]:WCDMa:ACQUisition:RRCFilter:STATe?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command queries the root raised cosine filter state.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:ACQU:RRCF:STAT?`

Modes: WCDMA

Associated Web Services Methods: `GetRRCFilter (WCDMA)`

[:SENSe<1|2>]:WCDMa:DISPlay:FORMat

Parameters: QPVSummary | QPVector | QPCSummary | QPConstellation | QPPTSummary | QPPTime | QPETSummary | QPETime | QPMETSummary | QPMETime QPPETSummary | QPPETime | QPAETSummary | QPAETime | QPISummary | QPIeye | QPQSummary | QPQeye | QPIQSummary | QPIQeye | CPVSummary | CPVector | CPConstellationCPPTSummary | CPPTime | CPETS | CPETime | CPMETSummary | CPMETime | CPPETSummary | CPPETime | CPAETSummary | CPAETime | CPISummary | CPI | CPQSummary | CPQeye | CPIQSummary | CPIQeye | CDPOwer | CDPSUmmary | CDPZOom | CDPZSummary | CDERror | CDESummary | CDEZOom | CDEZSummary | CDVSingle | CDVEctor | CDCSingle | CDCOnstellation | CDSSingle | CDSUmmary | PVSSingle | PVSLot | EVSSingle | EVSLot | EVTSingle | EVTIme | MEVTSingle | MEVTIme | PEVTSingle | PEVTIme | AEVTSingle | CDISingle | CDIEye | CDQSingle | CDQEye | CDIQSingle | CDQEye | CDIQSingle | CDIQEye

Parameter Form: <char>

Default: CDPOwer

Description: This command sets the WCDMA display graph type.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DISPlay:FORMat?`

Example: `:WCDM:DISP:FORM TPT`

Modes: WCDMA

Associated Web Services Methods: `SetGraphType (WCDMA)`

[:SENSe<1|2>]:WCDMa:DISPlay:FORMat?

Return Parameters: QPVSummary | QPVector | QPCSummary | QPConstellation | QPPTSummary | QPPTime | QPETSummary | QPETime | QPMETSummary | QPMETime | QPPETSummary | QPPETime | QPAETSummary | QPAETime | QPISummary | QPIeye | QPQSummary | QPQeye | QPIQSummary | QPIQeye | CPVSummary | CPVector | CPConstellationCPPTSummary | CPPTime | CPETS | CPETime | CPMETSummary | CPMETime | CPPETSummary | CPPETime | CPAETSummary | CPAETime | CPISummary | CPI | CPQSummary | CPQeye | CPIQSummary | CPIQeye | CDPOwer | CDPSUmmary | CDPZOom | CDPZSummary | CDERror | CDESummary | CDEZOom | CDEZSummary | CDVSingle | CDVEctor | CDCSingle | CDCOnstellation | CDSSingle | CDSUmmary | PVSSingle | PVSLot | EVSSingle | EVSLot | EVTSingle | EVTIme | MEVTSingle | MEVTIme | PEVTSingle | PEVTIme | AEVTSingle | CDISingle | CDIEye | CDQSingle | CDQEye | CDIQSingle | CDQEye | CDIQSingle | CDIQEye

Parameter Form: <char>

Description: This command queries the WCDMA display graph type.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DISP:FORM?`

Modes: WCDMA

Associated Web Services Methods: None

[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels:THReshold

Parameters: <nv>DB

Parameter Form: <char>

Default: –33DB

Description: This command sets the WCDMA threshold level. Ranges from –50 dB to –10 dB with a resolution of 1 dB.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels:THReshold?`

Example: `:WCDM:DMOD:ACT:THR –35DB`

Modes: WCDMA

Associated Web Services Methods: `SetActiveChannelThreshold (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels:THReshold?

Return Parameters: <nv>DB

Parameter Form: <char>

Default: –33DB

Description: This command queries the WCDMA threshold level.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:ACT:THRe?`

Modes: WCDMA

Associated Web Services Methods: `GetActiveChannelThresholdIndB (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels[:CODE]

Parameters: AUTO|MANual <PathSpecifier>|TESTModel <TestModelSpecifier>

Parameter Form: <char>

Default: AUTO

Description: This command sets the WCDMA active channel code.

When MANual is selected, the path specifier includes the qualified path to the .tm file with the colon character (:) replaced with the backslash character (\).

When TESTmodel is selected, the TestModelSpecifier includes the name of the WCDMA test model desired. Allowable TestModelSpecifiers are:

TM1_16DPCH|TM1_32DPCH|TM1_64DPCH|TM2|TM3_16DPCH|TM3_32DPCH|TM4_CPICH|TM4_NO_CPICH|TM5_2HSPDSCH|TM5_4HSPDSCH|TM5_8HSPDSCH

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels[:CODE]?`

Examples:

```
:WCDM:DMOD:ACT MANual
C\\ChannelTables\TestModel1_16DPCH.tm
:WCDM:DMOD:ACT TEST TM1_64DPCH
:WCDM:DMOD:ACT AUTO
:WCDM:DMOD:ACT LAST
```

Modes: WCDMA

Associated Web Services Methods: `SetActiveCodeChannelType (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels[:CODE]?

Return Parameters: AUTO|MANual <PathSpecifier>|TESTModel <TestModelSpecifier>

Parameter Form: <char>

Default: AUTO

Description: This command queries the WCDMA active channel code.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:ACT?`

Modes: WCDMA

Associated Web Services Methods: `GetActiveCodeChannelType (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode:CODEchannel

Parameters: 0 to n
where n = Compressed mode Spreadfactor – 1

Parameter Form: <nv>

Default: 1

Description: This command sets the WCDMA compressed code channel number.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode:CODEchannel?`

Example: `:WCDM:DMOD:COMP:CODE 2`

Modes: WCDMA

Associated Web Services Methods: None

[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode:CODEchannel?

Return Parameters: 0 to n
where n = Compressed mode Spreadfactor – 1

Parameter Form: <nv>

Default: 1

Description: This command queries the WCDMA compressed code channel number.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:COMP:CODE?`

Modes: WCDMA

Associated Web Services Methods: None

[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode:SPFactor

Parameters: 1|2|4|8|16|32|64|128|256|MAX

Parameter Form: <char>

Default: 1

Description: This command sets the WCDMA compressed mode spreading factor.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode:SPFactor?`

Example: `:WCDM:DMOD:COMP:SPF 128`

Modes: WCDMA

Associated Web Services Methods: None

[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode:SPFactor?

Return Parameters: 1|2|4|8|16|32|64|128|256|MAX

Parameter Form: <char>

Default: 1

Description: This command queries the WCDMA compressed mode spreading factor.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:COMP:SPF?`

Modes: WCDMA

Associated Web Services Methods: None

[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode[:MODE]

Parameters: OFF|AUTO|MANual

Parameter Form: <char>

Default: OFF

Description: This command sets the WCDMA compressed mode.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode[:MODE]?`

Example: `:WCDM:DMOD:COMP MAN`

Modes: WCDMA

Associated Web Services Methods: `SetDisplayCompressedModeSignalsMode (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode[:MODE]?

Return Parameters: OFF|AUTO|MANual

Parameter Form: <char>

Default: OFF

Description: This command queries the WCDMA compressed mode.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:COMP?`

Modes: WCDMA

Associated Web Services Methods: `GetDisplayCompressedModeSignalsMode (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:MAXSpreadfactor

Parameters: 256|512

Parameter Form: <nv>

Default: 256

Description: This command sets the WCDMA maximum spreading factor.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:MAXSpreadfactor?`

Example: `:WCDM:DMOD:MAXS 512`

Modes: WCDMA

Associated Web Services Methods: `SetMaxSpreadFactor (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:MAXSpreadfactor?

Return Parameters:	256|512
Parameter Form:	<nv>
Default:	256
Description:	This command queries the WCDMA maximum spreading factor.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:DMOD:MAXS?`
Modes:	WCDMA
Associated Web Services Methods:	`GetMaxSpreadFactor (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:ROTAtion

Parameters:	0|45
Parameter Form:	<nv>
Default:	0
Description:	This command sets the WCDMA display rotation.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:WCDMa:DMODulation:ROTAtion?`
Example:	`:WCDM:DMOD:ROTA 45`
Modes:	WCDMA
Associated Web Services Methods:	`SetIQDisplayRotation (WCDMA)`

`[:SENSe<1|2>]:WCDMa:DMODulation:ROTAtion?`

Return Parameters: 0|45

Parameter Form: <nv>

Default: 0

Description: This command queries the WCDMA display rotation.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:ROTA?`

Modes: WCDMA

Associated Web Services Methods: `GetIQDisplayRotation (WCDMA)`

`[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode`

Parameters: 0 to 3FFFF

Parameter Form: <nv>

Default: 3FFFF

Description: This command sets the WCDMA scrambling code.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode?`

Example: `:WCDM:DMOD:SCRA 3FFF0`

Modes: WCDMA

Associated Web Services Methods: `SetScramblingCode (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode?

Return Parameters:	
Parameter Form:	<nv>
Default:	3FFFF
Description:	This command queries the WCDMA scrambling code.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:DMOD:SCRA?`
Modes:	WCDMA
Associated Web Services Methods:	`GetScramblingCode (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:AUTO

Parameters:	ON|OFF
Parameter Form:	<boolean>
Default:	ON
Description:	This command sets the WCDMA scrambling code mode.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:AUTO?`
Example:	`:WCDM:DMOD:SCRA:AUTO OFF`
Modes:	WCDMA
Associated Web Services Methods:	`SetScramblingCodeMode (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:AUTO?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: ON

Description: This command queries the WCDMA scrambling code mode.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:SCRA:AUTO?`

Modes: WCDMA

Associated Web Services Methods: `GetScramblingCodeMode (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:COMPressedchannel

Parameters: ORDinary|LEFTalternate|RIGHtalternate

Parameter Form: <char>

Default: ORD

Description: This command sets the WCDMA scrambling code compressed channel number.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:COMPressedchannel?`

Example: `:WCDM:DMOD:SCRA:COMP LEFT`

Modes: WCDMA

Associated Web Services Methods: `SetScramblingCodeForCompressedChannel (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:COMPressedchannel?

Return Parameters:	ORD|LEFT|RIGH
Parameter Form:	<char>
Default:	ORD
Description:	This command queries the WCDMA scrambling code compressed channel number.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:DMOD:SCRA:COMP?`
Modes:	WCDMA
Associated Web Services Methods:	`GetScramblingCodeForCompressedChannel (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:TYPE

Parameters:	LONG|SHORt
Parameter Form:	<char>
Default:	LONG
Description:	This command sets the WCDMA scrambling code type.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:TYPE?`
Example:	`:WCDM:DMOD:SCRA:TYPE SHOR`
Modes:	WCDMA
Associated Web Services Methods:	None

[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:TYPE?

Return Parameters: LONG|SHORT

Parameter Form: <char>

Default: LONG

Description: This command queries the WCDMA scrambling code type.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:SCRA:TYPE?`

Modes: WCDMA

Associated Web Services Methods: None

[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference:CODEchannel

Parameters: 0 to n
where n = Syncreference Spreadfactor – 1

Parameter Form: <nv>

Default: 0

Description: This command sets the WCDMA synchronization reference code channel number.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference:CODEchannel?`

Example: `:WCDM:DMOD:SYNC:CODE 2`

Modes: WCDMA

Associated Web Services Methods: None

[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference:CODEchannel?

Return Parameters:	0 to n where n = Syncreference Spreadfactor – 1
Parameter Form:	<nv>
Default:	0
Description:	This command queries the WCDMA synchronization reference code channel number.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:DMOD:SYNC:CODE?`
Modes:	WCDMA
Associated Web Services Methods:	None

[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference:SPFactor

Parameters:	4|8|16|32|64|128|256|512
Parameter Form:	<char>
Default:	128
Description:	This command sets the WCDMA synchronization reference spreading factor.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference:SPFactor?`
Example:	`:WCDM:DMOD:SYNC:SPF 128`
Modes:	WCDMA
Associated Web Services Methods:	`SetSyncReferenceDownlinkManual (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference:SPFactor?

Return Parameters:	4|8|16|32|64|128|256|512
Parameter Form:	<char>
Default:	128
Description:	This command queries the WCDMA synchronization reference spreading factor.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:DMOD:SYNC:SPF?`
Modes:	WCDMA
Associated Web Services Methods:	`GetSyncReferenceDownlinkManual (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference[:REFerence]

Parameters:	PCPIch|MANual
Parameter Form:	<char>
Default:	PCPI
Description:	This command sets the WCDMA synchronization reference.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	`[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference[:REFerence]?`
Example:	`:WCDM:DMOD:SYNC MAN`
Modes:	WCDMA
Associated Web Services Methods:	`SetSyncReferenceDownlink (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference[:REFerence]?

Return Parameters: PCPICH|MANUAL

Parameter Form: <char>

Default: PCPICH

Description: This command queries the WCDMA synchronization reference.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:SYNC?`

Modes: WCDMA

Associated Web Services Methods: `GetSyncReferenceDownlink (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity[:ANTenna]

Parameters: OFF|ANT1|ANT2

Parameter Form: <char>

Default: OFF

Description: This command sets the WCDMA transmit diversity.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity[:ANTenna]?`

Example: `:WCDM:DMOD:TRAN ANT1`

Modes: WCDMA

Associated Web Services Methods: `SetTransmitDiversity (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity[:ANTenna]?

Return Parameters: OFF|ANT1|ANT2

Parameter Form: <char>

Default: OFF

Description: This command queries the WCDMA transmit diversity.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:TRAN?`

Modes: WCDMA

Associated Web Services Methods: `GetTransmitDiversity (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity:TYPE

Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command sets the WCDMA transmit diversity type.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity:TYPE ?`

Example: `:WCDM:DMOD:TRAN:TYPE ON`

Modes: WCDMA

Associated Web Services Methods: `SetTransmitDiversityType (WCDMA)`

[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity:TYPE?

Return Parameters: ON|OFF

Parameter Form: <boolean>

Default: OFF

Description: This command queries the WCDMA transmit diversity type.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:DMOD:TRAN:TYPE?`

Modes: WCDMA

Associated Web Services Methods: `GetTransmitDiversityType (WCDMA)`

[:SENSe<1|2>]:WCDMa:LINK:STATe

Parameters: ON | OFF

Parameter Form: <boolean>

Default: OFF

Description: Sets measurement type to Downlink or Uplink for WCDMA measurement.

Query Form: `[:SENSe<1|2>]:WCDMa:LINK:STATe?`

Example: `:WCDM:LINK:STAT ON`

Modes: WCDMA

Associated Web Service: `SetDownlinkUplink (WCDMA)`

[:SENSe<1|2>]:WCDMa:LINK:STATe?

Return Parameters:	ON | OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	gets measurement type Downlink or Uplink for WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:LINK:STAT?`
Modes:	WCDMA
Associated Web Service:	`GetDownlinkUplink (WCDMA)`

[:SENSe<1|2>]:WCDMa:MARKer<1|2>:CDPError?

Return Parameters:	<nv>
Parameter Form:	<nv>
Default:	N/A
Description:	This command queries the specified WCDMA marker's code domain power error value.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:MARK1:CDPE?`
Modes:	WCDMA
Associated Web Services Methods:	None

[:SENSe<1|2>]:WCDMa:MARKer<1|2>:CDPLevel?

Return Parameters:	<nv>
Parameter Form:	<nv>
Default:	N/A
Description:	This command queries the specified WCDMA marker's code domain power level value.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	:WCDM:MARK1:CDPL?
Modes:	WCDMA
Associated Web Services Methods:	None

[:SENSe<1|2>]:WCDMa:MARKer<1|2>:POSition

Parameters:	
Parameter Form:	<nv>
Default:	
Description:	This command sets the specified WCDMA marker's position.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	[:SENSe<1|2>]:WCDMa:MARKer<1|2>:POSition?
Example:	:WCDM:MARK1:POS 128
Modes:	WCDMA
Associated Web Services Methods:	SetCDPMarkerPosition (WCDMA)

[:SENSe<1|2>]:WCDMa:MARKer<1|2>:POSition?

Return Parameters:

Parameter Form: <nv>

Default:

Description: This command queries the specified WCDMA marker’s position.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: N/A

Example: `:WCDM:MARK1:POS?`

Modes: WCDMA

Associated Web Services Methods: `GetCDPMarkerPosition (WCDMA)`

[:SENSe<1|2>]:WCDMa:MODanalysis:CDPower:REFerence

Parameters: ABSolute|RELative

Parameter Form: <char>

Default: REL

Description: This command sets whether the code domain power readings are relative to total power or in absolute power.

Precondition: The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.

Query Form: `[:SENSe<1|2>]:WCDMa:MODanalysis:CDPower:REFerence?`

Example: `:WCDM:MOD:CDP:REF ABS`

Modes: WCDMA

Associated Web Services Methods: None

[:SENSe<1|2>]:WCDMa:MODanalysis:CDPower:REFerence?

Parameters:	ABSolute|RELative
Parameter Form:	<char>
Default:	RELative
Description:	This command queries whether the code domain power readings are relative to total power or in absolute power.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:MOD:CDP:REF?`
Modes:	WCDMA
Associated Web Services Methods:	None

[:SENSe<1|2>]:WCDMa:SUMMary?

Return Parameters:	**Input:** FERR|RHO|EVM|MCEVM|PEVM|PHERR|AERR|IQOFF|SCODE|SPWR|PSPWR|SSPWR|PCDE **Output:** <nv>
Parameter Form:	<char>
Default:	N/A
Description:	This command outputs the specified summary data.
Precondition:	The WCDMA measurement mode must be active. Use the :INSTrument<1|2> Subsystem to select the appropriate mode.
Query Form:	N/A
Example:	`:WCDM:SUMM? FERR`
Modes:	WCDMA
Associated Web Services Methods:	`GetWCDMACompositeSummaryData (WCDMA)`

[:SENSe<1|2>]:WCDMa:RACH:CAPTure[:LENGth]

Parameters: 10MSEC to 100MSEC

Parameter Form: <char>

Default: 50MSEC

Description: Sets RACH Capture Length for the WCDMA measurement.

Query Form: `[:SENSe<1|2>]:WCDMa:RACH:CAPTure[:LENGth]?`

Example: `:WCDM:RACH:CAPT 30MSEC`

Modes: WCDMA

Associated Web Service: `SetRACHCaptureLength (WCDMA)`

[:SENSe<1|2>]:WCDMa:RACH:CAPTure[:LENGth]?

Return Parameters: 10MSEC to 100MSEC

Parameter Form: <char>

Default: 50MSEC

Description: gets RACH Capture Length for the WCDMA measurement.

Query Form: N/A

Example: `:WCDM:RACH:CAPT?`

Modes: WCDMA

Associated Web Service: `GetRACHCaptureLength (WCDMA)`

[:SENSe<1|2>]:WCDMa:RACH:ANAlysis[:LENGth]

Parameters:	400USECS to 1066.5USECS
Parameter Form:	<char>
Default:	1066.5USECS
Description:	Sets RACH Analysis Length for the WCDMA measurement.
Query Form:	`[:SENSe<1|2>]:WCDMa:RACH:ANAlysis[:LENgth]?`
Example:	`:WCDM:RACH:ANA 400USEC`
Modes:	WCDMA
Associated Web Service:	`SetRACHAnalysisLength (WCDMA)`

[:SENSe<1|2>]:WCDMa:RACH:ANAlysis[:LENGth]?

Return Parameters:	400USECS to 1066.5USECS
Parameter Form:	<nv>
Default:	1066.5USECS
Description:	gets RACH Analysis Length for the WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:RACH:ANA?`
Modes:	WCDMA
Associated Web Service:	`GetRACHAnalysisLength (WCDMA)`

[:SENSe<1|2>]:WCDMa:RACH:SUMMary?

Return Parameters: PreamblesDisplayed | PreambleSignature | TimeOffset | OnOffRatio | FilteredPowerOn | FilteredPowerOff_Leading | FilteredPowerOffTrailing | FilteredPowerOffAverage

Parameter Form: <char>

Default: N/A

Description: gets RACH Summary Data for the WCDMA measurement.

Query Form: N/A

Example: `:WCDM:RACH:SUMM?`

Modes: WCDMA

Associated Web Service: `GetRACHSummaryData (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:TIMe[:RANGe]

Parameters: 1FRAME to 8FRAME | 0SLOT to 12SLOT

Parameter Form: <char>

Default: 1FRAME

Description: Sets TPC time range for the WCDMA measurement.

Query Form: `[:SENSe<1|2>]:WCDMa:TPC:TIMe[:RANGe]?`

Example: `:WCDM:TPC:TIM 1FRAM`

Modes: WCDMA

Associated Web Service: `SetTPCTimeRange (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:TIMe[:RANGe]?

Return Parameters:	1FRAME to 8FRAME | 0SLOT to 12SLOT
Parameter Form:	<char>
Default:	1FRAME
Description:	gets TPC time range for the WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:TPC:TIM?`
Modes:	WCDMA
Associated Web Service:	`GetTPCTimeRange (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:AVERage:TIMeperslot

Parameters:	400USECS to 666.7USECS
Parameter Form:	<char>
Default:	666.7USECS
Description:	Sets TPC averaging time per slot for the WCDMA measurement.
Query Form:	`[:SENSe<1|2>]:WCDMa:TPC:AVERage:TIMeperslot?`
Example:	`:WCDM:TPC:AVER:TIM`
Modes:	WCDMA
Associated Web Service:	`SetTPCAveragingTimePerSlot (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:AVERage:TIMeperslot?

Return Parameters: 400USECS to 666.7USECS

Parameter Form: <char>

Default: 666.7USECS

Description: gets TPC averaging time per slot for the WCDMA measurement.

Query Form: N/A

Example: `:WCDM:TPC:AVER:TIM?`

Modes: WCDMA

Associated Web Service: `GetTPCAveragingTimePerSlot (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:LIMitcheck:STATe

Parameters: ON | OFF

Parameter Form: <boolean>

Default: OFF

Description: Sets TPC limit check for the WCDMA measurement.

Query Form: `[:SENSe<1|2>]:WCDMa:TPC:LIMitcheck:STATe?`

Example: `:WCDM:TPC:LIM:STAT ON`

Modes: WCDMA

Associated Web Service: `SetTPCLimitCheck (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:LIMitcheck:STATe?

Return Parameters:	ON | OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	gets TPC limit check for the WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:TPC:LIM:STAT?`
Modes:	WCDMA
Associated Web Service:	`GetTPCLimitCheck (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:STEP:SIZe

Parameters:	1DB | 2DB | 3DB
Parameter Form:	<char>
Default:	1DB
Description:	Sets TPC step size for the WCDMA measurement.
Query Form:	`[:SENSe<1|2>]:WCDMa:TPC:STEP:SIZe?`
Example:	`:WCDM:TPC:STEP:SIZ 2dB`
Modes:	WCDMA
Associated Web Service:	`SetTPCStepSize (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:STEP:SIZe?

Return Parameters: 1DB | 2DB | 3DB

Parameter Form: <char>

Default: 1DB

Description: gets TPC step size for the WCDMA measurement.

Query Form: N/A

Example: `:WCDM:TPC:STEP:SIZ?`

Modes: WCDMA

Associated Web Service: `GetTPCStepSize (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:COMMand:PATTern

Parameters: –1 | 0 | 1

Parameter Form: <nv>

Default: –1

Description: Sets TPC command pattern for the WCDMA measurement.

Query Form: `[:SENSe<1|2>]:WCDMa:TPC:COMMand:PATTern?`

Example: `:WCDM:TPC:COMM:PATT –1`

Modes: WCDMA

Associated Web Service: `SetTPCComandPattern (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:COMMand:PATTern?

Return Parameters:	–1 | 0 | 1
Parameter Form:	<nv>
Default:	–1
Description:	gets TPC command pattern for the WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:TPC:COMM:PATT?`
Modes:	WCDMA
Associated Web Service:	`GetTPCComandPattern (WCDMA)`

[:SENSe<1|2>]:WCDMa:SCH:SLOTnumber:DATAbits

Parameters:	1 to 15
Parameter Form:	<nv>
Default:	1
Description:	Sets SCH slot number for databits for the WCDMA measurement.
Query Form:	`[:SENSe<1|2>]:WCDMa:SCH:SLOTnumber:DATAbits?`
Example:	`:WCDM:SCH:SLOT:DATA 2`
Modes:	WCDMA
Associated Web Service:	`SetSCHSlotNumberforDataBits (WCDMA)`

[:SENSe<1|2>]:WCDMa:SCH:SLOTnumber:DATAbits?

Return Parameters:	1 to 15
Parameter Form:	<nv>
Default:	1
Description:	gets SCH slot number for databits for the WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:SCH:SLOT:DATA?`
Modes:	WCDMA
Associated Web Service:	`GetSCHSlotNumberforDataBits (WCDMA)`

[:SENSe<1|2>]:WCDMa:SCH:DISPlay:FORMat

Parameters:	BINARY | HEX
Parameter Form:	<char>
Default:	HEX
Description:	Sets SCH data display format for the WCDMA measurement.
Query Form:	`[:SENSe<1|2>]:WCDMa:SCH:DISPlay:FORMat?`
Example:	`:WCDM:SCH:DISP:FORM HEX`
Modes:	WCDMA
Associated Web Service:	`SetSCHDataDisplayFormat (WCDMA)`

[:SENSe<1|2>]:WCDMa:SCH:DISPlay:FORMat?

Return Parameters:	BINARY | HEX
Parameter Form:	<char>
Default:	HEX
Description:	gets SCH data display format for the WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:SCH:DISP:FORM?`
Modes:	WCDMA
Associated Web Service:	`GetSCHDataDisplayFormat (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:SCHCompensation:STATe

Parameters:	ON | OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	Sets SCH compensation for the WCDMA measurement.
Query Form:	`[:SENSe<1|2>]:WCDMa:TPC:SCHCompensation:STATe?`
Example:	`:WCDM:TPC:SCHC:STAT ON`
Modes:	WCDMA
Associated Web Service:	`SetSCHCompensation (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:SCHCompensation:STATe?

Return Parameters:	ON | OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	gets SCH compensation for the WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:TPC:SCHC:STAT?`
Modes:	WCDMA
Associated Web Service:	`GetSCHCompensation (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:PICHcompensation:STATe

Parameters:	ON | OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	Sets PICH compensation for the WCDMA measurement.
Query Form:	`[:SENSe<1|2>]:WCDMa:TPC:PICHcompensation:STATe?`
Example:	`:WCDM:TPC:PICH:STAT ON`
Modes:	WCDMA
Associated Web Service:	`SetPICHCompensation (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:PICHcompensation:STATe?

Return Parameters:	ON | OFF
Parameter Form:	<boolean>
Default:	OFF
Description:	gets PICH compensation for the WCDMA measurement.
Query Form:	N/A
Example:	`:WCDM:TPC:PICH:STAT?`
Modes:	WCDMA
Associated Web Service:	`GetPICHCompensation (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:PICH:CHANnelnumber

Parameters:	0 to 255
Parameter Form:	<nv>
Default:	16
Description:	Sets PICH channel number for the WCDMA measurement.
Query Form:	`[:SENSe<1|2>]:WCDMa:TPC:PICH:CHANnelnumber?`
Example:	`:WCDM:TPC:PICH:CHAN 20`
Modes:	WCDMA
Associated Web Service:	`SetPICHChannelNumber (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:PICH:CHANnelnumber?

Return Parameters: 0 to 255

Parameter Form: <nv>

Default: 16

Description: gets PICH channel number for the WCDMA measurement.

Query Form: N/A

Example: `:WCDM:TPC:PICH:CHAN?`

Modes: WCDMA

Associated Web Service: `GetPICHChannelNumber (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:PICH:TIMIngoffset

Parameters: 0 to 149

Parameter Form: <nv>

Default: 120

Description: Sets PICH timing offset for the WCDMA measurement.

Query Form: `[:SENSe<1|2>]:WCDMa:TPC:PICH:TIMIngoffset?`

Example: `:WCDM:TPC:PICH:TIMI 120`

Modes: WCDMA

Associated Web Service: `SetPICHTimingOffset (WCDMA)`

[:SENSe<1|2>]:WCDMa:TPC:PICH:TIMIngoffset?

Return Parameters: 0 to 149

Parameter Form: <nv>

Default: 120

Description: gets PICH timing offset for the WCDMA measurement.

Query Form: N/A

Example: `:WCDM:TPC:PICH:TIMI?`

Modes: WCDMA

Associated Web Service: `GetPICHTimingOffset (WCDMA)`

[:SENSe<1|2>]:WCDMa:SCH:SUMMary?

Return Parameters: **Input:** SCHState | SecondarySynchronizationCode | ScramblingCodeGroup | ScramblingCodeNumber | TotalSCHPowerAbsolute | PSCHPowerAbsolute | SSCHPowerAbsolute | TotalSCHPowerRelative | PSCHPowerRelative | SSCHPowerRelative | TransmitPower
Output: <nv>

Parameter Form: <char>

Default: N/A

Description: gets SCH summary data for the WCDMA measurement.

Query Form: N/A

Example: `:WCDM:SCH:SUMM?`

Modes: WCDMA

Associated Web Service: `GetSCHSummaryData (WCDMA)`

2-26 Programming Examples

This section provides programming examples represented in the VB6 programming environment. Figure 2-1 shows a basic example of a GPIB program using the SCPI command set documented in this manual.

```
Const GPIBBoard = 0
Const GPIBAddress = 1
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Private Sub GPIB_Click()
    Dim Signature As Integer
    Dim Buffer As String * 50
    Dim Frequency As String * 16
    Dim Power As String * 16

    'get handle to Signature
    Signature = ildev(GPIBBoard, GPIBAddress, 0, T10s, 1, 0)
    'Send Preset command
    Call ibwrt(Signature, "System:Preset" & vbCrLf)
    'wait for preset to complete
    Call Sleep(2000)
    'set to single sweep
    Call ibwrt(Signature, "Initiate:Continuous Off")
    'Enable 50MHz calibrator
    Call ibwrt(Signature, "Diagnostic:Service:Input Calibration" & vbCrLf)
    'Set Center Frequency to 50 MHz
    Call ibwrt(Signature, "Frequency:Center 50 MHz" & vbCrLf)
    'Set Span to 2 MHz
    Call ibwrt(Signature, "Frequency:Span 2 MHz" & vbCrLf)
    'trigger a sweep
    Call ibwrt(Signature, "Initiate:Continuous Off")
    'wait for sweep to complete

    Do
        Call ibwrt(Signature, "Initiate:Sweep?")
        'Read the reply
        Buffer = ""
        Call ibrd(Signature, Buffer)

    Loop While Left(Buffer, 1) <> "1"
    'Enable Marker #1

    Call ibwrt(Signature, "Calculate:Marker1 ON" & vbCrLf)
    'Send Marker #1 to the peak signal
    Call ibwrt(Signature, "Calculate:Marker1:Maximum" & vbCrLf)
    'Query the Marker Frequency
    Call ibwrt(Signature, "Calculate:Marker1:X?" & vbCrLf)
    'Read the reply
    Buffer = ""
    Call ibrd(Signature, Buffer)
    Frequency = Left(Buffer, 16)
    'Query the Marker Power
    Call ibwrt(Signature, "Calculate:Marker1:Y?" & vbCrLf)
    'Read the reply

    Buffer = ""
    Call ibrd(Signature, Buffer)
    Power = Left(Buffer, 16)
    'Output Result
    MsgBox "Marker Frequency = " & Frequency & vbCrLf & "Marker Power = " & Power

    'Close GPIB Interface

    Call ibonl(Signature, 0)

End Sub
```

Figure 2-1. GPIB Program Example

Chapter 3
Web Services Programming Methods

3-1 Introduction

This chapter contains all of the Web Services methods that are implemented in the instrument. The Web Services methods are grouped by their respective service classes and are described in detail.

Method Descriptions

The complete details of the methods are given following a brief description of the service class. If applicable, an example for each method and the range of values are given. The modes for which a method can be used are indicated by the following abbreviations:

- SYS: SignatureSystem Class
- SPA: Spectrum Analysis and SignatureSpectrum Class
- VSA: Vector Signal Analysis and SignatureModulation Class
- WCDMA: Wideband Code Division Multiple Access Modulation Analysis

Note: The spectrum analysis mode is implemented in the basic unit. For the VSA and WCDMA modes, the corresponding instrument options are required. Refer to the operation manual for details about these options.

3-2 Signature System Control Class

The SignatureSystemControl class provides access to system level controls and queries.

The examples provided in this section require the appropriate header code as follows:

VB6 Example Header Code

```
Dim SignatureSystemControl As New MSSOAPLib30.SoapClient30
SignatureSystemControl.MSSoapInit "http://SN123456/SignatureSystemControl/" & _
"SignatureSystemControl.asmx?wsdl"
'Enter SignatureSystemControl VB6 Example Code here to remotely program the
'instrument.
```

C#.Net Example Header Code

```
using System;
namespace SampleWSClient

{

/// <summary>
/// This is a sample web service client that demonstrates how to use the following
/// Anritsu web services in a C# .NET environment.
/// SignatureSystemControl.
/// </summary>

class SampleClient

{

[STAThread]
static void Main(string[] args)

{

//Creating the different Anritsu web service objects below.
SampleWSClient.SignatureSystemControl.SignatureSystem SigSystemObj = new
SampleWSClient.SignatureSystemControl.SignatureSystem();

{
//Enter SignatureSystemControl C# Example Code here to remotely program the
//instrument.
}

}

}

}
```

AsyncStartSweep (SYS)

Description: This method triggers a sweep when in the single sweep mode. This is a non-blocking call.

Precondition: This call works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API: `public void AsyncStartSweep()`

Arguments: None

VB6 Example: `Call SignatureSystemControl.AsyncStartSweep`

C#.NET Example:

```
//Call to start a sweep when in the Capture IQ mode.
SigSystemObj.AsyncStartSweep();
```

Associated GPIB Commands: None

ClearSignatureErrorLog (SYS)

Description: This method clears the instrument's error log and command history.

API: `public void ClearSignatureErrorLog()`

Arguments: None

VB6 Example: `Call SignatureSystemControl.ClearSignatureErrorLog`

C#.NET Example:

```
//Call to clear the contents of the Signature error
log.
SigSystemObj.ClearSignatureErrorLog();
```

Associated GPIB Commands: `:SYSTem:ERRor:CLEar:ALL`

GetAntiAliasingFilterState (SYS)

Description: This method queries the anti-aliasing filter toggle setting.

API:

```
public void GetAntiAliasingFilterState(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains the boolean switch with the following values:

True - anti-aliasing filter is On
False - anti-aliasing filter is Off

VB6 Example:

```
Dim bSwitchOn_out As Boolean
Dim bSwitchOn_out = _
SignatureSystemControl.GetAntiAliasingFilterState
```

C#.NET Example:

```
//Call to get the state of the Anti-Alias filter in the
system.
bool bFilterState = false; //Anti-Alias filter state.
bFilterState =
SigSystemObj.GetAntiAliasingFilterState();
```

Associated GPIB Commands:

```
:SYSTem:FILTer:AALias?
```

GetCaptureIQDataMode (SYS)

Description: This method queries the Capture IQ Data mode setting.

API:
```
public void GetCaptureIQDataMode(out bool
bCurrentCaptureIQDataMode)
```

Arguments: **bCurrentCaptureIQDataMode**

Contains a boolean switch with the following values:

True - capture IQ data mode is On
False - capture IQ data mode is Off

VB6 Example:
```
Dim bIQMode As Boolean
bIQMode = SignatureSystemControl.GetCaptureIQDataMode
```

C#.NET Example:
```
//Call to check if the system is in the the CaptureIQ
Data mode or not.
bool bTrue = true;
bTrue = SigSystemObj.GetCaptureIQDataMode();
//If bTrue == true (The system is in CaptureIQ Data
mode)
//If bTrue == false (The system is in non CaptureIQ
Data mode)
```

Associated GPIB Commands: None

GetCaptureIQDIFSampleRateBandwidth (SYS)

Description: This method queries the capture IQ sample rate and bandwidth setting.

API:
```
public void GetCaptureIQDIFSampleRateBandwidth(out
DDFNarrowTable enumDDFTable)
```

Arguments: **enumDDFTable**

An output parameter that contains the capture IQ selection with different choices for sample rate and bandwidth, and is defined as an enumeration constant with the following vaues:

"Tbl_21p4M_10M"
"Tbl_21p4M_5M"
"Tbl_12p84M"
"Tbl_8p56M"
"Tbl_4p28M"
"Tbl_2p14M"
"Tbl_1p28M"
"Tbl_856K"
"Tbl_428K"

VB6 Example:
```
Dim sDDFTable As String
sDDFTable =
SignatureSystemControl.GetCaptureIQDIFSampleRateBandwi
dth
```

C#.NET Example:
```
//Call to read the Sampling Rate from the system when
in the Capture IQ mode.
SignatureSystemControl.DDFTable enumDDFTable
=SignatureSystemControl.DDFTable.Tbl_428K;
enumDDFTable =
SigSystemObj.GetCaptureIQDIFSampleRateBandwidth();
```

GetCaptureIQSampleRateBandwidth (SYS)

Description: This method queries the capture IQ sample rate and bandwidth setting.

API:
```
public void GetCaptureIQSampleRateBandwidth(out
DDFTable enumDDFTable)
```

Arguments: **enumDDFTable**

An output parameter that contains the capture IQ selection with different choices for sample rate and bandwidth, and is defined as an enumeration constant with the following vaues:

"Tbl_50M_NoHBF" for sampling rate/bandwidth of 50M/25M
"Tbl_50M" for sampling rate/bandwidth of 50M/20M
"Tbl_25M" for sampling rate/bandwidth of 25M/10M
"Tbl_12p5M" for sampling rate/bandwidth of 12.5M/5M
"Tbl_6p25M" for sampling rate/bandwidth of 6.25M/2M
"Tbl_3p1215M" for sampling rate/bandwidth of 3.125M/1M
"Tbl_2M" for sampling rate/bandwidth of 2M/800K
"Tbl_1M" for sampling rate/bandwidth of 1M/400K
"Tbl_500k" for sampling rate/bandwidth of 500K/200K
"Tbl_400k" for sampling rate/bandwidth of 400K/150K
"Tbl_200k" for sampling rate/bandwidth of 200K/80K
"Tbl_100k" for sampling rate/bandwidth of 100K/40K

VB6 Example:
```
Dim sDDFTable As String
sDDFTable =
SignatureSystemControl.GetCaptureIQSampleRateBandwidth
```

C#.NET Example:
```
//Call to read the Sampling Rate from the system when
in the Capture IQ mode.
SignatureSystemControl.DDFTable enumDDFTable =
SignatureSystemControl.DDFTable.Tbl_50M;
enumDDFTable =
SigSystemObj.GetCaptureIQSampleRateBandwidth();
```

Associated GPIB Commands: None

GetCaptureIQSweepMode (SYS)

Description: This method queries the sweep mode setting when in the Capture IQ measurement mode.

Precondition: This API works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:

```
public void GetCaptureIQSweepMode(out bool
bContinuous_out)
```

Arguments: **bContinuous_out**

Contains a boolean value when the call returns with the following values:

True - continuous sweep mode
False - single sweep mode

VB6 Example:

```
Dim bContinuous As Boolean
bContinuous = _
SignatureSystemControl.GetCaptureIQSweepMode
```

C#.NET Example:

```
//Call to read the Capture IQ Sweep Mode from the
system.
bool bContinuous = false;
bContinuous = SigSystemObj.GetCaptureIQSweepMode();
//If bContinuous = true (the system is in the
Continuous sweep mode.)
//If bContinuous = false (the system is in the Single
sweep mode.)
```

Associated GPIB Commands: None

`GetCaptureStatus` (SYS)

Description: This method queries the status of a triggered capture.

API:

```
public void GetCaptureStatus (out bool bCaptureStatus)
```

Arguments: **bCaptureStatus**

Contains a boolean value when the call returns with the following values:

True -capture successful
False - capture unsuccessful

VB6 Example:

```
Dim bCaptureStatus As Boolean
bCaptureStatus = _
SignatureSystemControl.GetCaptureStatus
```

C#.NET Example:

```
//Call to read a triggered capture status from the
system.
bool bCaptureStatus = false;
bCaptureStatus = SigSystemObj.GetCaptureStatus();
//If bCaptureStatus = true (the system has a
successful capture.)
//If bContinuous = false (the system has an
unsuccessful capture.)
```

Associated GPIB Commands: None

`GetExternalReferenceLockStatus` (SYS)

Description: This method queries the lock status of the instrument.

Precondition: This API works only when the system is set to the External Reference mode.

API:

```
public void GetExternalReferenceLockStatus(out bool
bLockStatus)
```

Arguments: **bLockStatus**

Contains a boolean value when the call returns with the following values:

True - locked
False - unlocked

VB6 Example:

```
Dim bLockStatus As Boolean
bLockStatus =
SignatureSystemControl.GetExternalReferenceLockStatus
```

C#.NET Example:

```
//Call to read the external frequency reference lock
status of the system
bool bLockStatus = false;
bLockStatus =
SigSystemObj.GetExternalReferenceLockStatus();
```

Associated GPIB Commands: None

`GetInputSignal` (SYS)

Description: This method queries the input signal type setting.

API:

```
public void GetInputSignal(out InputSignal
isInputSignal_out)
```

Arguments: **isInputSignal_out**

Contains the input signal setting when the call is sent and is defined as an enumeration constant with the following values:

"IQDiffLow"
"IQDiffHigh"
"IQSingleLow"
"IQSingleHigh"
"IFInputWideBand"
"IFInputNarrowBand"

VB6 Example:

```
Dim sInputType As String
sInputType = SignatureSystemControl.GetInputSignal
```

C#.NET Example:

```
//Call to read the Input Signal from the system when in
the RF mode.
SignatureSystemControl.InputSignal enumInputSignal =
SignatureSystemControl.InputSignal.IQDiffLow;
enumInputSignal = SigSystemObj.GetInputSignal();
```

Associated GPIB Commands: None

`GetInstrumentIdentification` (SYS)

Description: This method queries the instrument's identification string.

API:
```
public void GetInstrumentIdentification(out string
strIDNString_out)
```

Arguments: **strIDNString_out**

Contains the instrument identification when the call returns and is defined as a string. For example:

"Anritsu,MS2781B,SN123456,R1.00"

which corresponds to:

"Company Name, Model Number, Serial Number, Software Version Number "

VB6 Example:
```
Dim strIDNString_out As String
strIDNString_out = _
SignatureSystemControl.GetInstrumentIdentification
```

C#.NET Example:
```
//Call to retrieve the instrument identification
string.
string strInstrumentId = null;
//Instrument identification.
strInstrumentId =
SigSystemObj.GetInstrumentIdentification();
```

Associated GPIB Commands: `*IDN? (Identification Query)`

GetInstrumentOptions (SYS)

Description: This method queries the instrument's installed options list. Refer to the technical data sheet, part number 11410-00333, for a complete list and description of available options for Signature.

API:
```
public void GetInstrumentOptions(out string
strOptionsString_out)
```

Arguments: **strOptionsString_out**

Contains the instrument's installed options list when the call returns and is defined as a string with the following format:

"3,40" (the GPIB and MATLAB options are installed) or
"3" (only the GPIB option is installed)

VB6 Example:
```
Dim strOptionsString_out As String
strOptionsString_out = _
SignatureSystemControl.GetInstrumentOptions
```

C#.NET Example:
```
//Call to retrieve information about the installed
options in the system.
string strOptions = null; //Installed options.
strOptions = SigSystemObj.GetInstrumentOptions();
```

Associated GPIB Commands: `*OPT? (Option Identification Query)`

GetInstrumentState (SYS)

Description: This method queries the instrument's current measurement state, such as overload or unlocked, etc.

API:

```
public void GetInstrumentState(out string
strInstrument_out)
```

Arguments: **strInstrument_out**

Contains the instrument's measurement state when the call returns and is defined as a string with the following format:

"NarrowBandIfOverLoaded = 0,WideBandIFOverloaded = 0,LoUnlocked = 0,RFOverloaded = 0"

Where 0 = False and 1 = True

VB6 Example:

```
Dim strInstrument_out As String
strInstrument_out = _
SignatureSystemControl.GetInstrumentState
```

C#.NET Example:

```
//Call to retrieve the instrument state.
string strInstrumentState = null;
//Instrument state.
strInstrumentState =
SigSystemObj.GetInstrumentState();
```

Associated GPIB Commands:

```
:STATus:QUEStionable:POWer:CONDition?
```

GetIQCapturePath (SYS)

Description: This method queries the IQ capture path setting.

API:
```
public void GetIQCapturePath(out CaptureIQPath
enumCaptureIQPath)
```

Arguments: **enumCaptureIQPath**

An output parameter that contains the capture IQ path selection when the call returns and is defined as an enumeration constant with the following values:

"NarrowBand"
"WideBand"

VB6 Example:
```
Dim sIQPath As String
sIQPath = SignatureSystemControl.GetIQCapturePath
```

C#.NET Example:
```
//Call to read the IQCapturePath from the the system.
SignatureSystemControl.CaptureIQPath enumCaptureIQPath
= SignatureSystemControl.CaptureIQPath.WideBand;
enumCaptureIQPath = SigSystemObj.GetIQCapturePath();
```

Associated GPIB Commands: None

GetIQCaptureTime (SYS)

Description: This method queries the IQ capture time setting.

Precondition: This API works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:
```
public void GetIQCaptureTime(out double dValue, out
TimeUnits enumTimeUnits)
```

Arguments: **dValue**

Contains the raw IQ data capture time duration when the call returns and is defined as a double.

enumTimeUnits

Contains the IQ capture time units when the call returns and is defined as an enumeration constant with the following values:

"ns" for nanoseconds
"us" for microseconds
"ms" for milliseconds
"s" for seconds
"ks" for kiloseconds

VB6 Example:
```
Dim dCaptureTime As Double
Dim sUnits As String
dCaptureTime =
SignatureSystemControl.GetIQCaptureTime(sUnits)
```

C#.NET Example:
```
//Call to read the CaptureTime from the system.
double dValue = 0.0;
SignatureSystemControl.TimeUnits enumTimeUnits =
SignatureSystemControl.TimeUnits.ms;
dValue = SigSystemObj.GetIQCaptureTime(out
enumTimeUnits);
```

Associated GPIB Commands: None

GetLastError (SYS)

Description: This method retrieves the most recent error from the instrument's error log.

API: `public void GetLastError(out string strLastError_out)`

Arguments: **strLastError_out**

Contains the most recent instrument error when the call returns and is defined as a string, for example:

"UI -Error while bringing GPIB service: The server threw an exception, 3/22/2005 10:54:03 AM"

VB6 Example:

```
Dim strLastError_out As String
strLastError_out = SignatureSystemControl.GetLastError
```

C#.NET Example:

```
//Call to retrieve the most recent error from the
system's error log.
string strLastError = null;
//The last logged error in the error log.
strLastError = SigSystemObj.GetLastError();
```

Associated GPIB Commands: `:SYSTem:ERRor?`

GetPhaseNoiseOptimizationMode (SYS)

Description: This method queries the phase noise optimization mode setting of the system.

API:
```
public void GetPhaseNoiseOptimizationMode(out
PhaseNoiseOptimizationMode
enumPhaseNoiseOptimizationMode)
```

Arguments: **enumPhaseNoiseOptimizationMode**

An output parameter that contains the phase noise optimization mode selection when the call returns and is defined as an enumeration constant with the following values:

“Auto”
“Manual”

VB6 Example:
```
Dim sPhaseNoiseOptimizationMode As String
sPhaseNoiseOptimizationMode =
SignatureSystemControl.GetPhaseNoiseOptimizationMode
```

C#.NET Example:
```
//Call to get the phase noise optimization mode from
the system.
SignatureSystemControl.PhaseNoiseOptimizationMode
enumPhaseNoiseOptimizationMode = Signature
SystemControl.PhaseNoiseOptimizationMode.Auto;
enumCaptureIQPath =
SigSystemObj.GetPhaseNoiseOptimizationMode();
```

Associated GPIB Commands:
```
:SYSTem:NOISeoptimize:AUTO?
```

GetPhaseNoiseOptimizationOffset (SYS)

Description: This method queries the manual phase noise optimization setting.

API:
```
public void GetPhaseNoiseOptimizationOffset(out
PhaseNoiseOptimizationOffset
enumPhaseNoiseOptimizationOffset)
```

Arguments: **enumPhaseNoiseOptimizationOffset**

An output parameter that contains the phase noise optimization offset selection and is defined as an enumeration constant with the following vaues:

"GreaterThan85kHz"
"LessThan85kHz"

VB6 Example:
```
Dim sPhaseNoiseOptimizationOffset As String
sPhaseNoiseOptimizationOffset =
SignatureSystemControl.GetPhaseNoiseOptimizationOffset
```

C#.NET Example:
```
//Call to read the phase noise optimization selection
from the system.
SignatureSystemControl.PhaseNoiseOptimizationOffset
enumPhaseNoiseOptimizationOffset =
SignatureSystemControl.PhaseNoiseOptimizationOffset.Gr
eaterThan85kHz;
enumPhaseNoiseOptimizationOffset  =
SigSystemObj.GetPhaseNoiseOptimizationOffset();
```

Associated GPIB Commands:
```
:SYSTem:NOISeoptimize:OFFSet?
```

GetRawIQVectorData (SYS)

Description: This method returns the raw IQ vector data for the specified data window in samples.

Precondition: This API works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:

```
public void GetRawIQVectorData(long
lOffsetInSampleNum, long lDataWindowInSamples, out
float[] fIData_out, out float[] fQData_out)
```

Arguments:

lOffsetInSampleNum

An input parameter that contains the sample offset number when the call is made and is defined as a long.

lDataWindowInSamples

An input parameter that contains the data window samples when the call is made and is defined as a long. This is the value that gets returned from the GetRawIQVectorDataSize API.

fIData_out

Contains the "I" data array when the call returns and is defined as a floating point array.

fQData_out

Contains the "Q" data array when the call returns and is defined as a floating point array.

VB6 Example:

```
Dim lOffsetSamples As Long, lDataWindowSamples As Long
Dim fIData() As Single, fQData() As Single
Dim lDataSize As Long
lOffsetSamples = 100 'Start at the 101st Sample
lDataWindowSamples = 300 'Read 300 Samples
lDataSize =
SignatureSystemControl.GetRawIQVectorDataSize _
 (lOffsetSamples, lDataWindowSamples)
'Rediminsion the data arrays to the expected data size
ReDim fIData(lDataSize)
ReDim fQData(lDataSize)
fIData = SignatureSystemControl.GetRawIQVectorData _
(lOffsetSamples, lDataWindowSamples, fQData)
```

C#.NET Example:

```
//Call to read 300 or available number of samples
(whichever is lower) of RawIQVector Data starting from
sample 101.
long lOffsetSampleNumber = 100;
long lNumberOfSamplesRequested = 300;
long lReturnableNumberOfSamples = 0;
lReturnableNumberOfSamples =
SigSystemObj.GetRawIQVectorDataSize(lOffsetSampleNumbe
r, lNumberOfSamplesRequested);
System.Single[] sIData = new
System.Single[lReturnableNumberOfSamples];
System.Single[] sQData = new
System.Single[lReturnableNumberOfSamples];
sIData =
SigSystemObj.GetRawIQVectorData(lOffsetSampleNumber,
lReturnableNumberOfSamples, out sQData);
```

Associated GPIB Commands: None

GetRawIQVectorDataSize (SYS)

Description: This method queries the raw IQ vector data size for the specified data window.

Precondition: This API works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:
```
public void GetRawIQVectorDataSize(long
lOffsetInSampleNum, long lDataWindowInSamples, out
long lDataSizeInSamples)
```

Arguments:

lOffsetInSampleNum

An input parameter that contains the sample offset number when the call is made and is defined as a long.

lDataWindowInSamples

An input parameter that contains the data window samples when the call is made and is defined as a long.

lDataSizeInSamples

An output parameter that contains the data sample size when the call returns and is defined as a long.

VB6 Example:
```
Dim lOffsetSamples As Long, lDataWindowSamples As
Long, lDataSize As Long
lDataSize =
SignatureSystemControl.GetRawIQVectorDataSize _
(lOffsetSamples, lDataWindowSamples)
```

C#.NET Example:
```
//Call to read the Raw IQ Vector Data size for a
specified data window (in the below example we want to
// read 300 samples starting at the 101th samples. If
there are less than 300 samples available to be
returned,
//this API returns the available number of samples.).
//This call usually precedes the GetRawIQVectorData
call
long lOffsetSampleNumber = 100;
long lNumberOfSamplesRequested = 300;
long lReturnableNumberOfSamples = 0;
lReturnableNumberOfSamples =
SigSystemObj.GetRawIQVectorDataSize(lOffsetSampleNumbe
r, lNumberOfSamplesRequested);
```

Associated GPIB Commands: None

GetReferenceType (SYS)

Description: This method queries the instrument's reference input setting.

API:
```
public void GetReferenceType(out double
dFrequencyValueMHz_out, out ReferenceSelect
enumReferenceSelect_out)
```

Arguments: **dFrequencyValueMHz_out**

Contains the reference frequency value when the call returns and is defined as a double.

Range: 1 MHz to 25 MHz in increments of 1 MHz, and 1.544 or 2.048 MHz.

enumReferenceSelect_out

Contains the reference units when the call returns and is defined as an enumeration constant with the following values:

"Ref_No_External" for internal reference
"Ref_1_25MHZ_Integer" for external reference
"Ref_2p048MHZ" for 2.048 MHz external reference
"Ref_1p544MHZ" for 1.544 external reference

VB6 Example:
```
Dim enumReferenceSelect_out As String
Dim dFrequencyValueMHz_out As Double
dFrequencyValueMHz_out = SignatureSystemControl. _
GetReferenceType(enumReferenceSelect_out)
```

C#.NET Example:
```
//Call to read the reference type from the system.
double dFrequencyValue = 0.0;
//If the reference type is Ref_1_25MHZ_Integer, then
this variable will contain the reference frequency
after the below call gets executed.
SignatureSystemControl.ReferenceSelect rsRefType;
//Reference type.
dFrequencyValue = SigSystemObj.GetReferenceType(out
rsRefType);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ROSCillator:SOURce?
```

GetSerialNumber (SYS)

Description: This method queries the instrument's serial number.

API:

```
public void GetSerialNumber(out string
strSerialNum_out)
```

Arguments: **strSerialNum_out**

Contains the instrument serial number when the call returns and is defined as a string, for example:

"SN123456"

VB6 Example:

```
Dim strSerialNum_out As String
strSerialNum_out =
SignatureSystemControl.GetSerialNumber
```

C#.NET Example:

```
//Call to read the serial number from the system.
string strSerialNumber;
//Serial number
strSerialNumber = SigSystemObj.GetSerialNumber();
```

Associated GPIB Commands: `*IDN? (Identification Query)`

GetSignatureErrorLog (SYS)

Description: This method queries the instrument's error log and command history.

API:

```
public void GetSignatureErrorLog(out string
strErrorLog_out)
```

Arguments: **strErrorLog_out**

Contains the instrument's error log and command history when the call returns and is defined as a string.

VB6 Example:

```
Dim strErrorLog_out As String
strErrorLog_out =
SignatureSystemControl.GetSignatureErrorLog
```

C#.NET Example:

```
//Call to read all of the errors from the error log in
a stringized format.
string strErrors = null;
// String data form of all of the errors logged in the
system.
strErrors = SigSystemObj.GetSignatureErrorLog();
```

Associated GPIB Commands: `:SYSTem:ERRor:LIST?`

GetSoftwareVersionNumber (SYS)

Description: This method queries the instrument's software version number.

API:
```
public void GetSoftwareVersionNumber(out string
strSoftwareVersionNum_out)
```

Arguments: **strSoftwareVersionNum_out**

Contains the instrument software version number when the call returns and is defined as a string, for example:

"R1.00"

VB6 Example:
```
Dim strSoftwareVersionNum_out As String
strSoftwareVersionNum_out = _
SignatureSystemControl.GetSoftwareVersionNumber
```

C#.NET Example:
```
//Call to read the software version number from the
system.
string strSoftwareVersion = null;
//Signature software version number
strSoftwareVersion =
SigSystemObj.GetSoftwareVersionNumber();
```

Associated GPIB Commands: `*IDN? (Identification Query)`

GetStandardType (SYS)

Description: This method queries the instrument's current measurement standard type.

API:
```
public void GetStandardType(out Standard
enumStandardType_out)
```

Arguments: **enumStandardType_out**

Contains the instrument's standard type setting when the call returns and is defined as an enumeration constant with the following values:

"Generic"
"WCDMA"

VB6 Example:
```
Dim enumStandardType_out As String
enumStandardType_out =
SignatureSystemControl.GetStandardType
```

C#.NET Example:
```
//Call to read the standard type set in the system.
SignatureSystemControl.Standard stStandard;
//Standard type.
stStandard = SigSystemObj.GetStandardType();
```

Associated GPIB Commands: `:SYSTem:STANdard?`

IsCaptureComplete (SYS)

Description: This method queries the capture complete status of a single sweep initiated by the AsyncStartSweep (SYS) web service.

Precondition: This call works only when the system is set to the CaptureIQ measurement mode and a single sweep is initiated by the AsyncStartSweep (SYS) web service. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:
```
public void IsCaptureComplete(out bool
bCaptureComplete)
```

Arguments: **bCaptureComplete**

Contains a boolean value when the call returns with the following values:

True - capture is complete
False - capture is not complete

VB6 Example:
```
Dim bCaptureComplete As Boolean
bCaptureComplete = _
SignatureSystemControl.IsCaptureComplete
```

C#.NET Example:
```
//Call to read the capture complete status.
bool bCaptureComplete;
bCaptureComplete = SigSystemObj.IsCaptureComplete();
```

Associated GPIB Commands: None

PerformIFCal (SYS)

Description: This method starts the instrument’s IF calibration routine. Note that this calibration takes approximately two minutes and an appropriate timeout must be implemented.

API:
```
public void PerformIFCal()
```

Arguments: None

VB6 Example:
```
'Set the Soap Client Timeout for 200 seconds _
(200000 milliseconds).
SignatureSystemControl.ConnectorProperty _
("Timeout") = 200000
Call SignatureSystemControl.PerformIFCal
```

C#.NET Example:
```
//Call to trigger an IF calibration.
SigSystemObj.PerformIFCal();
//A 200 second timeout value should be set for the
PerformIFCal as it is a timeconsuming call.
SigSystemObj.Timeout = 200000
```

Associated GPIB Commands: None

Preset (SYS)

Description: This method resets the instrument’s measurement parameters to factory default values.

API:
```
public void Preset()
```

Arguments: None

VB6 Example:
```
Call SignatureSystemControl.Preset
```

C#.NET Example:
```
//Call to trigger a system preset.
SigSystemObj.Preset();
```

Associated GPIB Commands: `:SYSTem:PRESet`

SetActiveMeasurement (SYS)

Description: This method sets the measurement mode of either Frequency Sweep (SPA) or Digital Demodulation (VSA).

API:
```
public void SetActiveMeasurement(MeasurementType
enumMeasurementType_in)
```

Arguments: **enumMeasurementType_in**

Contains the measurement type when the call is sent and is defined as an enumeration constant with the following values:

"FrequencySweep" for SPA mode
"DigitalDemod" for VSA mode
"WCDMA" for WCDMA mode
"CaptureIQ" for CaptureIQ mode

VB6 Example:
```
Call SignatureSystemControl. _
SetActiveMeasurement("FrequencySweep")
```

C#.NET Example:
```
//Call to set a frequency sweep as the active
measurement in the system.
SigSystemObj.SetActiveMeasurement(SignatureSystem _
Control.MeasurementType.FrequencySweep);
```

Associated GPIB Commands:
```
:INSTrument<1|2>:NSELect
:INSTrument<1|2>:SELect
```

SetCaptureIQDataMode (SYS)

Description: This method allows the user to set the Capture IQ Data mode.

API:
```
public void SetCaptureIQDataMode(bool bTrue)
```

Arguments: **bTrue**

Contains the boolean switch when the call returns with the following values:

True - capture IQ data mode On
False - capture IQ data mode Off

VB6 Example:
```
Dim bIQMode As Boolean
bIQMode = True
Call SignatureSystemControl.SetCaptureIQDataMode(True)
```

C#.NET Example:
```
//Call to set the system to the CaptureIQ Data mode.
bool bTrue = true;
SigSystemObj.SetCaptureIQDataMode(bTrue);
```

Associated GPIB Commands: None

`SetCaptureIQDIFSampleRateBandwidth` (SYS)

Description: This method sets the capture IQ sample rate and bandwidth setting.

Precondition: This API works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode web service to set the active measurement mode.

API:
```
public void
SetCaptureIQDIFSampleRateBandwidth(DDFNarrowTable
enumDDFTable)
```

Arguments: **enumDDFTable**

An input parameter that contains the capture IQ selection with different choices for sample rate and bandwidth, and is defined as an enumeration constant with the following vaues:

Tbl_21p4M_10M
Tbl_21p4M_5M
Tbl_12p84M
Tbl_8p56M
Tbl_4p28M
Tbl_2p14M
Tbl_1p28M
Tbl_856K
Tbl_428K

VB6 Example:
```
Dim sDDFTable As String
sDDFTable = "Tbl_428K"
Call
SignatureSystemControl.SetCaptureIQDIFSampleRateBandwi
dth(sDDFTable)
```

C#.NET Example:
```
Capture IQ mode.SignatureSystemControl.DDFTable
enumDDFTable =
SignatureSystemControl.DDFTable.Tbl_428K;
SigSystemObj.SetCaptureIQDIFSampleRateBandwidth(enumDD
FTable);
```

Associated GPIB Commands: None

SetCaptureIQSampleRateBandwidth (SYS)

Description: This method sets the capture IQ sample rate and bandwidth setting.

Precondition: This API works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:
```
public void SetCaptureIQSampleRateBandwidth(DDFTable
enumDDFTable)
```

Arguments: **enumDDFTable**

An input parameter that contains the capture IQ selection with different choices for sample rate and bandwidth, and is defined as an enumeration constant with the following vaues:

"Tbl_50M_NoHBF" for sampling rate/bandwidth of 50M/25M
"Tbl_50M" for sampling rate/bandwidth of 50M/20M
"Tbl_25M" for sampling rate/bandwidth of 25M/10M
"Tbl_12p5M" for sampling rate/bandwidth of 12.5M/5M
"Tbl_6p25M" for sampling rate/bandwidth of 6.25M/2M
"Tbl_3p1215M" for sampling rate/bandwidth of 3.125M/1M
"Tbl_2M" for sampling rate/bandwidth of 2M/800K
"Tbl_1M" for sampling rate/bandwidth of 1M/400K
"Tbl_500k" for sampling rate/bandwidth of 500K/200K
"Tbl_400k" for sampling rate/bandwidth of 400K/150K
"Tbl_200k" for sampling rate/bandwidth of 200K/80K
"Tbl_100k" for sampling rate/bandwidth of 100K/40K

VB6 Example:
```
Dim sDDFTable As String
sDDFTable = "Tbl_50M"
Call SignatureSystemControl.SetCaptureIQSampleRate _
Bandwidth(sDDFTable)
```

C#.NET Example:
```
//Call to set the Sampling Rate to 50 MHz when in the
Capture IQ mode.
SignatureSystemControl.DDFTable enumDDFTable =
SignatureSystemControl.DDFTable.Tbl_50M;
SigSystemObj.SetCaptureIQSampleRateBandwidth(enumDDFTa
ble);
```

Associated GPIB Commands: None

SetCaptureIQSweepMode (SYS)

Description: This method sets the sweep mode when in the Capture IQ measurement mode.

Precondition: This API works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:
```
public void SetCaptureIQSweepMode(bool bContinuous_in)
```

Arguments: **bContinuous_in**

An input parameter that contains a boolean value when the call is made with one of the following values:

True - continuous sweep mode
False - single sweep mode

VB6 Example:
```
Dim bContinuous As Boolean
bContinuous = True
Call SignatureSystemControl.SetCaptureIQSweepMode _
(bContinuous)
```

C#.NET Example:
```
//Call to set the CaptureIQ Sweep Mode to Single.
bool bContinuous = false;
SigSystemObj.SetCaptureIQSweepMode(bContinuous);
```

Associated GPIB Commands: None

`SetInputSignal` (SYS)

Description: This method sets the input signal type.

Precondition: This API works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:
```
public void SetInputSignal(InputSignal
isInputSignal_in)
```

Arguments: **isInputSignal_in**

An input parameter that contains the input signal setting when the call is made and is defined as an enumeration constant with one of the following values:

"IQDiffLow"
"IQDiffHigh"
"IQSingleLow"
"IQSingleHigh"
"IFInputWideBand"
"IFInputNarrowBand"

VB6 Example:
```
Dim sInputType As String
sInputType = "IQSingleLow"
Call SignatureSystemControl.SetInputSignal(sInputType)
```

C#.NET Example:
```
//Call to set the Input Signal to RF.
SignatureSystemControl.InputSignal enumInputSignal =
SignatureSystemControl.InputSignal.IQDiffLow;
SigSystemObj.SetInputSignal(enumInputSignal);
```

Associated GPIB Commands: None

SetInstrumentMode (SYS)

Description: This method sets the instrument to either the local or the remote mode of operation. When the system is switched back to the local mode, the previously active menu is displayed.

API:
```
public void SetInstrumentMode(InstrumentMode
enumInstrumentMode)
```

Arguments: **enumInstrumentMode**

An input parameter that contains the instrument mode setting when the call is made and is defined as an enumeration constant with one of the following values:

"Local"
"Remote"

VB6 Example:
```
Dim sInstrumentMode As String
sInstrumentMode = "Local"
Call SignatureSystemControl.SetInstrumentMode _
(sInstrumentMode)
```

C#.NET Example:
```
//Call to set the remote/local instrument mode of the
system.
SignatureSystemControl.InstrumentMode
enumInstrumentMode  =
SignatureSystemControl.InstrumentMode.Local;
SigSystemObj.SetInstrumentMode(enumInstrumentMode);
```

Associated GPIB Commands: `:SYSTem:RTL`

SetIQCapturePath (SYS)

Description: This method sets the IQ capture path setting.

API:
```
public void SetIQCapturePath(CaptureIQPath
enumCaptureIQPath)
```

Arguments: **enumCaptureIQPath**

An input parameter that contains the capture IQ path when the call is made and is defined as an enumeration constant with one of the following values:

“NarrowBand”
“WideBand”

VB6 Example:
```
Dim sIQPath As String
sIQPath = "WideBand"
Call SignatureSystemControl.SetIQCapturePath(sIQPath)
```

C#.NET Example:
```
//Call to set the IQCapturePath to the Wideband mode
in the system.
SignatureSystemControl.CaptureIQPath enumCaptureIQPath
= SignatureSystemControl.CaptureIQPath.WideBand;
SigSystemObj.SetIQCapturePath(enumCaptureIQPath);
```

Associated GPIB Commands: None

`SetIQCaptureTime` (SYS)

Description: This method sets the IQ data capture time.

API:
```
public void SetIQCaptureTime(double dValue, TimeUnits
enumTimeUnits)
```

Arguments: **dValue**

Contains the raw IQ data capture time duration when the call is made and is defined as a double.

enumTimeUnits

Contains the IQ data capture time units when the call is sent and is defined as an enumeration constant with the following values:

"ns" for nanoseconds
"us" for microseconds
"ms" for milliseconds
"s" for seconds
"ks" for kiloseconds

VB6 Example:
```
Dim dCaptureTime As Double
Dim sUnits As String
dCaptureTime = 10
sUnits = "ms"
Call SignatureSystemControl.SetIQCaptureTime _
(dCaptureTime, sUnits)
```

C#.NET Example:
```
//Call to set the IQ CaptureTime to 2 milli secs.
double dValue = 2.0;
SignatureSystemControl.TimeUnits enumTimeUnits =
SignatureSystemControl.TimeUnits.ms;
SigSystemObj.SetIQCaptureTime(dValue, enumTimeUnits);
```

Associated GPIB Commands: None

SetPhaseNoiseOptimizationMode (SYS)

Description: This method sets the phase noise optimization mode setting of the system.

API:
```
public void
SetPhaseNoiseOptimizationMode(PhaseNoiseOptimizationMode enumPhaseNoiseOptimizationMode)
```

Arguments: **enumPhaseNoiseOptimizationMode**

An input parameter that contains the phase noise optimization mode selection when the call is sent and is defined as an enumeration constant with the following values:

"Auto"
"Manual"

VB6 Example:
```
Dim sPhaseNoiseOptimizationMode As String
sPhaseNoiseOptimizationMode  = "Auto"
Call SignatureSystemControl.SetPhaseNoise _
OptimizationMode(sPhaseNoiseOptimizationMode)
```

C#.NET Example:
```
//Call to set the phase noise optimization mode of the system.
SignatureSystemControl.PhaseNoiseOptimizationMode
enumPhaseNoiseOptimizationMode  =
SignatureSystemControl.PhaseNoiseOptimizationMode.Auto;
SigSystemObj.SetPhaseNoiseOptimizationMode(enumPhaseNoiseOptimizationMode);
```

Associated GPIB Commands: `:SYSTem:NOISeoptimize:AUTO`

SetReferenceType (SYS)

Description: This method sets the frequency reference source and reference frequency value.

API:
```
public void SetReferenceType(double
dFrequencyValueMHz_in, ReferenceSelect
enumReferenceSelect_in)
```

Arguments: **dFrequencyValueMHz_in**

Contains the reference frequency value and is defined as a double. Ranges from 1 MHz to 25 MHz in increments of 1 MHz, and 1.544 or 2.048 MHz.

enumReferenceSelect_in

Contains the reference input type and is defined as an enumeration constant with the following values:

"Ref_No_External" for internal reference
"Ref_1_25MHZ_Integer" for external reference
"Ref_2p048MHZ" for 2.048 MHz external reference
"Ref_1p544MHZ" for 1.544 external reference

VB6 Example:
```
Call SignatureSystemControl. _
SetReferenceType(10#, "Ref_1_25MHZ_Integer")
```

C#.NET Example:
```
//Call to set an external reference frequency of
10 MHz.
double dValvalueMHz_in = 10;
SigSystemObj.SetReferenceType(dValvalueMHz_in,
SignatureSystemControl.ReferenceSelect.Ref_1_25MHZ_Int
eger);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ROSCillator:SOURce
```

SetStandardType (SYS)

Description: This method sets the instrument's current measurement standard type.

API:
```
public void SetStandardType(Standard
enumStandardType_in)
```

Arguments: **enumStandardType_in**

Contains the measurement type when the call is sent and is defined as an enumeration constant with the following values:

"Generic"
"WCDMA"

VB6 Example:
```
Call SignatureSystemControl.SetStandardType("WCDMA")
```

C#.NET Example:
```
//Call to set the measurement standard type to WCDMA
in the system.
SigSystemObj.SetStandardType(SignatureSystemControl.St
andard.WCDMA);
```

Associated GPIB Commands: `:SYSTem:STANdard`

StartSweep (SYS)

Description: This method triggers a sweep when in the single sweep mode. This is a blocking call and does not return until the sweep is complete.

API:
```
public void StartSweep()
```

Arguments: None

VB6 Example:
```
Call SignatureSystemControl.StartSweep
```

C#.NET Example:
```
//Call to start a sweep when in the Capture IQ mode.
SigSystemObj.StartSweep();
```

Associated GPIB Commands: `:INITiate<1|2>[:IMMediate]`

SwitchOnCalibratorSignal (SYS)

Description: This method toggles the internal 50 MHz calibrator On or Off.

API:
```
public void SwitchOnCalibratorSignal(bool
bCalibratorSignalOn_in)
```

Arguments: **bCalibratorSignalOn_in**

Contains the boolean switch with the following values:

True - calibrator signal On
False - calibrator signal Off

VB6 Example:
```
Call
SignatureSystemControl.SwitchOnCalibratorSignal(True)
```

C#.NET Example:
```
//Call to switch on the calibrator signal in the
system.
bool bSwitchOnCalibratorSignal = true;
SigSystemObj.SwitchOnCalibratorSignal(bSwitchOnCalibra
torSignal);
```

Associated GPIB Commands: `:DIAGnostic:SERVice:INPut[:SELect]`

ToggleAntiAliasingFilterState (SYS)

Description: This method toggles the anti-aliasing filter On or Off.

Precondition: This API only controls the anti-alias filter when the system is in the QAM/PSK Modulation mode and has no affect in other measurement modes.

API:
```
public void ToggleAntiAliasingFilterState(bool
bSwitchOn_in)
```

Arguments: **bSwitchOn_in**

Contains the boolean switch with the following values:

True - anti-aliasing filter On
False - anti-aliasing filter Off

VB6 Example:
```
Call SignatureSystemControl. _
ToggleAntiAliasingFilterState(True)
```

C#.NET Example:
```
//Call to switch on the anti-aliasing filter in the
system.
bool bSwitchOn_in = true;
SigSystemObj.ToggleAntiAliasingFilterState(bSwitchOn_i
n);
```

Associated GPIB Commands: `:SYSTem:FILTer:AALias`

3-3 SignatureSpectrum Class

The SignatureSpectrum class provides access to spectrum analysis controls and queries.

The examples provided in this section require the appropriate header code as follows:

VB6 Example Header Code

```
Dim SignatureSpectrum As New MSSOAPLib30.SoapClient30
SignatureSpectrum.MSSoapInit "http://SN123456/SignatureSpectrum/" &_
"SignatureSpectrum.asmx?wsdl"
'Enter SignatureSpectrum VB6 Example Code here to remotely program the instrument.
```

C#.Net Example Header Code

```
using System;
namespace SampleWSClient

{

/// <summary>
/// This is a sample web service client that demonstrates how to use the following
/// Anritsu web services in a C# .NET environment.
/// SignatureSpectrum.
/// </summary>

class SampleClient

{

[STAThread]
static void Main(string[] args)

{

//Creating the different Anritsu web service objects below.
SampleWSClient.SignatureSpectrum.SignatureSpectrum SigSpectrumObj = new
SampleWSClient.SignatureSpectrum.SignatureSpectrum();

{
//Enter SignatureSpectrum C# Example Code here to remotely program the
//instrument.
}

}

}

}
```

AsyncStartSweep (SPA)

Description: This method triggers a sweep when in the single sweep mode. This is a non-blocking call.

Precondition: This call works only when the system is set to the CaptureIQ measurement mode. Refer to the SetCaptureIQDataMode (SYS) web service to set the active measurement mode.

API:
```
public void AsyncStartSweep()
```

Arguments: None

VB6 Example:
```
Call SignatureSpectrum.AsyncStartSweep
```

C#.NET Example:
```
//Call to start a sweep when in the Capture IQ mode.
SigSpectrumObj.AsyncStartSweep();
```

Associated GPIB Commands: None

GetACPAdjacentChannelBandwidthInHz (SPA)

Description: This method queries the adjacent channel power, adjacent channel bandwidth parameter.

API:
```
public void GetACPAdjacentChannelBandwidthInHz(out
double dBandwidthinHz_out)
```

Arguments: **dBandwidthinHz_out**

Contains the adjacent channel power, adjacent channel bandwidth value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

VB6 Example:
```
Dim dBandwidthinHz_out As Double
dBandwidthinHz_out = _
SignatureSpectrum.GetACPAdjacentChannelBandwidthInHz
```

C#.NET Example:
```
//Call to read the adjacent channel power, adjacent
channel bandwidth from the system.
double dBandWidth = 0.0;
//ACPR adjacent channel bandwidth.
dBandWidth =
SigSpectrumObj.GetACPAdjacentChannelBandwidthInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ADJacent:CHBandwidth?
```

GetACPAdjacentChannelSpacingInHz (SPA)

Description: This method queries the adjacent channel power, adjacent channel spacing parameter.

API:
```
public void GetACPAdjacentChannelSpacingInHz(out
double dBandwidthinHz_out)
```

Arguments: **dBandwidthinHz_out**

Contains the adjacent channel power, adjacent channel spacing value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

VB6 Example:
```
Dim dBandwidthinHz_out As Double
dBandwidthinHz_out = _
SignatureSpectrum.GetACPAdjacentChannelSpacingInHz
```

C#.NET Example:
```
//Call to read the adjacent channel power, adjacent
channel spacing setting.
double dBandWidth = 0.0;
//ACPR adjacent channel spacing.
dBandWidth =
SigSpectrumObj.GetACPAdjacentChannelSpacingInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ADJacent:CHSPacing?
```

GetACPAdjacentChannelState (SPA)

Description: This method queries the adjacent channel power, adjacent channel toggle setting.

API:
```
public void GetACPAdjacentChannelState(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - ACP adjacent channel is On
False - ACP adjacent channel is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetACPAdjacentChannelState
```

C#.NET Example:
```
//Call to read the ACP adjacent channel state from the
system.
bool bAdjacentChannelState_out = false;
bAdjacentChannelState_out =
SigSpectrumObj.GetACPAdjacentChannelState();
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:ADJacent:STATe?`

GetACPAlternateChannel1BandwidthInHz (SPA)

Description: This method queries the adjacent channel power, alternate channel one bandwidth parameter.

API:
```
public void GetACPAlternateChannel1BandwidthInHz(out
double dBandwidthinHz_out)
```

Arguments: **dBandwidthinHz_out**

Contains the adjacent channel power, alternate channel one bandwidth value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

VB6 Example:
```
Dim dBandwidthinHz_out As Double
dBandwidthinHz_out = _
SignatureSpectrum.GetACPAlternateChannel1BandwidthInHz
```

C#.NET Example:
```
//Call to read the adjacent channel power, alternate
channel one bandwidth from the system.
double dBandWidth = 0.0;
//ACPR alternate channel1 bandwidth
dBandWidth =
SigSpectrumObj.GetACPAlternateChannel1BandwidthInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:CHBandwidth?
```

GetACPAlternateChannel1SpacingInHz (SPA)

Description: This method queries the adjacent channel power, alternate channel one spacing parameter.

API:
```
public void GetACPAlternateChannel1SpacingInHz(out
double dBandwidthinHz_out)
```

Arguments: **dBandwidthinHz_out**

Contains the adjacent channel power, alternate channel one spacing value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 10 MHz.

VB6 Example:
```
Dim dBandwidthinHz_out As Double
dBandwidthinHz_out = _
SignatureSpectrum.GetACPAlternateChannel1SpacingInHz
```

C#.NET Example:
```
//Call to read the adjacent channel power, alternate
channel one spacing from the system.
double dBandWidth = 0.0;
//ACPR alternate channel1 spacing.
dBandWidth =
SigSpectrumObj.GetACPAlternateChannel1SpacingInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:CHSPacing?
```

GetACPAlternateChannel1State (SPA)

Description: This method queries the adjacent channel power, alternate channel one toggle setting.

API:
```
public void GetACPAlternateChannel1State(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - ACP alternate channel one is On
False - ACP alternate channel one is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetACPAlternateChannel1State
```

C#.NET Example:
```
//Call to read the ACP alternate channel one state.
bool bSwitchOn = false;
//Channel one state.
bSwitchOn =
SigSpectrumObj.GetACPAlternateChannel1State();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:STATe?
```

GetACPAlternateChannel2BandwidthInHz (SPA)

Description: This method queries the adjacent channel power, alternate channel two bandwidth parameter.

API:
```
public void GetACPAlternateChannel2BandwidthInHz(out
double dBandwidthinHz_out)
```

Arguments: **dBandwidthinHz_out**

Contains the adjacent channel power, alternate channel two bandwidth value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

VB6 Example:
```
Dim dBandwidthinHz_out As Double
dBandwidthinHz_out = _
SignatureSpectrum.GetACPAlternateChannel2BandwidthInHz
```

C#.NET Example:
```
//Call to read the adjacent channel power, alternate
channel two bandwidth from the system.
double dBandWidth = 0.0;
//ACPR alternate channel2 bandwidth
dBandWidth =
SigSpectrumObj.GetACPAlternateChannel2BandwidthInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:CHBandwidth?
```

GetACPAlternateChannel2SpacingInHz (SPA)

Description: This method queries the adjacent channel power, alternate channel two spacing parameter.

API:
```
public void GetACPAlternateChannel2SpacingInHz(out
double dBandwidthinHz_out)
```

Arguments: **dBandwidthinHz_out**

Contains the adjacent channel power, alternate channel two spacing value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 15 MHz.

VB6 Example:
```
Dim dBandwidthinHz_out As Double
dBandwidthinHz_out = _
SignatureSpectrum.GetACPAlternateChannel2SpacingInHz
```

C#.NET Example:
```
//Call to read the adjacent channel power, alternate
channel two spacing from the system.
double dBandwidth = 0.0;
dBandwidth =
SigSpectrumObj.GetACPAlternateChannel2SpacingInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:CHSPacing?
```

GetACPAlternateChannel2State (SPA)

Description: This method queries the adjacent channel power, alternate channel two toggle setting.

API:
```
public void GetACPAlternateChannel2State(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - ACP alternate channel two is On
False - ACP alternate channel two is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetACPAlternateChannel2State
```

C#.NET Example:
```
//Call to read the ACP alternate channel two state.
bool bSwitchOn = false;
//Channel two state.
bSwitchOn =
SigSpectrumObj.GetACPAlternateChannel2State();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:STATe?
```

GetACPChannelBandwidthInHz (SPA)

Description: This method queries the adjacent channel power channel bandwidth parameter.

API:
```
public void GetACPChannelBandwidthInHz(out double
dBandwidthinHz_out)
```

Arguments: **dBandwidthinHz_out**

Contains the adjacent channel power channel bandwidth value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

VB6 Example:
```
Dim dBandwidthinHz_out As Double
dBandwidthinHz_out = _
SignatureSpectrum.GetACPChannelBandwidthInHz
```

C#.NET Example:
```
//Call to read the adjacent channel power channel
bandwidth from the system.
double dBandwidth = 0.0;
//ACP channel bandwidth.
dBandwidth =
SigSpectrumObj.GetACPChannelBandwidthInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:CHBandwidth?`

GetACPDivisionPerHzState (SPA)

Description: This method queries the adjacent channel power division/Hz toggle setting.

API:
```
public void GetACPDivisionPerHzState(out bool
bDisplay_out)
```

Arguments: **bDisplay_out**

Contains a boolean value when the call returns with the following values:

True - ACP division/Hz state is On
False - ACP division/Hz state is Off

VB6 Example:
```
Dim bDisplay_out As Boolean
bDisplay_out = _
SignatureSpectrum.GetACPDivisionPerHzState
```

C#.NET Example:
```
//Call to read the ACP division per Hz state setting.
bool bDisplay = false;
bDisplay = SigSpectrumObj.GetACPDivisionPerHzState();
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:HZ:STATe?`

GetACPFFTState (SPA)

Description: This method queries the adjacent channel power fast fourier transform toggle setting.

API:
```
public void GetACPFFTState(out bool bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - ACP FFT is On
False - ACP FFT is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetACPFFTState
```

C#.NET Example:
```
//Call to read the ACP FFT state setting.
bool bSwitchOn = false;
bSwitchOn = SigSpectrumObj.GetACPFFTState();
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:FFT:STATe?`

GetACPNoiseCompensationState (SPA)

Description: This method queries the adjacent channel power noise compensation toggle setting.

API:
```
public void GetACPNoiseCompensationState(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - noise compensation is On
False - noise compensation is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetACPNoiseCompensationState
```

C#.NET Example:
```
//Call to read the adjacent channel power noise
compensation state.
bool bSwitchOn = false;
bSwitchOn =
SigSpectrumObj.GetACPNoiseCompensationState();
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:NOISecomp:STATe?`

GetACPOBMMode (SPA)

Description: This method queries the current adjacent channel power one-button-measurement (OBM) mode setting.

API:
```
public void GetACPOBMMode(out OBMMode enumOBMMode_out)
```

Arguments: **enumOBMMode_out**

Contains the adjacent channel power OBM mode setting when the call returns and is defined as an enumeration constant with the following value:

"Relative"
"Absolute"

VB6 Example:
```
Dim enumOBMMode_out As String
enumOBMMode_out = SignatureSpectrum.GetACPOBMMode
```

C#.NET Example:
```
//Call to read the adjacent channel power OBM mode
setting.
SignatureSpectrum.OBMMode enumOBMMode;
// ACP OBM mode (relative or absolute)
enumOBMMode = SigSpectrumObj.GetACPOBMMode();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:TYPE?
```

`GetACPRAdjacentChannelResults` (SPA)

Description: This method queries the adjacent channel power measurement results. If the measurement is invalid, the returned result is 0.0 dBm.

API:

```
public void GetACPRAdjacentChannelResults(out bool
bValid_out, out double dACPRight_out, out
AmplitudeUnits enumAmpRUnits_out, out double
dACPLeft_out, out AmplitudeUnits enumAmpLUnits_out)
```

Arguments: **bValid_out**

Contains a boolean value when the call returns with the following values:

True - valid ACP measurement
False - invalid ACP measurement

dACPRight_out

Contains the right adjacent channel power value when the call returns and is defined as a double.

enumAmpRUnits_out

Contains the right adjacent channel power amplitude units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

dACPLeft_out

Contains the left adjacent channel power value when the call returns and is defined as a double.

enumAmpLUnits_out

Contains the left adjacent channel power amplitude units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:

```
Dim bValid_out As Boolean
Dim dUpperResult_out As Double
Dim sEnumUpperUnits_out As String
Dim dLowerResult_out As Double
Dim sEnumLowerUnits_out As String
bValid_out = SignatureSpectrum. _
GetACPRAdjacentChannelResults(dUpperResult_out, _
sEnumUpperUnits_out, dLowerResult_out, _
sEnumLowerUnits_out)
```

C#.NET Example:

```
//Call to retrieve the adjacent channel power results.
bool bValidResults = false;
//true if the results are valid, false otherwise.
double dUpper = 0.0;//Upper channel value
double dLower = 0.0;//Lower channel value
SignatureSpectrum.AmplitudeUnits auUpperUnits;
//Upper value units
SignatureSpectrum.AmplitudeUnits auLowerUnits;
//Lower value units
bValidResults =
SigSpectrumObj.GetACPRAdjacentChannelResults(out
dUpper, out auUpperUnits, out dLower, out
auLowerUnits);
```

Associated GPIB Commands:

```
:CALCulate<1|2>:ACP:ADJacent:RESult?
```

GetACPRAlternateChannel1Results (SPA)

Description: This method queries the alternate channel one power measurement results. If the measurement is invalid, the returned result is 0.0 dBm.

API:

```
public void GetACPRAlternateChannel1Results(out bool
bValid_out, out double dAlt1Right_out, out
AmplitudeUnits enumAmpRUnits_out, out double
dAlt1Left_out, out AmplitudeUnits enumAmpLUnits_out)
```

Arguments:

bValid_out

Contains a boolean value when the call returns with the following values:

True - valid ACP measurement
False - invalid ACP measurement

dAlt1Right_out

Contains the right alternate channel one power value when the call returns and is defined as a double.

enumAmpRUnits_out

Contains the right alternate channel one power amplitude units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

dAlt1Left_out

Contains the left alternate channel one power value when the call returns and is defined as a double.

enumAmpLUnits_out

Contains the left alternate channel one power amplitude units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:

```
Dim bValid_out As Boolean
Dim dUpperResult_out As Double
Dim sEnumUpperUnits_out As String
Dim dLowerResult_out As Double
Dim sEnumLowerUnits_out As String
bValid_out = SignatureSpectrum. _
GetACPRAlternateChannel1Results(dUpperResult_out, _
sEnumUpperUnits_out, dLowerResult_out, _
sEnumLowerUnits_out)
```

C#.NET Example:

```
//Call to retrieve alternate channel 1 results.
bool bValidResults = false;
//true if the results are valid, false otherwise.
double dUpper = 0.0;//Upper channel value
double dLower = 0.0;//Lower channel value
SignatureSpectrum.AmplitudeUnits auUpperUnits;
//Upper value units
SignatureSpectrum.AmplitudeUnits auLowerUnits;
//Lower value units
bValidResults =
SigSpectrumObj.GetACPRAlternateChannel1Results(out
dUpper, out auUpperUnits, out dLower, out
auLowerUnits);
```

Associated GPIB Commands:

```
:CALCulate<1|2>:ACP:ALT<1|2>:RESult?
```

GetACPRAlternateChannel2Results (SPA)

Description: This method queries the alternate channel two power measurement results. If the measurement is invalid, the returned result is 0.0 dBm.

API:
```
public void GetACPRAlternateChannel2Results(out bool
bValid_out, out double dAlt2Right_out, out
AmplitudeUnits enumAmpRUnits_out, out double
dAlt2Left_out, out AmplitudeUnits enumAmpLUnits_out)
```

Arguments: **bValid_out**

Contains a boolean value when the call returns with the following values:

True - valid ACP measurement
False - invalid ACP measurement

dAlt2Right_out

Contains the right alternate channel two power value when the call returns and is defined as a double.

enumAmpRUnits_out

Contains the right alternate channel two power amplitude units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

dAlt2Left_out

Contains the left alternate channel two power value when the call returns and is defined as a double.

enumAmpLUnits_out

Contains the left alternate channel two power amplitude units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:

```
Dim bValid_out As Boolean
Dim dUpperResult_out As Double
Dim sEnumUpperUnits_out As String
Dim dLowerResult_out As Double
Dim sEnumLowerUnits_out As String
bValid_out = SignatureSpectrum. _
GetACPRAlternateChannel2Results(dUpperResult_out, _
sEnumUpperUnits_out, dLowerResult_out, _
sEnumLowerUnits_out)
```

C#.NET Example:

```
//Call to retrieve alternate channel 2 results.
bool bValidResults = false;
//true if the results are valid, false otherwise.
double dUpper = 0.0;//Upper channel value
double dLower = 0.0;//Lower channel value
SignatureSpectrum.AmplitudeUnits auUpperUnits;
//Upper value units
SignatureSpectrum.AmplitudeUnits auLowerUnits;
//Lower value units
bValidResults =
SigSpectrumObj.GetACPRAlternateChannel2Results(out
dUpper, out auUpperUnits, out dLower, out
auLowerUnits);
```

Associated GPIB Commands:

```
:CALCulate<1|2>:ACP:ALT<1|2>:RESult?
```

GetACPRMainChannelResults (SPA)

Description: This method queries the main adjacent channel power measurement results. If the measurement is invalid, the returned result is 0.0 dBm.

API:
```
public void GetACPRMainChannelResults(out bool
bValid_out, out double dCP_out, out AmplitudeUnits
enumAmpUnits_out)
```

Arguments:

bValid_out

Contains a boolean value when the call returns with the following values:

True - valid ACP measurement
False - invalid ACP measurement

dCP_out

Contains the main adjacent channel power value when the call returns and is defined as a double.

enumAmpUnits_out

Contains the main adjacent channel power amplitude units when the call returns and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Dim dACPR_out As Double
Dim sAmplitudeUnits As String
Dim bValid_out As Boolean
bValid_out = SignatureSpectrum. _
GetACPRMainChannelResults(dACPR_out, sAmplitudeUnits)
```

C#.NET Example:
```
//Call to retrieve main channel results.
bool bValidResults = false;
//true if the results are valid, false otherwise.
double dChannelPower = 0.0;//main channel power
SignatureSpectrum.AmplitudeUnits auMainCPUnits;
//upper value units
bValidResults =
SigSpectrumObj.GetACPRMainChannelResults(out
dChannelPower, out auMainCPUnits);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:ACP:MAIN:RESult?
```

GetACPRollOffFactor (SPA)

Description: This method queries the adjacent channel power roll-off factor setting.

API:
```
public void GetACPRollOffFactor(out float
fRollOffFactor_out)
```

Arguments: **fRollOffFactor_out**

Contains the adjacent channel power roll off factor value when the call returns and is defined as a floating point number. Ranges from 0.1 to 1.0 with a default value of 0.22.

VB6 Example:
```
Dim fRollOffFactor_out As Single
fRollOffFactor_out = _
SignatureSpectrum.GetACPRollOffFactor
```

C#.NET Example:
```
//Call to get the adjacent channel power roll-off
factor value from the system.
Single fRollOffFactor_out = 0.0F;//ACP Roll off factor
fRollOffFactor_out =
SigSpectrumObj.GetACPRollOffFactor();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:FACTor:ROLLoff?
```

GetACPRRCFilterState (SPA)

Description: This method queries the adjacent channel power, root raised cosine filter toggle setting.

API:
```
public void GetACPRRCFilterState(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - ACP RRC is On
False - ACP RRC is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetACPRRCFilterState
```

C#.NET Example:
```
//Call to read the ACP RRC state.
bool bSwitchOn_out = false;
bSwitchOn_out = SigSpectrumObj.GetACPRRCFilterState();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:FILTer:RRC?
```

GetACPSymbolRateInHz (SPA)

Description: This method queries the adjacent channel power symbol rate setting.

API:
```
public void GetACPSymbolRateInHz(out double
dSymbolRateinHz_out)
```

Arguments: **dSymbolRateinHz_out**

Contains the adjacent channel power symbol rate value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 3.84 MHz.

VB6 Example:
```
Dim dSymbolRateinHz_out As Double
dSymbolRateinHz_out = _
SignatureSpectrum.GetACPSymbolRateInHz
```

C#.NET Example:
```
//Call to read the adjacent channel power symbol rate.
double dSymbolRateinHz_out = 0.0;
dSymbolRateinHz_out =
SigSpectrumObj.GetACPSymbolRateInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:SRATe?`

GetAmplitudeUnits (SPA)

Description: This method queries the current amplitude units setting.

API:
```
public void GetAmplitudeUnits(out ActiveAmpUnits
enumAmpUnits_out)
```

Arguments: **enumAmpUnits_out**

Contains the amplitude units when the call returns and is defined as an enumeration constant with the following values:

"dBm"
"dBmV"
"dBuV"
"W"
"V"

VB6 Example:
```
Dim enumAmpUnits_out As String
enumAmpUnits_out = SignatureSpectrum.GetAmplitudeUnits
```

C#.NET Example:
```
//Call to read the amplitude units from the system.
SignatureSpectrum.ActiveAmpUnits AmpUnits_out;
AmpUnits_out = SigSpectrumObj.GetAmplitudeUnits();
```

Associated GPIB Commands: `:CALCulate<1|2>:UNIT:POWer?`

GetAttenuationIndB (SPA)

Description: This method queries the current attenuation setting.

API:
```
public void GetAttenuationIndB(out int newValue_out)
```

Arguments: **iValue_out**

Contains the attenuation value when the call returns and is defined as an integer. Ranges from 0 dB to 62 dB with Auto set as default.

VB6 Example:
```
Dim iValue_out As Integer
iValue_out = SignatureSpectrum.GetAttenuationIndB
```

C#.NET Example:
```
//Call to read the attenuation value from the system.
int iValue_out;
iValue_out = SigSpectrumObj.GetAttenuationIndB();
```

Associated GPIB Commands:
```
:INPut<1|2>:ATTenuation?
```

GetAttenuationModeAuto (SPA)

Description: This method queries the attenuation mode toggle setting.

API:
```
public void GetAttenuationModeAuto(out bool bAuto)
```

Arguments: **bAuto**

Contains a boolean value when the call returns with the following values:

True - auto mode selected
False - manual mode selected

VB6 Example:
```
Dim bAuto As Boolean
bAuto = SignatureSpectrum.GetAttenuationModeAuto
```

C#.NET Example:
```
//Call to read the attenuation mode setting.
bool bAuto_out = false;
bAuto_out = SigSpectrumObj.GetAttenuationModeAuto();
```

Associated GPIB Commands:
```
:INPut<1|2>:ATTenuation?
```

GetCCDFRBW (SPA)

Description: Gets CCDF RBW for the spectrum measurement.

API:
```
public void GetCCDFRBW(out double dVal, out
_FrequencyUnits enumFrequencyUnits)
```

Arguments: **dVal**

Contains CCDF RBW value when the call is returned.

enumFrequencyUnits

Contains the frequency units when the call is returned and is defined as an enumeration constant with the following value:

“Hz”, “KHz”, “MHz”, “GHz”

VB6 Example:
```
Dim dVal as Double Dim enumFrequencyUnits as String
dVal =
SignatureSpectrum.GetCCDFRBW(enumFrequencyUnits)
```

C#.NET Example:
```
double dVal = 0.0; SignatureWCDMA._FrequencyUnits
enumFrequencyUnits =
SignatureWCDMA._FrequencyUnits.KHz; dVal =
SigSpectrumObj.GetCCDFRBW(out enumFrequencyUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CCDF:RBW?
```

GetCCDFResults (SPA)

Description: Gets CCDF results for the spectrum measurement.

API:

```
public void GetCCDFResults(out double Mean,out double
Peak,out double Crest,out double TargetSamples,out
double AcquiredSamples,out double p1_prob,out double
p01_prob,out double p001_prob,out double
p0001_prob,out double p00001_prob,out double
p000001_prob,out double p0000001_prob,out bool
Valid,out AmplitudeUnits enumAmplitudeUnits)
```

Arguments: The arguments get the following CCDF results:
Mean, Peak, Crest, TargetSamples, AcquiredSamples, p1_prob, p01_prob, p001_prob, p0001_prob, p00001_prob, p000001_prob, p0000001_prob, Valid, enumAmplitudeUnits

VB6 Example:

```
Dim Mean as double Dim Peak as Double Dim Crest as
Double Dim TargetSamples as Double Dim AcquiredSamples
as Double Dim p1_prob as Double Dim p01_prob as Double
Dim p001_prob as Double Dim p0001_prob as Double Dim
p00001_prob as Double Dim p000001_prob as Double Dim
p0000001_prob as Double Dim Valid as Boolean Dim
enumAmplitudeUnits as String Mean =
SignatureSpectrum.GetCCDFResults(Mean, Peak,Crest,
TargetSamples, AcquiredSamples,p1_prob,p01_prob,
p001_prob,p0001_prob,p00001_prob,p000001_prob,
p0000001_prob, Valid, enumAmplitudeUnits)
```

C#.NET Example:

```
double Mean = 0.0; double Peak = 0.0; double Crest =
0.0; double TargetSamples = 0.0; double
AcquiredSamples = 0.0; double p001_prob=0.0 double
p0001_prob = 0.0; double p00001_prob = 0.0; double
p000001_prob = 0.0; double p0000001_prob = 0.0; bool
Valid = true; SignatureWCDMA.AmplitudeUnits
enumAmplitudeUnits =
SignatureWCDMA.AmplitudeUnits.dBm;
SigSpectrumObj.GetCCDFResultsout(out Mean,out Peak,out
Crest,out  TargetSamples,out AcquiredSamples,out
p1_prob,out p01_prob,out p001_prob,out  p0001_prob,out
p00001_prob,out p000001_prob,out p0000001_prob,out
Valid,out  enumAmplitudeUnits);
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:CCDF:RESults?
```

`GetCenterFrequencyInHz` (SPA)

Description: This method queries the center frequency setting.

API:
```
public void GetCenterFrequencyInHz(out double
dnewValueinHz_out)
```

Arguments: **dnewValueinHz_out**

Contains the center frequency value when the call returns and is defined as a double. Ranges from 5 Hz to 8.079999995 GHz.

VB6 Example:
```
Dim dnewValueinHz_out As Double
dnewValueinHz_out = _
SignatureSpectrum.GetCenterFrequencyInHz
```

C#.NET Example:
```
//Call to read the center frequency from the system.
double dValue_out = 0.0;
//The dValue_out contains the frequency value when the
following call gets executed.
dValue_out = SigSpectrumObj.GetCenterFrequencyInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:FREQuency:CENTer?
```

GetCFStepSizeInHz (SPA)

Description: This method queries the current center frequency step size setting.

API:
```
public void GetCFStepSizeInHz(out CFStepSize
enumCFStepSize_out, out double dnewValueinHz_out)
```

Arguments: **enumCFStepSize_out**

Contains the center frequency step size setting when the call returns and is defined as an enumeration constant with the following values:

"DefaultCF"
"OneTenth"
"OneTenthRbw"
"OneHalf"
"OneHalfRbw"
"PercentageofSpan"
"PercentageofRbw"
"EqualsCenter"
"EqualsMarker"
"EqualsSpan"
"ManualCF"

dnewValueinHz_out

Contains the center frequency step size value when the call returns and is defined as a double. The default value is One-Tenth of frequency (when in the frequency domain mode) or OneTenth of RBW (when in the time domain mode).

VB6 Example:
```
Dim dnewValueinHz_out As Double
Dim enumCFStepSize_out As String
dnewValueinHz_out = SignatureSpectrum. _
GetCFStepSizeInHz(enumCFStepSize_out)
```

C#.NET Example:
```
//Call to read the center frequency step size setting
from the system.
SignatureSpectrum.CFStepSize enumCFStepSizeMode_out;
double dStepSize = 0.0; //Step Size in Hz.
dStepSize = SigSpectrumObj.GetCFStepSizeInHz(out
enumCFStepSizeMode_out);
```

Associated GPIB Commands: None

GetChannelPowerBandwidthInHz (SPA)

Description: This method queries the channel power bandwidth setting.

API:
```
public void GetChannelPowerBandwidthInHz(out double
dBandwidthinHz_out)
```

Arguments: **dBandwidthInHz_out**

Contains the channel power bandwidth value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

VB6 Example:
```
Dim dBandwidthinHz_out As Double
dBandwidthinHz_out = _
SignatureSpectrum.GetChannelPowerBandwidthInHz
```

C#.NET Example:
```
//Call to read the channel power bandwidth setting
from the system.
double dBandwidthInHz_out;
dBandwidthInHz_out =
SigSpectrumObj.GetChannelPowerBandwidthInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CHP:BANDwidth?
```

GetChannelPowerDivisionPerHzState (SPA)

Description: This method queries the channel power division/Hz toggle setting.

API:
```
public void GetChannelPowerDivisionPerHzState(out bool
bDisplay_out)
```

Arguments: **bDisplay_out**

Contains a boolean value when the call returns with the following values:

True - channel power division/Hz is On
False - channel power division/Hz is Off

VB6 Example:
```
Dim bDisplay_out As Boolean
bDisplay_out = _
SignatureSpectrum.GetChannelPowerDivisionPerHzState
```

C#.NET Example:
```
//Call to read the channel power division/Hz state.
bool bDisplay_out = false
bDisplay_out =
SigSpectrumObj.GetChannelPowerDivisionPerHzState();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CHP:HZ:STATe?
```

GetChannelPowerFFTState (SPA)

Description: This method queries the channel power fast fourier transform toggle setting.

API:
```
public void GetChannelPowerFFTState(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - FFT is On
False - FFT is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetChannelPowerFFTState
```

C#.NET Example:
```
//Call to read the channel power FFT state.
bool bSwitchOn_out = false;
bSwitchOn_out =
SigSpectrumObj.GetChannelPowerFFTState();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CHP:FFT:STATe?
```

GetChannelPowerMeasurementDomain (SPA)

Description: This method queries the channel power measurement domain mode.

API:
```
public void GetChannelPowerMeasurementDomain(out
MeasurementDomain enumMeasurementDomain_out)
```

Arguments: **enumMeasurementDomain_out**

Contains the channel power measurement domain setting when the call returns and is defined as an enumeration constant with the following value:

"Freq_Domain"

VB6 Example:
```
Dim enumMeasurementDomain_out As String
enumMeasurementDomain_out = _
SignatureSpectrum.GetChannelPowerMeasurementDomain
```

C#.NET Example:
```
//Call to read the channel power measurement domain
setting from the system.
SignatureSpectrum.MeasurementDomain
enumMeasurementDomain_out;//Frequency.
enumMeasurementDomain_out =
SigSpectrumObj.GetChannelPowerMeasurementDomain();
```

Associated GPIB Commands: None

GetChannelPowerOBMMode (SPA)

Description: This method queries the channel power one-button-measurement mode.

API:
```
public void GetChannelPowerOBMMode(out OBMMode
enumOBMMode_out)
```

Arguments: **enumOBMMode_out**

Contains the channel power one-button-measurement mode setting when the call returns and is defined as an enumeration constant with the following values:

"Relative"
"Absolute"

VB6 Example:
```
Dim enumOBMMode_out As String
enumOBMMode_out = _
SignatureSpectrum.GetChannelPowerOBMMode
```

C#.NET Example:
```
//Call to read the channel power one-button-
measurement mode from the system.
SignatureSpectrum.OBMMode enumOBMMode_out;
//Absolute or Relative
enumOBMMode_out =
SigSpectrumObj.GetChannelPowerOBMMode();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CHP:TYPE?
```

GetChannelPowerResults (SPA)

Description: This method queries the channel power measurement results.

API:

```
public void GetChannelPowerResults(out bool
bResultsValid_out, out AmplitudeUnits
enumAmplitudeUnits_out, out double dPowerLevel_out)
```

Arguments:

bResultsValid_out

Contains a boolean value when the call returns with the following values:

True - measurement passed
False - measurement failed

enumAmplitudeUnits_out

Contains the channel power result units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

dPowerLevel_out

Contains the channel power result value when the call returns and is defined as a double.

VB6 Example:

```
Dim bResultsValid_out As Boolean
Dim enumAmplitudeUnits_out As String
Dim dPowerLevel_out As Double
bResultsValid_out = _
SignatureSpectrum.GetChannelPowerResults _
(enumAmplitudeUnits_out, dPowerLevel_out)
```

C#.NET Example:

```
//Call to retrieve the channel power results from the
system.
bool bValidResults = false;
//true if the results are valid, false otherwise.
double dChannelPower = 0.0;//main channel power.
SignatureSpectrum.AmplitudeUnits auMainCPUnits;
//Upper value units.
bValidResults =
SigSpectrumObj.GetChannelPowerResults(out
auMainCPUnits, out dChannelPower);
```

Associated GPIB Commands:

```
:CALCulate<1|2>:CHP:RESult?
```

GetChannelPowerRollOffFactor (SPA)

Description: This method queries the channel power roll-off factor setting.

API:
```
public void GetChannelPowerRollOffFactor(out float
fRollOffFactor_out)
```

Arguments: **fRollOffFactor_out**

Contains the channel power roll off factor value when the call returns and is defined as a floating point number. Ranges from 0.1 to 1.0 with a default value of 0.22.

VB6 Example:
```
Dim fRollOffFactor_out As Single
fRollOffFactor_out = _
SignatureSpectrum.GetChannelPowerRollOffFactor
```

C#.NET Example:
```
//Call to get the channel power roll-off factor value
from the system.
Single fRollOffFactor_out = 0.0F;
fRollOffFactor_out =
SigSpectrumObj.GetChannelPowerRollOffFactor();
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:FACTor:ROLLoff?`

GetChannelPowerRRCFilterState (SPA)

Description: This method queries the channel power root raised cosine filter setting.

API:
```
public void GetChannelPowerRRCFilterState(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - RRC is On
False - RRC is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetChannelPowerRRCFilterState
```

C#.NET Example:
```
//Call to read the channel power RRC filter setting.
bool bSwitchOn_out = false;
bSwitchOn_out =
SigSpectrumObj.GetChannelPowerRRCFilterState();
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:FILTer:RRC?`

GetChannelPowerSymbolRateInHz (SPA)

Description: This method queries the current channel power symbol rate setting.

API:
```
public void GetChannelPowerSymbolRateInHz(double
dSymbolRateinHz_out)
```

Arguments: **dSymbolRateinHz_out**

Contains the channel power symbol rate value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 3.84 MHz.

VB6 Example:
```
Dim dSymbolRateinHz_out As Double
dSymbolRateinHz_out = _
SignatureSpectrum.GetChannelPowerSymbolRateInHz
```

C#.NET Example:
```
//Call to get the channel power symbol rate setting
from the system.
double dSymbolRateinHz_out = 0.0;
dSymbolRateinHz_out =
SigSpectrumObj.GetChannelPowerSymbolRateInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:SRATe?`

GetCPNoiseCompensationState (SPA)

Description: This method queries the channel power noise compensation toggle setting.

API:
```
public void GetCPNoiseCompensationState(out bool
bSwitchOn_out)
```

Arguments: **bSwitchOn_out**

Contains a boolean value when the call returns with the following values:

True - noise compensation is On
False - noise compensation is Off

VB6 Example:
```
Dim bSwitchOn_out As Boolean
bSwitchOn_out = _
SignatureSpectrum.GetCPNoiseCompensationState
```

C#.NET Example:
```
//Call to read the channel power noise compensation
state.
bool bSwitchOn_out = false;
bSwitchOn_out =
SigSpectrumObj.GetCPNoiseCompensationState();
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:NOISecomp:STATe?`

GetCurrentMarker (SPA)

Description: This method queries the current active marker number.

API: `public void GetCurrentMarker(out int iMarkerNum_out)`

Arguments: **sMarkerNum_out**

Contains the active marker number when the call returns and is defined as an integer. Ranges from 1 to 5. When all of the markers are off, the sMarkerNum_out paremeters is equal to "–1."

VB6 Example:

```
Dim iMarkerNum_out As Integer
iMarkerNum_out = SignatureSpectrum.GetCurrentMarker
```

C#.NET Example:

```
//Call to get the current active marker in the SPA
mode.
int iMarkerNum_out = 0; //Valid marker number: 1 to 5.
iMarkerNum_out = SigSpectrumObj.GetCurrentMarker();
```

Associated GPIB Commands: `:CALCulate<1|2>:MARKer:ACTive?`

GetExternalTriggerLevelInVolts (SPA)

Description: This method queries the current external trigger voltage level setting.

API: `public void GetExternalTriggerLevelInVolts(out double dnewValueinVolts_out)`

Arguments: **dnewValueinVolts_out**

Contains the external trigger level value when the call returns and is defined as a double. Ranges from –10V to 10V with a default value of 1.4V TTL.

VB6 Example:

```
Dim dnewValueinVolts_out As Double
dnewValueinVolts_out = _
SignatureSpectrum.GetExternalTriggerLevelInVolts
```

C#.NET Example:

```
//Call to get the external trigger level value from
the system.
double dnewValueinVolts_out = 0.0;
dnewValueinVolts_out =
SigSpectrumObj.GetExternalTriggerLevelInVolts();
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:LEVel:EXTernal?`

GetFrequencyMarkerPositionInHz (SPA)

Description: This method queries the frequency position of the indicated marker.

Precondition: The queried marker should be turned on before making this call using SetMarkerState. The behavior otherwise is not defined.

API:
```
public void GetFrequencyMarkerPositionInHz(short
sMarkerNum_in, out double XPositioninHz_out)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

XPositioninHz_out

Contains the frequency value when the call returns and is defined as a double. Ranges from the frequency range of the instrument.

VB6 Example:
```
Dim XPositioninHz_out As Double
Const sMarkerNum_in = 1
XPositioninHz_out = _
SignatureSpectrum.GetFrequencyMarkerPositionInHz _
(sMarkerNum_in)
```

C#.NET Example:
```
//Call to read marker number three’s frequency from
the system when in the SPA mode.
short sMarkerNum_in = 3;
double dValue_out = 0.0;
dValue_out =
SigSpectrumObj.GetFrequencyMarkerPositionInHz(sMarkerN
um_in);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:X?
```

GetFrequencyOffsetInHz (SPA)

Description: This method queries the center frequency offset parameter.

API:
```
public void GetFrequencyOffsetInHz(out double
dValueinHz_out)
```

Arguments: **dValueinHz_out**

Contains the frequency offset value when the call returns and is defined as a double. Ranges from –100 GHz to +100 GHz with a default value of 0 Hz.

VB6 Example:
```
Dim dValueinHz_out As Double
dValueinHz_out = _
SignatureSpectrum.GetFrequencyOffsetInHz
```

C#.NET Example:
```
//Call to read the frequency offset value from the
system when in the SPA mode.
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetFrequencyOffsetInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:OFFSet?`

GetFrequencySpanInHz (SPA)

Description: This method queries the current frequency span.

API:
```
public void GetFrequencySpanInHz(out double
dnewValueinHz_out)
```

Arguments: **dnewValue_out**

Contains the frequency span value when the call returns and is defined as a double. Ranges from 10 Hz to 8 GHz with a default value of 8 GHz.

VB6 Example:
```
Dim dnewValueinHz_out As Double
dnewValueinHz_out = _
SignatureSpectrum.GetFrequencySpanInHz
```

C#.NET Example:
```
//Call to read the frequency span setting from the
system.
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetFrequencySpanInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:SPAN?`

GetGaussianReferenceLineStatus (SPA)

Description: Gets CCDF Gaussian reference line status for the spectrum measurement.

API:
```
public void GetGaussianReferenceLineStatus(out bool
bGaussianReferenceLineStatus)
```

Arguments: **bGaussianReferenceLineStatus**

Contains the boolean value when the call is returned.

VB6 Example:
```
Dim bGaussianReferenceLineStatus as Boolean
bGaussianReferenceLineStatus =
SignatureSpectrum.GetGaussianReferenceLineStatus
```

C#.NET Example:
```
bool bGaussianReferenceLineStatus = true;
bGaussianReferenceLineStatus =
SigSpectrumObj.GetGaussianReferenceLineStatus();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CCDF:GAUSsianreferenceline:STATe?
```

GetMarkerAmplitude (SPA)

Description: This method queries the indicated marker amplitude value. Turns On the marker identified by the input parameter, if it was Off.

API:
```
public void GetMarkerAmplitude(short sMarkerNum_in,
out double YPosition_out, out AmplitudeUnits
enumAmplitudeUnits_out)
```

Arguments:

sMarkerNum_in

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

YPosition_out

Contains the marker amplitude value when the call returns and is defined as a double.

enumAmplitudeUnits_out

Contains the marker amplitude units when the call returns and is defined as an enumeration constant with the following values:

"dBm", "dBmV", "dBuV", "W", "mW", "uW", "nW", "pW", "fW", "aW", "zW", "yW", "V", "mV", "uV", "nV", "pV"

VB6 Example:
```
Dim dMarkerAmplitude As Double
Dim enumAmplitudeUnits_out As String
Const sMarkerNum_in = 1
dMarkerAmplitude = _
SignatureSpectrum.GetMarkerAmplitude _
(sMarkerNum_in, enumAmplitudeUnits_out)
```

C#.NET Example:
```
//Call to read the marker amplitude of marker number
two from the system when in the SPA mode.
SignatureSpectrum.AmplitudeUnits AmpUnits_out;
short sMarkerNum_in = 2;
double dValue_out = 0.0;
dValue_out =
SigSpectrumObj.GetMarkerAmplitude(sMarkerNum_in, out
AmpUnits_out);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:Y?
```

GetMarkerMode (SPA)

Description: This method queries the indicated marker mode setting.

API:
```
public void GetMarkerMode(short sMarkerNum_in, out
MarkerMode enumMarkerMode_out)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

enumMarkerMode_out

Contains the marker mode setting when the call returns and is defined as an enumeration constant with the following values:

"DeltaMarker"
"NormalMarker"

VB6 Example:
```
Dim enumMarkerMode_out As String
Const sMarkerNum_in = 1
enumMarkerMode_out = _
SignatureSpectrum.GetMarkerMode(sMarkerNum_in)
```

C#.NET Example:
```
//Call to read the marker mode of marker number two.
SignatureSpectrum.MarkerMode markerMode_out;
short sMarkerNum_in = 2;
markerMode_out =
SigSpectrumObj.GetMarkerMode(sMarkerNum_in);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<2 to 5>:MODE?
```

GetMarkerState (SPA)

Description: This method queries the indicated marker state setting.

API:
```
public void GetMarkerState(short sMarkerNum_in, out
MarkerState enumMarkerState_out)
```

Arguments:

sMarkerNum_in

Contains the marker number when the call is sent and is defined as a short.

enumMarkerState_out

Contains the marker state setting when the call returns and is defined as an enumeration constant with the following values:

"MarkerOn"
"MarkerOff"

VB6 Example:
```
Dim enumMarkerState_out As String
Const sMarkerNum_in = 1
enumMarkerState_out = _
SignatureSpectrum.GetMarkerState(sMarkerNum_in)
```

C#.NET Example:
```
//Call to read the marker state of marker number two.
SignatureSpectrum.MarkerState enumMarkerState_out;
short sMarkerNum_in = 2;
enumMarkerState_out =
SigSpectrumObj.GetMarkerState(sMarkerNum_in);
```

Associated GPIB Commands: None

GetMixerLevel (SPA)

Description: This method queries the current mixer level setting of the system. The return value and unit is always returned in the active unit setting indicated by the GetAmplitudeUnits (SPA) call.

API:
```
public void GetMixerLevel(out double dnewValue_out,
out AmplitudeUnits enumAmplitudeUnits_out)
```

Arguments: **dnewValue_out**

Contains the mixer level value when the call returns and is defined as a double. Ranges from:

0 dBm to –50 dBm
Default value: –10 dBm

The additional units listed below are supported with the proper conversion.

enumAmplitudeUnits_out

Contains the mixer level units when the call returns and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Dim dnewValue_out As Double
Dim enumAmplitudeUnits_out As String
dnewValue_out = SignatureSpectrum. _
GetMixerLevel(enumAmplitudeUnits_out)
```

C#.NET Example:
```
//Call to read the mixer Level from the system when in
the SPA mode.
SignatureSpectrum.AmplitudeUnits AmpUnits_out;
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetMixerLevel(out
AmpUnits_out);
```

Associated GPIB Commands:
```
:INPut<1|2>:MIXer[:POWer]?
```

GetNumberOfAverages (SPA)

Description: This method queries the number of trace averages setting.

API:
```
public void GetNumberOfAverages(int
iNumOfAverages_out)
```

Arguments: **iNumOfAverages_out**

Contains the number of trace averages value when the call returns and is defined as an integer. Ranges from 1 to 10,000 with a default value of 10.

VB6 Example:
```
Dim iNumOfAverages_out As Integer
iNumOfAverages_out = _
SignatureSpectrum.GetNumberOfAverages
```

C#.NET Example:
```
//Call to read the number of trace averages from the
system when in the SPA mode.
int iNumOfAverages_out;
iNumOfAverages_out =
SigSpectrumObj.GetNumberOfAverages();
```

Associated GPIB Commands: `[:SENSe<1|2>]:SWEep:AVERage?`

GetOBWBandwidthIndB (SPA)

Description: This method queries the current occupied bandwidth setting of the system.

API:
```
public void GetOBWBandwidthIndB(out double
dBandwidthindB_out)
```

Arguments: **dBandwidthindB_out**

Contains the occupied bandwidth value when the call returns and is defined as a double. Ranges from 0.1 dB to 100 dB with a default value of 26 dB.

VB6 Example:
```
Dim dBandwidthindB_out As Double
dBandwidthindB_out = _
SignatureSpectrum.GetOBWBandwidthIndB
```

C#.NET Example:
```
//Call to read the occupied bandwidth setting from the
system.
double dBandwidthindB_out = 0.0;
dBandwidthindB_out =
SigSpectrumObj.GetOBWBandwidthIndB();
```

Associated GPIB Commands: `[:SENSe<1|2>]:OBW:XDBS?`

GetOBWPercentagePower (SPA)

Description: This method queries the occupied bandwidth percentage value.

API:
```
public void GetOBWPercentagePower(out float
fPercentagePower_out)
```

Arguments: **fPercentagePower_out**

Contains the occupied bandwidth percentage value when the call returns and is defined as a floating point number. Ranges from 10% to 100% with a default value of 99%.

VB6 Example:
```
Dim fPercentagePower_out As Single
fPercentagePower_out = _
SignatureSpectrum.GetOBWPercentagePower
```

C#.NET Example:
```
//Call to read the occupied bandwidth percentage
setting from the system.
Single fPercentagePower_out = 0.0F;
fPercentagePower_out =
SigSpectrumObj.GetOBWPercentagePower();
```

Associated GPIB Commands: `[:SENSe<1|2>]:OBW:POWer:PERCent?`

GetOccupiedBWResultsInHz (SPA)

Description: This method queries the occupied bandwidth results of the last measurement.

API:

```
public void GetOccupiedBWResultsInHz(out double
dBWPecentage_out, out double dXdBBW_out )
```

Arguments:

dBWPecentage_out

Contains the percentage occupied bandwidth frequency value when the call returns and is defined as a double.

dXdBBW_out

Contains the XdB occupied bandwidth frequency value when the call returns and is defined as a double.

VB6 Example:

```
Dim dBWPecentage_out As Double
Dim dXdBBW_out As Double
dXdBBW_out = _
SignatureSpectrum.GetOccupiedBWResultsInHz(dBWPecentag
e_out)
```

C#.NET Example:

```
//Call to read the occupied bandwidth results from the
system.
double dOBW_out = 0.0;
double dXdBBW_out = 0.0;
dOBW_out = SigSpectrumObj.GetOccupiedBWResultsInHz(out
dXdBBW_out);
```

Associated GPIB Commands:

```
:CALCulate<1|2>:OBW:POWer:RESult?
```

GetRBWAuto (SPA)

Description: This method queries the resolution bandwidth mode setting.

API: `public void GetRBWAuto(out bool bAuto_out)`

Arguments: **bAuto_out**

Contains a boolean value when the call returns with the following values:

True - auto mode
False - manual mode

VB6 Example:

```
Dim bAuto_out As Boolean
bAuto_out = SignatureSpectrum.GetRBWAuto
```

C#.NET Example:

```
//Call to read the resolution bandwidth mode setting.
bool bAuto_out = false;
bAuto_out = SigSpectrumObj.GetRBWAuto();
```

Associated GPIB Commands: `[:SENSe<1|2>]:BANDwidth[:RESolution]:AUTO?`

GetRBWInHz (SPA)

Description: This method queries the current resolution bandwidth setting.

API: `public void GetRBWInHz(out double dnewValueinHz_out)`

Arguments: **dnewValueinHz_out**

Contains the resolution bandwidth value when the call returns and is defined as a double. Ranges from 10 Hz to 8 MHz with Auto as default.

VB6 Example:

```
Dim dnewValueinHz_out As Double
dnewValueinHz_out = SignatureSpectrum.GetRBWInHz
```

C#.NET Example:

```
//Call to read the RBW value from the system.
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetRBWInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:BANDwidth[:RESolution]?`

`GetReferenceLevel` (SPA)

Description: This method queries the reference level setting.

API:
```
public void GetReferenceLevel(out double
dnewValue_out, out AmplitudeUnits enumAmpUnits_out)
```

Arguments: **dnewValue_out**

Contains the reference level value when the call returns and is defined as a double. Range from 30 dBm to –150 dBm with a default value of 0 dBm.

enumAmpUnits_out

Contains the reference level units when the call returns and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Dim dnewValue_out As Double
Dim enumAmpUnits_out As String
dnewValue_out = SignatureSpectrum. _
GetReferenceLevel("enumAmpUnits_out")
```

C#.NET Example:
```
//Call to read the reference level from the system.
SignatureSpectrum.AmplitudeUnits AmpUnits_out;
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetReferenceLevel(out
AmpUnits_out);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel?
```

GetReferenceLevelOffsetIndB (SPA)

Description: This method queries the reference level offset setting.

API:
```
public void GetReferenceLevelOffsetIndB(out double
dRefLvlOffsetValueindB_out)
```

Arguments: **dRefLvlOffsetValueindB_out**

Contains the reference level offset value when the call returns and is defined as a double. Ranges from –300 dB to +300 dB.

VB6 Example:
```
Dim dnewValue_out As Double
dnewValue_out = _
SignatureSpectrum.GetReferenceLevelOffsetIndB
```

C#.NET Example:
```
//Call to read the reference level offset from the
system.
double dValue_out = 0.0;
dValue_out =
SigSpectrumObj.GetReferenceLevelOffsetIndB();
```

Associated GPIB Commands: `:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel:OFFSet?`

GetScaleTypeLinear (SPA)

Description: This method queries the scale type setting.

API:
```
public void GetScaleTypeLinear(out bool
bScaleType_out)
```

Arguments: **bScaleType_out**

Contains a boolean value when the call returns with the following values:

True - linear scale type
False - logarithmic scale type

VB6 Example:
```
Dim bScaleType_out As Boolean
bScaleType_out = SignatureSpectrum.GetScaleTypeLinear
```

C#.NET Example:
```
//Call to read the scale type setting.
bool bScaleType_out = false;
//true if linear, false otherwise.
bScaleType_out = SigSpectrumObj.GetScaleTypeLinear();
```

Associated GPIB Commands: `:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y:SPACing?`

GetScalingPerDivision (SPA)

Description: This method queries the vertical graticule scaling per division setting.

API:
```
public void GetScalingPerDivision(out float
fValuePerDivision_out)
```

Arguments: **fValuePerDivision_out**

Contains the scaling/division value when the call returns and is defined as a floating point number. Ranges from 0.1 dB to 20 dB with a default value of 10 dB. The resolution is 0.1 dB for the 0.1 dB to 1 dB range and 1 dB for the 1 dB to 20 dB range.

VB6 Example:
```
Dim fValuePerDivision_out As Single
fValuePerDivision_out = _
SignatureSpectrum.GetScalingPerDivision
```

C#.NET Example:
```
//Call to get the current scale resolution of the
system.
Single fScalingPerDiv = 0.0F;
fScalingPerDiv =
SigSpectrumObj.GetScalingPerDivision();
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to
5>:Y[:SCALe]:PDIVision?
```

GetSpanToRBWRatio (SPA)

Description: This method queries the span/RBW ratio setting.

API:
```
public void GetSpanToRBWRatio(out double
dnewValue_out, out Multiplier enumMulUnit_out)
```

Arguments: **dnewValue_out**

Contains the span/RBW ratio value when the call returns and is defined as a double. Ranges from 2 to 10,000 with a default value of 50.

enumMulUnit_out

Contains the span/RBW ratio multiplier when the call returns with the following value:

"e00" for a multiplier of 1

VB6 Example:
```
Dim dnewValue_out As Double
dnewValue_out = _
SignatureSpectrum.GetSpanToRBWRatio
```

C#.NET Example:
```
//Call to get the span to RBW ratio from the system.
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetSpanToRBWRatio();
```

Associated GPIB Commands: `[:SENSe<1|2>]:BANDwidth[:RESolution]:RATio?`

GetStartFrequencyInHz (SPA)

Description: This method queries the current start frequency of the sweep.

API:
```
public void GetStartFrequencyInHz(out double
dnewValueinHz_out)
```

Arguments: **dnewValue_out**

Contains the start frequency value when the call returns and is defined as a double. Ranges from 0 Hz to (Stop Frequency – Minimum Span).

VB6 Example:
```
Dim dnewValueinHz_out As Double
dnewValueinHz_out = _
SignatureSpectrum.GetStartFrequencyInHz
```

C#.NET Example:
```
//Call to read the start frequency from the system.
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetStartFrequencyInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:STARt?`

GetStopFrequencyInHz (SPA)

Description: This method queries the stop frequency of the sweep.

API:
```
public void GetStopFrequencyInHz(out double
dnewValueinHz_out)
```

Arguments: **dnewValue_out**

Contains the stop frequency value when the call returns and is defined as a double. Ranges from (Start Frequency – Minimum Span) to 8.08 GHz.

VB6 Example:
```
Dim dnewValueinHz_out As Double
dnewValueinHz_out = _
SignatureSpectrum.GetStopFrequencyInHz
```

C#.NET Example:
```
//Call to read the stop frequency from the system.
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetStopFrequencyInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:FREQuency:STOP?
```

GetSweepCount (SPA)

Description: This method queries the current sweep count when trace averaging is selected.

API:
```
public void GetSweepCount(out int
iMaxAverageCount_out, out int iSweepCount_out, out
bool bAverageModeExists_out)
```

Arguments:

iMaxAverageCount_out

Contains the averaging count value when the call returns and is defined as an integer. Ranges from 1 to 10,000 with a default value of 10.

iSweepCount_out

Contains the current sweep count value when the call returns and is defined as an integer. Ranges from 1 to 10,000 with a default value of 10.

bAverageModeExists_out

Contains the boolean switch when the call returns with the following values:

True - averaging
False - no averaging

VB6 Example:
```
Dim iMaxAverageCount_out As Integer
Dim iSweepCount_out As Integer
Dim bAverageModeExists_out As Boolean
iSweepCount_out = _
SignatureSpectrum.GetSweepCount(iMaxAverageCount_out,
bAverageModeExists_out)
```

C#.NET Example:
```
//Call to read the current sweep count value from the
system.
int iCurrentSweepCount_out = 0;
//Currently completed sweep count.
int iNumOfAverages_out = 0;
//User specified number of trace averages.
bool bTraceAveragingMode = false;
//Is the system in the trace averaging mode?
iNumOfAverages_out = SigSpectrumObj.GetSweepCount(out
iCurrentSweepCount_out, out bTraceAveragingMode);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:SWEep:AVERage?
```

GetSweepMode (SPA)

Description: This method queries the sweep mode setting.

API:
```
public void GetSweepMode(out SweepMode
enumSweepMode_out)
```

Arguments: **enumSweepMode_out**

Contains the sweep mode when the call returns and is defined as an enumeration constant with the following values:

"Continuous"
"Single"

VB6 Example:
```
Dim enumSweepMode_out As String
enumSweepMode_out = SignatureSpectrum.GetSweepMode
```

C#.NET Example:
```
//Call to read the SPA sweep mode from the system.
SignatureSpectrum.SweepMode SwMode_out;
//Single or Continuous
SwMode_out = SigSpectrumObj.GetSweepMode();
```

Associated GPIB Commands: `:INITiate<1|2>:CONTinuous?`

GetSweepTimeAuto (SPA)

Description: This method queries the sweep time mode setting.

API:
```
public void GetSweepTimeAuto(out bool bAuto_out)
```

Arguments: **bAuto_out**

Contains a boolean value when the call returns with the following values:

True - auto mode
False - manual mode

VB6 Example:
```
Dim bAuto_out As Boolean
bAuto_out = SignatureSpectrum.GetSweepTimeAuto
```

C#.NET Example:
```
//Call to read the sweep time setting of manual or
automatic.
bool bSweepTimeAuto = false;
bSweepTimeAuto = SigSpectrumObj.GetSweepTimeAuto();
```

Associated GPIB Commands: `[:SENSe<1|2>]:SWEep:TIME:AUTO?`

`GetSweepTimeInSecs` (SPA)

Description: This method queries the current sweep time.

API:
```
public void GetSweepTimeInSecs(out double
dnewValueinSecs_out)
```

Arguments: **dnewValueinSecs_out**

Contains the sweep time value when the call returns and is defined as a double. Ranges from 5 ms to 10 ks with Auto as default.

VB6 Example:
```
Dim dnewValueinSecs_out As Double
dnewValueinSecs_out = _
SignatureSpectrum.GetSweepTimeInSecs
```

C#.NET Example:
```
//Call to read the sweep time from the system.
double dSweepTimeInSecs_out = 0.0;
dSweepTimeInSecs_out =
SigSpectrumObj.GetSweepTimeInSecs();
```

Associated GPIB Commands: `[:SENSe<1|2>]:SWEep:TIME?`

`GetSweepTimeMode` (SPA)

Description: This method queries the current sweep time mode setting.

API:
```
public void GetSweepTimeMode(out SweepTimeMode
enumSpeedTimeMode_out)
```

Arguments: **enumSpeedTimeMode_out**

Contains the sweep time mode when the call returns and is defined as an enumeration constant with the following values:

"Speed"
"Accuracy"

VB6 Example:
```
Dim enumSpeedTimeMode_out As String
enumSpeedTimeMode_out = _
SignatureSpectrum.GetSweepTimeMode
```

C#.NET Example:
```
//Call to read the sweep time mode from the system.
SignatureSpectrum.SweepTimeMode enumSTMode_out;
//Accuracy or Speed.
enumSTMode_out = SigSpectrumObj.GetSweepTimeMode();
```

Associated GPIB Commands: `[:SENSe<1|2>]:SWEep:TIME:AUTO:COUPling?`

GetSweepType (SPA)

Description: This method queries the sweep type. The settings are Swept or FFT. Wideband FFT is not covered.

API:
```
public void GetSweepType(out SweepType
enumSweepType_out)
```

Arguments: **enumSweepType_out**

Contains the sweep type when the call returns and is defined as an enumeration constant with the following values:

"Normal"
"FFT"

VB6 Example:
```
Dim enumSweepType_out As String
enumSweepType_out = SignatureSpectrum.GetSweepType
```

C#.NET Example:
```
//Call to the read the sweep type from the system.
SignatureSpectrum.SweepType swType_out;
//FFT or Normal
swType_out = SigSpectrumObj.GetSweepType();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:BANDwidth[:RESolution]:TYPE?
```

GetTargetNumberOfSamples (SPA)

Description: Gets CCDF target number of samples for the spectrum measurement.

API:
```
public void GetTargetNumberofSamples(out double
dSamples)
```

Arguments: **dSamples**

Contains CCDF target number of samples when the call is returned.

VB6 Example:
```
Dim dSamples as Double dSamples =
SignatureSpectrum.GetTargetNumberOfSamples
```

C#.NET Example:
```
double dSamples = 0.0; dSamples =
SigSpectrumObj.GetTargetNumberOfSamples();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CCDF:TARget:SAMPles?
```

GetTimeMarkerPositionInSecs (SPA)

Description: This method queries the indicated marker's time position in the zero span mode.

API:
```
public void GetTimeMarkerPositionInSecs(short
sMarkerNum_in, out double XPositioninSecs_out)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from1 to 5.

XPositioninSecs_out

Contains the marker time value when the call returns and is defined as a double. Ranges from 0.1 ms to 10 ks.

VB6 Example:
```
Dim XPositioninSecs_out As Double
Const sMarkerNum_in = 1
XPositioninSecs_out = _
SignatureSpectrum.GetTimeMarkerPositionInSecs _
(sMarkerNum_in)
```

C#.NET Example:
```
//Call to read the time marker value for marker number
two when in the zero span mode.
short sMarkerNum_in = 2;
double dValue_out;
dValue_out =
SigSpectrumObj.GetTimeMarkerPositionInSecs(sMarkerNum_
in);
```

Associated GPIB Commands: None

GetTOIResults (SPA)

Description: This method queries the third order intercept result of the last measurement.

API:
```
public void GetTOIResults(out double dPowerLevel_out)
```

Arguments: **dPowerLevel_out**

Contains the third order intercept result value when the call returns and is defined as a double.

VB6 Example:
```
Dim dPowerLevel_out As Double
dPowerLevel_out = SignatureSpectrum.GetTOIResults
```

C#.NET Example:
```
//Call to read the TOI results from the system.
double dPowerLevel_out = 0.0;
dPowerLevel_out = SigSpectrumObj.GetTOIResults();
```

Associated GPIB Commands:
```
:CALCulate<1|2>:TOI:RESult?
```

GetTraceAttachedToMarker (SPA)

Description: This method queries the trace data for a specified marker.

API:
```
public void GetTraceAttachedToMarker(short
sMarkerNum_in, out short sTraceNum_out)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

sTraceNum_out

Contains the trace number when the call is sent and is defined as a short. Ranges from 1 to 5.

VB6 Example:
```
Dim sTraceNum_out As Integer
Const sMarkerNum_in = 1
sTraceNum_out = _
SignatureSpectrum.GetTraceAttachedToMarker _
(sMarkerNum_in)
```

C#.NET Example:
```
//Call to read the trace attached to Marker number
two.
short sMarkerNum_in = 2;
short sTraceNum_out = 0;
sTraceNum_out =
SigSpectrumObj.GetTraceAttachedToMarker(sMarkerNum_in)
;
```

Associated GPIB Commands: `:CALCulate<1|2>:MARKer<1 to 5>:TRACe?`

GetTraceData (SPA)

Description: This method queries the trace data for a specified trace.

API:
```
public void GetTraceData(short sTraceNum_in, out
float[] fTraceData_out)
```

Arguments: **sTraceNum_in**

Contains the trace number when the call is sent and is defined as a short. Ranges from 1 to 5.

fTraceData_out

Contains the trace data when the call is sent and is defined as a floating point number.

VB6 Example:
```
Dim fTraceData_out() As Single
Dim iTraceDataSize As Integer
Const sTraceNum_in = 1
iTraceDataSize = _
SignatureSpectrum.GetTraceDataSize(sTraceNum_in)
ReDim fTraceData_out(iTraceDataSize)
fTraceData_out = _
SignatureSpectrum.GetTraceData(sTraceNum_in)
```

C#.NET Example:
```
//Call to read the trace data corresponding to trace
number two from the system.
short sTraceNum_in = 2;
int iTraceDataSize = 0;
iTraceDataSize =
SigSpectrumObj.GetTraceDataSize(sTraceNum_in);
//Get the trace data size for Trace number two.
System.Single[] sTraceData = new
System.Single[iTraceDataSize];
sTraceData =
SigSpectrumObj.GetTraceData(sTraceNum_in);
//Get the tracedata corresponding to Trace number two
from the system.
```

Associated GPIB Commands:
```
:TRACe<1 to 5>?
```

GetTraceDataSize (SPA)

Description: This method queries the trace data size for a specified trace.

API:
```
public void GetTraceDataSize(short sTraceNum_in, out
int iDataSize_out)
```

Arguments:

sTraceNum_in

Contains the trace number when the call returns and is defined as a short. Ranges from 1 to 5.

iDataSize_out

Contains the trace data size value when the call returns and is defined as an integer.

VB6 Example:
```
Dim iDataSize_out as Integer
Const sTraceNum_in = 1
iDataSize_out = _
SignatureSpectrum.GetTraceDataSize(sTraceNum_in)
```

C#.NET Example:
```
//Call to read the trace data size corresponding to
trace number two from the system.
short sTraceNum_in = 2;
int iTraceDataSize = 0;
iTraceDataSize =
SigSpectrumObj.GetTraceDataSize(sTraceNum_in);
//Get the trace data size for trace number two.
```

Associated GPIB Commands: None

GetTraceDetectionType (SPA)

Description: This method queries a specified trace’s detection type.

API:
```
public void GetTraceDetectionType(short sTraceNum_in,
out TraceDetectionType enumTraceDetType_out)
```

Arguments: **sTraceNum_in**

Contains the trace number when the call returns and is defined as a short. Ranges from 1 to 5.

enumTraceDetType_out

Contains the trace detection type when the call returns and is defined as an enumeration constant with the following values:

“DetAuto”
“DetNormal”
“DetMaxPeak”
“DetMinPeak”
“DetSample”
“DetAverage”
“DetRMS”

VB6 Example:
```
Dim enumTraceDetType_out as String
Const sTraceNum_in = 1
enumTraceDetType_out = _
SignatureSpectrum.GetTraceDetectionType(sTraceNum_in)
```

C#.NET Example:
```
//Call to read trace one’s detection type from the
system.
SignatureSpectrum.TraceDetectionType
enumTraceDetType_out;
short sTraceNum_in = 1;
enumTraceDetType_out =
SigSpectrumObj.GetTraceDetectionType(sTraceNum_in);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DETector<1 to 5>?
```

GetTraceMode (SPA)

Description: This method queries the specified trace's mode.

API:
```
public void GetTraceMode(short sTraceNum_in, out
TraceMode enumTraceMode_out)
```

Arguments: **sTraceNum_in**

Contains the trace number when the call returns and is defined as a short. Ranges from 1 to 5.

enumTraceMode_out

Contains the trace mode when the call returns and is defined as an enumeration constant with the following values:

"ClearWrite"
"MaxHold"
"MinHold"
"Average"
"WriteHold"
"Off"

VB6 Example:
```
Dim enumTraceMode_out as String
Const sTraceNum_in = 1
enumTraceMode_out = _
SignatureSpectrum.GetTraceMode(sTraceNum_in)
```

C#.NET Example:
```
//Call to read trace one's mode from the system.
SignatureSpectrum.TraceMode enumTraceMode_out;
short sTraceNum_in = 1;
enumTraceMode_out =
SigSpectrumObj.GetTraceMode(sTraceNum_in);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:MODE?
```

GetTriggerDelayInSecs (SPA)

Description: This method queries the trigger delay setting.

API:
```
public void GetTriggerDelayInSecs(out double
dnewValueinSecs_out)
```

Arguments: **dnewValueinSecs_out**

Contains the trigger delay value when the call returns and is defined as a double. Ranges from 0 ms to 65.5 ms with a default value of 0 ms.

VB6 Example:
```
Dim dnewValueinSecs_out As Double
dnewValueinSecs_out = _
SignatureSpectrum.GetTriggerDelayInSecs
```

C#.NET Example:
```
//Call to read the trigger delay from the system.
double dTriggerDelayInSecs_out = 0.0;
dTriggerDelayInSecs_out =
SigSpectrumObj.GetTriggerDelayInSecs();
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:HOLDoff?
```

`GetTriggerSource` **(SPA)**

Description: This method queries the trigger source setting.

API:
```
public void GetTriggerSource(out SPATriggerSource
enumSPATriggerSource_out)
```

Arguments: **enumSPATriggerSource_out**

Contains the trigger source setting when the call returns and is defined as an enumeration constant with the following values:

"FreeRun"
"WideIF"
"Line"
"External"
"Video"
"ExternalTTL"

VB6 Example:
```
Dim enumSPATriggerSource_out As String
enumSPATriggerSource_out = _
SignatureSpectrum.GetTriggerSource
```

C#.NET Example:
```
//Call to read the trigger source from the system.
SignatureSpectrum.SPATriggerSource
enumSPATriggerSource_out;
enumSPATriggerSource_out =
SigSpectrumObj.GetTriggerSource();
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:SOURce?
```

GetVBWAuto (SPA)

Description: This method queries the video bandwidth mode setting.

API: `public void GetVBWAuto(out bool bAuto_out)`

Arguments: **bAuto_out**

Contains a boolean value when the call returns with the following values:

True - auto mode
False - manual mode

VB6 Example:

```
Dim bAuto_out As Boolean
bAuto_out = SignatureSpectrum.GetVBWAuto
```

C#.NET Example:

```
//Call to read the video bandwidth mode setting.
bool bAuto_out = false;
bAuto_out = SigSpectrumObj.GetVBWAuto();
```

Associated GPIB Commands: `[:SENSe<1|2>]:BANDwidth:VIDeo:AUTO?`

GetVBWInHz (SPA)

Description: This method queries the current video bandwidth setting. A return value of 50 MHz VBW implies that the VBW method is disabled.

API: `public void GetVBWInHz(out double dnewValueinHz_out)`

Arguments: **dnewValueinHz_out**

Contains the video bandwidth value when the call returns and is defined as a double. Ranges from 1 Hz to 10 MHz with Auto as default.

VB6 Example:

```
Dim dnewValueinHz_out As Double
dnewValueinHz_out = SignatureSpectrum.GetVBWInHz
```

C#.NET Example:

```
//Call to read the VBW value from the system.
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetVBWInHz();
```

Associated GPIB Commands: None

GetVBWToRBWRatio (SPA)

Description: This method queries the current VBW/RBW ratio.

API:
```
public void GetVBWToRBWRatio(out double dnewValue_out)
```

Arguments: **dnewValue_out**

Contains the VBW/RBW ratio value when the call returns and is defined as a double. Ranges from 0.001 to 1,000 with a default value of 5.

VB6 Example:
```
Dim dnewValue_out As Double
dnewValue_out = _
SignatureSpectrum.GetVBWToRBWRatio
```

C#.NET Example:
```
//Call to read the VBW to RBW ratio from the system.
double dValue_out = 0.0;
dValue_out = SigSpectrumObj.GetVBWToRBWRatio();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:BANDwidth:VIDeo:RATio?
```

GetVideoTriggerLevel (SPA)

Description: This method queries the current video trigger level setting.

API:
```
public void GetVideoTriggerLevel(out double
dnewValue_out, out AmplitudeUnits
enumAmplitudeUnits_out)
```

Arguments: **dnewValue_out**

Contains the video trigger level value when the call returns and is defined as a double. Ranges from:
Reference Level to (Reference Level – 10 x Scale/Div)
with a default value of:
Reference Level – 0.5 x (10 x Scale/Div)

enumAmplitudeUnits_out

Contains the video trigger level units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Dim dnewValue_out As Double
Dim enumAmplitudeUnits_out As String
dnewValue_out = SignatureSpectrum. _
GetVideoTriggerLevel(enumAmplitudeUnits_out)
```

C#.NET Example:
```
//Call to read the video trigger level value from the
system.
double dValue_out = 0.0;
SignatureSpectrum.AmplitudeUnits enumAmpUnits_out;
dValue_out = SigSpectrumObj.GetVideoTriggerLevel(out
enumAmpUnits_out);
```

Associated GPIB Commands: None

GetWideIFTriggerLevel (SPA)

Description: This method queries the current wideband IF power trigger level setting.

Precondition: The trigger source must be set to IF Power (Wideband) before the call is sent.

API:
```
public void GetWideIFTriggerLevel(out double
dnewValue_out, out AmplitudeUnits enumAmpUnits_out)
```

Arguments: **dnewValue_out**

Contains the wideband IF power trigger level value when the call returns and is defined as a double.

enumAmplitudeUnits_out

Contains the wideband IF power trigger level units when the call returns and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Dim dnewValue_out As Double
Dim enumAmplitudeUnits_out As String
dnewValue_out = SignatureSpectrum. _
GetWideIFTriggerLevel(enumAmplitudeUnits_out)
```

C#.NET Example:
```
//Call to read the wideband IF power trigger level
value from the system.
double dnewValue_out = 0.0;
SignatureSpectrum.AmplitudeUnits enumAmpUnits_out;
dnewValue_out =
SigSpectrumObj.GetWideIFTriggerLevel(out
enumAmpUnits_out);
```

Associated GPIB Commands: None

GetXAxisMaximum (SPA)

Description: Gets CCDF X axis maximum for the spectrum measurement.

API:
```
public void GetXAxisMaximum(out int iXAxisMaximum)
```

Arguments: **iXAxisMaximum**

Contains CCDF X axis maximum value when the call is returned.

VB6 Example:
```
Dim iXAxisMaximum as Integer iXAxisMaximum =
SignatureSpectrum.GetXAxisMaximum
```

C#.NET Example:
```
double iXAxisMaximum = 0.0;iXAxisMaximum =
SigSpectrumObj.GetXAxisMaximum();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CCDF:XAXis:MAXimum?
```

IsNoiseMarker (SPA)

Description: This method queries the marker type setting of the system.

API:
```
public void IsNoiseMarker(short sMarkerNum_in, out
bool bNoiseMarker_out)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

bNoiseMarker_out

Contains the boolean switch when the call returns with the following values:

True - noise marker
False - normal marker

VB6 Example:
```
Dim bNoiseMarker_out As Boolean
Const sMarkerNum_in = 1
bNoiseMarker_out = _
SignatureSpectrum.IsNoiseMarker(sMarkerNum_in)
```

C#.NET Example:
```
//Call to read the marker type setting for marker
number two from the system.
bool bNoiseMarker = false;
short sMarkerNum_in = 2;
bNoiseMarker =
SigSpectrumObj.IsNoiseMarker(sMarkerNum_in);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:FUNCtion:TYPE?
```

IsSweepComplete (SPA)

Description: This method queries the sweep complete status of a single sweep initiated by the AsyncStartSweep (SPA) web service.

Precondition: This call works only when a single sweep is initiated by the AsyncStartSweep (SPA) web service. The behavior is undefined in the Continuous sweep mode.

API:
```
public void IsSweepComplete(out bool bSweepComplete)
```

Arguments: **bSweepComplete**

Contains a boolean value when the call returns with the following values:

True - sweep is complete
False - sweep is not complete

VB6 Example:
```
Dim bSweepComplete As Boolean
bSweepComplete = _
SignatureSpectrum.IsSweepComplete
```

C#.NET Example:
```
//Call to read the sweep complete status.
bool bSweepComplete;
bSweepComplete = SigSpectrumObj.IsSweepComplete();
```

Associated GPIB Commands: None

IsTraceAveragingComplete (SPA)

Description: This method queries the trace averaging status.

API:
```
public void IsTraceAveragingComplete(out bool
bTraceAveragingComplete_out)
```

Arguments: **bTraceAveragingComplete_out**

Contains a boolean value when the call returns with the following values:

True - trace averaging is On and averaging is complete, or trace averaging is Off
False - trace averaging is On and the averaging is not complete

VB6 Example:
```
Dim bTraceAveragingComplete_out As Boolean
bTraceAveragingComplete_out = _
SignatureSpectrum.IsTraceAveragingComplete
```

C#.NET Example:
```
//Call to read the trace averaging complete status.
bool bTraceAveragingComplete_out;
bTraceAveragingComplete_out =
SigSpectrumObj.IsTraceAveragingComplete();
```

Associated GPIB Commands:
```
:INITiate<1|2>:SWEep:AVERage?
```

IsTriggerEdgeRising (SPA)

Description: This method queries the trigger edge setting of the system.

API:
```
public void IsTriggerEdgeRising(out bool
bTriggerEdge_out)
```

Arguments: **bTriggerEdge_out**

Contains the boolean switch when the call returns with the following values:

True - rising edge triggering
False - falling edge triggering

VB6 Example:
```
Dim bTriggerEdge_out As Boolean
bTriggerEdge_out = _
SignatureSpectrum.IsTriggerEdgeRising
```

C#.NET Example:
```
//Call to read the trigger edge setting from the
system.
bool bTriggerEdgeSlope_out;
bTriggerEdgeSlope_out =
SigSpectrumObj.IsTriggerEdgeRising();
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:SLOPe?
```

SetACPAdjacentChannelSpacing (SPA)

Description: This method sets the adjacent channel power, adjacent channel spacing parameter.

API:
```
public void SetACPAdjacentChannelSpacing(double
dBandwidth_in, FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dBandwidth_in**

Contains the adjacent channel power, adjacent channel spacing value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

enumFrequencyUnits_in

Contains the adjacent channel power, adjacent channel spacing units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum. _
SetACPAdjacentChannelSpacing(5#, "MHz")
```

C#.NET Example:
```
//Call to set the adjacent channel power, adjacent
channel spacing to 5 MHz.
double dBandwidth_in = 5.0;
SigSpectrumObj.SetACPAdjacentChannelSpacing(dBandwidth
_in, SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ADJacent:CHSPacing
```

SetACPAdjChannelBandwidth (SPA)

Description: This method sets the adjacent channel power, adjacent channel bandwidth parameter.

API:
```
public void SetACPAdjChannelBandwidth(double
dBandwidth_in, FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dBandwidth_in**

Contains the adjacent channel power, adjacent channel bandwidth value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

enumFrequencyUnits_in

Contains the adjacent channel power, adjacent channel bandwidth units when the call is sent and is defined as an enumeration constant with the following values:

“Hz”
“KHz”
“MHz”
“GHz”

VB6 Example:
```
Call SignatureSpectrum. _
SetACPAdjChannelBandwidth(5#, "MHz")
```

C#.NET Example:
```
//Call to set the adjacent channel power, adjacent
channel bandwidth to 5 MHz.
double dBandwidth_in = 5.0;
SigSpectrumObj.SetACPAdjChannelBandwidth(dBandwidth_in
, SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ADJacent:CHBandwidth
```

SetACPAlternateChannel1Bandwidth (SPA)

Description: This method sets the adjacent channel power, alternate channel one bandwidth parameter.

API:
```
public void SetACPAlternateChannel1Bandwidth(double
dBandwidth_in, FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dBandwidth_in**

Contains the adjacent channel power, alternate channel one bandwidth value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

enumFrequencyUnits_in

Contains the adjacent channel power, alternate channel one bandwidth units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum. _
SetACPAlternateChannel1Bandwidth(5#, "MHz")
```

C#.NET Example:
```
//Call to set the adjacent channel power, alternate
channel one bandwidth to 5 MHz.
double dBandwidth_in = 5.0;
SigSpectrumObj.SetACPAlternateChannel1Bandwidth(dBandw
idth_in, SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:CHBandwidth
```

`SetACPAlternateChannel1Spacing` (SPA)

Description: This method sets the adjacent channel power, alternate channel one spacing parameter.

API:
```
public void SetACPAlternateChannel1Spacing(double
dBandwidth_in, FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dBandwidth_in**

Contains the adjacent channel power, alternate channel one spacing value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 10 MHz.

enumFrequencyUnits_in

Contains the adjacent channel power, alternate channel one spacing units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum. _
SetACPAlternateChannel1Spacing(5#, "MHz")
```

C#.NET Example:
```
//Call to set the adjacent channel power, alternate
channel one spacing to 5 MHz.
double dBandwidth_in = 5.0;
SigSpectrumObj.SetACPAlternateChannel1Spacing(dBandwid
th_in, SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:CHSPacing
```

SetACPAlternateChannel2Bandwidth (SPA)

Description: This method sets the adjacent channel power, alternate channel two bandwidth parameter.

API:
```
public void SetACPAlternateChannel2Bandwidth(double
dBandwidth_in, FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dBandwidth_in**

Contains the adjacent channel power, alternate channel two bandwidth value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

enumFrequencyUnits_in

Contains the adjacent channel power, alternate channel two bandwidth units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum. _
SetACPAlternateChannel2Bandwidth(5#, "MHz")
```

C#.NET Example:
```
//Call to set the adjacent channel power, alternate
channel two bandwidth to 5 MHz.
double dBandwidth_in = 5.0;
SigSpectrumObj.SetACPAlternateChannel2Bandwidth(dBandw
idth_in, SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:CHBandwidth
```

SetACPAlternateChannel2Spacing (SPA)

Description: This method sets the adjacent channel power, alternate channel two spacing parameter.

API:
```
public void SetACPAlternateChannel2Spacing(double
dBandwidth_in, FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dBandwidth_in**

Contains the adjacent channel power, alternate channel two spacing value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 15 MHz.

enumFrequencyUnits_in

Contains the adjacent channel power, alternate channel two spacing units when the call is sent and is defined as an enumeration constant with the following values:

“Hz”
“KHz”
“MHz”
“GHz”

VB6 Example:
```
Call SignatureSpectrum. _
SetACPAlternateChannel2Spacing(5#, "MHz")
```

C#.NET Example:
```
//Call to set the adjacent channel power, alternate
channel two spacing to 5 MHz.
double dBandwidth_in = 5.0;
SigSpectrumObj.SetACPAlternateChannel2Spacing(dBandwid
th_in, SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:ALT<1|2>:CHSPacing
```

SetACPChannelBandwidth (SPA)

Description: This method sets the adjacent channel power channel bandwidth parameter.

API:
```
public void SetACPChannelBandwidth(double
dBandwidth_in, FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dBandwidth_in**

Contains the adjacent channel power channel bandwidth value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

enumFrequencyUnits_in

Contains the adjacent channel power channel bandwidth units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum. _
SetACPChannelBandwidth(5#, "MHz")
```

C#.NET Example:
```
//Call to set the adjacent channel power channel
bandwidth to 5 MHz.
double dBandwidth_in = 5.0;
SigSpectrumObj.SetACPChannelBandwidth(dBandwidth_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:CHBandwidth`

SetACPDivisionPerHzState (SPA)

Description: This method sets the adjacent channel power division/Hz toggle setting.

API: `public void SetACPDivisionPerHzState(bool bDisplay_in)`

Arguments: **bDisplay_in**

Contains the boolean switch when the call is sent with the following values:

True - adjacent channel power division/Hz state On
False - adjacent channel power division/Hz state Off

VB6 Example:

```
Call SignatureSpectrum. _
SetACPDivisionPerHzState(True)
```

C#.NET Example:

```
//Call to enable the adjacent channel power division/
Hz state.
bool bDisplay_in = true;
SigSpectrumObj.SetACPDivisionPerHzState(bDisplay_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:HZ:STATe`

SetACPOBMMode (SPA)

Description: This method sets the adjacent channel power one-button-measurement mode.

API: `public void SetACPOBMMode(OBMMode enumOBMMode_in)`

Arguments: **enumOBMMode_in**

Contains the adjacent channel power one-button-measurement mode setting when the call returns and is defined as an enumeration constant with the following values:

“Relative”
“Absolute”

VB6 Example:

```
Call SignatureSpectrum.SetACPOBMMode("Absolute")
```

C#.NET Example:

```
//Call to set the adjacent channel power one-button-
measurement mode.
SigSpectrumObj.SetACPOBMMode(SignatureSpectrum.OBMMode
.Absolute); //Absolute or Relative.
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:TYPE`

SetACPRollOffFactor (SPA)

Description: This method sets the adjacent channel power roll-off factor parameter.

API:

```
public void SetACPRollOffFactor(float
fRollOffFactor_in)
```

Arguments: **fRollOffFactor_in**

Contains the adjacent channel power roll-off factor value when the call is sent and is defined as a floating point number. Ranges from 0.1 to 1.0 with a default value of 0.22.

VB6 Example:

```
Call SignatureSpectrum.SetACPRollOffFactor(0.22!)
```

C#.NET Example:

```
//Call to set the adjacent channel power roll-off
factor to 0.22.
Single fRollOffFactor_in = 0.22F;
SigSpectrumObj.SetACPRollOffFactor(fRollOffFactor_in);
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:ACP:FACTor:ROLLoff
```

`SetACPSymbolRate` (SPA)

Description: This method sets the adjacent channel power symbol rate parameter.

API:
```
public void SetACPSymbolRate(double dSymbolRate_in,
FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dSymbolRate_in**

Contains the adjacent channel power symbol rate value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 3.84 MHz.

enumFrequencyUnits_in

Contains the adjacent channel power symbol rate units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum.SetACPSymbolRate(5#, "MHz")
```

C#.NET Example:
```
//Call to set the adjacent channel power symbol rate
to 5 MHz.
double dSymbolRate_in = 5.0;
SigSpectrumObj.SetACPSymbolRate(dSymbolRate_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ACP:SRATe
```

SetAmplitudeUnits (SPA)

Description: This method sets the amplitude units setting.

API:
```
public void SetAmplitudeUnits(ActiveAmpUnits
enumAmpUnits_in)
```

Arguments: **enumAmpUnits_in**

Contains the amplitude units when the call is sent and is defined as an enumeration constant with the following values:

"dBm"
"dBmV"
"dBuV"
"W"
"V"

VB6 Example:
```
Call SignatureSpectrum.SetAmplitudeUnits("dBm")
```

C#.NET Example:
```
//Call to set dBm as the active amplitude units in the
system.
SigSpectrumObj.SetAmplitudeUnits(SignatureSpectrum.Act
iveAmpUnits.dBm);
```

Associated GPIB Commands: `:CALCulate<1|2>:UNIT:POWer`

SetAsNoiseMarker (SPA)

Description: This method sets the indicated marker as a noise marker.

API:
```
public void SetAsNoiseMarker(short sMarkerNum_in, bool
bNoiseMarker_in)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

bNoiseMarker_in

Contains the boolean switch when the call is sent with the following values:

True - noise marker On
False - noise marker Off

VB6 Example:
```
Call SignatureSpectrum.SetAsNoiseMarker(1, "True")
```

C#.NET Example:
```
//Call to set marker number two as a noise marker
short sMarkerNum_in = 2;
SigSpectrumObj.SetAsNoiseMarker(sMarkerNum_in, true);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:FUNCtion:TYPE
```

SetAttenuation (SPA)

Description: This method sets the attenuation level parameter.

API:
```
public void SetAttenuation(int newValue_in,
AttenuationUnits enumAttunits_in)
```

Arguments: **newValue_in**

Contains the attenuation level value when the call returns and is defined as an integer. Ranges from 0 dB to 62 dB with Auto as default.

enumAttunits_in

Contains the attenuation level units when the call returns and is defined as an enumeration constant with the following value:

"dB"

VB6 Example:
```
Call SignatureSpectrum.SetAttenuation(10, "dB")
```

C#.NET Example:
```
//Call to set 20 dB of attenuation in the system.
int iValue_in = 20;
SigSpectrumObj.SetAttenuation(iValue_in,
SignatureSpectrum.AttenuationUnits.dB);
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation`

SetAttenuationModeAuto (SPA)

Description: This method sets the auto attenuation mode.

API:
```
public void SetAttenuationModeAuto(bool bAuto_in)
```

Arguments: **bAuto_in**

Contains the boolean switch when the call is sent with the following values:

True - auto mode On
False - auto mode Off

VB6 Example:
```
Call SignatureSpectrum.SetAttenuationModeAuto(True)
```

C#.NET Example:
```
//Call to set the attenuation mode to auto.
bool bAuto_in = true;
SigSpectrumObj.SetAttenuationModeAuto(bAuto_in);
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation:AUTO`

SetCCDFRBW (SPA)

Description: Sets CCDF RBW for the spectrum measurement.

API:
```
public void SetCCDFRBW(double dVal, _FrequencyUnits enumFrequencyUnits)
```

Arguments: **dVal**

Contains CCDF RBW value when the call is sent.

enumFrequencyUnits

Contains the frequency units when the call is sent and is defined as an enumeration constant with the following value:

“Hz”, “KHz”, “MHz”, “GHz”

VB6 Example:
```
Call SignatureSpectrum.SetCCDFRBW(2,"MHz")
```

C#.NET Example:
```
SigSpectrumObj.SetCCDFRBW(2,SignatureWCDMA._FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CCDF:RBW
```

`SetCenterFrequency` (SPA)

Description: This method sets the center frequency parameter.

API:
```
public void SetCenterFrequency(double dnewValue_in,
FrequencyUnits enumFreqUnits_in)
```

Arguments: **dnewValue_in**

Contains the center frequency value when the call is sent and is defined as a double. Ranges from 5 Hz to 8.079999995 GHz with 4 GHz as the default value. Constrained to:
(MinStart + MinSpan ÷ 2) to (MaxStop – MinSpan ÷ 2)

enumFreqUnits_in

Contains the center frequency units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum. _
SetCenterFrequency(4325#, "MHz")
```

C#.NET Example:
```
//Call to set 100 MHz as the center frequency in the
system.
double dValue_in = 100.0;
SigSpectrumObj.SetCenterFrequency(dValue_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:CENTer`

`SetCenterFrequencyStepSize` (SPA)

Description: This method sets the center frequency parameter step size parameter.

API:
```
public void SetCenterFrequencyStepSize(double
dnewValue_in, FrequencyUnits enumFreqUnits_in)
```

Arguments: **dnewValue_in**

Contains the center frequency step size value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 1/10 Span (1/10 RBW for Zero Span).

enumFreqUnits_in

Contains the center frequency step size units when the call is sent and is defined as an enumeration constant with the following values:

“Hz”
“KHz”
“MHz”
“GHz”

VB6 Example:
```
Call SignatureSpectrum. _
SetCenterFrequencyStepSize(10#, "MHz")
```

C#.NET Example:
```
//Call to set the center frequency step size to
10 MHz.
double dValue_in = 10.0;
SigSpectrumObj.SetCenterFrequencyStepSize(dValue_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands: None

SetCenterToMarkerFreq (SPA)

Description: This method sets the center frequency to a selected marker's frequency.

API: `public void SetCenterToMarkerFreq(short sMarkerNum_in)`

Arguments: **sMarkerNum_in**

Contains the marker number to apply the center frequency setting when the call is sent and is defined as a short. Ranges from 1 to 5.

VB6 Example:

```
Call SignatureSpectrum.SetCenterToMarkerFreq(1)
```

C#.NET Example:

```
//Call to set marker number two's frequency as the
center frequency.
short sMarkerNum_in = 2;
SigSpectrumObj.SetCenterToMarkerFreq(sMarkerNum_in);
```

Associated GPIB Commands: `:CALCulate<1|2>:MARKer<1 to 5>:FUNCtion:CENTer`

SetCFStepSizeByEnumeration (SPA)

Description: This method sets the center frequency parameter to a selected mode.

API:
```
public void
SetCFStepSizeByEnumeration(CFStepSizeByEnumeration
enumStepSize_in)
```

Arguments: **enumCFStepSize_in**

Contains the center frequency step size setting when the call is sent and is defined as an enumeration constant with the following values:

"OneTenthSpan"
"OneHalfSpan"
"EqualsCenter"
"EqualsMarker"
"EqualsSpan"
"DefaultCF"
"OneTenthRbw"
"OneHalfRbw"

VB6 Example:
```
Call SignatureSpectrum. _
SetCFStepSizeByEnumeration("OneTenthSpan")
```

C#.NET Example:
```
//Call to set the CF step size to be equal to the
center frequency.
SigSpectrumObj.SetCFStepSizeByEnumeration(SignatureSpe
ctrum.CFStepSizeByEnumeration.EqualsCenter);
```

Associated GPIB Commands: None

SetCFStepSizeBySpanPercentage (SPA)

Description: This method sets the center frequency parameter to a specified percentage of the sweep span.

API:
```
public void SetCFStepSizeBySpanPercentage(double
dnewValue_in)
```

Arguments: **dnewValue_in**

Contains the center frequency step size value when the call is sent and is defined as a double. Ranges from 1% to 100% with a default value of 10%.

VB6 Example:
```
Call SignatureSpectrum. _
SetCFStepSizeBySpanPercentage(10#)
```

C#.NET Example:
```
//Call to set the CF step size to be 10% of the
frequency span.
SigSpectrumObj.SetCFStepSizeBySpanPercentage(10);
//Value should be in the range from 1 to 100.
```

Associated GPIB Commands: None

SetChannelPowerBandwidth (SPA)

Description: This method sets the channel power bandwidth parameter.

API:
```
public void SetChannelPowerBandwidth(double
dBandwidth_in, FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dBandwidth_in**

Contains the channel power bandwidth value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 5 MHz.

enumFreqUnits_in

Contains the channel power bandwidth units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum. _
SetChannelPowerBandwidth(2#, "MHz")
```

C#.NET Example:
```
//Call to set the channel power bandwidth to 2 MHz.
double dnewValue_in = 2.0;
SigSpectrumObj.SetChannelPowerBandwidth(dnewValue_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:BANDwidth`

SetChannelPowerDivisionPerHzState (SPA)

Description: This method sets the channel power division/Hz toggle setting.

API:
```
public void SetChannelPowerDivisionPerHzState(bool
bDisplay_in)
```

Arguments: **bDisplay_in**

Contains a boolean value when the call is sent with the following values:

True - channel power division/Hz On
False - channel power division/Hz Off

VB6 Example:
```
Call SignatureSpectrum. _
SetChannelPowerDivisionPerHzState(True)
```

C#.NET Example:
```
//Call to set the channel power division/Hz state.
bool bDisplay_in = true;
SigSpectrumObj.SetChannelPowerDivisionPerHzState(bDisp
lay_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:HZ:STATe`

SetChannelPowerMeasurementDomain (SPA)

Description: This method sets the channel power measurement domain mode.

API:
```
public void
SetChannelPowerMeasurementDomain(MeasurementDomain
enumMeasurementDomain_in)
```

Arguments: **enumMeasurementDomain_in**

Contains the channel power measurement domain setting when the call is sent and is defined as an enumeration constant with the following values:

“Freq_Domain”
“Time_Domain”

VB6 Example:
```
Call SignatureSpectrum. _
SetChannelPowerMeasurementDomain("Time_Domain")
```

C#.NET Example:
```
//Call to set the channel power measurement domain
mode to time domain.
SigSpectrumObj.SetChannelPowerMeasurementDomain(Signat
ureSpectrum.MeasurementDomain.Time_Domain);
```

Associated GPIB Commands: None

SetChannelPowerOBMMode (SPA)

Description: This method sets the channel power one-button-measurement mode.

API:
```
public void SetChannelPowerOBMMode(OBMMode
enumOBMMode_in)
```

Arguments: **enumOBMMode_in**

Contains the channel power one-button-measurement mode when the call is sent and is defined as an enumeration constant with the following values:

"Relative"
"Absolute"

VB6 Example:
```
Call SignatureSpectrum. _
SetChannelPowerOBMMode("Absolute")
```

C#.NET Example:
```
//Call to set the channel power one-button-measurement
mode to absolute.
SigSpectrumObj.SetChannelPowerOBMMode(SignatureSpectru
m.OBMMode.Absolute);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:TYPE`

SetChannelPowerRollOffFactor (SPA)

Description: This method sets the channel power roll-off factor parameter.

API:
```
public void SetChannelPowerRollOffFactor(float
fRollOffFactor_in)
```

Arguments: **fRollOffFactor_in**

Contains the channel power roll-off factor value when the call is sent and is defined as a floating point number. Ranges from 0.1 to 1.0 with a default value of 0.22.

VB6 Example:
```
Call SignatureSpectrum. _
SetChannelPowerRollOffFactor(0.22!)
```

C#.NET Example:
```
//Call to set the channel power roll-off factor to
0.22.
Single fRollOffFactor_in = 0.22F;
SigSpectrumObj.SetChannelPowerRollOffFactor(fRollOffFa
ctor_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:FACTor:ROLLoff`

`SetChannelPowerSymbolRate` (SPA)

Description: This method sets the channel power symbol rate parameter.

API:
```
public void SetChannelPowerSymbolRate(double
dSymbolRate_in, FrequencyUnits enumFreqUnits_in)
```

Arguments: **dSymbolRate_in**

Contains the channel power symbol rate value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 3.84 MHz.

enumFreqUnits_in

Contains the channel power symbol rate units when the call is sent and is defined as an enumeration constant with the following values:

"GHz"
"MHz"
"kHz"
"Hz"

VB6 Example:
```
Call SignatureSpectrum. _
SetChannelPowerSymbolRate(2#, "MHz")
```

C#.NET Example:
```
//Call to set the channel power symbol rate to 2 MHz.
double dSymbolRate_in = 2.0;
SigSpectrumObj.SetChannelPowerSymbolRate(dSymbolRate_i
n, SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:SRATe`

SetExternalTriggerLevel (SPA)

Description: This method sets the external trigger level parameter.

API:
```
public void SetExternalTriggerLevel(double
dnewValue_in, AmpUnits enumAmpUnits_in)
```

Arguments: **dnewValue_in**

Contains the external trigger level value when the call is sent and is defined as a double. Ranges from –10V to 10V with a default value of 1.4V TTL.

enumAmpUnits_in

Contains the external trigger level units when the call is sent and is defined as an enumeration constant with the following values:

"Volt"
"mVolt"

VB6 Example:
```
Call SignatureSpectrum. _
SetExternalTriggerLevel(1#, "Volt")
```

C#.NET Example:
```
//Call to set the external trigger level to 1V.
double dnewValue_in = 1.0;
SigSpectrumObj.SetExternalTriggerLevel(dnewValue_in,
SignatureSpectrum.AmpUnits.Volt);
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:LEVel:EXTernal
```

SetFrequencyMarkerPosition (SPA)

Description: This method sets the indicated marker to the indicated frequency position.

Precondition: The corresponding marker should be turned on before making this call using SetMarkerState. The behavior otherwise is not defined.

API:
```
public void SetFrequencyMarkerPosition(short
sMarkerNum_in, double XPosition_in, FrequencyUnits
enumFreqUnits_in)
```

Arguments:

sMarkerNum_in

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

XPosition_in

Contains the marker frequency when the call is sent and is defined as a double. Ranges from start frequency to stop frequency.

enumFreqUnits_in

Contains the frequency units when the call is sent and is defined as an enumeration constant with the following values:

"GHz"
"MHz"
"KHz"
"Hz"

VB6 Example:
```
Call SignatureSpectrum. _
SetFrequencyMarkerPosition(1, 10#, "MHz")
```

C#.NET Example:
```
//Call to set 100 MHz as marker number two's
frequency.
short sMarkerNum_in = 2;
double dValue_in = 100.0;
SigSpectrumObj.SetFrequencyMarkerPosition(sMarkerNum_i
n, dValue_in, SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:X
```

SetFrequencyOffset (SPA)

Description: This method sets the frequency offset value.

API:

```
public void SetFrequencyOffset(double newValue_in,
FrequencyUnits enumfreqUnits_in)
```

Arguments: **newValue_in**

Contains the frequency offset value when the call is sent and is defined as a double. Ranges from –100 GHz to +100 GHz with a default value of 0 Hz.

enumfreqUnits_in

Contains the frequency offset units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:

```
Call SignatureSpectrum.SetFrequencyOffset(872#, "Hz")
```

C#.NET Example:

```
//Call to set 20 MHz as the frequency offset in the
system.
double dValue_in = 20.0;
SigSpectrumObj.SetFrequencyOffset(dValue_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:FREQuency:OFFSet
```

SetFrequencySpan (SPA)

Description: This method sets the frequency span parameter.

API:
```
public void SetFrequencySpan(double dnewValue_in,
FrequencyUnits enumFreqUnits_in)
```

Arguments: **dnewValue_in**

Contains the frequency span value when the call is sent and is defined as a double. Ranges from 10 Hz to 8 GHz.

enumFreqUnits_in

Contains the frequency span units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum.SetFrequencySpan(1.2#, "GHz")
```

C#.NET Example:
```
//Call to set the frequency span to 4 GHz in the
system.
double dnewValue_in = 4.0;
SigSpectrumObj.SetFrequencySpan(dnewValue_in,
SignatureSpectrum.FrequencyUnits.GHz);
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:SPAN`

SetGaussianReferenceLineStatus (SPA)

Description: Sets CCDF Gaussian reference line status for the spectrum measurement.

API:
```
public void SetGaussianReferenceLineStatus(bool
bGaussianReferenceLineStatus)
```

Arguments: **bGaussianReferenceLineStatus**

Contains the boolean value when the call is sent.

VB6 Example:
```
Call
SignatureSpectrum.SetGaussianReferenceLineStatus(True)
```

C#.NET Example:
```
SigSpectrumObj.SetGaussianReferenceLineStatus(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CCDF:GAUSsianreferenceline:STATe`

`SetMarkerMode` (SPA)

Description: This method sets the marker mode. Marker 1 cannot be set to DeltaMarker.

API:
```
public void SetMarkerMode(short sMarkerNum_in,
MarkerMode enumMarkerMode_out)
```

Arguments:

sMarkerNum_in

Contains the marker number to apply the marker mode to when the call is sent and is defined as a short. Ranges from 2 to 5.

enumMarkerMode

Contains the marker mode when the call is sent and is defined as an enumeration constant with the following values:

"DeltaMarker"
"NormalMarker"

VB6 Example:
```
Call SignatureSpectrum.SetMarkerMode(2, "DeltaMarker")
```

C#.NET Example:
```
//Call to set marker number two as a normal marker.
short sMarkerNum_in = 2;
SigSpectrumObj.SetMarkerMode(sMarkerNum_in,SignatureSp
ectrum.MarkerMode.NormalMarker);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<2 to 5>:MODE
```

SetMarkerState (SPA)

Description: This method sets the indicated marker on or off.

API:
```
public void SetMarkerState(short sMarkerNum_in,
MarkerState enumMarkerState_in)
```

Arguments: **sMarkerNum_in**

Contains the marker number to apply the detection type when the call is sent and is defined as a short. Ranges from 1 to 5.

enumMarkerState_in

Contains the marker state when the call is sent and is defined as an enumeration constant with the following values:

"MarkerOn"
"MarkerOff"

VB6 Example:
```
Call SignatureSpectrum.SetMarkerState(1, "MarkerOn")
```

C#.NET Example:
```
//Call to turn ON Marker number two.
short sMarkerNum_in = 2;
SigSpectrumObj.SetMarkerState(sMarkerNum_in,SignatureS
pectrum.MarkerState.MarkerOn);
```

Associated GPIB Commands: None

SetMarkerToCenterFreq (SPA)

Description: This method sends the selected marker to the center frequency value.

API:
```
public void SetMarkerToCenterFreq(short sMarkerNum_in)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

VB6 Example:
```
Call SignatureSpectrum.SetMarkerToCenterFreq(1)
```

C#.NET Example:
```
//Call to set marker number one to the center
frequency.
short sMarkerNum_in = 1;
SigSpectrumObj.SetMarkerToCenterFreq(sMarkerNum_in);
```

Associated GPIB Commands: None

SetMarkerToNextPeak (SPA)

Description: This method sends the selected marker to the next trace peak.

API:
```
public void SetMarkerToNextPeak(short sMarkerNum_in)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

VB6 Example:
```
Call SignatureSpectrum.SetMarkerToNextPeak(1)
```

C#.NET Example:
```
//Call to set marker number two to the next available
peak.
short sMarkerNum_in = 2;
SigSpectrumObj.SetMarkerToNextPeak(sMarkerNum_in);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:MAXimum:NEXT
```

SetMarkerToPeak (SPA)

Description: This method sends the selected marker to the trace peak.

API:
```
public void SetMarkerToPeak(short sMarkerNum_in)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

VB6 Example:
```
Call SignatureSpectrum.SetMarkerToPeak(1)
```

C#.NET Example:
```
//Call to set marker number two to the trace peak.
short sMarkerNum_in = 2;
SigSpectrumObj.SetMarkerToPeak(sMarkerNum_in);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:MAXimum[:PEAK]
```

`SetMarkerToTrace` **(SPA)**

Description: This method sets the indicated marker to the indicated trace.

API:
```
public void SetMarkerToTrace(short sMarkerNum_in,
short sTraceNum_in)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

sTraceNum_in

Contains the trace number when the call is sent and is defined as a short. Ranges from 1 to 5.

VB6 Example:
```
Call SignatureSpectrum.SetMarkerToTrace(1, 2)
```

C#.NET Example:
```
//Call to attach marker number two to trace three.
short sMarkerNum_in = 2;
short sTraceNum_in = 3;
SigSpectrumObj.SetMarkerToTrace(sMarkerNum_in,
sTraceNum_in);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:TRACe
```

`SetMixerLevel` (SPA)

Description: This method sets the mixer level of the system. The value and unit must be in the active unit setting indicated by the GetAmplitudeUnits (SPA) call.

API:
```
public void SetMixerLevel(double dnewValue_in,
AmplitudeUnits enumAmpUnits_in)
```

Arguments:

dnewValue_in

Contains the mixer level value when the call is sent and is defined as a double. Ranges from:

0 dBm to –50 dBm
Default value: –10 dBm

The additional units listed below are supported with the proper conversion.

enumAmpUnits_in

Contains the mixer level units when the call is sent and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Call SignatureSpectrum.SetMixerLevel(1#, "W")
```

C#.NET Example:
```
//Call to set the mixer level to 1 dB in the system.
double dnewValue_in = -1.0;
SigSpectrumObj.SetMixerLevel(dnewValue_in,
SignatureSpectrum.AmplitudeUnits.dBm);
```

Associated GPIB Commands:
```
:INPut<1|2>:MIXer[:POWer]
```

`SetNumberOfAverages` (SPA)

Description: This method sets the number of averages parameter.

API: `public void SetNumberOfAverages(int iNumOfAverages_in)`

Arguments: **iNumOfAverages_in**

Contains the number-of-averages value when the call is sent and is defined as an integer. Ranges from 1 to 10,000 with a default value of 10.

VB6 Example: `Call SignatureSpectrum.SetNumberOfAverages(10)`

C#.NET Example:

```
//Call to set the number of trace averages in the
system.
int iNumOfAverages_in = 8;
SigSpectrumObj.SetNumberOfAverages(iNumOfAverages_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:SWEep:AVERage`

`SetOBWBandwidth` (SPA)

Description: This method sets the occupied bandwidth parameter of the system.

API:

```
public void SetOBWBandwidth(double dBandwidth_in,
AttenuationUnits enumAttUnits_in)
```

Arguments: **dBandwidth_in**

Contains the occupied bandwidth value when the call is sent and is defined as a double. Ranges from 0.1 dB to 100 dB with a default value of 26 dB.

enumAttUnits_in

Contains the occupied bandwidth units when the call is sent and is defined as an enumeration constant with the following values:

“dB”

VB6 Example: `Call SignatureSpectrum.SetOBWBandwidth(1#, "dB")`

C#.NET Example:

```
//Call to set the OBW bandwidth to 10 dB.
double dBandWidth_in = 10.0;
SigSpectrumObj.SetOBWBandwidth(dBandWidth_in,
SignatureSpectrum.AttenuationUnits.dB);
```

Associated GPIB Commands: `[:SENSe<1|2>]:OBW:XDBS`

`SetOBWPercentagePower` (SPA)

Description: This method sets the occupied bandwidth percentage power parameter.

API:
```
public void SetOBWPercentagePower(float
fPercentagePower_in)
```

Arguments: **fPercentagePower_in**

Contains the occupied bandwidth percentage power value when the call is sent and is defined as a floating point number. Ranges from 10% to 100% with a default value of 99%.

VB6 Example:
```
Call SignatureSpectrum.SetOBWPercentagePower(50!)
```

C#.NET Example:
```
//Call to set the occupied bandwidth that includes the
most channel energy.
Single fPercentagePower_in = 15.0F;
//99 is the default.
SigSpectrumObj.SetOBWPercentagePower(fPercentagePower_
in);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:OBW:POWer:PERCent
```

`SetPeakToCenter` (SPA)

Description: This method sets the center frequency of the system to the peak of the specified marker.

API:
```
public void SetPeakToCenter(short sMarkerNum_in)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

VB6 Example:
```
Call SignatureSpectrum.SetPeakToCenter(1)
```

C#.NET Example:
```
//Call to set the peak frequency as the center
frequency.
short sMarkerNum_in = 3;
//Positions marker 3 to the peak and sets the peak
frequency as the center frequency.
SigSpectrumObj.SetPeakToCenter(sMarkerNum_in);
```

Associated GPIB Commands:
```
:CALCulate<1|2>:MARKer<1 to 5>:MAXimum:CENTer
```

SetRBW (SPA)

Description: This method sets the resolution bandwidth value.

API:
```
public void SetRBW(double dnewValue_in, FrequencyUnits
enumFreqUnits_in)
```

Arguments: **dnewValue_in**

Contains the resolution bandwidth value when the call is sent and is defined as a double. Ranges from 10 Hz to 8 MHz with Auto as default (3 MHz is the maximum in Auto mode).

enumFreqUnits_in

Contains the resolution bandwidth units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum.SetRBW(1#, "KHz")
```

C#.NET Example:
```
//Call to set the RBW to 4 MHz.
double dValue_in = 4.0;
SigSpectrumObj.SetRBW(dValue_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:BANDwidth[:RESolution]
```

SetRBWAuto (SPA)

Description: This method sets the auto resolution bandwidth mode.

API: `public void SetRBWAuto(bool bAuto_in)`

Arguments: **bAuto_in**

Contains a boolean value when the call is sent with the following values:

True - auto RBW mode
False - manual RBW mode

VB6 Example:
```
Call SignatureSpectrum.SetRBWAuto(True)
```

C#.NET Example:
```
//Call to allow the system to automatically set the
RBW value.
bool bAuto_in = true;
SigSpectrumObj.SetRBWAuto(bAuto_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:BANDwidth[:RESolution]:AUTO`

SetReferenceLevel (SPA)

Description: This method sets the current reference level value.

API:
```
public void SetReferenceLevel(double dnewValue_in,
AmplitudeUnits enumAmpUnits_in)
```

Arguments: **dnewValue_in**

Contains the reference level value when the call is sent and is defined as a double. Ranges from 30 dBm to –150 dBm with a default value of 0 dBm.

enumAmpUnits_in

Contains the reference level units when the call is sent and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Call SignatureSpectrum.SetReferenceLevel(-20#, "dBm")
```

C#.NET Example:
```
//Call to set the reference level in the system to
20 dBm.
double dValue_in = 20.0;
SigSpectrumObj.SetReferenceLevel(dValue_in,
SignatureSpectrum.AmplitudeUnits.dBm);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel
```

SetReferenceLevelOffset (SPA)

Description: This method sets the reference level offset value.

API:
```
public void SetReferenceLevelOffset(double
dnewValue_in, AttenuationUnits enumAttUnits_in)
```

Arguments: **dnewValue_in**

Contains the reference level offset value when the call is sent and is defined as a double. Ranges from 300 dB to –300 dB with a default value of 0 dB.

enumAttUnits_in

Contains the reference level offset units when the call is sent and is defined as an enumeration constant with the following values:

“dB”

VB6 Example:
```
Call SignatureSpectrum. _
SetReferenceLevelOffset(10#, "dB")
```

C#.NET Example:
```
//Call to set the reference level offset to 10 dB.
double dValue_in = 10.0;
SigSpectrumObj.SetReferenceLevelOffset(dValue_in,
SignatureSpectrum.AttenuationUnits.dB);
```

Associated GPIB Commands: `:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel:OFFSet`

SetScaleTypeLinear (SPA)

Description: This method sets the scale type setting to linear or logarithmic.

API:
```
public void SetScaleTypeLinear(bool bScaleType_in)
```

Arguments: **bScaleType_in**

Contains the boolean switch when the call is sent with the following values:

True - linear scale type
False - logarithmic scale type

VB6 Example:
```
Call SignatureSpectrum.SetScaleTypeLinear(True)
```

C#.NET Example:
```
//Call to set the scale type to linear.
bool bScaleType_in = true;
SigSpectrumObj.SetScaleTypeLinear(bScaleType_in);
```

Associated GPIB Commands: `:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y:SPACing`

SetScalingPerDivision (SPA)

Description: This method sets the vertical graticule scaling value.

API:
```
public void SetScalingPerDivision(int
iValuePerDivision_in)
```

Arguments: **iValuePerDivision_in**

Contains the scale/division value when the call is sent and is defined as an integer. Ranges from 0.1 dB to 20 dB with a default value of 10 dB. The resolution is 0.1 dB for the 0.1 dB to 1 dB range and 1 dB for the 1 dB to 20 dB range.

VB6 Example:
```
Call SignatureSpectrum.SetScalingPerDivision(10)
```

C#.NET Example:
```
//Call to set the scale resolution on the display.
int iValue_in = 10;
SigSpectrumObj.SetScalingPerDivision(iValue_in);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to
5>:Y[:SCALe]:PDIVision
```

SetSpanToRBWRatio (SPA)

Description: This method sets the span/RBW ratio.

API:
```
public void SetSpanToRBWRatio(double dnewValue_in,
Multiplier enumMulUnit_in)
```

Arguments: **dnewValue_in**

Contains the span/RBW ratio value when the call is sent and is defined as a double. Ranges from 2 to 10,000 with a default value of 50.

enumMulUnit_in

Contains the span/RBW ratio multiplier when the call is sent with the following value:

“e00” for a multiplier of 1

VB6 Example:
```
Call SignatureSpectrum.SetSpanToRBWRatio(2, "e00")
```

C#.NET Example:
```
//Call to set the span to RBW ratio to 4.
double dValue_in = 4.0;
SigSpectrumObj.SetSpanToRBWRatio(dValue_in,
SignatureSpectrum.Multiplier.e00);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:BANDwidth[:RESolution]:RATio
```

`SetStartFrequency` (SPA)

Description: This method sets the sweep start frequency.

API:
```
public void SetStartFrequency(double dnewValue_in,
FrequencyUnits enumFreqUnits_in)
```

Arguments: **dnewValue_in**

Contains the start frequency value when the call is sent and is defined as a double. Ranges from 0 Hz to 8.079999990 with a default value of 0 Hz.

enumFreqUnits_in

Contains the start frequency units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum.SetStartFrequency(513.5, "KHz")
```

C#.NET Example:
```
//Call to set the start frequency to 100 MHz in the
system.
double dValue_in = 100.0;
SigSpectrumObj.SetStartFrequency(dValue_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:STARt`

SetStopFrequency (SPA)

Description: This method sets the sweep stop frequency.

API:
```
public void SetStopFrequency(double dnewValue_in,
FrequencyUnits enumFreqUnits_in)
```

Arguments: **dnewValue_in**

Contains the stop frequency value when the call is sent and is defined as a double. Ranges from 10 Hz to 8.08 GHz with a default value of 8 GHz.

enumFreqUnits_in

Contains the stop frequency units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum.SetStopFrequency(8#, "GHz")
```

C#.NET Example:
```
//Call to set the stop frequency to 5 GHz in the
system.
double dValue_in = 5.0;
SigSpectrumObj.SetStopFrequency(dValue_in,
SignatureSpectrum.FrequencyUnits.GHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:FREQuency:STOP
```

`SetSweepMode` (SPA)

Description: This method sets the sweep mode. The settings are single or continuous.

API: `public void SetSweepMode(SweepMode enumSweepMode_in)`

Arguments: **enumSweepMode_in**

Contains the sweep mode when the call is sent and is defined as an enumeration constant with the following values:

"Continuous"
"Single"

VB6 Example:

```
Call SignatureSpectrum.SetSweepMode("Single")
```

C#.NET Example:

```
//Call to set the sweep mode to single sweep.
SigSpectrumObj.SetSweepMode(SignatureSpectrum.SweepMod
e.Single);
```

Associated GPIB Commands: `:INITiate<1|2>:CONTinuous`

SetSweepTime (SPA)

Description: This method sets the sweep time value.

API:
```
public void SetSweepTime(double dnewValue_in,
TimeUnits enumTimeUnits_in)
```

Arguments: **dnewValue_in**

Contains the sweep time value when the call is sent and is defined as a double. Ranges from 5 ms to 10 ks with Auto as default.

enumTimeUnits_in

Contains the sweep time units when the call is sent and is defined as an enumeration constant with the following valuesan enumeration:

"ns" for nanoseconds
"us" for microseconds
"ms" for milliseconds
"s" for seconds
"ks" for kiloseconds

VB6 Example:
```
Call SignatureSpectrum.SetSweepTime(20#, "ms")
```

C#.NET Example:
```
//Call to set a 100 msec sweep time in the system.
double dValue_in = 100.0;
SigSpectrumObj.SetSweepTime(dValue_in,
SignatureSpectrum.TimeUnits.ms);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:SWEep:TIME
```

SetSweepTimeAuto (SPA)

Description: This method sets the auto sweep time mode to auto or manual.

API: `public void SetSweepTimeAuto(bool bAuto_in)`

Arguments: **bAuto_in**

Contains a boolean value when the call is sent with the following values:

True - auto sweep time mode
False - manual sweep time mode

VB6 Example: `Call SignatureSpectrum.SetSweepTimeAuto(True)`

C#.NET Example:

```
//Call to allow the system to automatically set the
sweep time.
bool bAuto_in = true;
SigSpectrumObj.SetSweepTimeAuto(bAuto_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:SWEep:TIME:AUTO`

SetSweepTimeMode (SPA)

Description: This method sets the sweep time mode setting.

API: `public void SetSweepTimeMode(SweepTimeMode enumSpeedTimeMode_in)`

Arguments: **enumSpeedTimeMode_in**

Contains the sweep time mode when the call is sent and is defined as an enumeration constant with the following values:

“Speed”
“Accuracy”

VB6 Example: `Call SignatureSpectrum.SetSweepTimeMode("Accuracy")`

C#.NET Example:

```
//Call to set the sweep time mode to accuracy.
SigSpectrumObj.SetSweepTimeMode(SignatureSpectrum.Swee
pTimeMode.Accuracy);
```

Associated GPIB Commands: `[:SENSe<1|2>]:SWEep:TIME:AUTO:COUPling`

SetSweepType (SPA)

Description: This method sets the sweep type. The settings are Swept or FFT. Wideband FFT is not covered.

API: `public void SetSweepType(SweepType enumSweepType_in)`

Arguments: **enumSweepType_in**

Contains the sweep type when the call is sent and is defined as an enumeration constant with the following values:

"Normal"
"FFT"

VB6 Example:
```
Call SignatureSpectrum.SetSweepType("FFT")
```

C#.NET Example:
```
//Call to set the sweep type to FFT in the system.
SigSpectrumObj.SetSweepType(SignatureSpectrum.SweepTyp
e.FFT);
```

Associated GPIB Commands: `[:SENSe<1|2>]:BANDwidth[:RESolution]:TYPE`

SetTargetNumberOfSamples (SPA)

Description: Sets CCDF target number of samples for the spectrum measurement.

API: `public void SetTargetNumberOfSamples(double dSamples)`

Arguments: **dSamples**

Contains CCDF target number of samples when the call is sent.

VB6 Example:
```
Call SignatureSpectrum.SetTargetNumberOfSamples(6000)
```

C#.NET Example:
```
SigSpectrumObj.SetTargetNumberOfSamples(6000);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CCDF:TARget:SAMPles`

SetTimeMarkerPosition (SPA)

Description: This method sets the indicated marker to the indicated time position.

API:
```
public void SetTimeMarkerPosition(short sMarkerNum_in,
double XPosition_in, TimeUnits enumTimeUnits_in)
```

Arguments: **sMarkerNum_in**

Contains the marker number when the call is sent and is defined as a short. Ranges from 1 to 5.

XPosition_in

Contains the marker time when the call is sent and is defined as a double. Ranges from (Trigger Delay) to (Sweep Time + Trigger Delay).

enumFreqUnits_in

Contains the time units when the call is sent and is defined as an enumeration constant with the following values:

"ns" for nanoseconds
"us" for microseconds
"ms" for milliseconds
"s" for seconds
"ks" for kiloseconds

VB6 Example:
```
Call SignatureSpectrum. _
SetTimeMarkerPosition(1, 100#, "ms")
```

C#.NET Example:
```
//Call to set the marker time to 100 msec when in the
zero span mode.
double dValue_in = 100.0;
short sMarkerNum_in = 2 ;
SigSpectrumObj.SetTimeMarkerPosition(sMarkerNum_in,
dValue_in, SignatureSpectrum.TimeUnits.ms);
```

Associated GPIB Commands: None

SetTraceDetectionType (SPA)

Description: This method sets the trace detection type.

API:
```
public void SetTraceDetectionType(short sTraceNum_in,
TraceDetectionType enumTraceDetType_in)
```

Arguments: **sTraceNum_in**

Contains the trace number to apply the detection type when the call is sent and is defined as a short. Ranges from 1 to 5.

enumTraceDetType_in

Contains the trace detection type when the call is sent and is defined as an enumeration constant with the following values:

"DetAuto"
"DetNormal"
"DetMaxPeak"
"DetMinPeak"
"DetSample"
"DetAverage"
"DetRMS"

VB6 Example:
```
Call SignatureSpectrum. _
SetTraceDetectionType(1, "DetRMS")
```

C#.NET Example:
```
//Call to set the trace one detection type to RMS.
short sTraceNum_in = 1;
SigSpectrumObj.SetTraceDetectionType(sTraceNum_in,
SignatureSpectrum.TraceDetectionType.DetRMS);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DETector<1 to 5>
```

`SetTraceMode` (SPA)

Description: This method sets the specified trace's mode.

API:
```
public void SetTraceMode(short sTraceNum_in, TraceMode
enumTraceMode_in)
```

Arguments: **sTraceNum_in**

Contains the trace number to apply the detection type when the call is sent and is defined as a short. Ranges from 1 to 5.

enumTraceMode_in

Contains the trace mode when the call is sent and is defined as an enumeration constant with the following values:

"ClearWrite"
"MaxHold"
"MinHold"
"Average"
"WriteHold"
"Off"

VB6 Example:
```
Call SignatureSpectrum.SetTraceMode(1, "Average")
```

C#.NET Example:
```
//Call to set max hold as the trace mode for trace
number two.
short sTraceNum_in = 2;
SigSpectrumObj.SetTraceMode(sTraceNum_in,
SignatureSpectrum.TraceMode.MaxHold);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:MODE
```

`SetTriggerDelay` (SPA)

Description: This method sets the trigger delay parameter.

API:
```
public void SetTriggerDelay(double dnewValue_in,
TimeUnits enumTimeUnits_in)
```

Arguments: **dnewValue_in**

Contains the trigger delay value when the call is sent and is defined as a double. Ranges from 0 ms to 65.5 ms with a default value of 0 ms.

enumTimeUnits_in

Contains the trigger delay units when the call is sent and is defined as an enumeration constant with the following values:

"ns" for nanoseconds
"us" for microseconds
"ms" for milliseconds
"s" for seconds
"ks" for kiloseconds

VB6 Example:
```
Call SignatureSpectrum.SetTriggerDelay(10#, "ms")
```

C#.NET Example:
```
//Call to set the trigger delay to 50 ms.
double dDelay = 50.0;
SigSpectrumObj.SetTriggerDelay(dDelay,
SignatureSpectrum.TimeUnits.ms);
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:HOLDoff`

SetTriggerEdgeRising (SPA)

Description: This method sets the edge triggering to rising or falling.

API: `public void SetTriggerEdgeRising(bool bTriggerEdge_in)`

Arguments: **bTriggerEdge_in**

Contains the boolean switch when the call is sent with the following values:

True - rising edge triggering
False - falling edge triggering

VB6 Example:

```
Call SignatureSpectrum.SetTriggerEdgeRising(True)
```

C#.NET Example:

```
//Call to set the trigger edge to rising edge
triggering.
SigSpectrumObj.SetTriggerEdgeRising(true);
//Passing false would make it falling edge triggering.
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:SLOPe`

SetTriggerSource (SPA)

Description: This method sets the trigger source.

API: `public void SetTriggerSource(SPATriggerSource enumSPATriggerSource_in)`

Arguments: **enumSPATriggerSource_in**

Contains the trigger source when the call is sent and is defined as an enumeration constant with the following values:

“FreeRun”
“WideIF”
“Line”
“External”
“Video”
“ExternalTTL”

VB6 Example:

```
Call SignatureSpectrum.SetTriggerSource("Line")
```

C#.NET Example:

```
//Call to set the trigger source as external.
SigSpectrumObj.SetTriggerSource(SignatureSpectrum.SPAT
riggerSource.External);
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:SOURce`

SetVBW (SPA)

Description: This method sets the video bandwidth value.

API:
```
public void SetVBW(double dnewValue_in, FrequencyUnits
enumFreqUnits_in)
```

Arguments: **dnewValue_in**

Contains the video bandwidth value when the call is sent and is defined as a double. Ranges from 1 Hz to 10 MHz with Auto as default.

enumFreqUnits_in

Contains the video bandwidth units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureSpectrum.SetVBW(1#, "KHz")
```

C#.NET Example:
```
//Call to set the VBW to 10 MHz.
double dValue_in = 10.0;
SigSpectrumObj.SetVBW(dValue_in,
SignatureSpectrum.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:BANDwidth:VIDeo
```

SetVBWAuto (SPA)

Description: This method sets the video bandwidth mode to auto or manual.

API: `public void SetVBWAuto(bool bAuto_in)`

Arguments: **bAuto_in**

Contains the boolean switch with the following values:

True - auto VBW mode
False - manual VBW mode

VB6 Example: `Call SignatureSpectrum.SetVBWAuto(True)`

C#.NET Example:

```
//Call to allow the system to automatically set the
VBW value.
bool bAuto_in = true;
SigSpectrumObj.SetVBWAuto(bAuto_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:BANDwidth:VIDeo:AUTO`

SetVBWToRBWRatio (SPA)

Description: This method sets the video bandwidth to resolution bandwidth ratio.

API:

```
public void SetVBWToRBWRatio(double dnewValue_in,
Multiplier enumMulUnit_in)
```

Arguments: **dnewValue_in**

Contains the VBW/RBW ratio value when the call is sent and is defined as a double. Ranges from 0.001 to 1,000 with a default value of 5.

enumMulUnit_in

Contains the VBW/RBW ratio multiplier when the call is sent with the following values:

“e00” for a multiplier of 1

VB6 Example: `Call SignatureSpectrum.SetVBWToRBWRatio(1#, "e00")`

C#.NET Example:

```
//Call to set the VBW to RBW ratio to three in the
system.
double dValue_in = 3.0;
SigSpectrumObj.SetVBWToRBWRatio(dValue_in,
SignatureSpectrum.Multiplier.e00);
```

Associated GPIB Commands: `[:SENSe<1|2>]:BANDwidth:VIDeo:RATio`

`SetVideoTriggerLevel` (SPA)

Description: This method sets the video trigger level parameter.

API:
```
public void SetVideoTriggerLevel(double dnewValue_in,
AbsoluteAmpUnits enumAmpUnits_in)
```

Arguments: **dnewValue_in**

Contains the video trigger level value when the call is sent and is defined as a double. Ranges from Reference Level to (Reference Level – 10 x Scale/Div) with a default value of Reference Level – 0.5 x (10 x Scale/Div).

enumAmpUnits_in

Contains the video trigger level units when the call is sent with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Call SignatureSpectrum.SetVideoTriggerLevel(3#, "dBm")
```

C#.NET Example:
```
//Call to set the video trigger level value to 10 dB.
double dTriggerValue = 10.0;
SigSpectrumObj.SetVideoTriggerLevel(dTriggerValue,
SignatureSpectrum.AbsoluteAmpUnits.dB);
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:LEVel:VIDeo
```

`SetWideIFTriggerLevel` (SPA)

Description: This method sets the wideband IF power trigger level setting.

Precondition: The trigger source must be set to IF Power (Wideband) before the call is sent.

API:
```
public void SetWideIFTriggerLevel(out double
dnewValue_in, out AmplitudeUnits enumAmpUnits_in)
```

Arguments: **dnewValue_in**

Contains the wideband IF power trigger level value when the call is sent and is defined as a double.

enumAmplitudeUnits_in

Contains the wideband IF power trigger level units when the call is sent and is defined as an enumeration constant with the following values:

“dB”, “dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Call SignatureSpectrum.SetWideIFTriggerLevel(10#,
"dBm")
```

C#.NET Example:
```
//Call to set the wideband IF power trigger level
value to 10 dB.
double dTriggerValue = 10.0;
SigSpectrumObj.SetWideIFTriggerLevel(dTriggerValue,
SignatureSpectrum.AmplitudeUnits.dB);
```

Associated GPIB Commands: None

`SetXAxisMaximum` (SPA)

Description: Sets CCDF X axis maximum for the spectrum measurement.

API:
```
public void SetXAxisMaximum(int iXAxisMaximum)
```

Arguments: **iXAxisMaximum**

Contains CCDF X axis maximum value when the call is sent.

VB6 Example:
```
Call SignatureSpectrum.SetXAxisMaximum(6)
```

C#.NET Example:
```
SigSpectrumObj.SetXAxisMaximum(6);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CCDF:XAXis:MAXimum`

`StartSweep` (SPA)

Description: This method triggers a sweep when in the single sweep mode. This is a blocking call and does not return until the sweep is complete.

API: `public void StartSweep()`

Arguments: None

VB6 Example: `Call SignatureSpectrum.StartSweep`

C#.NET Example:
```
//Call to start a sweep.
SigSpectrumObj.StartSweep();
```

Associated GPIB Commands: `:INITiate<1|2>[:IMMediate]`

`SwitchOffAllMarkers` (SPA)

Description: This method switches off all markers.

API: `public void SwitchOffAllMarkers()`

Arguments: None

VB6 Example: `Call SignatureSpectrum.SwitchOffAllMarkers`

C#.NET Example:
```
//Call to switch Off all markers.
SigSpectrumObj.SwitchOffAllMarkers();
```

Associated GPIB Commands: `:CALCulate<1|2>:MARKer:AOFF`

`ToggleACPAdjacentChannelState` (SPA)

Description: This method sets the adjacent channel power, adjacent channel toggle setting.

API: `public void ToggleACPAdjacentChannelState(bool bSwitchOn_in)`

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - ACP adjacent channel On
False - ACP adjacent channel Off

VB6 Example:
```
Call SignatureSpectrum. _
ToggleACPAdjacentChannelState(True)
```

C#.NET Example:
```
//Call to switch On the ACP adjacent channel state.
SigSpectrumObj.ToggleACPAdjacentChannelState(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:ADJacent:STATe`

`ToggleACPAlternateChannel1State` (SPA)

Description: This method sets the adjacent channel power, alternate channel one toggle setting.

API: `public void ToggleACPAlternateChannel1State(bool bSwitchOn_in)`

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - ACP alternate channel one On
False - ACP alternate channel one Off

VB6 Example:
```
Call SignatureSpectrum. _
ToggleACPAlternateChannel1State(True)
```

C#.NET Example:
```
//Call to switch On the ACP alternate channel one state.
SigSpectrumObj.ToggleACPAlternateChannel1State(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:ALT<1|2>:STATe`

`ToggleACPAlternateChannel2State` (SPA)

Description: This method sets the adjacent channel power, alternate channel two toggle setting.

API: `public void ToggleACPAlternateChannel2State(bool bSwitchOn_in)`

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - ACP alternate channel two On
False - ACP alternate channel two Off

VB6 Example:
```
Call SignatureSpectrum. _
ToggleACPAlternateChannel2State(True)
```

C#.NET Example:
```
//Call to switch On the ACP alternate channel two
state.
SigSpectrumObj.ToggleACPAlternateChannel2State(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:ALT<1|2>:STATe`

`ToggleACPFFTState` (SPA)

Description: This method sets the adjacent channel power, fast fourier transform toggle setting.

API: `public void ToggleACPFFTState(bool bSwitchOn_in)`

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - ACP FFT On
False - ACP FFT Off

VB6 Example:
```
Call SignatureSpectrum.ToggleACPFFTState(True)
```

C#.NET Example:
```
//Call to switch On the ACP FFT state.
SigSpectrumObj.ToggleACPFFTState(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:FFT:STATe`

ToggleACPNoiseCompensationState (SPA)

Description: This method sets the adjacent channel power noise compensation toggle setting.

API:
```
public void ToggleACPNoiseCompensationState(bool
bSwitchOn_in)
```

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - noise compensation On
False - noise compensation Off

VB6 Example:
```
Call SignatureSpectrum. _
ToggleACPNoiseCompensationState(True)
```

C#.NET Example:
```
//Call to switch On the ACP noise compensation state.
SigSpectrumObj.ToggleACPNoiseCompensationState(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:NOISecomp:STATe`

ToggleACPRRCFilterState (SPA)

Description: This method sets the adjacent channel power, RRC filter toggle setting.

API:
```
public void ToggleACPRRCFilterState(bool bSwitchOn_in)
```

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - ACP RRC filter On
False - ACP RRC filter Off

VB6 Example:
```
Call SignatureSpectrum. _
ToggleACPRRCFilterState(True)
```

C#.NET Example:
```
//Call to switch On the ACP RRC filter state.
SigSpectrumObj.ToggleACPRRCFilterState(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:ACP:FILTer:RRC`

ToggleChannelPowerFFTState (SPA)

Description: This method sets the channel power fast fourier transform toggle setting.

API:
```
public void ToggleChannelPowerFFTState(bool
bSwitchOn_in)
```

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - FFT On
False - FFT Off

VB6 Example:
```
Call SignatureSpectrum. _
ToggleChannelPowerFFTState(True)
```

C#.NET Example:
```
//Call to switch On the channel power FFT state.
SigSpectrumObj.ToggleChannelPowerFFTState(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:FFT:STATe`

ToggleChannelPowerRRCFilterState (SPA)

Description: This method sets the channel power root raised cosine filter toggle setting.

API:
```
public void ToggleChannelPowerRRCFilterState(bool
bSwitchOn_in)
```

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - RRC On
False - RRC Off

VB6 Example:
```
Call SignatureSpectrum. _
ToggleChannelPowerRRCFilterState(True)
```

C#.NET Example:
```
//Call to switch On the channel power RRC filter
state.
SigSpectrumObj.ToggleChannelPowerRRCFilterState(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:CHP:FILTer:RRC`

ToggleCPNoiseCompensationState (SPA)

Description: This method sets the channel power noise compensation toggle setting.

API:
```
public void ToggleCPNoiseCompensationState(bool
bSwitchOn_in)
```

Arguments: **bSwitchOn_in**

Contains a boolean value when the call is sent with the following values:

True - noise compensation On
False - noise compensation Off

VB6 Example:
```
Call SignatureSpectrum. _
ToggleCPNoiseCompensationState(True)
```

C#.NET Example:
```
//Call to switch On the channel power noise
compensation state.
SigSpectrumObj.ToggleCPNoiseCompensationState(true);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:CHP:NOISecomp:STATe
```

3-4 SignatureModulation Class

The SignatureModulation class provides access to Vector Signal Analysis controls and queries.

The examples provided in this section require the appropriate header code as follows:

VB6 Example Header Code

```
Dim SignatureModulation As New MSSOAPLib30.SoapClient30
SignatureModulation.MSSoapInit "http://SN123456/SignatureModulation/" &_
"SignatureModulation.asmx?wsdl"
'Enter SignatureModulation VB6 Example Code here to remotely program the
'instrument.
```

C#.Net Example Header Code

```
using System;
namespace SampleWSClient

{
/// <summary>
/// This is a sample web service client that demonstrates how to use the following
/// Anritsu web services in a C# .NET environment.
/// SignatureModulation.
/// </summary>

class SampleClient

{

[STAThread]
static void Main(string[] args)

{

SampleWSClient.SignatureModulation.SignatureModulation SigDemodulationObj = new
SampleWSClient.SignatureModulation.SignatureModulation();

{
//Enter SignatureModulation C# Example Code here to remotely program the
//instrument.
}

}

}

}
```

GetAmplitudeUnits (VSA)

Description: This method queries the amplitude units of the graticule.

API:
```
public void GetAmplitudeUnits(out ActiveAmpUnits
enumAmpUnits_out)
```

Arguments: **enumAmpUnits_out**

Contains the amplitude units value when the call returns and is defined as an enumeration constant with the following values:

"dBm"
"dBmV"
"dBuV"
"W"
"V"

VB6 Example:
```
Dim enumAmpUnits_out As String
enumAmpUnits_out = _
SignatureModulation.GetAmplitudeUnits
```

C#.NET Example:
```
//Call to read the amplitude units from the system
when in the modulation mode.
SignatureModulation.ActiveAmpUnits AmpUnits_out;
AmpUnits_out = SigModulationObj.GetAmplitudeUnits();
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation?`

GetAttenuationIndB (VSA)

Description: This method queries the attenuation level.

API:
```
public void GetAttenuationIndB(out int
iAttValueindB_out)
```

Arguments: **iAttValueindB_out**

Contains the attenuation value when the call returns and is defined as an integer. Ranges from 0 dB to 62 dB.

VB6 Example:
```
Dim iAttValueindB_out As Integer
iAttValueindB_out = _
SignatureModulation.GetAttenuationIndB
```

C#.NET Example:
```
//Call to read the attenuation setting from the
system.
int iValue_out = 0;
iValue_out = SigModulationObj.GetAttenuationIndB();
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation?`

`GetCaptureTimeInSecs` (VSA)

Description: This method queries the modulation capture time parameter. The return value is in seconds.

API:
```
public void GetCaptureTimeInSecs(out double
dTimeValinSecs_out)
```

Arguments: **dTimeValinSecs_out**

Contains the capture time value when the call returns and is defined as a double. Ranges from 1s to 5 µs. Constrained to:
10,000 ≥ (Symbol Rate x Capture Time ≥ 100

VB6 Example:
```
Dim dTimeValinSecs_out As Double
dTimeValinSecs_out = _
SignatureModulation.GetCaptureTimeInSecs
```

C#.NET Example:
```
//Call to read the capture time from the system.
double dTimeValinSecs_out = 0.0;
dTimeValinSecs_out =
SigModulationObj.GetCaptureTimeInSecs();
```

Associated GPIB Commands: `[:SENSe<1|2>]:TCAPture:LENGth?`

`GetCarrierFrequencyError` (VSA)

Description: This method queries the carrier frequency error value.

API:
```
public void GetCarrierFrequencyError(out float
fCarrierFreqErr_out)
```

Arguments: **fCarrierFreqErr_out**

Contains the carrier frequency error value when the call returns and is defined as a double.

VB6 Example:
```
Dim fCarrierFreqErr_out As Single
fCarrierFreqErr_out = _
SignatureModulation.GetCarrierFrequencyError
```

C#.NET Example:
```
//Call to read the carrier frequency error from the
system.
Single fCarrierFreqErr_out = 0.0F;
fCarrierFreqErr_out =
SigModulationObj.GetCarrierFrequencyError();
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:RESult?`

GetCenterFrequencyInHz (VSA)

Description: This method queries the center frequency value.

API:
```
public void GetCenterFrequencyInHz(out double
dFreqValueinHz_out)
```

Arguments: **dFreqValueinHz_out**

Contains the center frequency value when the call returns and is defined as a double. Ranges from 5 Hz to 8.079999995 GHz with 4 GHz as the default value. Constrained to:
(MinStart + MinSpan ÷ 2) to (MaxStop – MinSpan ÷ 2).

VB6 Example:
```
Dim dFreqValueinHz_out As Double
dFreqValueinHz_out = _
SignatureModulation.GetCenterFrequencyInHz
```

C#.NET Example:
```
//Call to read the center frequency from the unit.
double dValue_out = 0.0;
dValue_out =
SigModulationObj.GetCenterFrequencyInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:FREQuency:CENTer?
```

GetConstellationDiagramIQMarkerPosition (VSA)

Description: This method queries the constellation diagram's IQ marker position.

API:

```
public void
GetConstellationDiagramIQMarkerPosition(int
iMarkerNum_in, out double dIValue_out, out double
dQValue_out, out double dSymbolNum_out)
```

Arguments:

iMarkerNum_in

Contains the marker number when the call returns and is defined as an integer. Ranges from 1 to 2

dIValue_out

Contains the marker's I value when the call returns and is defined as a double.

dQValue_out

Contains the marker's Q value when the call returns and is defined as a double.

dSymbolNum_out

Contains the symbol number value when the call returns and is defined as a double.

VB6 Example:

```
Dim dIValue_out As Double
Dim dQValue_out As Double
Dim dSymbolNum_out As Double
Const iMarkerNum_in = 1
dIValue_out = SignatureModulation. _
GetConstellationDiagramIQMarkerPosition _
(iMarkerNum_in, dQValue_out, dSymbolNum_out)
```

C#.NET Example:

```
//Call to read marker number one's position from the
constellation diagram.
int iMarkerNum_in = 1;
double dIvalue_out = 0.0;
double dQvalue_out = 0.0;
double dSymbolNum_out = 0.0;
dIvalue_out =
SigModulationObj.GetConstellationDiagramIQMarkerPositi
on(iMarkerNum_in, out dQvalue_out, out
dSymbolNum_out);
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:DDEMod:MARKer<1|2>:Y?
```

GetEVM (VSA)

Description: This method queries the EVM data.

API:
```
public void GetEVM(out float evmValue)
```

Arguments: **evmValue**

Contains the error vector magnitude data when the call returns and is defined as a floating point number.

VB6 Example:
```
Dim evmValue As Single
evmValue = SignatureModulation.GetEVM
```

C#.NET Example:
```
//Call to read the EVM value from the system.
Single sValue_out = 0.0F;
sValue_out = SigModulationObj.GetEVM();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:RESult?
```

GetEVMDiagramMarkerPosition (VSA)

Description: This method queries the indicated marker's position on the EVM diagram.

API:
```
public void GetEVMDiagramMarkerPosition(int
iMarkerNum_in, out double dEVMValue_out, out double
dSymbolNum_out)
```

Arguments:

iMarkerNum_in

Contains the marker number when the call returns and is defined as an integer. Ranges from 1 to 2.

dEVMValue_out

Contains the marker's EVM value when the call returns and is defined as a double.

dSymbolNum_out

Contains the symbol number value when the call returns and is defined as a double.

VB6 Example:
```
Dim dEVMValue_out As Double
Dim dSymbolNum_out As Double
Const iMarkerNum_in = 1
dEVMValue_out = _
SignatureModulation.GetEVMDiagramMarkerPosition _
(iMarkerNum_in, dSymbolNum_out)
```

C#.NET Example:
```
//Call to read marker number one's position from the
Evm diagram.
int iMarkerNum_in = 1;
double dEvmValue_out = 0.0;
double dSymbolNum_out = 0.0;
dEvmValue_out =
SigModulationObj.GetEVMDiagramMarkerPosition(iMarkerNu
m_in, out dSymbolNum_out);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:MARKer<1|2>:Y?
```

GetEye_IDiagramIMarkerPosition (VSA)

Description: This method queries the indicated I-marker's position on the Eye diagram.

API:
```
public void GetEye_IDiagramIMarkerPosition(int
iMarkerNum_in, out double dIValue_out, out double
dSymbolNum_out)
```

Arguments:

iMarkerNum_in

Contains the marker number when the call returns and is defined as an integer. Ranges from 1 to 2.

dIValue_out

Contains the marker's I value when the call returns and is defined as a double.

dSymbolNum_out

Contains the symbol number value when the call returns and is defined as a double.

VB6 Example:
```
Dim dIValue_out As Double
Dim dSymbolNum_out As Double
Const iMarkerNum_in = 1
dIValue_out = _
SignatureModulation.GetEye_IDiagramIMarkerPosition _
(iMarkerNum_in, dSymbolNum_out)
```

C#.NET Example:
```
//Call to read marker number one's position from the
EyeI diagram.
int iMarkerNum_in = 1;
double dIvalue_out = 0.0;
double dSymbolNum_out = 0.0;
dIvalue_out =
SigModulationObj.GetEye_IDiagramIMarkerPosition(iMarke
rNum_in, out dSymbolNum_out);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:MARKer<1|2>:Y?
```

GetEye_QDiagramQMarkerPosition (VSA)

Description: This method queries the indicated Q-marker's position on the Eye diagram.

API:
```
void GetEye_QDiagramQMarkerPosition(int iMarkerNum_in,
out double dQValue_out, out double dSymbolNum_out)
```

Arguments:

iMarkerNum_in

Contains the marker number when the call returns and is defined as an integer. Ranges from 1 to 2.

dQValue_out

Contains the marker's I value when the call returns and is defined as a double.

dSymbolNum_out

Contains the symbol number value when the call returns and is defined as a double.

VB6 Example:
```
Dim dQValue_out As Double
Dim dSymbolNum_out As Double
Const iMarkerNum_in = 1
dQValue_out = _
SignatureModulation.GetEye_QDiagramQMarkerPosition _
(iMarkerNum_in, dSymbolNum_out)
```

C#.NET Example:
```
//Call to read marker number one's position from the
EyeQ diagram.
int iMarkerNum_in = 1;
double dQvalue_out = 0.0;
double dSymbolNum_out = 0.0;
dQvalue_out =
SigModulationObj.GetEye_QDiagramQMarkerPosition(iMarke
rNum_in, out dSymbolNum_out);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:MARKer<1|2>:Y?
```

`GetFilterRollOffFactor` (VSA)

Description: This method queries the current filter roll-off factor (a).

API:
```
public void GetFilterRollOffFactor(out float
fRollOffFactor_out)
```

Arguments: **fRollOffFactor_out**

Contains the filter rolloff factor when the call returns and is defined as a floating point number. Ranges from 0.0 to 1.0 with a default value of 0.22.

VB6 Example:
```
Dim fRollOffFactor_out As Single
fRollOffFactor_out = _
SignatureModulation.GetFilterRollOffFactor
```

C#.NET Example:
```
//Call to read the filter roll-off factor from the
system.
Single sRollOffFactor_out = 0.0F;
sRollOffFactor_out =
SigModulationObj.GetFilterRollOffFactor();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:FILTer:ALPHa?
```

`GetFilterType` (VSA)

Description: This method queries the current filter type.

API:
```
public void GetFilterType(out ReceiveFilterType
enumFilterType_out)
```

Arguments: **enumFilterType_out**

Contains the filter type when the call returns and is defined as an enumeration constant with the following values:

"LowPassFlt"
"NyquistFlt"
"RootNyquistFlt"

VB6 Example:
```
Dim enumFilterType_out As String
enumFilterType_out = SignatureModulation.GetFilterType
```

C#.NET Example:
```
//Call to read the filter type from the system.
SignatureModulation.ReceiveFilterType
enumFilterType_out;
enumFilterType_out = SigModulationObj.GetFilterType();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:FILTer:MEASurement?
```

GetFrequencyOffsetInHz (VSA)

Description: This method queries the frequency offset value.

API:
```
public void GetFrequencyOffsetInHz(out double
dFreqValueinHz_out)
```

Arguments: **dFreqValueinHz_out**

Contains the center frequency value when the call returns and is defined as a double. Ranges from –100 GHz to +100 GHz with a default value of 0 Hz.

VB6 Example:
```
Dim dFreqValueinHz_out As Double
dFreqValueinHz_out = _
SignatureModulation.GetFrequencyOffsetInHz
```

C#.NET Example:
```
//Call to read the frequency offset value from the
system when in the modulation mode.
double dValue_out = 0.0;
dValue_out =
SigModulationObj.GetFrequencyOffsetInHz();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:FREQuency:OFFSet?
```

GetGraphType (VSA)

Description: This method queries the current graph type.

API:
```
public void GetGraphType(out GraphType
enumGraphType_out)
```

Arguments: **enumGraphType_out**

Contains the graph type when the call returns and is defined as an enumeration constant with the following values:

"Power_Vs_Time"
"Constellation"
"Vector"
"Evm_Vs_Time"
"Eye_I"
"Eye_Q"

VB6 Example:
```
Dim enumGraphType_out As String
enumGraphType_out = SignatureModulation.GetGraphType
```

C#.NET Example:
```
//Call to read the VSA graph type from the system.
SignatureModulation.GraphType enumGraphType_out;
enumGraphType_out = SigModulationObj.GetGraphType();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:DISPlay:FORMat?
```

GetInputSignal (VSA)

Description: This method queries the current input source. The settings are IFInput, IQDiffLow, IQDiffHigh, IQSingleLow, or IQSingleHigh.

API:
```
public void GetInputSignal(out InputSignal
isInputSignal_out)
```

Arguments: **isInputSignal_out**

Contains the input signal type when the call returns and is defined as an enumeration constant with the following values:

"IQDiffLow"
"IQDiffHigh"
"IQSingleLow"
"IQSingleHigh"
"IFInput"

VB6 Example:
```
Dim isInputSignal_out As String
isInputSignal_out = SignatureModulation.GetInputSignal
```

C#.NET Example:
```
//Call to read the input signal from the system.
SignatureModulation.InputSignal enumInputSignal_out;
enumInputSignal_out =
SigModulationObj.GetInputSignal();
```

Associated GPIB Commands:
```
:INPut<1|2>:MODE?
```

GetIQVectorData (VSA)

Description: This method queries the IQ vector data.

API:
```
public void GetIQVectorData(int iStartSymbol_in, int
iNumOfSymbols_in, out float[] farrDataArray_out)
```

Arguments:

iStartSymbol_in

Contains the start symbol value when the call returns and is defined as an integer. Ranges from 1 to the number of symbols where the number of symbols is constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iNumOfSymbols_in

Contains the number of symbols value when the call returns and is defined as an integer. Constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

farrDataArray_out

Contains the data array when the call returns and is defined as a floating point number.

VB6 Example:
```
Dim lStartSymbol_in As Long
Dim lNumOfSymbols_in As Long
Dim lDataArraySize_out As Long
Dim farrDataArray_out() As Single
'Pick the symbol starting point
lStartSymbol_in = 10
'Pick the number of symbols to return
lNumOfSymbols_in = 1000
'Determine exactly how much data will be returned
lDataArraySize_out = _
SignatureModulation.GetIQVectorDataSize _
(lStartSymbol_in, lNumOfSymbols_in)
'Make the array large enough to hold the data returned
ReDim fDataArray_out(lDataArraySize_out - 1)
farrDataArray_out = _
SignatureModulation.GetIQVectorData _
(lStartSymbol_in, lNumOfSymbols_in)
```

C#.NET Example:

```
//Call to read the IQ vector data from the system.
int iStartSymbol_in = 10; //Start with Symbol 10
int iSymbolWindowWidth_in = 30;
//Get 30 symbols worth of data.
int iDataSize =
SigModulationObj.GetIQVectorDataSize(iStartSymbol_in,i
SymbolWindowWidth_in);
//Get the float array data size.
System.Single[] sArrIQData = new
System.Single[iDataSize];
sArrIQData =
SigModulationObj.GetIQVectorData(iStartSymbol_in,iSymb
olWindowWidth_in); //sArrIQdata contains the IQ vector
data after sending this call.
```

Associated GPIB Commands: `:TRACe:DDEMod:DATA:IQV?`

GetIQVectorDataSize (VSA)

Description: This method queries the IQ vector data size.

API:
```
public void GetIQVectorDataSize(int iStartSymbol_in,
int iNumOfSymbols_in, out int iDataArraySize_out)
```

Arguments: **iStartSymbol_in**

Contains the start symbol value when the call returns and is defined as an integer. Ranges from 1 to the number of symbols where the number of symbols is constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iNumOfSymbols_in

Contains the number of symbols value when the call returns and is defined as an integer. Constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iDataArraySize_out

Contains the data array size value when the call returns and is defined as an integer.

VB6 Example:
```
Dim lStartSymbol_in As Long
Dim lNumOfSymbols_in As Long
Dim lDataArraySize_out As Long
lDataArraySize_out = _
SignatureModulation.GetIQVectorDataSize _
(lStartSymbol_in, lNumOfSymbols_in)
```

C#.NET Example:
```
//Call to read the data array size for a specified
symbol window width.
int iStartSymbol_in = 10; //Start with Symbol 10
int iSymbolWindowWidth_in = 30;
//Get 30 symbols worth of data.
int iDataSize =
SigModulationObj.GetIQVectorDataSize(iStartSymbol_in,i
SymbolWindowWidth_in);
//Get the float array data size.
```

Associated GPIB Commands: :TRACe:DDEMod:DATA:IQV?

GetLockStatus (VSA)

Description: This method queries the current lock status.

API:
```
public void GetLockStatus(out bool bLockStatus_out)
```

Arguments: **bLockStatus_out**

Contains the boolean switch when the call returns with the following values:

True - locked status
False - unlocked status

VB6 Example:
```
Dim bLockStatus_out As Boolean
bLockStatus_out = SignatureModulation.GetLockStatus
```

C#.NET Example:
```
//Call to read the lock status from the system.
bool bLockStatus = false;
bLockStatus = SigModulationObj.GetLockStatus();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:ROSCillator:EXT:STATe?
```

GetMarkerPosition (VSA)

Description: This method queries the indicated marker's current position.

API:
```
public void GetMarkerPosition(GraphType
enumGraphType_in, int iMarkerNum_in, out double
dSymbolNumber_out)
```

Arguments: **enumGraphType_in**

Contains the graph type when the call is sent and is defined as an enumeration constant with the following values:

"Power_Vs_Time"
"Constellation"
"Vector"
"Evm_Vs_Time"
"Eye_I"
"Eye_Q"

iMarkerNum_in

Contains the marker number value when the call returns and is defined as an integer. Ranges from 1 to 2.

dSymbolNumber_out

Contains the symbol number value when the call returns and is defined as a double. Ranges from 1 to the total number of symbols where the total number of symbols is constrained to:
10,000 ≥ (Symbol Rate x Capture Time ≥ 100

VB6 Example:
```
Const enumGraphType_in = "Vector"
Const iMarkerNum_in = 1
Dim dSymbolNumber_out As Integer
dSymbolNumber_out = _
SignatureModulation.GetMarkerPosition _
(enumGraphType_in, iMarkerNum_in)
```

C#.NET Example:
```
//Call to read the symbol number for marker number one
in the vector diagram.
int iMarkerNum_in = 1;
double dMarkerPos_out = 0.0;
dMarkerPos_out =
SigModulationObj.GetMarkerPosition(SignatureModulation
.GraphType.Vector, iMarkerNum_in);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:MARKer<1|2>:X?
```

GetMixerLevel (VSA)

Description: This method queries the current mixer level setting of the system. The return value and unit is always returned in the active unit setting indicated by the GetAmplitudeUnits (SPA) call.

API:
```
public void GetMixerLevel(out double dMixerLevel_out,
out AmplitudeUnits enumAmpUnits_out)
```

Arguments: **dMixerLevel_out**

Contains the mixer level value when the call returns and is defined as a double. Ranges from:

0 dBm to –50 dBm
Default value: –10 dBm

The additional units listed below are supported with the proper conversion.

enumAmpUnits_out

Contains the mixer level amplitude units when the call returns and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Dim dMixerLevel_out As Double
Dim enumAmpUnits_out As String
dMixerLevel_out = _
SignatureModulation.GetMixerLevel(enumAmpUnits_out)
```

C#.NET Example:
```
//Call to read the mixer level from the system when in
the Modulation mode.
SignatureModulation.AmplitudeUnits AmpUnits_out;
double dValue_out = 0.0;
dValue_out = SigModulationObj.GetMixerLevel(out
AmpUnits_out);
```

Associated GPIB Commands:
```
:INPut<1|2>:MIXer[:POWer]?
```

GetModEvmTimeData (VSA)

Description: This method queries the modulation EVM time data.

API:
```
public void GetModEvmTimeData(int iStartSymbol_in, int
iNumOfSymbols_in, out float[] farrDataArray_out)
```

Arguments:

iStartSymbol_in

Contains the start symbol value when the call returns and is defined as an integer. Ranges from 1 to the number of symbols where the number of symbols is constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iNumOfSymbols_in

Contains the number of symbols value when the call returns and is defined as an integer. Constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

farrDataArray_out

Contains the data array when the call returns and is defined as a floating point number.

VB6 Example:
```
Dim lStartSymbol_in As Long
Dim lNumOfSymbols_in As Long
Dim lDataArraySize_out As Long
Dim farrDataArray_out() As Single
'Pick the symbol starting point
lStartSymbol_in = 10
'Pick the number of symbols to return
lNumOfSymbols_in = 1000
'Determine exactly how much data will be returned
lDataArraySize_out = _
SignatureModulation.GetModEvmTimeDataSize _
(lStartSymbol_in, lNumOfSymbols_in)
'Make the array large enough to hold the data returned
ReDim fDataArray_out(lDataArraySize_out - 1)
farrDataArray_out = _
SignatureModulation.GetModEvmTimeData _
(lStartSymbol_in, lNumOfSymbols_in)
```

C#.NET Example:

```
//Call to read the EVM time data from the system.
int iStartSymbol_in = 10; //Start with Symbol 10
int iSymbolWindowWidth_in = 30;
//Get 30 symbols worth of data.
int iDataSize =
SigModulationObj.GetModEvmTimeDataSize(iStartSymbol_in
,iSymbolWindowWidth_in);
//Get the float array data size.
System.Single[] sArrIQData = new
System.Single[iDataSize];
sArrIQData =
SigModulationObj.GetModEvmTimeData(iStartSymbol_in,iSy
mbolWindowWidth_in);
//sArrIQdata contains the EVM time data after sending
this call.
```

Associated GPIB Commands: `:TRACe:DDEMod:DATA:EVMT?`

GetModEvmTimeDataSize (VSA)

Description: This method queries the modulation EVM time data size.

API:
```
public void GetModEvmTimeDataSize(int iStartSymbol_in,
int iNumOfSymbols_in, out int iDataArraySize_out)
```

Arguments: **iStartSymbol_in**

Contains the start symbol value when the call returns and is defined as an integer. Ranges from 1 to the number of symbols where the number of symbols is constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iNumOfSymbols_in

Contains the number of symbols value when the call returns and is defined as an integer. Constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iDataArraySize_out

Contains the data array size value when the call returns and is defined as an integer.

VB6 Example:
```
Dim lStartSymbol_in As Long
Dim lNumOfSymbols_in As Long
Dim lDataArraySize_out As Long
lDataArraySize_out = _
SignatureModulation.GetModEvmTimeDataSize _
(lStartSymbol_in, lNumOfSymbols_in)
```

C#.NET Example:
```
//Call to read the data array size for a specified
symbol window width.
int iStartSymbol_in = 10; //Start with Symbol 10
int iSymbolWindowWidth_in = 30;
//Get 30 symbols worth of data.
int iDataSize =
SigModulationObj.GetModEvmTimeDataSize(iStartSymbol_in
,iSymbolWindowWidth_in);
//Get the float array data size
```

Associated GPIB Commands: `:TRACe:DDEMod:DATA:EVMT?`

GetModPowerWaveformData (VSA)

Description: This method queries the modulation power waveform data.

API:
```
public void GetModPowerWaveformData(int
iStartSymbol_in, int iNumOfSymbols_in, out float[]
farrDataArray_out)
```

Arguments: **iStartSymbol_in**

Contains the start symbol value when the call returns and is defined as an integer. Ranges from 1 to the number of symbols where the number of symbols is constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iNumOfSymbols_in

Contains the number of symbols value when the call returns and is defined as an integer. Constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

farrDataArray_out

Contains the data array when the call returns and is defined as a floating point number.

VB6 Example:
```
Dim lStartSymbol_in As Long
Dim lNumOfSymbols_in As Long
Dim lDataArraySize_out As Long
Dim farrDataArray_out() As Single
'Pick the symbol starting point
lStartSymbol_in = 10
'Pick the number of symbols to return
lNumOfSymbols_in = 1000
'Determine exactly how much data will be returned
lDataArraySize_out = _
SignatureModulation.GetModPowerWaveformDataSize _
(lStartSymbol_in, lNumOfSymbols_in)
'Make the array large enough to hold the data returned
ReDim fDataArray_out(lDataArraySize_out - 1)
farrDataArray_out = _
SignatureModulation.GetModPowerWaveformData _
(lStartSymbol_in, lNumOfSymbols_in)
```

C#.NET Example:

```
//Call to read the modulation power waveform data from
the system.
int iStartSymbol_in = 10; //Start with Symbol 10
int iSymbolWindowWidth_in = 30;
//Get 30 symbols worth of data.
int iDataSize =
SigModulationObj.GetModPowerWaveformDataSize(iStartSym
bol_in,iSymbolWindowWidth_in);
//Get the float array data size.
System.Single[] sArrIQData = new
System.Single[iDataSize];
sArrIQData =
SigModulationObj.GetModPowerWaveformData(10,30);
//sArrIQdata contains the power waveform data after
sending this call.
```

Associated GPIB Commands: `:TRACe:DDEMod:DATA:POWertime?`

GetModPowerWaveformDataSize (VSA)

Description: This method queries the modulation power waveform data size.

API:

```
public void GetModPowerWaveformDataSize(int
iStartSymbol_in, int iNumOfSymbols_in, out int
iDataArraySize_out)
```

Arguments:

iStartSymbol_in

Contains the start symbol value when the call returns and is defined as an integer. Ranges from 1 to the number of symbols where the number of symbols is constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iNumOfSymbols_in

Contains the number of symbols value when the call returns and is defined as an integer. Constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iDataArraySize_out

Contains the data array size value when the call returns and is defined as an integer.

VB6 Example:

```
Dim lStartSymbol_in As Long
Dim lNumOfSymbols_in As Long
Dim lDataArraySize_out As Long
lDataArraySize_out = _
SignatureModulation.GetModPowerWaveformDataSize _
(lStartSymbol_in, lNumOfSymbols_in)
```

C#.NET Example:

```
//Call to read the data array size for a specified
symbol window width.
int iStartSymbol_in = 10; //Start with Symbol 10.
int iSymbolWindowWidth_in = 30;
//Get 30 symbols worth of data.
int iDataSize =
SigModulationObj.GetModPowerWaveformDataSize(iStartSym
bol_in,iSymbolWindowWidth_in);
//Get the float array data size.
```

Associated GPIB Commands: `:TRACe:DDEMod:DATA:POWertime?`

GetModulationBitStream (VSA)

Description: This method queries the modulation bit stream data.

API:

```
public void GetModulationBitStream(int
iStartSymbol_in, int iNumOfSymbols_in, out float[]
farrDataArray_out)
```

Arguments: **iStartSymbol_in**

Contains the start symbol value when the call returns and is defined as an integer. Ranges from 1 to the number of symbols where the number of symbols is constrained to:
10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iNumOfSymbols_in

Contains the number of symbols value when the call returns and is defined as an integer. Constrained to:
10,000 ≥ (Symbol Rate x Capture Time ≥ 100

farrDataArray_out

Contains the data array when the call returns and is defined as a floating point number.

VB6 Example:

```
Dim lStartSymbol_in As Long
Dim lNumOfSymbols_in As Long
Dim lDataArraySize_out As Long
Dim farrDataArray_out() As Single
'Pick the symbol starting point
lStartSymbol_in = 10
'Pick the number of symbols to return
lNumOfSymbols_in = 1000
'Determine exactly how much data will be returned
lDataArraySize_out = _
SignatureModulation.GetModulationBitStreamSize _
(lStartSymbol_in, lNumOfSymbols_in)
'Make the array large enough to hold the data returned
ReDim fDataArray_out(lDataArraySize_out - 1)
farrDataArray_out = _
SignatureModulation.GetModulationBitStream _
(lStartSymbol_in, lNumOfSymbols_in)
```

C#.NET Example:

```
//Call to read the modulation bit stream data from the
system.
int iStartSymbol_in = 10; //Start with Symbol 10.
int iSymbolWindowWidth_in = 30;
//Get 30 symbols worth of data.
int iDataSize =
SigModulationObj.GetModulationBitStreamSize(iStartSymb
ol_in,iSymbolWindowWidth_in);
//Get the float array data size.
System.Single[] sArrIQData = new
System.Single[iDataSize];
sArrIQData =
SigModulationObj.GetModulationBitStream(iStartSymbol_i
n,iSymbolWindowWidth_in);
//sArrIQdata contains the modulation bit stream data
after sending this call.
```

Associated GPIB Commands: :TRACe:DDEMod:DATA:BITStream?

GetModulationBitStreamSize (VSA)

Description: This method queries the modulation bit stream data.

API:
```
public void GetModulationBitStreamSize(int
iStartSymbol_in, int iNumOfSymbols_in, out int
iDataArraySize_out)
```

Arguments: **iStartSymbol_in**

Contains the start symbol value when the call returns and is defined as an integer. Ranges from 1 to the number of symbols where the number of symbols is constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iNumOfSymbols_in

Contains the number of symbols value when the call returns and is defined as an integer. Constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

iDataArraySize_out

Contains the data array size value when the call returns and is defined as an integer.

VB6 Example:
```
Dim lStartSymbol_in As Long
Dim lNumOfSymbols_in As Long
Dim lDataArraySize_out As Long
lDataArraySize_out = _
SignatureModulation.GetModulationBitStreamSize _
(lStartSymbol_in, lNumOfSymbols_in)
```

C#.NET Example:
```
//Call to read the data array size for a specified
symbol window width.
int iStartSymbol_in = 10; //Start with Symbol 10.
int iSymbolWindowWidth_in = 30;
//Get 30 symbols worth of data.
int iDataSize =
SigModulationObj.GetModulationBitStreamSize(iStartSymb
ol_in,iSymbolWindowWidth_in);
//Get the float array data size.
```

Associated GPIB Commands: `:TRACe:DDEMod:DATA:BITStream?`

GetModulationSummaryData (VSA)

Description: This method queries the modulation summary data.

API:

```
public void GetModulationSummaryData(out string
strSummaryString_out)
```

Arguments: **strSummaryString_out**

Contains the modulation summary data when the call returns and is defined as a string.

VB6 Example:

```
Dim strSummaryString_out As String
strSummaryString_out = _
SignatureModulation.GetModulationSummaryData
```

C#.NET Example:

```
//Call to read the modulation summary data from the
system.
string strModulationSummary_out;
strModulationSummary_out =
SigModulationObj.GetModulationSummaryData();
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:DDEMod:RESult?
```

GetModulationType (VSA)

Description: This method queries the modulation type setting.

API:
```
public void GetModulationType(out ModulationType
enumModType_out)
```

Arguments: **enumModType_out**

Contains the modulation type setting when the call returns and is defined as an enumeration constant with the following values:

"M_BPSK"
"M_QPSK"
"M_Pi4QPSK"
"M_8PSK"
"M_3Pi8PSK"
"M_16QAM"
"M_32QAM"
"M_64QAM"
"M_128QAM"
"M_256QAM"

VB6 Example:
```
Dim enumModType_out As String
enumModType_out = _
SignatureModulation.GetModulationType
```

C#.NET Example:
```
//Call to read the modulation type from the system.
SignatureModulation.ModulationType
enumModulationType_out;
enumModulationType_out =
SigModulationObj.GetModulationType();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:FORMat?
```

GetNumOfTaps (VSA)

Description: This method queries the current number of taps (for digital signal processing) for the filter.

API:

```
public void GetNumOfTaps(out int iNumOfTaps_out)
```

Arguments: **iNumOfTaps_out**

Contains the number of taps value when the call returns and is defined as an integer. Ranges from 1 to 2048 with a default value of 256.

VB6 Example:

```
Dim iNumOfTaps_out As Integer
iNumOfTaps_out = SignatureModulation.GetNumOfTaps
```

C#.NET Example:

```
//Call to read the number of taps from the system.
int iNumTaps_out;
iNumTaps_out = SigModulationObj.GetNumOfTaps();
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:DDEMod:NUMTap?
```

GetPowerDiagramMarkerPosition (VSA)

Description: This method queries the indicated marker's position on the power diagram.

API:
```
public void GetPowerDiagramMarkerPosition(int
iMarkerNum_in, out double dPowerValue_out, out double
dSymbolNum_out)
```

Arguments:

iMarkerNum_in

Contains the marker number when the call returns and is defined as an integer. Ranges from 1 to 2.

dPowerValue_out

Contains the marker's power level value when the call returns and is defined as a double.

dSymbolNum_out

Contains the symbol number value when the call returns and is defined as a double. Ranges from 100 to 10,000.

VB6 Example:
```
Const iMarkerNum_in = 1
Dim dPowerValue_out As Double
Dim dSymbolNum_out As Double
dPowerValue_out = SignatureModulation. _
GetPowerDiagramMarkerPosition _
(iMarkerNum_in, dSymbolNum_out)
```

C#.NET Example:
```
//Call to read the power diagram marker position.
int iMarkerNum_in = 1;
double dPowerValue_out;
double dSymbolNum_out;
dPowerValue_out =
SigModulationObj.GetPowerDiagramMarkerPosition(iMarker
Num_in, out dSymbolNum_out);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:MARKer<1|2>:Y?
```

GetReferenceLevel (VSA)

Description: This method queries the current reference level.

API:
```
public void GetReferenceLevel(out double
dRefLevel_out, out AmplitudeUnits
enumAmplitudeUnits_out)
```

Arguments: **dRefLevel_out**

Contains the reference level value when the call returns and is defined as a double. Ranges from –150 dBm to 30 dBm with a default value of 0 dBm.

enumAmplitudeUnits_out

Contains the reference level units when the call returns and is defined as an enumeration constant with the following values:

"dBm", "dBmV", "dBuV", "W", "mW", "uW", "nW", "pW", "fW", "aW", "zW", "yW", "V", "mV", "uV", "nV", "pV"

VB6 Example:
```
Dim dRefLevel_out As Double
Dem enumAmplitudeUnits_out As String
dRefLevel_out = SignatureModulation. _
GetReferenceLevel(enumAmplitudeUnits_out)
```

C#.NET Example:
```
//Call to read the reference level from the system.
SignatureModulation.AmplitudeUnits AmpUnits_out;
double dValue_out = 0.0;
dValue_out = SigModulationObj.GetReferenceLevel(out
AmpUnits_out);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel?
```

GetReferenceLevelOffsetIndB (VSA)

Description: This method queries the current reference level offset. Return values are in dB.

API:
```
public void GetReferenceLevelOffsetIndB(out double
dRefLvlOffsetValueindB_out)
```

Arguments: **dRefLvlOffsetValueindB_out**

Contains the reference level offset value when the call returns and is defined as a double. Ranges from 30 dBm to –150 dBm with a default value of 0 dBm.

VB6 Example:
```
Dim dRefLvlOffsetValueindB_out As Double
dRefLvlOffsetValueindB_out = _
SignatureModulation.GetReferenceLevelOffsetIndB
```

C#.NET Example:
```
//Call to read the reference level offset from the
system.
double dValue_out = 0.0;
dValue_out =
SigModulationObj.GetReferenceLevelOffsetIndB();
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to
5>:Y[:SCALe]:RLEVel:OFFSet?
```

GetSweepMode (VSA)

Description: This method queries the sweep mode setting.

API:
```
public void GetSweepMode(out bool bContinuous)
```

Arguments: **bContinuous**

Contains the boolean switch when the call returns with the following values:

True - continuous sweep
False - single sweep

VB6 Example:
```
Dim bContinuous As Boolean
bContinuous = SignatureModulation.GetSweepMode
```

C#.NET Example:
```
//Call to read the sweep mode from the system.
bool bSweepMode = false;
//true if continuous, false otherwise.
bSweepMode = SigModulationObj.GetSweepMode();
```

Associated GPIB Commands:
```
:INITiate<1|2>:CONTinuous?
```

GetSymbolRateError (VSA)

Description: This method queries the symbol rate error value.

API:
```
public void GetSymbolRateError(out float
fSymbolRateErr_out)
```

Arguments: **fSymbolRateErr_out**

Contains the symbol rate error value when the call returns and is defined as a double.

VB6 Example:
```
Dim fSymbolRateErr_out As Single
fSymbolRateErr_out = _
SignatureModulation.GetSymbolRateError
```

C#.NET Example:
```
//Call to read the symbol rate error from the system.
Single sSymbolRateErr_out = 0.0F;
sSymbolRateErr_out =
SigModulationObj.GetSymbolRateError();
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:RESult?`

GetSymbolRateInHz (VSA)

Description: This method queries the symbol rate setting.

API:
```
public void GetSymbolRateInHz(out double
dSymbolRateinHz_out)
```

Arguments: **dSymbolRateinHz_out**

Contains the symbol rate value when the call returns and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 3.84 MHz. Constrained to:
10,000 ≥ (Symbol Rate x Capture Time ≥ 100

VB6 Example:
```
Dim dSymbolRateinHz_out As Single
dSymbolRateinHz_out = _
SignatureModulation.GetSymbolRateInHz
```

C#.NET Example:
```
//Call to read the symbol rate from the system.
double dSymbolRate_out = 0.0;
dSymbolRate_out =
SigModulationObj.GetSymbolRateInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:SRATe?`

GetTrackingFlag (VSA)

Description: This method queries the tracking flag toggle setting.

API:
```
public void GetTrackingFlag(out bool
bTrackingFlag_out)
```

Arguments: **bTrackingFlag_out**

Contains the boolean switch when the call returns with the following values:

True - tracking flag is On
False - tracking flag is Off

VB6 Example:
```
Dim bTrackingFlag_out As Boolean
bTrackingFlag_out = _
SignatureModulation.GetTrackingFlag
```

C#.NET Example:
```
//Call to read the tracking flag state from the
system.
bool bTrackingFlag = false;
bTrackingFlag = SigModulationObj.GetTrackingFlag();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:RANGe:TRACking?
```

GetTriggerSource (VSA)

Description: This method queries the current trigger source of the receiver.

API:
```
public void GetTriggerSource(out TriggerSource
enumTriggerSource_out)
```

Arguments: **enumTriggerSource_out**

Contains the trigger source when the call returns and is defined as an enumeration constant with the following values:

"FreeRun"
"WideIF"
"Line"
"External"
"Video"
"ExternalTTL"

VB6 Example:
```
Dim enumTriggerSource_out As String
enumTriggerSource_out = _
SignatureModulation.GetTriggerSource
```

C#.NET Example:
```
//Call to read the trigger source from the system.
SignatureModulation.TriggerSource
enumTriggerSource_out;
enumTriggerSource_out =
SigModulationObj.GetTriggerSource();
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:SOURce?
```

GetVectorDiagramIQMarkerPosition (VSA)

Description: This method queries the vector diagram's IQ marker position.

API:
```
public void GetVectorDiagramIQMarkerPosition(int
iMarkerNum_in, out double dIValue_out, out double
dQValue_out, out double dSymbolNum_out)
```

Arguments:

iMarkerNum_in

Contains the marker number when the call returns and is defined as an integer. Ranges from 1 to 2.

dIValue_out

Contains the marker's I value when the call returns and is defined as a double.

dQValue_out

Contains the marker's Q value when the call returns and is defined as a double.

dSymbolNum_out

Contains the symbol number value when the call returns and is defined as a double. Ranges from 100 to 10,000.

VB6 Example:
```
Dim dIValue_out As Double
Dim dQValue_out As Double
Dim _dSymbolNum_out As Double
Const iMarkerNum_in = 1
dIValue_out = SignatureModulation. _
GetVectorDiagramIQMarkerPosition(iMarkerNum_in, _
dQValue_out, dSymbolNum_out)
```

C#.NET Example:
```
//Call to read the vector diagram IQ marker position.
int iMarkerNum_in = 1;
double dIValue_out = 0.0;
double dQValue_out = 0.0;
double dSymbolNum_out = 0.0;
dIValue_out =
SigModulationObj.GetVectorDiagramIQMarkerPosition(iMar
kerNum_in, out dQValue_out, out dSymbolNum_out);
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:MARKer<1|2>:Y?`

IsDifferentialEncodingOn (VSA)

Description: This method queries the differential data encoding toggle setting. The settings are True or False.

API:
```
public void IsDifferentialEncodingOn(out bool
bDiffCode_out)
```

Arguments: **bDiffCode_out**

Contains the boolean switch when the call returns with the following values:

True - differential encoding is On
False - differential encoding is Off

VB6 Example:
```
Dim bDiffCode_out As Boolean
bDiffCode_out = _
SignatureModulation.IsDifferentialEncodingOn
```

C#.NET Example:
```
//Call to read the differential encoding state from
the system.
bool bDiffEncodingOn = false;
bDiffEncodingOn =
SigModulationObj.IsDifferentialEncodingOn();
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:DIFFcode[:STATe]?`

IsSweepComplete (VSA)

Description: This method queries the sweep complete status of the system.

Precondition: This call works only when a single sweep is initiated by the Async-StartSweep (SYS) web service. The behavior is undefined in the Continuous sweep mode.

API:

```
public void IsSweepComplete(out bool
bSweepComplete_out)
```

Arguments: **bSweepComplete_out**

Contains the boolean switch when the call returns with the following values:

True - sweep is complete
False - sweep is not complete

VB6 Example:

```
Dim bSweepComplete_out As Boolean
bSweepComplete_out = _
SignatureModulation.IsSweepComplete
```

C#.NET Example:

```
//Call to check if the sweep is complete.
bool bSweepComplete = false;
bSweepComplete = SigModulationObj.IsSweepComplete();
```

Associated GPIB Commands:

```
:INITiate<1|2>:SWEep?
```

IsTriggerEdgeRising (VSA)

Description: This method queries the edge triggering of the system.

API:
```
public void IsTriggerEdgeRising(out bool bRising_out)
```

Arguments: **bRising_out**

Contains the boolean switch when the call returns with the following values:

True - rising edge triggering
False - falling edge triggering

VB6 Example:
```
Dim bRising_out As Boolean
bRising_out = _
SignatureModulation.IsTriggerEdgeRising
```

C#.NET Example:
```
//Call to read the trigger edge setting from the
system.
bool bTriggerEdgeSlope_out;
bTriggerEdgeSlope_out =
SigModulationObj.IsTriggerEdgeRising();
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:SLOPe?
```

SetAmplitudeUnits (VSA)

Description: This method sets the amplitude units of the graticule.

API:
```
public void SetAmplitudeUnits(out ActiveAmpUnits
enumAmpUnits_in)
```

Arguments: **enumAmpUnits_in**

Contains the amplitude units when the call is sent and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Call SignatureModulation.SetAmplitudeUnits("dBuV")
```

C#.NET Example:
```
//Call to set dBm as the active amplitude units in the
system.
SigModulationObj.SetAmplitudeUnits(SignatureModulation
.ActiveAmpUnits.dBm);
```

Associated GPIB Commands:
```
:INPut<1|2>:ATTenuation
```

SetAttenuation (VSA)

Description: This method sets the RF input attenuation level.

API:
```
public void SetAttenuation(int iAttValue_in,
AttenuationUnits enumAttUnits_in)
```

Arguments: **iAttValue_in**

Contains the attenuation value when the call is sent and is defined as an integer. Ranges from 0 dB to 62 dB with a default value of 10 dB.

enumAttUnits_in

Contains the attenuation units when the call is sent and is defined as an enumeration constant with the following value:

“dB”

VB6 Example:
```
Call SignatureModulation.SetAttenuation(10, "dB")
```

C#.NET Example:
```
//Call to set the attenuation to 10 dB in the system.
int iAttVal_in = 10;
SigModulationObj.SetAttenuation(iAttVal_in,
SignatureModulation.AttenuationUnits.dB);
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation`

SetCaptureTime (VSA)

Description: This method sets the modulation capture time parameter.

API:
```
public void SetCaptureTime(double dTimeVal_in,
TimeUnits enumTimeUnits_in)
```

Arguments: **dTimeVal_in**

Contains the capture time value when the call is sent and is defined as a double. Ranges from 1s to 5 µs. Constrained to: 10,000 ≥ (Symbol Rate x Capture Time ≥ 100

enumTimeUnits_in

Contains the capture time units when the call is sent and is defined as an enumeration constant with the following values:

"ns" for nanoseconds
"us" for microseconds
"ms" for milliseconds
"s" for seconds
"ks" for kiloseconds

VB6 Example:
```
Call SignatureModulation.SetCaptureTime(90#, "us")
```

C#.NET Example:
```
//Call to set the capture time to 100 microseconds in
the system.
double dValue_in = 100;
SigModulationObj.SetCaptureTime(dValue_in,
SignatureModulation.TimeUnits.us);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:TCAPture:LENGth
```

`SetCenterFrequency` (VSA)

Description: This method sets the center frequency of the measurement.

API:
```
public void SetCenterFrequency(double dFreqValue_in,
FrequencyUnits enumFreqUnits_in)
```

Arguments:

dFreqValue_in

Contains the center frequency value when the call is sent and is defined as a double. Ranges from 5 Hz to 8.079999995 GHz with a default value of 4 GHz.

enumFreqUnits_in

Contains the center frequency units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureModulation. _
SetCenterFrequency(500#, "MHz")
```

C#.NET Example:
```
//Call to set the center frequency to 50 MHz.
double dValue_in = 50.0;
SigModulationObj.SetCenterFrequency(dValue_in,
SignatureModulation.FrequencyUnits.MHz);
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:CENTer`

`SetDifferentialEncodingOn` (VSA)

Description: This method sets the differential data encoding toggle setting.

API:
```
public void SetDifferentialEncodingOn(bool
bDiffCode_in)
```

Arguments: **bDiffCode_in**

Contains the boolean switch when the call is sent with the following values:

True - differential data encoding On
False - differential data encoding Off

VB6 Example:
```
Call SignatureModulation. _
SetDifferentialEncodingOn(True)
```

C#.NET Example:
```
//Call to turn On differential encoding.
SigModulationObj.SetDifferentialEncodingOn(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:DIFFcode[:STATe]`

`SetFilterRollOffFactor` (VSA)

Description: This method sets the filter roll-off factor (a).

API:
```
public void SetFilterRollOffFactor(float
fRollOffFactor_in)
```

Arguments: **fRollOffFactor_in**

Contains the roll-off factor value when the call is sent and is defined as a floating point number. Ranges from 0.1 to 1.0 with a default value of 0.22.

VB6 Example:
```
Call SignatureModulation.SetFilterRollOffFactor(0.22!)
```

C#.NET Example:
```
//Call to set the filter roll-off factor to 0.5.
Single sFilterRollOffFactor = 0.5F;
SigModulationObj.SetFilterRollOffFactor(sFilterRollOff
Factor);
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:FILTer:ALPHa`

SetFilterType (VSA)

Description: This method sets the filter type.

API:
```
public void SetFilterType(ReceiveFilterType
enumfilterType_in)
```

Arguments: **enumfilterType_in**

Contains the filter type when the call is sent and is defined as an enumeration constant with the following values:

"LowPassFlt"
"NyquistFlt"
"RootNyquistFlt"

VB6 Example:
```
Call SignatureModulation. _
SetFilterType("RootNyquistFlt")
```

C#.NET Example:
```
//Call to set the filter type to Nyquist filter.
SigModulationObj.SetFilterType(SignatureModulation.Rec
eiveFilterType.NyquistFlt);
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:FILTer:MEASurement`

SetFrequencyOffset (VSA)

Description: This method sets the center frequency offset for the system.

API:
```
public void SetFrequencyOffset(double dFreqValue_in,
FrequencyUnits enumFrequencyUnits_in)
```

Arguments: **dFreqValue_in**

Contains the center frequency value when the call is sent and is defined as a double. Ranges from –100 GHz to 100 GHz with a default value of 0 Hz.

enumFreqUnits_in

Contains the center frequency units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Call SignatureModulation. _
SetFrequencyOffset(500#, "MHz")
```

C#.NET Example:
```
//Call to set 20 MHz as the frequency offset in the
system.
double dValue_in = 20.0;
SigModulationObj.SetFrequencyOffset(dValue_in,
SignatureModulation.FrequencyUnits.MHz);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:FREQuency:OFFSet
```

SetGraphType (VSA)

Description: This method sets the graph type.

API:
```
public void SetGraphType(GraphType enumGraphType_in)
```

Arguments: **enumGraphType_in**

Contains the graph type when the call is sent and is defined as an enumeration constant with the following values:

"Power_Vs_Time"
"Constellation"
"Vector"
"Evm_Vs_Time"
"Eye_I"
"Eye_Q"

VB6 Example:
```
Call SignatureModulation.SetGraphType("Vector")
```

C#.NET Example:
```
//Call to set the graph type to vector.
SigModulationObj.SetGraphType(SignatureModulation.Grap
hType.Vector);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:DISPlay:FORMat
```

`SetInputSignal` (VSA)

Description: This method sets the current input source.

API:
```
public void SetInputSignal(InputSignal
isInputSignal_in)
```

Arguments: **isInputSignal_in**

Contains the input signal type when the call is sent and is defined as an enumeration constant with the following values:

"IQDiffLow"
"IQDiffHigh"
"IQSingleLow"
"IQSingleHigh"
"RFInput"
"RFInputWideBand"
"RFInputNarrowBand"

VB6 Example:
```
Call SignatureModulation.SetInputSignal("RFInput")
```

C#.NET Example:
```
//Call to set RF as the input signal in the system.
SigModulationObj.SetInputSignal(SignatureModulation.In
putSignal.RFInput);
```

Associated GPIB Commands:
```
:INPut<1|2>:MODE
```

`SetMarkerMode` (VSA)

Description: This method sets the indicated marker's mode.

API:
```
public void SetMarkerMode(GraphType enumGraphType_in,
int iMarkerNum_in, MarkerMode enumMarkerMode_in)
```

Arguments:

enumGraphType_in

Contains the graph type when the call is sent and is defined as an enumeration constant with the following values:

"None"
"Power_Vs_Time"
"Constellation"
"Vector"
"Evm_Vs_Time"
"Eye_I"
"Eye_Q"

iMarkerNum_in

Contains the marker number value when the call is sent and is defined as an integer. Ranges from 1 to 2.

enumMarkerMode_in

Contains the marker mode when the call is sent and is defined as an enumeration constant with the following values:

"MarkerOn"
"MarkerOff"

VB6 Example:
```
Dim sEnumGraphType_in As String
Dim sEnumMarkerMode_in as string
Const iMarkerNum_in = 1
sEnumGraphType_in = "Vector"
sEnumMarkerMode_in = "MarkerOn"
Call SignatureModulation. _
SetMarkerMode(sEnumGraphType_in, iMarkerNum_in, _
sEnumMarkerMode_in)
```

C#.NET Example:
```
//Call to turn On marker number two in the
constellation diagram.
int iMarkerNum_in = 2;
SigModulationObj.SetMarkerMode(SignatureModulation.Gra
phType.Constellation,iMarkerNum_in,SignatureModulation
.MarkerMode.MarkerOn);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:MARKer<1|2>:STATe
```

SetMarkerPosition (VSA)

Description: This method sets the indicated marker's position.

API:
```
public void SetMarkerPosition(GraphType
enumGraphType_in, int iMarkerNum_in, double
dSymbolNumber_in)
```

Arguments: **enumGraphType_in**

Contains the graph type when the call is sent and is defined as an enumeration constant with the following values:

"Power_Vs_Time"
"Constellation"
"Vector"
"Evm_Vs_Time"
"Eye_I"
"Eye_Q"

iMarkerNum_in

Contains the marker number value when the call is sent and is defined as an integer. Ranges from 1 to 2.

dSymbolNumber_in

Contains the symbol number value when the call is sent and is defined as a double. Ranges from 1 to the total number of symbols where the total number of symbols is constrained to:
10,000 ≥ (Symbol Rate x Capture Time ≥ 100

VB6 Example:
```
Dim sEnumGraphType_in As String
dim dSymbolNumber_in As Double
Const iMarkerNum_in = 1
sEnumGraphType_in = "Vector"
dSymbolNumber_in = 11
Call SignatureModulation. _
SetMarkerPosition(sEnumGraphType_in, iMarkerNum_in, _
dSymbolNumber _in)
```

C#.NET Example:
```
//Call to set the marker number two's position to 10
symbols in the vector diagram.
int iMarkerNum_in = 2;
double dSymbolNum_in = 10;
SigModulationObj.SetMarkerPosition(SignatureModulation
.GraphType.Vector, iMarkerNum_in, dSymbolNum_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:MARKer<1|2>:X`

SetMarkerToNextPeak (VSA)

Description: This method sets the indicated marker to the next trace peak.

API:
```
public void SetMarkerToNextPeak(GraphType
enumGraphType_in, int iMarkerNum_in)
```

Arguments: **enumGraphType_in**

Contains the attenuation units when the call is sent and is defined as an enumeration constant with the following values:

"None"
"Power_Vs_Time"
"Constellation"
"Vector"
"Evm_Vs_Time"
"Eye_I"
"Eye_Q"

iMarkerNum_in

Contains the marker number value when the call is sent and is defined as an integer. Ranges from 1 to 2.

VB6 Example:
```
Call SignatureModulation. _
SetMarkerToNextPeak("Vector", 1)
```

C#.NET Example:
```
//Call to set marker number two to the next trace peak.
int iMarkerNum_in = 2;
SigModulationObj.SetMarkerToNextPeak(SignatureModulati
on.GraphType.Vector, iMarkerNum_in);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:MARKer<1|2>:MAXimum:NEXT
```

SetMarkerToPeak (VSA)

Description: This method sets the indicated marker to the trace peak.

API:
```
public void SetMarkerToPeak(GraphType
enumGraphType_in, int iMarkerNum_in)
```

Arguments: **enumGraphType_in**

Contains the attenuation units when the call is sent and is defined as an enumeration constant with the following values:

"None"
"Power_Vs_Time"
"Constellation"
"Vector"
"Evm_Vs_Time"
"Eye_I"
"Eye_Q"

iMarkerNum_in

Contains the marker number value when the call is sent and is defined as an integer. Ranges from 1 to 2.

VB6 Example:
```
Call SignatureModulation.SetMarkerToPeak("Vector", 1)
```

C#.NET Example:
```
//Call to set the marker number two to the trace peak.
int iMarkerNum_in = 2;
SigModulationObj.SetMarkerToPeak(SignatureModulation.G
raphType.Vector, iMarkerNum_in);
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:MARKer<1|2>:MAXimum[:PEAK]`

SetModulationType (VSA)

Description: This method sets the modulation type setting.

API:
```
public void SetModulationType(ModulationType
enumModType_in)
```

Arguments: **enumModType_in**

Contains the modulation type setting when the call is sent and is defined as an enumeration constant with the following values:

“M_BPSK”
“M_QPSK”
“M_Pi4QPSK”
“M_8PSK”
“M_3Pi8PSK”
“M_16QAM”
“M_32QAM”
“M_64QAM”
“M_128QAM”
“M_256QAM”

VB6 Example:
```
Call SignatureModulation.SetModulationType("M_QPSK")
```

C#.NET Example:
```
//Call to set the modulation type to 128 QAM in the
system.
SigModulationObj.SetModulationType(SignatureModulation
.ModulationType.M_128QAM);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:FORMat
```

SetNumOfTaps (VSA)

Description: This method sets the current number of taps (digital signal processing parameter) for the filter.

API:
```
public void SetNumOfTaps(int iNumOfTaps_in)
```

Arguments: **iNumOfTaps_in**

Contains the number of taps value when the call is sent and is defined as an integer. Ranges from 1 to 2048 with a default value of 256.

VB6 Example:
```
Call SignatureModulation.SetNumOfTaps(256)
```

C#.NET Example:
```
//Call to set the number of taps to 256 in the system.
int iNumTaps = 256;
SigModulationObj.SetNumOfTaps(iNumTaps);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:DDEMod:NUMTap
```

SetReferenceLevel (VSA)

Description: This method sets the reference level parameter.

API:
```
public void SetReferenceLevel(double dRefLevel_in,
AmplitudeUnits enumAmplitudeUnits_in)
```

Arguments: **dRefLevel_in**

Contains the reference level value when the call is sent and is defined as a double. Ranges from –150 dBm to 30 dBm with a default value of 0 dBm.

enumAmplitudeUnits_in

Contains the reference level units when the call is sent and is defined as an enumeration constant with the following value:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Call SignatureModulation.SetReferenceLevel(10#, "dBm")
```

C#.NET Example:
```
//Call to set the reference level to 10 dBm in the
system.
double dValue_in = 10;
SigModulationObj.SetReferenceLevel(dValue_in,
SignatureModulation.AmplitudeUnits.dBm);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel
```

SetReferenceLevelOffset (VSA)

Description: This method sets the reference level offset parameter.

API:
```
public void SetReferenceLevelOffset(double
dRefLvlOffsetValue_in, AttenuationUnits
enumAttenuationUnits_in)
```

Arguments: **dRefLvlOffsetValue_in**

Contains the reference level offset value when the call is sent and is defined as a double. Ranges from –300 dB to 300 dB with a default value of 0 dB.

enumAttenuationUnits_in

Contains the reference level offset units when the call is sent and is defined as an enumeration constant with the following value:

"dB"

VB6 Example:
```
Call SignatureModulation. _
SetReferenceLevelOffset(10#, "dB")
```

C#.NET Example:
```
//Call to set the reference level offset to 10 dB in
the system.
double dValue_in = 10.0;
SigModulationObj.SetReferenceLevelOffset(dValue_in,
SignatureModulation.AttenuationUnits.dB);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to
5>:Y[:SCALe]:RLEVel:OFFSet
```

SetSweepMode (VSA)

Description: This method sets the sweep mode.

API:
```
public void SetSweepMode(bool bContinuous)
```

Arguments: **bContinuous**

Contains the boolean switch when the call is sent with the following values:

True - continuous sweep
False - single sweep

VB6 Example:
```
Call SignatureModulation.SetSweepMode(False)
```

C#.NET Example:
```
//Call to set the sweep mode to continuous.
SigModulationObj.SetSweepMode(true);
```

Associated GPIB Commands:
```
:INITiate<1|2>:CONTinuous
```

SetSymbolRate (VSA)

Description: This method sets the symbol rate value.

API:

```
public void SetSymbolRate(float fSymbolRate_in,
FrequencyUnits enumFreqUnits_in)
```

Arguments:

fSymbolRate_in

Contains the symbol rate value when the call is sent and is defined as a double. Ranges from 1 Hz to 8 GHz with a default value of 3.84 MHz. Constrained to:
10,000 ≥ (Symbol Rate x Capture Time ≥ 100

enumFreqUnits_in

Contains the symbol rate units when the call is sent and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:

```
Call SignatureModulation.SetSymbolRate(5#, "MHz")
```

C#.NET Example:

```
//Call to set the symbol rate to 10 MHz.
Single sSymbolRate = 10;
SigModulationObj.SetSymbolRate(sSymbolRate,
SignatureModulation.FrequencyUnits.MHz);
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:DDEMod:SRATe
```

SetTrackingFlag (VSA)

Description: This method sets the current tracking flag toggle setting of the receiver.

API: `public void SetTrackingFlag(bool bTrackingFlag_in)`

Arguments: **bTrackingFlag_in**

Contains the boolean switch when the call returns with the following values:

True - tracking flag On
False - tracking flag Off

VB6 Example:
```
Call SignatureModulation.SetTrackingFlag(True)
```

C#.NET Example:
```
//Call to set the tracking flag state to true.
SigModulationObj.SetTrackingFlag(true);
```

Associated GPIB Commands: `[:SENSe<1|2>]:DDEMod:RANGe:TRACking`

SetTriggerEdgeRising (VSA)

Description: This method sets the edge triggering of the system.

API: `public void SetTriggerEdgeRising(bool bRising_in)`

Arguments: **bRising_in**

Contains the boolean switch when the call returns with the following values:

True - rising edge triggering
False - falling edge triggering

VB6 Example:
```
Call SignatureModulation.SetTriggerEdgeRising(True)
```

C#.NET Example:
```
//Call to set the trigger edge to rising edge
triggering.
SigModulationObj.SetTriggerEdgeRising(true);
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:SLOPe`

SetTriggerSource (VSA)

Description: This method sets the current trigger source of the receiver.

API:
```
public void SetTriggerSource(TriggerSource
enumTriggerSource_in)
```

Arguments: **enumTriggerSource_in**

Contains the trigger source when the call is sent and is defined as an enumeration constant with the following values:

"FreeRun"
"WideIF"
"Line"
"External"
"Video"
"ExternalTTL"

VB6 Example:
```
Call SignatureModulation.SetTriggerSource("Video")
```

C#.NET Example:
```
//Call to set the trigger source to external.
SigModulationObj.SetTriggerSource(SignatureModulation.
TriggerSource.External);
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:SOURce
```

StartSweep (VSA)

Description: This method triggers a sweep when in the single sweep mode. This is a blocking call and does not return until the sweep is complete.

API:
```
public void StartSweep()
```

Arguments: None

VB6 Example:
```
Call SignatureModulation.StartSweep
```

C#.NET Example:
```
//Call to start a sweep when in the VSA single sweep
mode.
SigModulationObj.StartSweep();
```

Associated GPIB Commands:
```
:INITiate<1|2>[:IMMediate]
```

3-5 SignatureWCDMA Class

The SignatureWCDMA class provides access to Wideband Code Division Multiple Access modulation analysis controls and queries.

The examples provided in this section require the appropriate header code as follows:

VB6 Example Header Code

```
Dim SignatureWCDMA As New MSSOAPLib30.SoapClient30
SignatureWCDMA.MSSoapInit "http://SN123456/SignatureWCDMA/" &_
"SignatureWCDMA.asmx?wsdl"
'Enter SignatureWCDMA VB6 Example Code here to remotely program the instrument.
```

C#.Net Example Header Code

```
using System;
namespace SampleWSClient

{
/// <summary>
/// This is a sample web service client that demonstrates how to use the following
/// Anritsu web services in a C# .NET environment.
/// SignatureWCDMA.
/// </summary>

class SampleClient

{

[STAThread]
static void Main(string[] args)

{

SampleWSClient.SignatureWCDMA.SignatureWCDMA SigWCDMAObj = new
SampleWSClient.SignatureWCDMA.SignatureWCDMA();

{
//Enter SignatureWCDMA C# Example Code here to remotely program the
//instrument.
}

}

}

}
```

GetActiveChannelThresholdIndB (WCDMA)

Description: This method queries the active channel threshold level.

API:
```
public void GetActiveChannelThresholdIndB(out float
fValue)
```

Arguments: **fValue**

Contains the threshold value when the call returns and is defined as a floating point number. Ranges from –50 dB to –10 dB with a default value of –33 dB. The resolution is 1 dB.

VB6 Example:
```
Dim dValue As Double
dValue = SignatureWCDMA.GetActiveChannelThresholdIndB
```

C#.NET Example:
```
//Call to read the Active Channel Threshold.
double dValue = 0.0f;
dValue = SigWCDMAObj.GetActiveChannelThresholdIndB();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels:THResho
ld?
```

GetActiveCodeChannelType (WCDMA)

Description: This API queries the method used to detect the active code channel.

API:
```
public void GetActiveCodeChannelType(out
ActiveCodeChannelType enumActiveCodeChannelType)
```

Arguments: **enumActiveCodeChannelType**

Contains the active code channel type when the call returns and is defined as an enumeration constant with the following values:

"AutoChannelType"

VB6 Example:
```
Dim sActiveCodeChannelType As String
sActiveCodeChannelType =
SignatureWCDMA.GetActiveCodeChannelType
```

C#.NET Example:
```
//Call to read the Active Code Channel type from the
system.
SignatureWCDMA.ActiveCodeChannelType
enumActiveCodeChannelType =
SignatureWCDMA.ActiveCodeChannelType.AutoChannelType;
enumActiveCodeChannelType =
SigWCDMAObj.GetActiveCodeChannelType();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels[:CODE]?
```

GetAnalysisLength (WCDMA)

Description: This method queries the analysis length.

API:
```
public void GetAnalysisLength(out double dValue, out
WCDMA3GPPTimeUnits enumWCDMA3GPPTimeUnits)
```

Arguments: **dValue**

Contains the analysis length value when the call returns and is defined as a double.

enumWCDMA3GPPTimeUnits

Contains the WCDMA analysis length units when the call returns and is defined as an enumeration constant with the following values:

" _3GPP_Frame"
"_3GPP_Slot"
"_3GPP_Chip"
"_3GPP_s"
"_3GPP_ms"
"_3GPP_us"
"_3GPP_ns"

VB6 Example:
```
Dim dLength As Double
Dim sWCDMA3GPPTimeUnits As String
dLength =
SignatureWCDMA.GetAnalysisLength(sWCDMA3GPPTimeUnits)
```

C#.NET Example:
```
//Call to read the WCDMA Analysis Length value from
the system.
double dValue = 0.0;
SignatureWCDMA.WCDMA3GPPTimeUnits
enumWCDMA3GPPTimeUnits =
SignatureWCDMA.WCDMA3GPPTimeUnits._3GPP_Frame;
dValue = SigWCDMAObj.GetAnalysisLength(out
enumWCDMA3GPPTimeUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:LENgth?
```

GetAnalysisStart (WCDMA)

Description: This method queries the analysis start position.

API:
```
public void GetAnalysisStart(out double dValue, out
WCDMA3GPPTimeUnits enumWCDMA3GPPTimeUnits)
```

Arguments: **dValue**

Contains the WCDMA analysis start value when the call returns and is defined as a double.

enumWCDMA3GPPTimeUnits

Contains the WCDMA analysis start units when the call returns and is defined as an enumeration constant with the following values:

" _3GPP_Frame"
"_3GPP_Slot"
"_3GPP_Chip"
"_3GPP_s"
"_3GPP_ms"
"_3GPP_us"
"_3GPP_ns"

VB6 Example:
```
Dim dStart As Double
Dim sWCDMA3GPPTimeUnits As String
dStart =
SignatureWCDMA.GetanalysisStart(sWCDMA3GPPTimeUnits)
```

C#.NET Example:
```
//Call to read the WCDMA Analysis Start value from the
system.
double dValue = 0.0;
SignatureWCDMA.WCDMA3GPPTimeUnits
enumWCDMA3GPPTimeUnits =
SignatureWCDMA.WCDMA3GPPTimeUnits._3GPP_Frame;
dValue = SigWCDMAObj.GetAnalysisStart(out
enumWCDMA3GPPTimeUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:STARt?
```

GetAttenuationIndB (WCDMA)

Description: This method queries the input attenuation level.

API:
```
public void GetAttenuationIndB(out int
iAttValueindB_out)
```

Arguments: **iAttValueindB_out**

Contains the attenuation value when the call returns and is defined as an integer. Ranges from 0 dB to 62 dB.

VB6 Example:
```
Dim iAttenuation As Integer
iAttenuation = SignatureWCDMA.GetAttenuationIndB
```

C#.NET Example:
```
//Call to read the Input Attenuation Level from the
system when in WCDMA mode.
int iValue = 0;
iValue = SigWCDMAObj.GetAttenuationIndB();
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation?`

GetAttenuationModeAuto (WCDMA)

Description: This method queries the attenuation mode.

API:
```
public void GetAttenuationModeAuto(out bool bAuto)
```

Arguments: **bAuto**

Contains a boolean value when the call returns with the following values:

True - auto mode
False - manual mode

VB6 Example:
```
Dim bAutoMode As Boolean
bAutoMode = SignatureWCDMA.GetAttenuationModeAuto
```

C#.NET Example:
```
//Call to read the Attenuation mode from the system
when in WCDMA mode.
bool bAuto = false;
bAuto = SigWCDMAObj.GetAttenuationModeAuto();
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation?`

GetCaptureLength (WCDMA)

Description: This method queries the capture length duration.

API:
```
public void GetCaptureLength(out double dValue, out
WCDMA3GPPTimeUnits enumWCDMA3GPPTimeUnits)
```

Arguments: **dValue**

Contains the WCDMA capture length value when the call returns and is defined as a double.

enumWCDMA3GPPTimeUnits

Contains the WCDMA capture length units when the call returns and is defined as an enumeration constant with the following value:

" _3GPP_Frame"

VB6 Example:
```
Dim dCaptureLen As Double
Dim sWCDMA3GPPTimeUnits As String
dCaptureLen =
SignatureWCDMA.GetCaptureLength(sWCDMA3GPPTimeUnits)
```

C#.NET Example:
```
//Call to read the WCDMA Capture Length from the
system.
double dValue = 0.0;
SignatureWCDMA.WCDMA3GPPTimeUnits
enumWCDMA3GPPTimeUnits =
SignatureWCDMA.WCDMA3GPPTimeUnits._3GPP_Frame;
dValue = SigWCDMAObj.GetCaptureLength(out
enumWCDMA3GPPTimeUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:ACQUisition:CAPTure:LENgth?
```

`GetCDPMarkerPosition` (WCDMA)

Description: This method queries the specified code domain power marker's position and the code's power level.

API:

```
public void GetCDPMarkerPosition(CDPGraphType
enumCDPGraphType, int iMarkerNum, out int iCodeNum,
out SpreadingFactor enumSpreadingFactor, out float
fCDPLevel)
```

Arguments:

enumCDPGraphType

An input parameter that contains the graph type when the call is made and is defined as an enumeration constant with one the following values:

"CDM_CODEDOMAINPOWER"
"CDM_CODEDOMAINPOWER_ZOOM"
"CDM_CODEDOMAINERROR"
"CDM_CODEDOMAINERROR_ZOOM"

iMarkerNum

An input parameter that contains the active marker number when the call is made and is defined as an integer. Ranges from 1 to 2.

iCodeNum

Contains the code number when the call returns and is defined as an integer. Ranges from 0 to Maximum Spreading Factor minus one.

enumSpreadingFactor

Contains the spreading factor when the call returns and is defined as an enumeration constant with one of the following values:

"MaxSpreadFactor"
"SpreadFactor512"
"SpreadFactor256"
"SpreadFactor128"
"SpreadFactor64"
"SpreadFactor32"
"SpreadFactor16"
"SpreadFactor8"
"SpreadFactor4"

fCDPLevel

Contains the code domain power level value when the call returns and is defined as a floating point number.

VB6 Example:

```
Dim sCDPGraphType_In As String
Dim iMarkerNumber_In As Integer
Dim iCodeNumber_Out As Integer
Dim sSpreadFactor_Out As String
Dim sngCDPLevel_Out As Single
sCDPGraphType_In = "CDM_CODEDOMAINPOWER"
iMarkerNumber_In = 2
iCodeNumber_Out =
SignatureWCDMA.GetCDPMarkerPosition(sCDPGraphType_In,
iMarkerNumber_In, sSpreadFactor_Out, sngCDPLevel_Out)
```

C#.NET Example:

```
//Call to read the Code Domain Power Marker1 position.
SignatureWCDMA.CDPGraphType enumCDPGraphType =
SignatureWCDMA.CDPGraphType.CDM_CODEDOMAINPOWER;
int iMarkerNum = 1;
int iCodeNum = 0;
float fCDPLevel = 0.0f;
SignatureWCDMA.SpreadingFactor enumSpreadingFactor =
SignatureWCDMA.SpreadingFactor.SpreadFactor256;
iCodeNum =
SigWCDMAObj.GetCDPMarkerPosition(enumCDPGraphType,
iMarkerNum, out enumSpreadingFactor, out fCDPLevel);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:MARKer<1|2>:CDPLevel?`

GetCenterFrequencyInHz (WCDMA)

Description: This method queries the center frequency setting.

API:

```
public void GetCenterFrequencyInHz(out double
dFreqValueinHz_out)
```

Arguments: **dFreqValueinHz_out**

Contains the center frequency value when the call returns and is defined as a double.

VB6 Example:

```
Dim dCenterFrequencyInHz As Double
dCenterFrequencyInHz =
SignatureWCDMA.GetCenterFrequencyInHz
```

C#.NET Example:

```
//Call to read the Center Frequency setting from the
system when in the WCDMA mode.
double dValue = 0.0;
dValue = SigWCDMAObj.GetCenterFrequencyInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:CENTer?`

GetDisplayCompressedModeSignalsMode (WCDMA)

Description: This method queries the signal mode type.

API:
```
public void GetDisplayCompressedModeSignalsMode(out
CompressedModeSignalType enumCompressedModeSignalType)
```

Arguments: **enumCompressedModeSignalType**

Contains the compressed mode signal type and is defined as an enumeration constant with the following values:

"SignalOff"
"Auto_Code_Selection"
"Manual_Code_Selection"

VB6 Example:
```
Dim sCompressedModeSignalType As String
sCompressedModeSignalType =
SignatureWCDMA.GetDisplayCompressedModeSignalsMode
```

C#.NET Example:
```
//Call to read the Display Compressed Mode Signals
mode.
SignatureWCDMA.CompressedModeSignalType
enumCompressedModeSignalType =
SignatureWCDMA.CompressedModeSignalType.Auto_Code_Sele
ction;
enumCompressedModeSignalType =
SigWCDMAObj.GetDisplayCompressedModeSignalsMode();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode[:MODE]?`

GetDownlinkUplink (WCDMA)

Description: Gets measurement type Downlink or Uplink for WCDMA measurement.

API:
```
public void GetDownlinkUplink(out bool bTPCLimitCheck)
```

Arguments: **bDownlinkUplink**

Contains the boolean value when the call is returned. True for Uplink and False for Downlink.

VB6 Example:
```
Dim bDownlinkUplink as Boolean bDownlinkUplink =
SignatureWCDMA.GetDownlinkUplink
```

C#.NET Example:
```
bool bDownlinkUplink = true; bDownlinkUplink =
SigWCDMAObj.GetDownlinkUplink();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:LINK:STATe?`

GetExternalTriggerLevelInVolts (WCDMA)

Description: This method queries the trigger level setting in volts.

API:
```
public void GetExternalTriggerLevelInVolts(out double
dnewValueinVolts_out)
```

Arguments: **dnewValueinVolts_out**

Contains the external trigger level value when the call returns and is defined as a double. Ranges from –10V to 10V with a default value of 1.4V TTL.

VB6 Example:
```
Dim dTriggerLevelInVolts As Double
dTriggerLevelInVolts =
SignatureWCDMA.GetExternalTriggerLevelInVolts
```

C#.NET Example:
```
//Call to get the External Trigger Level when in the
WCDMA mode.
double dValue = 1.4;
dValue = SigWCDMAObj.GetExternalTriggerLevelInVolts();
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:LEVel:EXTernal?`

GetFrequencyOffsetInHz (WCDMA)

Description: This method queries the frequency offset value in Hertz.

API:
```
public void GetFrequencyOffsetInHz(out double
dFreqValueinHz_out)
```

Arguments: **dFreqValueinHz_out**

Contains the frequency offset value when the call returns and is defined as a double.

VB6 Example:
```
Dim dFrequencyOffsetInHz As Double
dFrequencyOffsetInHz =
SignatureWCDMA.GetFrequencyOffsetInHz
```

C#.NET Example:
```
//Call to read the Frequency Offset value from the
system when in the WCDMA mode.
double dValue = 0.0;
dValue = SigWCDMAObj.GetFrequencyOffsetInHz();
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:OFFSet?`

GetIQDisplayRotation (WCDMA)

Description: This method queries the rotation angle of the QPSK and Composite IQ displays.

API:
```
public void GetIQDisplayRotation(out RotationConstants
enumRotationConstants)
```

Arguments: **enumRotationConstants**

Contains the IQ display rotation constant when the call returns and is defined as an enumeration constant with the following values:

"Degrees_0"
"Degrees_45"

VB6 Example:
```
Dim sRotation As String
sRotation = SignatureWCDMA.GetIQDisplayRotation
```

C#.NET Example:
```
//Call to read the IQ Display Rotation setting from
the system.
SignatureWCDMA.RotationConstants enumRotationConstants
= SignatureWCDMA.RotationConstants.Degrees_0;
enumRotationConstants =
SigWCDMAObj.GetIQDisplayRotation();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:ROTAtion?
```

GetMarkerMode (WCDMA)

Description: This method queries the marker mode setting for the specified graph type and marker number.

API:
```
public void GetMarkerMode(GraphType enumGraphType, int iMarkerNum, out MarkerMode enumMarkerMode_out)
```

Arguments: **enumGraphType**

An input parameter that contains the graph type when the call is made and is defined as an enumeration constant with one of the following values:

"QPSK_VECTOR"
"QPSK_CONSTELLATION"
"QPSK_POWERVSTIME"
"QPSK_EVMVSTIME"
"QPSK_MAGNITUDEERRVSTIME"
"QPSK_PHASEERRVSTIME"
"QPSK_ALLERRSVSTIME"
"QPSK_EYEI"
"QPSK_EYEQ"
"QPSK_EYEIQ"
"COMPOSITE_VECTOR"
"COMPOSITE_CONSTELLATION"
"COMPOSITE_POWERVSTIME"
"COMPOSITE_EVMVSTIME"
"COMPOSITE_MAGNITUDEERRVSTIME"
"COMPOSITE_PHASEERRVSTIME"
"COMPOSITE_ALLERRSVSTIME"
"COMPOSITE_EYEI"
"COMPOSITE_EYEQ"
"COMPOSITE_EYEIQ"
"CDM_CODEDOMAINPOWER"
"CDM_CODEDOMAINPOWER_ZOOM"
"CDM_CODEDOMAINERROR"
"CDM_CODEDOMAINERROR_ZOOM"
"SINGLE_CHANNEL_VECTOR"
"SINGLE_CHANNEL_CONSTELLATION"
"SINGLE_CHANNEL_SUMMARY"
"SINGLE_CHANNEL_CODEPOWERVSSLOT"
"SINGLE_CHANNEL_CODEERRVSSLOT"
"SINGLE_CHANNEL_CODEEVMVSTIME"
"SINGLE_CHANNEL_MAGNITUDEERRVSTIME"
"SINGLE_CHANNEL_PHASEERRVSTIME"
"SINGLE_CHANNEL_EYEI"
"SINGLE_CHANNEL_EYEQ"
"SINGLE_CHANNEL_EYEIQ"

iMarkerNum

An input parameter that contains the marker number when the call is made and is defined as an integer. Ranges from 1 to 2.

enumMarkerMode_out

An output parameter that contains the marker mode when the call is made and is defined as an enumeration constant with the following values:

"MarkerOff"
"MarkerOn"

VB6 Example:

```
Dim sGraphType As String
Dim iMarkerNumber As Integer
Dim sMarkerMode As String
sGraphType = "COMPOSITE_CONSTELLATION"
iMarkerNumber = 2
sMarkerMode = SignatureWCDMA.GetMarkerMode(sGraphType,
iMarkerNumber)
```

C#.NET Example:

```
//Call to read the Marker1 mode from the Code Domain
Power Measurement.
SignatureWCDMA.GraphType enumGraphType =
SignatureWCDMA.GraphType.CDM_CODEDOMAINPOWER;
SignatureWCDMA.MarkerMode enumMarkerMode =
SignatureWCDMA.MarkerMode.MarkerOff;
int iMarkerNum = 1;
enumMarkerMode =
SigWCDMAObj.GetMarkerMode(enumGraphType, iMarkerNum);
```

Associated GPIB Commands:

```
:CALCulate<1|2>:WCDMa:MARKer<1|2>:STATe?
```

GetMaxSpreadFactor (WCDMA)

Description: This method queries the maximum spreading factor value.

API:
```
public void GetMaxSpreadFactor(out MaxSpreadFactor
enumMaxSpreadFactor)
```

Arguments: **enumMaxSpreadFactor**

Contains the maximum spreading factor when the call returns and is defined as an enumeration constant with the following values:

"SF256"
"SF512"

VB6 Example:
```
Dim sMaxSpreadFactor As String
sMaxSpreadFactor = SignatureWCDMA.GetMaxSpreadFactor
```

C#.NET Example:
```
//Call to read the Maximum Spread Factor from the
system.
SignatureWCDMA.MaxSpreadFactor enumMaxSpreadFactor =
SignatureWCDMA.MaxSpreadFactor.SF256;
enumMaxSpreadFactor =
SigWCDMAObj.GetMaxSpreadFactor();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:MAXSpreadfactor?`

GetPICHChannelNumber (WCDMA)

Description: Gets PICH channel number for the WCDMA measurement.

API:
```
public void GetPICHChannelNumber(out int
iPICHChannelNumber)
```

Arguments: **iPICHChannelNumber**

Contains the PICH channel number value when the call is returned.

VB6 Example:
```
Dim iPICHChannelNumber as Integer iPICHChannelNumber =
SignatureWCDMA.GetPICHChannelNumber
```

C#.NET Example:
```
double iPICHChannelNumber = 0.0; iPICHChannelNumber
=SigWCDMAObj.GetPICHChannelNumber();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:PICH:CHANnelnumber?`

GetPICHCompensation (WCDMA)

Description: Gets PICH compensation for the WCDMA measurement.

API:
```
public void GetPICHCompensation(out bool
bPICHCompensation)
```

Arguments: **bPICHCompensation**

Contains the boolean value when the call is returned.

VB6 Example:
```
Dim bPICHCompensation as Boolean bPICHCompensation =
SignatureWCDMA.GetPICHCompensation
```

C#.NET Example:
```
bool bPICHCompensation = true; bPICHCompensation
=SigWCDMAObj.GetPICHCompensation();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:PICHcompensation:STATe?`

GetPICHTimingOffset (WCDMA)

Description: Gets PICH timming offset for the WCDMA measurement.

API:
```
public void GetPICHTimingOffset(out int
iPICHTimingOffset)
```

Arguments: **iPICHTimingOffset**

Contains the PICH timing Offset value when the call is returned.

VB6 Example:
```
Dim iPICHTimingOffset as Interger iPICHTimingOffset =
SignatureWCDMA.GetPICHTimingOffset
```

C#.NET Example:
```
double iPICHTimingOffset = 0.0; iPICHTimingOffset =
SigWCDMAObj.GetPICHTimingOffset();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:PICH:TIMingoffset?`

`GetRACHAnalysisLength` (WCDMA)

Description: Gets RACH Analysis Length for the WCDMA measurement.

API:
```
public void GetRACHAnalysisLength(out double dVal, out
_3GPPTimeUnits enum3GPPTimeUnits)
```

Arguments: **dVal**

Contains the RACH analysis length value when the call is returned.

enum3GPPTimeUnits

Contains the time units when the call is returned and is defined as an enumeration constant with the following value:

"_3GPP_us"

VB6 Example:
```
Dim dVal as double Dim enum3GPPTimeUnits as String
dVal =
SignatureWCDMA.GetRACHAnalysisLength(enum3GPPTimeUnits
)
```

C#.NET Example:
```
double dVal = 0.0; SignatureWCDMA._3GPPTimeUnits
enum3GPPTimeUnits =
SignatureWCDMA._3GPPTimeUnits._3GPP_ms; dVal
=SigWCDMAObj.GetRACHAnalysisLength(out
enum3GPPTimeUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:RACH:ANAlysis[:LENGth]?
```

GetRACHCaptureLength (WCDMA)

Description: Gets RACH Capture Length for the WCDMA measurement.

API:
```
public void GetRACHCaptureLength(out double dVal, out
_3GPPTimeUnits enum3GPPTimeUnits)
```

Arguments: **dVal**

Contains the RACH capture length value when the call is returned.

enum3GPPTimeUnits

Contains the time units when the call is returned and is defined as an enumeration constant with the following value:

"_3GPP_ms"

VB6 Example:
```
Dim dVal as Double Dim enum3GPPTimeUnits as String
dVal =
SignatureWCDMA.GetRACHCaptureLength(senum3GPPTimeUnits
)
```

C#.NET Example:
```
double dVal = 0.0; SignatureWCDMA._3GPPTimeUnits
enum3GPPTimeUnits =
SignatureWCDMA._3GPPTimeUnits._3GPP_ms; dVal =
SigWCDMAObj.GetRACHCaptureLength(out
enum3GPPTimeUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:RACH:CAPTure[:LENGth]?
```

GetRACHSummaryData (WCDMA)

Description: Gets RACH Summary Data for the WCDMA measurement.

API:
```
public void GetRACHSummaryData(RachSummaryParameters
enumRachSummaryParameters, out float
fRequestedSummaryParameter)
```

Arguments: **enumRachSummaryParameters**

Requested RACH summary parameter with the following values:

"PreamblesDisplayed"
"PreambleSignature"
"TimeOffset"
"OnOffRatio"
"FilteredPowerOn"
"FilteredPowerOff_Leading"
"FilteredPowerOffTrailing"
"FilteredPowerOffAverage"

fRequestedSummaryParameter

Contains the requested RACH summary parameter when the call returns.

VB6 Example:
```
Dim fRequestedSummaryParameter as double
fRequestedSummaryParameter =
SignatureWCDMA.GetRACHSummaryData ("TimeOffset")
```

C#.NET Example:
```
double fRequestedSummaryParameter = 0.0;
SignatureWCDMA.RachSummaryParameters
enumRachSummaryParameters =
SignatureWCDMA.RachSummaryParameters.TimeOffset;
fRequestedSummaryParameter =
SigWCDMAObj.GetRACHSummaryData(enumRachSummaryParamete
rs);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:RACH:SUMMary?
```

`GetReferenceLevel` (WCDMA)

Description: This method queries the reference level setting.

API:
```
public void GetReferenceLevel(out double
dRefLevel_out, out AmplitudeUnits
enumAmplitudeUnits_out)
```

Arguments: **dRefLevel_out**

Contains the reference level value when the call returns and is defined as a double. Ranges from –150 dBm to 30 dBm with a default value of 0 dBm.

enumAmplitudeUnits_out

Contains the reference level units when the call returns and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:
```
Dim dReferenceLevel As Double
Dim sRefLevelUnits As String
dReferenceLevel =
SignatureWCDMA.GetReferenceLevel(sRefLevelUnits)
```

C#.NET Example:
```
//Call to read the Reference Level value from the
system when in the WCDMA mode.
double dValue = 0.0;
SignatureWCDMA.AmplitudeUnits enumAmplitudeUnits  =
SignatureWCDMA.AmplitudeUnits.dBm;
dValue = SigWCDMAObj.GetReferenceLevel(out
enumAmplitudeUnits);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel?
```

GetReferenceLevelOffsetIndB (WCDMA)

Description: This method queries the reference level offset setting.

API:
```
public void GetReferenceLevelOffsetIndB(out double
dRefLvlOffsetValueindB_out)
```

Arguments: **dRefLvlOffsetValueindB_out**

Contains the reference level offset value when the call returns and is defined as a double. Ranges from –300 dB to +300 dB.

VB6 Example:
```
Dim dReferenceLevelOffset As Double
dReferenceLevelOffset =
SignatureWCDMA.GetReferenceLevelOffsetIndB
```

C#.NET Example:
```
//Call to read the Reference Level Offset from the
system when in WCDMA mode.
double dValue = 0.0;
dValue = SigWCDMAObj.GetReferenceLevelOffsetIndB();
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to
5>:Y[:SCALe]:RLEVel:OFFSet?
```

GetRRCFilter (WCDMA)

Description: This method queries the root raised cosine filter state.

API:
```
public void GetRRCFilter(out bool bRRCFilterOn)
```

Arguments: **bRRCFilterOn**

Contains a boolean value when the call returns with the following values:

True - root raised cosine filter is On
False - root raised cosine filter is Off

VB6 Example:
```
Dim bRRCFilterOn As Boolean
bRRCFilterOn = SignatureWCDMA.GetRRCFilter
```

C#.NET Example:
```
//Call to read the status of the RRC Filter while in
the WCDMA mode.
bool bRRCFilterON = SigWCDMAObj.GetRRCFilter();
//If bRRCFilterON == true (RRC Filter is turned ON)
//If bRRCFilterON == false (RRC Filter is turned OFF)
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:ACQUisition:RRCFilter:STATe?
```

GetSCHCompensation (WCDMA)

Description: Gets SCH compensation for the WCDMA measurement.

API:
```
public void GetSCHCompensation(out bool
bSCHCompensation)
```

Arguments: **bSCHCompensation**

Contains the boolean value when the call is returned.

VB6 Example:
```
Dim bSCHCompensation as Boolean bSCHCompensation =
SignatureWCDMA.GetSCHCompensation
```

C#.NET Example:
```
bool bSCHCompensation = true; bSCHCompensation =
SigWCDMAObj.GetSCHCompensation();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:SCHCompensation:STATe?`

GetSCHDataDisplayFormat (WCDMA)

Description: Gets SCH data display format for the WCDMA measurement.

API:
```
public void GetSCHDataDisplayFormat(out
BitstreamFormatConstants newVal)
```

Arguments: **newVal**

Contains the data display value when the call is returned and is defined as an enumeration constant with the following value:

"Binary"
"Hex"

VB6 Example:
```
Dim newVal as String newVal =
SignatureWCDMA.GetSCHDataDisplayFormat
```

C#.NET Example:
```
SignatureWCDMA.BitstreamFormatConstants. newVal =
SignatureWCDMA.BitstreamFormatConstants.Hex; newVal =
SigWCDMAObj.GetSCHDataDisplayFormat();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:SCH:DISPlay:FORMat?`

GetSCHSlotNumberforDataBits (WCDMA)

Description: Gets SCH slot number for databits for the WCDMA measurement.

API:
```
public void GetSCHSlotNumberforDataBits(out int
iSCHSlotNumber)
```

Arguments: **iSCHSlotNumber**

Contains the Slot number for the data bits when the call is returned.

VB6 Example:
```
Dim iSCHSlotNumber as Integer iSCHSlotNumberl =
SignatureWCDMA.GetSCHSlotNumberforDataBits
```

C#.NET Example:
```
double iSCHSlotNumber = 0.0; iSCHSlotNumber =
SigWCDMAObj.GetSCHSlotNumberforDataBits();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:SCH:SLOTnumber:DATAbits?
```

GetSCHSummaryData (WCDMA)

Description: Gets SCH summary data for the WCDMA measurement.

API:
```
public void GetSCHSummaryData(SchSummaryParameters
enumSchSummaryParameters, out float
fRequestedSummaryParameter)
```

Arguments: **enumSchSummaryParameters**

Requested SCH summary parameter with the following values:

“SCHState”
“SecondarySynchronizationCode”
“ScramblingCodeGroup”
“ScramblingCodeNumber”
“TotalSCHPowerAbsolute”
“PSCHPowerAbsolute”
“SSCHPowerAbsolute”
“TotalSCHPowerRelative”
“PSCHPowerRelative”
“SSCHPowerRelative”
“TransmitPower”

fRequestedSummaryParameter

Contains the requested SCH summary parameter when the call returns.

VB6 Example:
```
Dim fRequestedSummaryParameter as double
fRequestedSummaryParameter =
SignatureWCDMA.GetSCHSummaryData("PSCHPowerAbsolute")
```

C#.NET Example:
```
double fRequestedSummaryParameter = 0.0;
SignatureWCDMA.SchSummaryParameters
enumSchSummaryParameters =
SignatureWCDMA.SchSummaryParameters.PSCHPowerAbsolute;
fRequestedSummaryParameter =
SigWCDMAObj.GetSCHSummaryData(enumSchSummaryParameters
);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:SCH:SUMMary?
```

GetScramblingCode (WCDMA)

Description: This method queries the scrambling code value.

API: `public void GetScramblingCode(out int iValue)`

Arguments: **iValue**

Contains the scrambling code value when the call returns and is defined as a long value.

VB6 Example:

```
Dim lCode As Long
lCode = SignatureWCDMA.GetScramblingCode
```

C#.NET Example:

```
//Call to read the Scrambling code from the system.
int iValue = 0;
iValue = SigWCDMAObj.GetScramblingCode();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode?`

GetScramblingCodeForCompressedChannel (WCDMA)

Description: This method queries the scrambling code type while in the compressed mode.

API: `public void GetScramblingCodeForCompressedChannel(out ScramblingCodeTypes enumScramblingCodeTypes)`

Arguments: **enumScramblingCodeTypes**

Contains the scrambling code type when the call returns and is defined as an enumeration constant with the following values:

"Ordinary"
"Left_Alternate"
"Right_Alternate"

VB6 Example:

```
Dim sEnumScramblingCodeType As String
sEnumScramblingCodeType =
SignatureWCDMA.GetScramblingCodeForCompressedChannel
```

C#.NET Example:

```
//Call to read the Scrambling Code Type for Compressed
Channels from the system.
SignatureWCDMA.ScramblingCodeTypes
enumScramblingCodeTypes =
SignatureWCDMA.ScramblingCodeTypes.Left_Alternate;
enumScramblingCodeTypes =
SigWCDMAObj.GetScramblingCodeForCompressedChannel();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:COMPressedchannel?`

GetScramblingCodeMode (WCDMA)

Description: This method queries the scrambling code mode.

API:
```
public void GetScramblingCodeMode(out bool
bScramblingCodeAuto)
```

Arguments: **bScramblingCodeAuto**

Contains a boolean value when the call returns with the following values:

True - auto mode
False - not auto mode

VB6 Example:
```
Dim bScramblingCodeAuto As Boolean
bScramblingCodeAuto =
SignatureWCDMA.GetScramblingCodeMode
```

C#.NET Example:
```
//Call to read the Scrambling code mode from the
system.
bool bAuto = false;
bAuto = SigWCDMAObj.GetScramblingCodeMode();
//if bAuto == true (ScramblingCode mode is set to
Auto)
//if bAuto == false (ScramblingCode mode is set to
Manual)
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:AUTO?`

GetSpectrumInversionMode (WCDMA)

Description: This method queries the spectrum inversion mode.

API:
```
public void GetSpectrumInversionMode(out
InversionConstants enumInversionConstants)
```

Arguments: **enumInversionConstants**

Contains the inversion constant when the call returns and is defined as an enumeration constant with the following values:

"Norm"
"invert"

VB6 Example:
```
Dim sMode As String
sMode = SignatureWCDMA.GetSpectrumInversionMode
```

C#.NET Example:
```
//Call to read the WCDMA Spectrum Inversion mode from
the system.
SignatureWCDMA.InversionConstants
enumInversionConstants =
SignatureWCDMA.InversionConstants.Norm;
enumInversionConstants =
SigWCDMAObj.GetSpectrumInversionMode();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:ACQUisition:INVersion?
```

GetSweepMode (WCDMA)

Description: This method queries the sweep mode.

API:
```
public void GetSweepMode(out bool bContinuous_out)
```

Arguments: **bContinuous_out**

Contains a boolean value when the call returns with the following values:

True - continuous sweep mode
False - single sweep mode

VB6 Example:
```
Dim bContinuous As String
bContinuous = SignatureWCDMA.GetSweepMode
```

C#.NET Example:
```
//Call to read the Sweep Mode from the system when in
the WCDMA mode.
bool bContinuous = true;
bContinuous = SigWCDMAObj.GetSweepMode();
//If bContinuous = true //Continuous sweep.
//If bContinuous = false //Single sweep.
```

Associated GPIB Commands:
```
:INITiate<1|2>:CONTinuous?
```

GetSyncReferenceDownlink (WCDMA)

Description: This method queries the downlink synchronization reference type.

API:
```
public void GetSyncReferenceDownlink(out
SyncRefTypeDownlink enumSyncRefTypeDownlink)
```

Arguments: **enumSyncRefTypeDownlink**

Contains the downlink synchronization reference type when the call returns and is defined as an enumeration constant with the following values:

"ManualSync"
"PCPICH"

VB6 Example:
```
Dim sSyncRefTypeDownlink As String
sSyncRefTypeDownlink =
SignatureWCDMA.GetSyncReferenceDownlink
```

C#.NET Example:
```
//Call to read the Synchronization Reference setting
from the system while in the downlink mode.
SignatureWCDMA.SyncRefTypeDownlink
enumSyncRefTypeDownlink =
SignatureWCDMA.SyncRefTypeDownlink.PCPICH;
enumSyncRefTypeDownlink =
SigWCDMAObj.GetSyncReferenceDownlink();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference[:REFeren
ce]?
```

`GetSyncReferenceDownlinkManual` (WCDMA)

Description: This method queries the downlink synchronization reference code number and spreading factor.

API:
```
public void GetSyncReferenceDownlinkManual(out int
iCodeNum, out SpreadingFactor enumSpreadingFactor)
```

Arguments: **iCodeNum**

Contains the code number when the call returns and is defined as an integer.

enumSpreadingFactor

Contains the spreading factor when the call returns and is defined as an enumeration constant with the following values:

"MaxSpreadFactor"
"SpreadFactor512"
"SpreadFactor256"
"SpreadFactor128"
"SpreadFactor64"
"SpreadFactor32"
"SpreadFactor16"
"SpreadFactor8"
"SpreadFactor4"
"SpreadFactor2"
"SpreadFactor1"

VB6 Example:
```
Dim lChannelCode As Long
Dim sSpreadFactor As String
lChannelCode =
SignatureWCDMA.GetSyncReferenceDownlinkManual(sSpreadF
actor)
```

C#.NET Example:
```
//Call to read the Synchronization Reference code
number and the spread factor while in the downlink
mode.
int iValue = 0;
SignatureWCDMA.SpreadingFactor enumSpreadingFactor =
SignatureWCDMA.SpreadingFactor.SpreadFactor256;
iValue =
SigWCDMAObj.GetSyncReferenceDownlinkManual(out
enumSpreadingFactor);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference:SPFactor
?
```

GetTPCAveragingTimePerSlot (WCDMA)

Description: Gets TPC averaging time per slot for the WCDMA measurement.

API:
```
public void GetTPCAveragingTimePerSlot(out float
fTimeVal, out _3GPPTimeUnitsAve timeUnit)
```

Arguments:

fTimeVal

Contains the averaging time per slot value when the call is returned.

timeUnit

Contains the time units when the call is returned and is defined as an enumeration constant with the following values:

"_3GPP_Aveus"
"_3GPP_Avems"

VB6 Example:
```
Dim fTimeVal as double Dim timeUnit as String fTimeVal
=
SignatureWCDMA.GetTPCAveragingTimePerSlot(timeUnit)
```

C#.NET Example:
```
double fTimeVal = 0.0 SignatureWCDMA._3GPPTimeUnitsAve
timeUnit =
SignatureWCDMA._3GPPTimeUnitsAve._3GPP_Avems; fTimeVal
= SigWCDMAObj.GetTPCAveragingTimePerSlot(out
timeUnit);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:TPC:AVERage:TIMeperslot?
```

GetTPCComandPattern (WCDMA)

Description: Gets TPC command pattern for the WCDMA measurement.

API: `public void GetTPCComandPattern(out TPCCommandPattern newVal)`

Arguments: **newVal**

Contains the Command Pattern value when the call is returned and is defined as an enumeration constant with the following value:

"MinusOne"
"Zero"
"PlusOne"

VB6 Example:
```
Dim newVal as String newVal =
SignatureWCDMA.GetTPCComandPattern
```

C#.NET Example:
```
SignatureWCDMA.TPCCommandPattern newVal =
SignatureWCDMA.TPCCommandPattern.Zero; newVal =
SigWCDMAObj.GetTPCComandPattern();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:COMMand:PATTern?`

GetTPCLimitCheck (WCDMA)

Description: Gets TPC limit check for the WCDMA measurement.

API: `public void GetTPCLimitCheck(out bool bTPCLimitCheck)`

Arguments: **bTPCLimitCheck**

Contains the boolean value when the call is returned.

VB6 Example:
```
Dim bTPCLimitCheck as Boolean bTPCLimitCheck =
SignatureWCDMA.GetTPCLimitCheck
```

C#.NET Example:
```
bool bTPCLimitCheck = true; bTPCLimitCheck =
SigWCDMAObj.GetTPCLimitCheck();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:LIMitcheck:STATe?`

GetTPCStepSize (WCDMA)

Description: Gets TPC step size for the WCDMA measurement.

API: `public void GetTPCStepSize(out TPCStepSize newVal)`

Arguments: **newVal**

Contains the Step Size value when the call is returned and is defined as an enumeration constant with the following value:

"One"
"Two"
"Three"

VB6 Example:
```
Dim newVal as String newVal =
SignatureWCDMA.GetTPCStepSize
```

C#.NET Example:
```
SignatureWCDMA.TPCStepSize newVal =
SignatureWCDMA.TPCStepSize.Two; newVal =
SigWCDMAObj.GetTPCStepSize();
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:STEP:SIZe?`

GetTPCTimeRange (WCDMA)

Description: Gets TPC time range for the WCDMA measurement.

API:
```
public void GetTPCTimeRange(out int nTimeVal, out
_3GPPTimeFrameSlotUnits timeUnit)
```

Arguments: **nTimeVal**

Contains the time range value when the call is returned.

timeUnit

Contains the time units when the call is returned and is defined as an enumeration constant with the following value:

"_3GPP_Frame _3GPP_Slot"

VB6 Example:
```
Dim nTimeVal as Integer Dim timeUnit as String
nTimeVal =   SignatureWCDMA.GetTPCTimeRange(timeUnit)
```

C#.NET Example:
```
double nTimeVal =0.0;
SignatureWCDMA._3GPPTimeFrameSlotUnits timeUnit =
SignatureWCDMA._3GPPTimeFrameSlotUnits._3GPP_Slot;
nTimeVal = SigWCDMAObj.GetTPCTimeRange(out timeUnit);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:TIMe[:RANGe]?`

GetTransmitDiversity (WCDMA)

Description: This method queries the transmit diversity setting.

API:
```
public void GetTransmitDiversity(out TransmitDiversity
enumTransmitDiversity)
```

Arguments: **enumTransmitDiversity**

Contains the transmit diversity setting when the call returns and is defined as an enumeration constant with the following values:

"TxDiversityOff"
"Antenna1"
"Antenna2"

VB6 Example:
```
Dim sTransmitDiversity As String
sTransmitDiversity =
SignatureWCDMA.GetTransmitDiversity
```

C#.NET Example:
```
//Call to read the Transmit Diversity setting from the
system.
SignatureWCDMA.TransmitDiversity enumTransmitDiversity
= SignatureWCDMA.TransmitDiversity.TxDiversityOff;
enumTransmitDiversity =
SigWCDMAObj.GetTransmitDiversity();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity[:ANT
enna]?
```

GetTransmitDiversityType (WCDMA)

Description: This method queries the transmit diversity type setting.

API:
```
public void GetTransmitDiversityType(out
TransmitDiversityType enumTransmitDiversityType)
```

Arguments: **enumTransmitDiversityType**

Contains the transmit diversity type setting when the call returns and is defined as an enumeration constant with the following values:

"STTD_OFF"
"STTD_ON"

VB6 Example:
```
Dim sTransmitDiversity As String
sTransmitDiversity =
SignatureWCDMA.GetTransmitDiversityType
```

C#.NET Example:
```
//Call to read the Transmit Diversity Type setting.
SignatureWCDMA.TransmitDiversityType
enumTransmitDiversityType =
SignatureWCDMA.TransmitDiversityType.STTD_OFF;
enumTransmitDiversityType =
SigWCDMAObj.GetTransmitDiversityType();
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity:TYPE
?
```

GetTriggerDelayInSecs (WCDMA)

Description: This method queries the trigger delay in seconds.

API:
```
public void GetTriggerDelayInSecs(out double
dnewValueinSecs_out)
```

Arguments: **dnewValueinSecs_out**

Contains the trigger delay value when the call returns and is defined as a double. Ranges from –65.5 ms to 65.5 ms with a default value of 0 ms.

VB6 Example:
```
Dim dTriggerDelayInSecs As Double
dTriggerDelayInSecs =
SignatureWCDMA.GetTriggerDelayInSecs
```

C#.NET Example:
```
//Call to read the Trigger delay setting when in the
WCDMA mode.
double dValue = 0.0;
dValue = SigWCDMAObj.GetTriggerDelayInSecs();
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:HOLDoff?
```

`GetTriggerSource` (WCDMA)

Description: This method queries the trigger source setting.

API:
```
public void GetTriggerSource(out TriggerSource
enumTriggerSource_out)
```

Arguments: **enumTriggerSource_out**

Contains the trigger source when the call returns and is defined as an enumeration constant with the following values:

"FreeRun"
"WideIF"
"Line"
"External"
"Video"
"ExternalTTL"

VB6 Example:
```
Dim sTriggerSource As String
sTriggerSource = SignatureWCDMA.GetTriggerSource
```

C#.NET Example:
```
//Call to read the Trigger Source setting from the
system when in the WCDMA mode.
SignatureWCDMA.TriggerSource enumTriggerSource =
SignatureWCDMA.TriggerSource.FreeRun;
enumTriggerSource = SigWCDMAObj.GetTriggerSource();
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:SOURce?
```

GetVideoTriggerLevel (WCDMA)

Description: This method queries the video trigger level setting.

API:

```
public void GetVideoTriggerLevel(out double
dnewValue_out, out AmplitudeUnits
enumAmplitudeUnits_out)
```

Arguments: **dnewValue_out**

Contains the video trigger level value when the call returns and is defined as a double. Ranges from:
Reference Level to (Reference Level – 10 x Scale/Div)
with a default value of:
Reference Level – 0.5 x (10 x Scale/Div)

enumAmplitudeUnits_out

Contains the video trigger level units when the call returns and is defined as an enumeration constant with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:

```
Dim dTriggerLevel As Double
Dim sTriggerUnits As String
dTriggerLevel =
SignatureWCDMA.GetVideoTriggerLevel(sTriggerUnits)
```

C#.NET Example:

```
//Call to read the Video Trigger Level setting from
the system when in the WCDMA mode.
double dValue = 0.0;
SignatureWCDMA.AmplitudeUnits enumAmplitudeUnits =
SignatureWCDMA.AmplitudeUnits.dBm;
dValue = SigWCDMAObj.GetVideoTriggerLevel(out
enumAmplitudeUnits);
```

Associated GPIB Commands:

```
:TRIGger<1|2>[:SEQuence]:LEVel:VIDeo?
```

GetWCDMACompositeSummaryData (WCDMA)

Description: This method queries the summary data.

Precondition: The active graph type must be set to any Composite graph with Summary turned On by using the front panel menus.

API:
```
public void
GetWCDMACompositeSummaryData(CompositeSummaryParameter
enumCompositeSummaryParameter, out float
fRequestedSummaryParameter_out)
```

Arguments: **enumCompositeSummaryParameter**

An input parameter that identifies the requested summary parameter when the call is made and is defined as an enumeration constant with the following values:

"FrequencyError", "RHO", "EVM"
"MaximumCompositeEVM"
"EVMPeakPosition"
"PhaseError"
"AmplitudeError"
"IQOffset"
"ScramblingCode"
"SCHPower"
"PSCHPower"
"SSCHPower"
"PeakCodeDomainError"

fRequestedSummaryParameter_out

Contains the requested summary parameter value when the call is made and is defined as a floating point number.

VB6 Example:
```
Dim sngVal As Single
Dim sCompositeSummaryParameter As String
sCompositeSummaryParameter = "FrequencyError"
sngVal = SignatureWCDMA.GetWCDMACompositeSummaryData _
(sCompositeSummaryParameter)
```

C#.NET Example:
```
//Call to read the WCDMA QPSK/Composite Summary data
from the system.
SignatureWCDMA.CompositeSummaryParameter
enumCompositeSummaryParameter =
SignatureWCDMA.CompositeSummaryParameter.EVM;
Single fRequestedSummaryParameter = 0.0f;
fRequestedSummaryParameter =
SigWCDMAObj.GetWCDMACompositeSummaryData(enumComposite
SummaryParameter);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:SUMMary?`

GetWCDMAQPSKSummaryData (WCDMA)

Description: This method queries the WCDMA QPSK summary data.

Precondition: The active graph type must be set to any QPSK graph with Summary turned On by using the front panel menus.

API:

```
public void
GetWCDMAQPSKSummaryData(QPSKSummaryParameter
enumQPSKSummaryParameter, out float
fRequestedSummaryParameter_out)
```

Arguments:

enumQPSKSummaryParameter

An input parameter that identifies the requested summary parameter when the call is made and is defined as an enumeration constant with the following values:

"FrequencyError"
"EVM"
"MaximumEVM"
"EVMPeakPosition"
"PhaseError"
"AmplitudeError"
"IQOffset"

fRequestedSummaryParameter_out

Contains the requested summary parameter value when the call is made and is defined as a floating point number.

VB6 Example:

```
Dim sngVal As Single
Dim sQPSKSummaryParameter As String
sQPSKSummaryParameter = "FrequencyError"
sngVal =
SignatureWCDMA.GetWCDMAQPSKSummaryData(sQPSKSummaryPar
ameter)
```

C#.NET Example:

```
//Call to read the WCDMA QPSK/Composite Summary data
from the system.
SignatureWCDMA.QPSKSummaryParameter
enumQPSKSummaryParameter =
SignatureWCDMA.QPSKSummaryParameter.EVM;
Single fRequestedSummaryParameter = 0.0f;
fRequestedSummaryParameter =
SigWCDMAObj.GetWCDMAQPSKSummaryData(enumQPSKSummaryPar
ameter);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:SUMMary?`

`IsTriggerEdgeRising` (WCDMA)

Description: This method queries the edge triggering state.

Precondition: The trigger source must not be set to FreeRun.

API:
```
public void IsTriggerEdgeRising(out bool bRising_out)
```

Arguments: **bRising_out**

Contains a boolean value when the call returns with the following values:

True - rising edge triggering
False - falling edge triggering

VB6 Example:
```
Dim bRising As Boolean
bRising = SignatureWCDMA.IsTriggerEdgeRising
```

C#.NET Example:
```
//Call to read the Trigger Slope setting from the
system when in the WCDMA mode.
bool bTriggerEdgeRising = true;
bTriggerEdgeRising =
SigWCDMAObj.IsTriggerEdgeRising();
//If bTriggerEdgeRising == true //Trigger Edge is
rising.
//If bTriggerEdgeRising == false //Trigger Edge is
falling.
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:SLOPe?
```

SetActiveChannelThreshold (WCDMA)

Description: This method sets the active channel threshold. This parameter is used for active and inactive code channel detection when the active code channel type is set to Auto.

API:
```
public void SetActiveChannelThreshold(float fValue,
AttenuationUnits enumAttenuationUnits)
```

Arguments: **fValue**

Contains the active channel threshold value when the call is sent and is defined as a floating point number. Ranges from –50 dB to –10 dB with a default value of –33 dB. The resolution is 1 dB.

enumAttenuationUnits

Contains the reference level offset units when the call is sent and is defined as an enumeration constant with the following value:

“dB”

VB6 Example:
```
Dim sngThreshold As Single
Dim sAttenUnits As String
sngThreshold = -33
sAttenUnits = "dB"
Call
SignatureWCDMA.SetActiveChannelThreshold(sngThreshold,
sAttenUnits)
```

C#.NET Example:
```
//Call to set the Active Channel Threshold to –30 dB.
Single sValue = –30.0f;
SignatureWCDMA.AttenuationUnits enumAttenuationUnits =
SignatureWCDMA.AttenuationUnits.dB;
SigWCDMAObj.SetActiveChannelThreshold(sValue,
enumAttenuationUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels:THResho
ld
```

SetActiveCodeChannelType (WCDMA)

Description: This API sets the method to identify the active and inactive code channels.

API:
```
public void
SetActiveCodeChannelType(ActiveCodeChannelType
enumActiveCodeChannelType)
```

Arguments: **enumActiveCodeChannelType**

Contains the active code channel type when the call returns and is defined as an enumeration constant with the following values:

"AutoChannelType"

VB6 Example:
```
Dim sActiveCodeChannelType As String
sActiveCodeChannelType = "AutoChannelType"
Call
SignatureWCDMA.SetActiveCodeChannelType(sActiveCodeCha
nnelType)
```

C#.NET Example:
```
//Call to set the Active Code Channel type to "Auto" in
the system.
SignatureWCDMA.ActiveCodeChannelType
enumActiveCodeChannelType =
SignatureWCDMA.ActiveCodeChannelType.AutoChannelType;
SigWCDMAObj.SetActiveCodeChannelType(enumActiveCodeCha
nnelType);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:ACTivechannels[:CODE]`

SetAnalysisLength (WCDMA)

Description: This method sets the time duration of the data to be analyzed.

API:
```
public void SetAnalysisLength(double dValue,
WCDMA3GPPTimeUnits enumWCDMA3GPPTimeUnits)
```

Arguments: **dValue**

Contains the analysis length value when the call is sent and is defined as a double. This parameter has a step size of 2 for QPSK measurements. For all other measurements, this parameter has a step size of Maximum Spreading Factor. The maximum analysis length is 2560 chips.

enumWCDMA3GPPTimeUnits

Contains the WCDMA analysis length units when the call is sent and is defined as an enumeration constant with the following values:

" _3GPP_Frame"
"_3GPP_Slot"
"_3GPP_Chip"
"_3GPP_s"
"_3GPP_ms"
"_3GPP_us"
"_3GPP_ns"

VB6 Example:
```
Dim dLength As Double
Dim sWCDMA3GPPTimeUnits As String
dLength = 2304
sWCDMA3GPPTimeUnits = "_3GPP_Chip"
Call SignatureWCDMA.SetAnalysisLength(dLength,
sWCDMA3GPPTimeUnits)
```

C#.NET Example:
```
//Call to set the WCDMA Analysis Length to 2304 chips.
double dValue = 2304.0;
SignatureWCDMA.WCDMA3GPPTimeUnits
enumWCDMA3GPPTimeUnits =
SignatureWCDMA.WCDMA3GPPTimeUnits._3GPP_Chip;
SigWCDMAObj.SetAnalysisLength(dValue,
enumWCDMA3GPPTimeUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:LENgth
```

SetAnalysisStart (WCDMA)

Description: This method sets the analysis start position. When the measuring object is set to anything other than QPSK, the frame beginning is used as the reference for the analysis start position.

API:
```
public void SetAnalysisStart(double dValue,
WCDMA3GPPTimeUnits enumWCDMA3GPPTimeUnits)
```

Arguments: **dValue**

Contains the analysis start value when the call is sent and is defined as a double. This parameter has a step size of 2 for QPSK measurements. For all other measurements, this parameter has a step size of Maximum Spreading Factor.

enumWCDMA3GPPTimeUnits

Contains the WCDMA analysis start units when the call is sent and is defined as an enumeration constant with the following values:

“ _3GPP_Frame”
“_3GPP_Slot”
“_3GPP_Chip”
“_3GPP_s”
“_3GPP_ms”
“_3GPP_us”
“_3GPP_ns”

VB6 Example:
```
Dim dStart As Double
Dim sWCDMA3GPPTimeUnits As String
dStart = 256
sWCDMA3GPPTimeUnits = "_3GPP_Chip"
Call SignatureWCDMA.SetAnalysisStart(dStart,
sWCDMA3GPPTimeUnits)
```

C#.NET Example:
```
//Call to set the WCDMA Analysis Start to 256 chips.
double dValue = 256.0;
SignatureWCDMA.WCDMA3GPPTimeUnits
enumWCDMA3GPPTimeUnits =
SignatureWCDMA.WCDMA3GPPTimeUnits._3GPP_Chip;
SigWCDMAObj.SetAnalysisStart(dValue,
enumWCDMA3GPPTimeUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:ACQUisition:ANAlysis:STARt
```

SetAttenuation (WCDMA)

Description: This method sets the input attenuation level.

API:
```
public void SetAttenuation(int iAttValue_in,
AttenuationUnits enumAttUnits_in)
```

Arguments: **iAttValue_in**

Contains the attenuation value when the call is sent and is defined as an integer. Ranges from 0 dB to 62 dB with a default value of 10 dB.

enumAttUnits_in

Contains the attenuation units when the call is sent and is defined as an enumeration constant with the following value:

“dB”

VB6 Example:
```
Dim iAttenuation As Integer
Dim sAttenuationUnits As String
iAttenuation = 10
sAttenuationUnits = "dB"
Call SignatureWCDMA.SetAttenuation(iAttenuation,
sAttenuationUnits)
```

C#.NET Example:
```
//Call to set the Input Attenuation Level to 10 dB when
in the WCDMA mode.
int iValue = 10;
SignatureWCDMA.AttenuationUnits enumAttenuationUnits =
SignatureWCDMA.AttenuationUnits.dB;
SigWCDMAObj.SetAttenuation(iValue,
enumAttenuationUnits);
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation`

SetAttenuationModeAuto (WCDMA)

Description: This method sets the attenuation mode.

API: `public void SetAttenuationModeAuto(bool bAuto_in)`

Arguments: **bAuto_in**

Contains a boolean value when the call is sent with the following values:

True - auto mode
False - manual mode

VB6 Example:

```
Dim bAutoMode As Boolean
bAutoMode = True
Call SignatureWCDMA.SetAttenuationModeAuto(bAutoMode)
```

C#.NET Example:

```
//Call to set the Attenuation mode to Auto when in the
WCDMA mode.
bool bAuto = true;
SigWCDMAObj.SetAttenuationModeAuto(bAuto);
```

Associated GPIB Commands: `:INPut<1|2>:ATTenuation:AUTO`

SetCaptureLength (WCDMA)

Description: This method sets the capture time duration.

API:
```
public void SetCaptureLength(double dValue,
WCDMA3GPPTimeUnits enumWCDMA3GPPTimeUnits)
```

Arguments: **dValue**

Contains the analysis capture length value when the call is sent and is defined as a double.

enumWCDMA3GPPTimeUnits

Contains the WCDMA capture length units when the call is sent and is defined as an enumeration constant with the following values:

“ _3GPP_Frame”

VB6 Example:
```
Dim dCaptureLen As Double
Dim sWCDMA3GPPTimeUnits As String
dCaptureLen = 1
sWCDMA3GPPTimeUnits = "_3GPP_Frame"
Call SignatureWCDMA.SetCaptureLength(dCaptureLen,
sWCDMA3GPPTimeUnits)
```

C#.NET Example:
```
//Call to set the WCDMA Capture Length to one Frame.
double dValue = 1.0;
SignatureWCDMA.WCDMA3GPPTimeUnits
enumWCDMA3GPPTimeUnits =
SignatureWCDMA.WCDMA3GPPTimeUnits._3GPP_Frame;
SigWCDMAObj.SetCaptureLength(dValue,
enumWCDMA3GPPTimeUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:ACQUisition:CAPTure:LENgth
```

SetCDPMarkerPosition (WCDMA)

Description: This method sets the specified code domain power marker's, code number, and spreading factor (position).

API:
```
public void SetCDPMarkerPosition(CDPGraphType
enumCDPGraphType, int iMarkerNum, int iCodeNum,
SpreadingFactor enumSpreadingFactor)
```

Arguments: **enumCDPGraphType**

An input parameter that contains the graph type when the call is made and is defined as an enumeration constant with one the following values:

"CDM_CODEDOMAINPOWER"
"CDM_CODEDOMAINPOWER_ZOOM"
"CDM_CODEDOMAINERROR"
"CDM_CODEDOMAINERROR_ZOOM"

iMarkerNum

An input parameter that contains the active marker number when the call is made and is defined as an integer. Ranges from 1 to 2.

iCodeNum

Contains the code number when the call returns and is defined as an integer.

enumSpreadingFactor

Contains the spreading factor when the call returns and is defined as an enumeration constant with one of the following values:

"MaxSpreadFactor"
"SpreadFactor512"
"SpreadFactor256"
"SpreadFactor128"
"SpreadFactor64"
"SpreadFactor32"
"SpreadFactor16"
"SpreadFactor8"
"SpreadFactor4"

VB6 Example:

```
Dim sCDPGraphType As String
Dim iMarkerNumber As Integer
Dim iCodeNumber As Integer
Dim sSpreadFactor As String
sCDPGraphType = "CDM_CODEDOMAINPOWER"
iMarkerNumber = 2
iCodeNumber = 125
sSpreadFactor = "SpreadFactor256"
Call
SignatureWCDMA.SetCDPMarkerPosition(sCDPGraphType,
iMarkerNumber, iCodeNumber, sSpreadFactor)
```

C#.NET Example:

```
//Call to set the Code Domain Power Marker1 position
to 10@256.
SignatureWCDMA.CDPGraphType enumCDPGraphType =
SignatureWCDMA.CDPGraphType.CDM_CODEDOMAINPOWER;
int iMarkerNum = 1;
int iCodeNum = 10;
SignatureWCDMA.SpreadingFactor enumSpreadingFactor =
SignatureWCDMA.SpreadingFactor.SpreadFactor256;
SigWCDMAObj.SetCDPMarkerPosition(enumCDPGraphType,
iMarkerNum, iCodeNum, enumSpreadingFactor);
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:WCDMa:MARKer<1|2>:CDPError?
```

SetCenterFrequency (WCDMA)

Description: This method sets the center frequency.

API:
```
public void SetCenterFrequency(double dFreqValue_in,
FrequencyUnits enumFreqUnits_in)
```

Arguments: **dFreqValue_in**

Contains the center frequency value when the call is made and is defined as a double.

enumFreqUnits_in

Contains the center frequency units when the call is made and is defined as an enumeration constant with the following values:

"Hz"
"KHz"
"MHz"
"GHz"

VB6 Example:
```
Dim dCenterFrequency As Double
Dim sFrequencyUnits As String
dCenterFrequency = 2000
sFrequencyUnits = "MHz"
Call
SignatureWCDMA.SetCenterFrequency(dCenterFrequency,
sFrequencyUnits)
```

C#.NET Example:
```
//Call to set the Center Frequency to 2 GHz when in the
WCDMA mode.
double dValue = 2.0;
SignatureWCDMA.FrequencyUnits enumFrequencyUnits =
SignatureWCDMA.FrequencyUnits.GHz;
SigWCDMAObj.SetCenterFrequency(dValue,
enumFrequencyUnits);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:FREQuency:CENTer
```

SetDisplayCompressedModeSignalsMode (WCDMA)

Description: This method sets the signal mode type.

API:
```
public void
SetDisplayCompressedModeSignalsMode(CompressedModeSign
alType enumCompressedModeSignalType)
```

Arguments: **enumCompressedModeSignalType**

Contains the compressed mode signal type when the call is sent and is defined as an enumeration constant with the following values:

"SignalOff"
"Auto_Code_Selection"
"Manual_Code_Selection"

"SignalOff" should be selected for normal mode.
"Auto_Code_Selection" or "Manual_Code_Selection" should be selected for compressed mode signal.
When "Auto_Code_Selection" is selected, the spreading factor and code number of the compressed signal is set to half the spreading factor and code number of an uncompressed signal.

VB6 Example:
```
Dim sCompressedModeSignalType As String
sCompressedModeSignalType = "SignalOff"
Call
SignatureWCDMA.SetDisplayCompressedModeSignalsMode(sCo
mpressedModeSignalType)
```

C#.NET Example:
```
//Call to set the Display Compressed Mode Signals mode
to "Auto".
SignatureWCDMA.CompressedModeSignalType
enumCompressedModeSignalType =
SignatureWCDMA.CompressedModeSignalType.Auto_Code_Sele
ction;
SigWCDMAObj.SetDisplayCompressedModeSignalsMode(enumCo
mpressedModeSignalType);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:COMPressedmode[:MODE]`

SetDownlinkUplink (WCDMA)

Description: Sets measurement type to Downlink or Uplink for WCDMA measurement.

API: `public void SetDownlinkUplink(bool bDownlinkUplink)`

Arguments: **bDownlinkUplink**

Contains the boolean value when the call is sent. True for Uplink and False for Downlink.

VB6 Example: `Call  SignatureWCDMA.SetDownlinkUplink(False)`

C#.NET Example: `SigWCDMAObj.SetDownlinkUplink(false);`

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:LINK:STATe`

SetExternalTriggerLevel (WCDMA)

Description: This method sets the external trigger level.

API:
```
public void SetExternalTriggerLevel(double
dnewValue_in, AmpUnits enumAmpUnits_in)
```

Arguments: **dnewValue_in**

Contains the external trigger level value when the call is sent and is defined as a double.

enumAmpUnits_in

Contains the amplitude units when the call is sent and is defined as an enumeration constant with the following values:

"Volt"
"mVolt"

VB6 Example:
```
Dim dTriggerLevel As Double
Dim sAmplitudeUnits As String
dTriggerLevel = 1.4
sAmplitudeUnits = "Volt"
Call
SignatureWCDMA.SetExternalTriggerLevel(dTriggerLevel,
sAmplitudeUnits)
```

C#.NET Example:
```
//Call to set the External Trigger Level to 1.4V.
double dValue = 1.4;
SignatureWCDMA.AmpUnits enumAmpUnits =
SignatureWCDMA.AmpUnits.Volt;
SigWCDMAObj.SetExternalTriggerLevel(dValue,
enumAmpUnits);
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:SOURce`

`SetFrequencyOffset` (WCDMA)

Description: This method sets the frequency offset value.

API:
```
public void SetFrequencyOffset(double dFreqValue_in,
FrequencyUnits enumFrequencyUnits_in)
```

Arguments:

dFreqValue_in

Contains the center frequency value when the call is sent and is defined as a double.

enumFrequencyUnits_in

Contains the frequency offset units when the call is sent and is defined as an enumeration constant with the following values:

“Hz”
“KHz”
“MHz”
“GHz”

VB6 Example:
```
Dim dFrequencyOffset As Double
Dim sFrequencyUnits As String
dFrequencyOffset = 1
sFrequencyUnits = "MHz"
Call
SignatureWCDMA.SetFrequencyOffset(dFrequencyOffset,
sFrequencyUnits)
```

C#.NET Example:
```
//Call to set the Frequency Offset to 0 Hz.
double dValue = 0.0;
SignatureWCDMA.FrequencyUnits enumFrequencyUnits =
SignatureWCDMA.FrequencyUnits.Hz;
SigWCDMAObj.SetFrequencyOffset(dValue,
enumFrequencyUnits);
```

Associated GPIB Commands: `[:SENSe<1|2>]:FREQuency:OFFSet`

SetGraphType (WCDMA)

Description: This method sets the graph type display.

API: `public void SetGraphType(GraphType enumGraphType_in)`

Arguments: **enumFrequencyUnits_in**

Contains the graph type when the call is sent and is defined as an enumeration constant with the following values:

"QPSK_VECTOR"
"QPSK_CONSTELLATION"
"QPSK_POWERVSTIME"
"QPSK_EVMVSTIME"
"QPSK_MAGNITUDEERRVSTIME"
"QPSK_PHASEERRVSTIME"
"QPSK_ALLERRSVSTIME"
"QPSK_EYEI"
"QPSK_EYEQ"
"QPSK_EYEIQ"
"COMPOSITE_VECTOR"
"COMPOSITE_CONSTELLATION"
"COMPOSITE_POWERVSTIME"
"COMPOSITE_EVMVSTIME"
"COMPOSITE_MAGNITUDEERRVSTIME"
"COMPOSITE_PHASEERRVSTIME"
"COMPOSITE_ALLERRSVSTIME"
"COMPOSITE_EYEI"
"COMPOSITE_EYEQ"
"COMPOSITE_EYEIQ"
"CDM_CODEDOMAINPOWER"
"CDM_CODEDOMAINPOWER_ZOOM"
"CDM_CODEDOMAINERROR"
"CDM_CODEDOMAINERROR_ZOOM"
"SINGLE_CHANNEL_VECTOR"
"SINGLE_CHANNEL_CONSTELLATION"
"SINGLE_CHANNEL_SUMMARY"
"SINGLE_CHANNEL_CODEPOWERVSSLOT"
"SINGLE_CHANNEL_CODEERRVSSLOT"
"SINGLE_CHANNEL_CODEEVMVSTIME"
"SINGLE_CHANNEL_MAGNITUDEERRVSTIME"
"SINGLE_CHANNEL_PHASEERRVSTIME"
"SINGLE_CHANNEL_EYEI"
"SINGLE_CHANNEL_EYEQ"
"SINGLE_CHANNEL_EYEIQ"

VB6 Example:

```
Dim sGraphType As String
sGraphType = "COMPOSITE_CONSTELLATION"
Call SignatureWCDMA.SetGraphType(sGraphType)
```

C#.NET Example:

```
//Call to set the WCDMA GraphType to Code Domain
Power.
SignatureWCDMA.GraphType enumGraphType =
SignatureWCDMA.GraphType.CDM_CODEDOMAINPOWER;
SigWCDMAObj.SetGraphType(enumGraphType);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DISPlay:FORMat`

SetIQDisplayRotation (WCDMA)

Description: This method sets the rotation angle of the QPSK and Composite IQ displays.

API:

```
public void SetIQDisplayRotation(RotationConstants
enumRotationConstants)
```

Arguments: **enumRotationConstants**

Contains the IQ display rotation constant when the call is sent and is defined as an enumeration constant with the following values:

"Degrees_0"
"Degrees_45"

VB6 Example:

```
Dim sRotation As String
sRotation = "Degrees_0"
Call SignatureWCDMA.SetIQDisplayRotation(sRotation)
```

C#.NET Example:

```
//Call to set the IQ display rotation to 0 degrees.
SignatureWCDMA.RotationConstants enumRotationConstants
= SignatureWCDMA.RotationConstants.Degrees_0;
SigWCDMAObj.SetIQDisplayRotation(enumRotationConstants
);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:ROTAtion`

SetMarkerMode (WCDMA)

Description: This method sets the specified marker's mode.

API:
```
public void SetMarkerMode(GraphType enumGraphType, int
iMarkerNum, MarkerMode enumMarkerMode_in)
```

Arguments: **enumGraphType**

An input parameter that contains the graph type when the call is made and is defined as an enumeration constant with one the following values:

"QPSK_VECTOR"
"QPSK_CONSTELLATION"
"QPSK_POWERVSTIME"
"QPSK_EVMVSTIME"
"QPSK_MAGNITUDEERRVSTIME"
"QPSK_PHASEERRVSTIME"
"QPSK_ALLERRSVSTIME"
"QPSK_EYEI"
"QPSK_EYEQ"
"QPSK_EYEIQ"
"COMPOSITE_VECTOR"
"COMPOSITE_CONSTELLATION"
"COMPOSITE_POWERVSTIME"
"COMPOSITE_EVMVSTIME"
"COMPOSITE_MAGNITUDEERRVSTIME"
"COMPOSITE_PHASEERRVSTIME"
"COMPOSITE_ALLERRSVSTIME"
"COMPOSITE_EYEI"
"COMPOSITE_EYEQ"
"COMPOSITE_EYEIQ"
"CDM_CODEDOMAINPOWER"
"CDM_CODEDOMAINPOWER_ZOOM"
"CDM_CODEDOMAINERROR"
"CDM_CODEDOMAINERROR_ZOOM"
"SINGLE_CHANNEL_VECTOR"
"SINGLE_CHANNEL_CONSTELLATION"
"SINGLE_CHANNEL_SUMMARY"
"SINGLE_CHANNEL_CODEPOWERVSSLOT"
"SINGLE_CHANNEL_CODEERRVSSLOT"
"SINGLE_CHANNEL_CODEEVMVSTIME"
"SINGLE_CHANNEL_MAGNITUDEERRVSTIME"
"SINGLE_CHANNEL_PHASEERRVSTIME"
"SINGLE_CHANNEL_EYEI"
"SINGLE_CHANNEL_EYEQ"
"SINGLE_CHANNEL_EYEIQ"

iMarkerNum

Contains the active marker number when the call returns and is defined as an integer. Ranges from 1 to 2.

enumMarkerMode_in

Contains the marker mode when the call is made and is defined as an enumeration constant with the following values:

“MarkerOn”
“MarkerOff”

VB6 Example:

```
Dim sGraphType As String
Dim iMarkerNumber As Integer
Dim sMarkerMode As String
sGraphType = "COMPOSITE_CONSTELLATION"
iMarkerNumber = 2
sMarkerMode = "MarkerOn"
Call SignatureWCDMA.SetMarkerMode(sGraphType,
iMarkerNumber, sMarkerMode)
```

C#.NET Example:

```
//Call to set the switch ON Marker 1 for the Code
Domain Power measurement.
SignatureWCDMA.GraphType enumGraphType =
SignatureWCDMA.GraphType.CDM_CODEDOMAINPOWER;
SignatureWCDMA.MarkerMode enumMarkerMode =
SignatureWCDMA.MarkerMode.MarkerOn;
int iMarkerNum = 1;
SigWCDMAObj.SetMarkerMode(enumGraphType, iMarkerNum,
enumMarkerMode);
```

Associated GPIB Commands:

```
:CALCulate<1|2>:WCDMa:MARKer<1|2>:STATe
```

SetMaxSpreadFactor (WCDMA)

Description: This method sets the maximum spreading factor.

API:
```
public void SetMaxSpreadFactor(MaxSpreadFactor
enumMaxSpreadFactor)
```

Arguments: **enumMaxSpreadFactor**

Contains the maximum spreading factor when the call is sent and is defined as an enumeration constant with the following values:

"SF256"
"SF512"

VB6 Example:
```
Dim sMaxSpreadFactor As String
sMaxSpreadFactor = "SF256"
Call
SignatureWCDMA.SetMaxSpreadFactor(sMaxSpreadFactor)
```

C#.NET Example:
```
//Call to set the Maximum Spread Factor to 256.
SignatureWCDMA.MaxSpreadFactor enumMaxSpreadFactor =
SignatureWCDMA.MaxSpreadFactor.SF256;
SigWCDMAObj.SetMaxSpreadFactor(enumMaxSpreadFactor);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:MAXSpreadfactor
```

SetPICHChannelNumber (WCDMA)

Description: Sets PICH channel number for the WCDMA measurement.

API:
```
public void SetPICHChannelNumber(int
iPICHChannelNumber)
```

Arguments: **iPICHChannelNumber**

Contains the PICH channel number value when the call is sent.

VB6 Example:
```
Call  SignatureWCDMA.SetPICHChannelNumber(20)
```

C#.NET Example:
```
SigWCDMAObj.SetPICHChannelNumber(20);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:TPC:PICH:CHANnelnumber
```

SetPICHCompensation (WCDMA)

Description: Sets PICH compensation for the WCDMA measurement.

API: `public void SetPICHCompensation(bool bPICHCompensation)`

Arguments: **bPICHCompensation**

Contains the boolean value when the call is sent.

VB6 Example: `Call  SignatureWCDMA.SetPICHCompensation(False)`

C#.NET Example: `SigWCDMAObj.SetPICHCompensation(false);`

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:PICHcompensation:STATe`

SetPICHTimingOffset (WCDMA)

Description: Sets PICH timming offset for the WCDMA measurement.

API: `public void SetPICHTimingOffset(int iPICHTimingOffset)`

Arguments: **iPICHTimingOffset**

Contains the PICH timing Offset value when the call is sent.

VB6 Example: `Call  SignatureWCDMA.SetPICHTimingOffset(136)`

C#.NET Example: `SigWCDMAObj.SetPICHTimingOffset(136);`

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:PICH:TIMIngoffset`

`SetRACHAnalysisLength` (WCDMA)

Description: Sets RACH Analysis Length for the WCDMA measurement.

API:
```
public void SetRACHAnalysisLength(double dVal,
_3GPPTimeUnits enum3GPPTimeUnits)
```

Arguments: **dVal**

Contains the RACH analysis length value when the call is sent.

enum3GPPTimeUnits

Contains the time units when the call is sent and is defined as an enumeration constant with the following value:

"_3GPP_us"

VB6 Example:
```
Call
SignatureWCDMA.SetRACHAnalysisLength(600,"_3GPP_us")
```

C#.NET Example:
```
SigWCDMAObj.SetRACHAnalysisLength(600,SignatureWCDMA._
3GPPTimeUnits._3GPP_us);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:RACH:ANAlysis[:LENGth]
```

`SetRACHCaptureLength` (WCDMA)

Description: Sets RACH Capture Length for the WCDMA measurement.

API:
```
public void SetRACHCaptureLength(double dVal,
_3GPPTimeUnits enum3GPPTimeUnits)
```

Arguments: **dVal**

Contains the RACH capture length value when the call is sent.

enum3GPPTimeUnits

Contains the time units when the call is sent and is defined as an enumeration constant with the following value:

"_3GPP_ms"

VB6 Example:
```
Call
SignatureWCDMA.SetRACHCaptureLength(50,"_3GPP_ms")
```

C#.NET Example:
```
SigWCDMAObj.SetRACHCaptureLength(50,SignatureWCDMA._3G
PPTimeUnits._3GPP_ms);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:RACH:CAPTure[:LENGth]
```

`SetReferenceLevel` (WCDMA)

Description: This method sets the reference level setting.

API:
```
public void SetReferenceLevel(double dRefLevel_in,
AmplitudeUnits enumAmplitudeUnits_in)
```

Arguments: **dRefLevel_in**

Contains the reference level value when the call is sent and is defined as a double. Ranges from –150 dBm to 30 dBm with a default value of 0 dBm.

enumAmplitudeUnits_in

Contains the reference level units when the call is sent and is defined as an enumeration constant with the following value:

"dBm", "dBmV", "dBuV", "W", "mW", "uW", "nW", "pW", "fW", "aW", "zW", "yW", "V", "mV", "uV", "nV", "pV"

VB6 Example:
```
Dim dReferenceLevel As Double
Dim sRefLevelUnits As String
dReferenceLevel = -10
sRefLevelUnits = "dBm"
Call SignatureWCDMA.SetReferenceLevel(dReferenceLevel,
sRefLevelUnits)
```

C#.NET Example:
```
//Call to set the Reference Level to 0 dBm.
double dValue = 0.0;
SignatureWCDMA.AmplitudeUnits enumAmplitudeUnits  =
SignatureWCDMA.AmplitudeUnits.dBm;
SigWCDMAObj.SetReferenceLevel(dValue,
enumAmplitudeUnits);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to 5>:Y[:SCALe]:RLEVel
```

SetReferenceLevelOffset (WCDMA)

Description: This method sets the reference level offset setting.

API:
```
public void SetReferenceLevelOffset(double
dRefLvlOffsetValue_in, AttenuationUnits
enumAttenuationUnits_in)
```

Arguments: **dRefLvlOffsetValue_in**

Contains the reference level offset value when the call is sent and is defined as a double. Ranges from –300 dB to 300 dB with a default value of 0 dB.

enumAttenuationUnits_in

Contains the reference level offset units when the call is sent and is defined as an enumeration constant with the following value:

“dB”

VB6 Example:
```
Dim dReferenceLevelOffset As Double
Dim dRefLevelUnits As String
dReferenceLevel = 2
sRefLevelUnits = "dB"
Call
SignatureWCDMA.SetReferenceLevelOffset(dReferenceLevel
, sRefLevelUnits)
```

C#.NET Example:
```
//Call to set the Reference Level Offset to 1 dB.
double dValue = 1.0;
SignatureWCDMA.AttenuationUnits enumAttenuationUnits =
SignatureWCDMA.AttenuationUnits.dB;
SigWCDMAObj.SetReferenceLevelOffset(dValue,
enumAttenuationUnits);
```

Associated GPIB Commands:
```
:DISPlay[:WINDow<1|2>]:TRACe<1 to
5>:Y[:SCALe]:RLEVel:OFFSet
```

SetRRCFilter (WCDMA)

Description: This method sets the root raised cosine filter state.

API: `public void SetRRCFilter(bool bRRCFilterOn)`

Arguments: **bRRCFilterOn**

Contains a boolean value when the call is sent with the following values:

True - root raised cosine filter On
False - root raised cosine filter Off

VB6 Example:

```
Dim bRRCFilterOn As Boolean
bRRCFilterOn = True
Call SignatureWCDMA.SetRRCFilter(bRRCFilterOn)
```

C#.NET Example:

```
//Call to switch ON the RRC Filter while in the WCDMA
measurement mode.
bool bSwitchON = true;
SigWCDMAObj.SetRRCFilter(bSwitchON);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:ACQUisition:RRCFilter:STATe`

SetSCHCompensation (WCDMA)

Description: Sets SCH compensation for the WCDMA measurement.

API: `public void SetSCHCompensation(bool bSCHCompensation)`

Arguments: **bSCHCompensation**

Contains the boolean value when the call is sent.

VB6 Example:

```
Call  SignatureWCDMA.SetSCHCompensation(False)
```

C#.NET Example:

```
SigWCDMAObj.SetSCHCompensation(false);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:SCHCompensation:STATe`

SetSCHDataDisplayFormat (WCDMA)

Description: Sets SCH data display format for the WCDMA measurement.

API:
```
public void
SetSCHDataDisplayFormat(BitstreamFormatConstants
newVal)
```

Arguments: **newVal**

Contains the data display value when the call is sent and is defined as an enumeration constant with the following value:

"Binary"
"Hex"

VB6 Example:
```
Call  SignatureWCDMA.SetSCHDataDisplayFormat("Binary")
```

C#.NET Example:
```
SigWCDMAObj.SetSCHDataDisplayFormat(SignatureWCDMA.Bit
streamFormatConstants.Binary);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:SCH:DISPlay:FORMat
```

SetSCHSlotNumberforDataBits (WCDMA)

Description: Sets SCH slot number for databits for the WCDMA measurement.

API:
```
public void SetSCHSlotNumberforDataBits(int
iSCHSlotNumber)
```

Arguments: **iSCHSlotNumber**

Contains the Slot number for the data bits when the call is sent.

VB6 Example:
```
Call  SignatureWCDMA.SetSCHSlotNumberforDataBits(5)
```

C#.NET Example:
```
SigWCDMAObj.SetSCHSlotNumberforDataBits(5);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:SCH:SLOTnumber:DATAbits
```

SetScramblingCode (WCDMA)

Description: This method sets the scrambling code value.

API:
```
public void SetScramblingCode(int iValue)
```

Arguments: **iValue**

Contains the scrambling code value when the call is sent and is defined as a long value.

VB6 Example:
```
Dim lCode As Long
lCode = &H3FFFF
Call SignatureWCDMA.SetScramblingCode(lCode)
```

C#.NET Example:
```
//Call to set the Scrambling code to 0x3def in the
system.
int iValue = 0x3def;
SigWCDMAObj.SetScramblingCode(iValue);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode
```

`SetScramblingCodeForCompressedChannel` (WCDMA)

Description: This method sets the scrambling code type while in the compressed mode.

API:
```
public void
SetScramblingCodeForCompressedChannel(ScramblingCodeTypes enumScramblingCodeTypes)
```

Arguments: **enumScramblingCodeTypes**

Contains the scrambling code type when the call returns and is defined as an enumeration constant with the following values:

"Ordinary"
"Left_Alternate"
"Right_Alternate"

VB6 Example:
```
sEnumScramblingCodeType = "Ordinary"
Call
SignatureWCDMA.SetScramblingCodeForCompressedChannel(sEnumScramblingCodeType)
```

C#.NET Example:
```
//Call to set the Scrambling Code for Compressed
Channel to Left Alternate in the system.
SignatureWCDMA.ScramblingCodeTypes
enumScramblingCodeTypes =
SignatureWCDMA.ScramblingCodeTypes.Left_Alternate;
SigWCDMAObj.SetScramblingCodeForCompressedChannel(enumScramblingCodeTypes);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:COMPressedchannel
```

SetScramblingCodeMode (WCDMA)

Description: This method sets the scrambling code mode. When set to false, the scrambling code needs to be manually set by the user using the Set-ScramblingCode API.

API:
```
public void SetScramblingCodeMode(bool
bScramblingCodeAuto)
```

Arguments: **bScramblingCodeAuto**

Contains a boolean value when the call is sent with the following values:

True - auto mode
False - manual mode

VB6 Example:
```
Dim bScramblingCodeAuto As Boolean
bScramblingCodeAuto = True
Call
SignatureWCDMA.SetScramblingCodeMode(bScramblingCodeAu
to)
```

C#.NET Example:
```
//Call to set the Scrambling code mode to Auto.
bool bAuto = true;
SigWCDMAObj.SetScramblingCodeMode(bAuto);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:SCRAmblingcode:AUTO`

SetSpectrumInversionMode (WCDMA)

Description: This method sets the spectrum inversion option.

API:
```
public void
SetSpectrumInversionMode(InversionConstants
enumInversionConstants)
```

Arguments: **enumInversionConstants**

Contains the inversion constant when the call is sent and is defined as an enumeration constant with the following values:

“Norm”
“invert”

VB6 Example:
```
Dim sMode As String
sMode = "Norm"
Call SignatureWCDMA.SetSpectrumInversionMode(sMode)
```

C#.NET Example:
```
//Call to set the Spectrum Inversion Mode to Normal.
SignatureWCDMA.InversionConstants
enumInversionConstants =
SignatureWCDMA.InversionConstants.Norm;
SigWCDMAObj.SetSpectrumInversionMode(enumInversionCons
tants);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:ACQUisition:INVersion`

SetSweepMode (WCDMA)

Description: This method sets the sweep mode.

API:
```
public void SetSweepMode(bool bContinuous_in)
```

Arguments: **bContinuous_in**

Contains a boolean value when the call is sent with the following values:

True - continuous sweep mode
False - single sweep mode

VB6 Example:
```
bContinuous = True
Call SignatureWCDMA.SetSweepMode(bContinuous)
```

C#.NET Example:
```
//Call to set the Sweep Mode to continuous.
bool bContinuous = true;
SigWCDMAObj.SetSweepMode(bContinuous);
```

Associated GPIB Commands: `:INITiate<1|2>:CONTinuous`

SetSyncReferenceDownlink (WCDMA)

Description: This method sets the downlink synchronization reference type. When set to "PCPICH," primary CPICH is used as the synchronization reference. When set to "ManualSync," the code number and spreading factor needs to be manually set by the user using the SetSyncReferenceDownlinkManual API.

API:
```
public void
SetSyncReferenceDownlink(SyncRefTypeDownlink
enumSyncRefTypeDownlink)
```

Arguments: **enumSyncRefTypeDownlink**

Contains the reference syncronization downlink type when the call is sent and is defined as an enumeration constant with the following values:

"ManualSync"
"PCPICH"

VB6 Example:
```
Dim sSyncRefTypeDownlink As String
sSyncRefTypeDownlink = "PCPICH"
Call
SignatureWCDMA.SetSyncReferenceDownlink(sSyncRefTypeDo
wnlink)
```

C#.NET Example:
```
//Call to set the Synchronization Reference to PCPICH
when the system is in the downlink mode.
SignatureWCDMA.SyncRefTypeDownlink
enumSyncRefTypeDownlink =
SignatureWCDMA.SyncRefTypeDownlink.PCPICH;
SigWCDMAObj.SetSyncReferenceDownlink(enumSyncRefTypeDo
wnlink);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference[:REFeren
ce]
```

SetSyncReferenceDownlinkManual (WCDMA)

Description: This method sets the downlink synchronization reference code number and the spreading factor. When the syc reference is set to “ManualSync,” the code number and spreading factor of any active channel can be used as the synchronization reference.

API:

```
public void SetSyncReferenceDownlinkManual(int
iCodeNum, SpreadingFactor enumSpreadingFactor)
```

Arguments:

iCodeNum

Contains the code number when the call returns and is defined as an integer.

enumSpreadingFactor

Contains the spreading factor when the call returns and is defined as an enumeration constant with the following values:

“MaxSpreadFactor”
“SpreadFactor512”
“SpreadFactor256”
“SpreadFactor128”
“SpreadFactor64”
“SpreadFactor32”
“SpreadFactor16”
“SpreadFactor8”
“SpreadFactor4”

VB6 Example:

```
lChannelCode = 0
sSpreadFactor = "SpreadFactor256"
Call
SignatureWCDMA.SetSyncReferenceDownlinkManual(lChannel
Code, sSpreadFactor)
```

C#.NET Example:

```
//Call to set the Synchronization Reference code
number and the spread factor to 10@256 while in the
downlink mode.
int iValue = 10;
SignatureWCDMA.SpreadingFactor enumSpreadingFactor =
SignatureWCDMA.SpreadingFactor.SpreadFactor256;
SigWCDMAObj.SetSyncReferenceDownlinkManual(iValue,
enumSpreadingFactor);
```

Associated GPIB Commands:

```
[:SENSe<1|2>]:WCDMa:DMODulation:SYNCreference:SPFactor
```

SetTPCAveragingTimePerSlot (WCDMA)

Description: Sets TPC averaging time per slot for the WCDMA measurement.

API:
```
public void SetTPCAveragingTimePerSlot(float fTimeVal,
_3GPPTimeUnitsAve timeUnit)
```

Arguments: **fTimeVal**

Contains the averaging time per slot value when the call is sent.

timeUnit

Contains the time units when the call is sent and is defined as an enumeration constant with the following values:

"_3GPP_Aveus"
"_3GPP_Avems"

VB6 Example:
```
Call
SignatureWCDMA.SetTPCAveragingTimePerSlot(500,"_3GPP_A
veus")
```

C#.NET Example:
```
SigWCDMAObj.SetTPCAveragingTimePerSlot(500,SignatureWC
DMA._3GPPTimeUnitsAve._3GPP_Aveus);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:AVERage:TIMeperslot`

SetTPCComandPattern (WCDMA)

Description: Sets TPC command pattern for the WCDMA measurement.

API:
```
public void SetTPCComandPattern(TPCCommandPattern
newVal)
```

Arguments: **newVal**

Contains the Command Pattern value when the call is sent and is defined as an enumeration constant with the following value:

"MinusOne"
"Zero"
"PlusOne"

VB6 Example:
```
Call  SignatureWCDMA.SetTPCComandPattern("MinusOne")
```

C#.NET Example:
```
SigWCDMAObj.SetTPCComandPattern(SignatureWCDMA.TPCComm
andPattern.MinusOne);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:COMMand:PATTern`

SetTPCLimitCheck (WCDMA)

Description: Sets TPC limit check for the WCDMA measurement.

API: `public void SetTPCLimitCheck (bool bTPCLimitCheck)`

Arguments: **bTPCLimitCheck**

Contains the boolean value when the call is sent.

VB6 Example: `Call  SignatureWCDMA.SetTPCLimitCheck(False)`

C#.NET Example: `SigWCDMAObj.SetTPCLimitCheck(false);`

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:LIMitcheck:STATe`

SetTPCStepSize (WCDMA)

Description: Sets TPC step size for the WCDMA measurement.

API: `public void SetTPCStepSize(TPCStepSize newVal)`

Arguments: **newVal**

Contains the Step Size value when the call is sent and is defined as an enumeration constant with the following values:

"One"
"Two"
"Three"

VB6 Example: `Call  SignatureWCDMA.SetTPCStepSize("One")`

C#.NET Example: `SigWCDMAObj.SetTPCStepSize(SignatureWCDMA.TPCStepSize.One);`

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:TPC:STEP:SIZe`

SetTPCTimeRange (WCDMA)

Description: Sets TPC time range for the WCDMA measurement.

API:
```
public void SetTPCTimeRange(int nTimeVal,
_3GPPTimeFrameSlotUnits timeUnit)
```

Arguments: **nTimeVal**

Contains the time range value when the call is sent.

timeUnit

Contains the time units when the call is sent and is defined as an enumeration constant with the following value:

"_3GPP_Frame _3GPP_Slot"

VB6 Example:
```
Call  SignatureWCDMA.SetTPCTimeRange(5,"_3GPP_Slot")
```

C#.NET Example:
```
SigWCDMAObj.SetTPCTimeRange(5,SignatureWCDMA._3GPPTime
FrameSlotUnits._3GPP_Slot);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:TPC:TIMe[:RANGe]
```

SetTransmitDiversity (WCDMA)

Description: This method sets the transmit diversity setting.

API:
```
public void SetTransmitDiversity(TransmitDiversity
enumTransmitDiversity)
```

Arguments: **enumTransmitDiversity**

Contains the transmit diversity setting when the call is sent and is defined as an enumeration constant with the following values:

"TxDiversityOff"
"Antenna1"
"Antenna2"

VB6 Example:
```
Dim sTransmitDiversity As String
sTransmitDiversity = "TxDiversityOff"
Call
SignatureWCDMA.SetTransmitDiversity(sTransmitDiversity
)
```

C#.NET Example:
```
//Call to set the Transmit Diversity to OFF.
SignatureWCDMA.TransmitDiversity enumTransmitDiversity
= SignatureWCDMA.TransmitDiversity.TxDiversityOff;
SigWCDMAObj.SetTransmitDiversity(enumTransmitDiversity
);
```

Associated GPIB Commands:
```
[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity[:ANT
enna]
```

SetTransmitDiversityType (WCDMA)

Description: This method sets the transmit diversity type.

API:
```
public void
SetTransmitDiversityType(TransmitDiversityType
enumTransmitDiversityType)
```

Arguments: **enumTransmitDiversityType**

Contains the transmit diversity type setting when the call is sent and is defined as an enumeration constant with the following values:

"STTD_OFF"
"STTD_ON"

VB6 Example:
```
Dim sTransmitDiversity As String
sTransmitDiversity = "STTD_OFF"
Call
SignatureWCDMA.SetTransmitDiversityType(sTransmitDiver
sity)
```

C#.NET Example:
```
//Call to set the Transmit Diversity Type to STTD_OFF.
SignatureWCDMA.TransmitDiversityType
enumTransmitDiversityType =
SignatureWCDMA.TransmitDiversityType.STTD_OFF;
SigWCDMAObj.SetTransmitDiversityType(enumTransmitDiver
sityType);
```

Associated GPIB Commands: `[:SENSe<1|2>]:WCDMa:DMODulation:TRANsmitdiversity:TYPE`

SetTriggerDelay (WCDMA)

Description: This method sets the trigger delay value.

API:
```
public void SetTriggerDelay(double dnewValue_in,
TimeUnits enumTimeUnits_in)
```

Arguments: **dnewValue_in**

Contains the trigger delay value when the call is sent and is defined as a double. Ranges from –65.5 ms to 65.5 ms with a default value of 0 ms.

enumTimeUnits_in

Contains the trigger delay units when the call is sent and is defined as an enumeration constant with the following values:

"ns" for nanoseconds
"us" for microseconds
"ms" for milliseconds
"s" for seconds
"ks" for kiloseconds

VB6 Example:
```
Dim dTriggerDelay As Double
Dim sDelayUnits As String
dTriggerDelay = 1
sDelayUnits = "ms"
Call SignatureWCDMA.SetTriggerDelay(dTriggerDelay,
sDelayUnits)
```

C#.NET Example:
```
//Call to set the Trigger Delay to 10 microseconds.
double dValue = 10.0;
SignatureWCDMA.TimeUnits enumTimeUnits =
SignatureWCDMA.TimeUnits.us;
SigWCDMAObj.SetTriggerDelay(dValue, enumTimeUnits);
```

Associated GPIB Commands:
```
:TRIGger<1|2>[:SEQuence]:HOLDoff
```

SetTriggerEdgeRising (WCDMA)

Description: This method sets the edge triggering to rising edge or falling edge.

API: `public void SetTriggerEdgeRising(bool bRising_in)`

Arguments: **bRising_in**

Contains a boolean value when the call is sent with the following values:

True - rising edge triggering
False - falling edge triggering

VB6 Example:
```
Call SignatureWCDMA.SetTriggerEdgeRising(True)
```

C#.NET Example:
```
//Call to set the Trigger Slope to Rising.
bool bTriggerEdgeRising = true;
SigWCDMAObj.SetTriggerEdgeRising(bTriggerEdgeRising);
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:SLOPe`

SetTriggerSource (WCDMA)

Description: This method sets the trigger source setting.

API:
```
public void SetTriggerSource(TriggerSource
enumTriggerSource_in)
```

Arguments: **enumTriggerSource_in**

Contains the trigger source when the call is sent and is defined as an enumeration constant with the following values:

"FreeRun"
"WideIF"
"Line"
"External"
"Video"
"ExternalTTL"

VB6 Example:
```
Dim sTriggerSource As String
sTriggerSource = "WideIF"
Call SignatureWCDMA.SetTriggerSource(sTriggerSource)
```

C#.NET Example:
```
//Call to set the Trigger Source to FreeRun.
SignatureWCDMA.TriggerSource enumTriggerSource =
SignatureWCDMA.TriggerSource.FreeRun;
SigWCDMAObj.SetTriggerSource(enumTriggerSource);
```

Associated GPIB Commands: `:TRIGger<1|2>[:SEQuence]:SOURce`

SetVideoTriggerLevel (WCDMA)

Description: This method sets the video trigger level value.

API:

```
public void SetVideoTriggerLevel(double dnewValue_in,
AmplitudeUnits enumAmpUnits_in)
```

Arguments:

dnewValue_in

Contains the video trigger level value when the call is sent and is defined as a double. Ranges from Reference Level to (Reference Level – 10 x Scale/Div) with a default value of Reference Level – 0.5 x (10 x Scale/Div).

enumAmpUnits_in

Contains the video trigger level units when the call is sent with the following values:

“dBm”, “dBmV”, “dBuV”, “W”, “mW”, “uW”, “nW”, “pW”, “fW”, “aW”, “zW”, “yW”, “V”, “mV”, “uV”, “nV”, “pV”

VB6 Example:

```
Dim dTriggerLevel As Double
Dim sTriggerUnits As String
dTriggerLevel = -10
sTriggerUnits = "dBm"
Call
SignatureWCDMA.SetVideoTriggerLevel(dTriggerLevel,
sTriggerUnits)
```

C#.NET Example:

```
//Call to set the Video Trigger Level to 0 dBm.
double dValue = 0.0;
SignatureWCDMA.AmplitudeUnits enumAmplitudeUnits =
SignatureWCDMA.AmplitudeUnits.dBm;
SigWCDMAObj.SetVideoTriggerLevel(dValue,
enumAmplitudeUnits);
```

Associated GPIB Commands:

```
:TRIGger<1|2>[:SEQuence]:LEVel:VIDeo
```

StartSweep (WCDMA)

Description: This method triggers a sweep when in the single sweep mode. This is a blocking call and does not return until the sweep is complete.

API:
```
public void StartSweep()
```

Arguments: None

VB6 Example:
```
Call SignatureWCDMA.StartSweep
```

C#.NET Example:
```
// Call to start a sweep. This is a blocking call.
SigWCDMAObj.StartSweep();
```

Associated GPIB Commands:
```
:INITiate<1|2>[:IMMediate]
```

3-6 Programming Examples

This section provides programming examples represented in the VB6 programming environment. Figure 3-1 shows a basic example of a Web Services program using the Web Services methods documented in this manual.

```
Const WSAddress = "SN040403P"
'Change this string to match Signature's computer name
Private Sub WS_Click()
    Dim SignatureSystemControl As New MSSOAPLib30.SoapClient30
    Dim SignatureSpectrum As New MSSOAPLib30.SoapClient30
    Dim XPosition_out As Double
    Dim YPosition_out As Double
    Dim enumAmplitudeUnits_out As String
    Const sMarkerNum_in = 1
    'Get handle to Signature
    SignatureSystemControl.MSSoapInit "http://" + WSAddress + _
    "/SignatureSystemControl/SignatureSystemControl.asmx?wsdl"
    SignatureSpectrum.MSSoapInit "http://" + WSAddress + _
    "/SignatureSpectrum/SignatureSpectrum.asmx?wsdl"
    'Send Preset method
    Call SignatureSystemControl.Preset
    'Enable 50MHz calibrator
    Call SignatureSystemControl.SwitchOnCalibratorSignal(True)
    'Set Center Frequency to 50 MHz
    Call SignatureSpectrum.SetCenterFrequency(50#, "MHz")
    'Set Span to 2 MHz
    Call SignatureSpectrum.SetFrequencySpan(2#, "MHz")
    'Set Sweep Mode to Single Sweep
    Call SignatureSpectrum.SetSweepMode("Single")
    'Take Sweep
    Call SignatureSpectrum.StartSweep
    'Enable Marker
    Call SignatureSpectrum.SetMarkerState(1, "MarkerOn")
    'Send Marker to the peak signal
    Call SignatureSpectrum.SetMarkerToPeak(1)
    'Query the Marker Frequency
    XPosition_out = SignatureSpectrum. _
    GetFrequencyMarkerPositionInHz(sMarkerNum_in)
    'Query the Marker Power
    YPosition_out = SignatureSpectrum.GetMarkerAmplitude _
    (sMarkerNum_in, enumAmplitudeUnits_out)
    'Output Result
    MsgBox "Marker Frequency = " & XPosition_out & " Hz" & vbCrLf & _
    "Marker Power = " & YPosition_out & " " & enumAmplitudeUnits_out
End Sub
```

Figure 3-1. Web Services Example Program

Appendix A
Quick Reference Guide

A-1 Introduction

The following sections provide a quick reference to the List of GPIB Commands and to the List of Web Services Methods, and are listed in alphabetical order.

A-2 List of GPIB Commands

A-3 List of Web Services Methods