

**Standard  
Commands for  
Programmable  
Instruments  
(SCPI)**

**Volume 1: Syntax and Style**

---

VERSION 1999.0  
May, 1999  
Printed in U.S.A.

## 1999 SCPI Syntax & Style

### NOTICE, STATEMENT of INTENT and AUTHORIZATION TO COPY

© Copyright 1999 SCPI Consortium

This document defines the Standard-Commands-for-Programmable-Instruments (SCPI) Consortium's SCPI standard. Although the Consortium hereby expressly disclaims any and all warranties whatsoever regarding the SCPI standard, the Consortium has published this document with the intent that the SCPI standard will be seriously considered and adopted, in whole, by the test and measurement marketplace. Consistent with that intent, permission is hereby granted by the Consortium to make copies of this entire document as a whole, PROVIDED THAT this **NOTICE, STATEMENT of INTENT and AUTHORIZATION TO COPY** shall prominently appear on each such whole copy. Further consistent with that intent, permission is hereby granted by the consortium to make copies of portions of this document, absent this **NOTICE, STATEMENT of INTENT and AUTHORIZATION TO COPY**, PROVIDED THAT each such portion-copies shall only be used in a manner which is consistent with the purposes of implementing, promoting, disseminating, or enhancing (as opposed to deviating from) the SCPI standard as herein defined, such as the inclusion of such a portion-copy in an instrument (or instrument-related) product manual which implements SCPI (as defined herein) or in use for design of such an instrument (or instrument related) product. However, under no circumstances may copies of this document, or any portion of this document, be made solely for purposes of sale of the copy or copies.

For information, contact:

Fred Bode, Executive Director  
SCPI Consortium  
2515 Camino del Rio South, Suite 340  
San Diego, CA 92108

Phone: (619) 297-1210  
Fax: (619) 297-5955  
Email: fbode@vxinl.com  
Web Page: <http://www.scpicconsortium.org>

European SCPI Consortium Contact:

John Pieper  
ACEA  
P.O. Box 134  
7640 AC Wierden  
The Netherlands

Phone: +31 546 57 79 94  
Fax: +31 546 57 55 75  
Email: [acea@compuserve.com](mailto:acea@compuserve.com)  
Web Page: <http://ourworld.compuserve.com/homepages/acea/>

# Foreword

---

Commercial computer-controlled test instruments introduced in the 1960s used a wide variety of non-standard, proprietary interfaces and communication protocols. In 1975, the Institute of Electrical and Electronic Engineers approved *IEEE 488-1975*. *IEEE 488* defined a standard electrical and mechanical interface for connectors and cables. It also defined handshaking, addressing, and general protocol for transmitting individual bytes of data to and from instruments and computers. This standard has been updated and is now *IEEE 488.1-1987*.

Although it solved the problem of how to send bytes of data between instruments and computers, *IEEE 488* did not specify the data bytes' meanings. Instrument manufacturers freely invented new commands as they developed new instruments. The format of data returned from instruments varied as well. By the early 1980s, work began on additional standards to specify how to interpret data sent via *IEEE 488*.

In 1987, the IEEE released *IEEE 488.2-1987, Codes, Formats, Protocols and Common Commands for Use with IEEE 488.1-1987*. This standard defined the roles of instruments and controllers in a measurement system and a structured scheme for communication. In particular, *IEEE 488.2* described how to send commands to instruments and how to send responses to controllers. It defined some frequently used "housekeeping" commands explicitly, but each instrument manufacturer was left with the task of naming any other types of command and defining their effect. *IEEE 488.2* specified how certain types of features should be implemented if they were included in an instrument. It generally did not specify which features or commands should be implemented for a particular instrument. Thus, it was possible that two similar instruments could each conform to *IEEE 488.2*, yet they could have an entirely different command set.

*Standard Commands for Programmable Instruments* (SCPI) is the new instrument command language for controlling instruments that goes beyond *IEEE 488.2* to address a wide variety of instrument functions in a standard manner. SCPI promotes consistency, from the remote programming standpoint, between instruments of the same class and between instruments with the same functional capability. For a given measurement function such as frequency or voltage, SCPI defines the specific command set that is available for that function. Thus, two oscilloscopes made by different manufacturers could be used to make frequency measurements in the same way. It is also possible for a SCPI counter to make a frequency measurement using the same commands as an oscilloscope.

SCPI commands are easy to learn, self-explanatory and account for both novice and expert programmer's usage. Once familiar with the organization and structure of SCPI, considerable efficiency gains can be achieved during control program development, independent of the control program language selected.

# Table of Contents

---

## Chapter 1 Introduction

1.1	Requirements .....	1-1
1.2	Organization .....	1-1
1.3	SCPI Goals .....	1-1
1.4	SCPI Usage .....	1-2
1.5	Instrument Interchangeability .....	1-3

## Chapter 2 References

## Chapter 3 Life Cycle

3.1	Adding a Capability .....	3-1
3.2	Obsoleting a Capability .....	3-1
3.3	Device Dependent Commands .....	3-2

## Chapter 4 SCPI Compliance Criteria

4.1	IEEE 488.2 Requirements .....	4-1
4.1.1	IEEE Mandated Commands .....	4-1
4.1.2	IEEE Optional Common Commands .....	4-1
4.1.3	IEEE Common Command Implications .....	4-1
4.1.3.1	Overlapped and Sequential Commands .....	4-2
4.1.3.2	*CLS .....	4-2
4.1.3.3	*OPC and *WAI .....	4-3
4.1.3.4	*OPC? .....	4-3
4.1.3.5	*RST .....	4-4
4.1.3.5.1	Interaction With the Synchronization Commands .....	4-4
4.1.3.5.2	Implications For *SAV and *RCL .....	4-4
4.1.3.5.3	*RST and *RCL as Overlapped Commands .....	4-4
4.1.3.6	*IDN? .....	4-5
4.2	SCPI Requirements .....	4-5
4.2.1	Required Commands .....	4-5
4.2.2	Optional Commands .....	4-6
4.2.3	Documentation Requirements .....	4-6

## Chapter 5 Notation

5.1	Interpreting Command Tables .....	5-1
5.2	Interpreting Syntax Flow Diagrams .....	5-2

# 1999 SCPI Syntax & Style

## Chapter 6 Program Headers

6.1	Common Command and Query Headers .....	6-1
6.2	Instrument-Control Headers .....	6-1
6.2.1	Mnemonic Generation Rules .....	6-1
6.2.2	Building the Command Tree .....	6-2
6.2.3	Queries .....	6-6
6.2.4	Traversal of the Header Tree .....	6-7
6.2.5	Multiple Capabilities and Numeric Keyword Suffixes .....	6-8
6.2.5.1	Single Instrument with Many Electrical Ports .....	6-8
6.2.5.2	Multiple Identical Capabilities .....	6-8
6.2.5.3	Logical Instruments .....	6-9

## Chapter 7 Parameters

7.1	Character Program Data .....	7-1
7.2	Decimal Numeric Program Data .....	7-1
7.2.1	<numeric_value> Definition .....	7-1
7.2.1.1	DEFault .....	7-1
7.2.1.2	MINimum MAXimum .....	7-2
7.2.1.3	UP/DOWN .....	7-2
7.2.1.3.1	STEP Subsystem Command Syntax .....	7-3
7.2.1.3.2	[:INCReement] <numeric_value> .....	7-3
7.2.1.3.3	:PDECade <numeric_value> .....	7-3
7.2.1.3.4	:MODE LINear LOGarithmic L125 L13 .....	7-3
7.2.1.3.5	:AUTO <Boolean> ONCE .....	7-4
7.2.1.3.6	STEP Subsystem Examples .....	7-4
7.2.1.4	INFinity and Negative INFinity (NINF) .....	7-4
7.2.1.5	Not A Number (NAN) .....	7-4
7.2.2	Unit Suffixes .....	7-5
7.3	Boolean Program Data .....	7-5
7.4	Coupling of Functions .....	7-5
7.4.1	Functional Coupling .....	7-6
7.4.2	Value Coupling .....	7-6
7.4.3	Automatic Coupling .....	7-7
7.5	Units of Measure and Suffixes .....	7-7
7.5.1	Units of Amplitude and Power .....	7-7
7.5.2	Expressing Unitless Quantities .....	7-9

## Chapter 8 Expressions

8.1	Function .....	8-1
8.2	Usage .....	8-1
8.3	Syntax .....	8-1
8.3.1	Numeric Expression .....	8-2
8.3.1.1	Syntax .....	8-2

## **1999 SCPI Syntax & Style**

8.3.1.2	Precedence Rules .....	8-3
8.3.1.3	Semantics .....	8-3
8.3.2	Channel Lists .....	8-3
8.3.3	Numeric Lists .....	8-6

## **Chapter 9 Status Reporting**

9.1	The Device-Dependent Register Model .....	9-3
9.2	Transition Filters .....	9-3
9.3	Operation Status Register .....	9-3
9.4	QUEStionable Data/Signal Status Register .....	9-4
9.5	Multiple Logical Instruments .....	9-5
9.6	Status Structure for the Expanded Capability Trigger Model .....	9-7

## **Chapter 10 \*RST Conditions**

## **Chapter 11 Naming Conventions**

11.1	:DEFIne <name>,<data> .....	11-2
11.2	:DEFIne? <name> .....	11-2
11.3	:DELet[e[:NAME] <name> .....	11-2
11.4	:DELet[e:ALL .....	11-2
11.5	:CATalog? .....	11-2

## **Chapter A Programming Tips**

# 1 Introduction

## 1.1 Requirements

This volume, “Syntax and Style,” is global in nature and shall be used with all other volumes of the SCPI standard.

## 1.2 Organization

The first five chapters of *Syntax and Style* are an introduction to the overall concept of SCPI 1999. They describe an overview of the standard, reference other standards, describe SCPI compliance criteria, and how to interpret command tables and syntax flow diagrams. The next three chapters in this volume describe the program headers, parameters and expressions from which SCPI commands and responses are built. Following this are chapters which describe the status reporting model, style guidelines for creating new commands, the effect of \*RST on parameter values, and facilities for managing named sets of data in an instrument.

This volume, Syntax and Style, is the first of the four-volume set that makes up the SCPI 1999 Standard. The second and largest volume is the SCPI Command Reference, which contains the actual language constructs which appear in instruments. The third volume, Data Interchange Format, defines a standard representation for data sets which may be used between instruments and applications, between applications, or directly between instruments. The fourth volume, Instrument Classes, defines the SCPI commands and behavior needed to implement functionality sets associated with common classes of instruments. These four volumes form the complete 1999 SCPI Standard and should be used as a set.

This document is intended to apply to “systems” instruments. It defines both organization and content of messages at the controller-to-instrument and instrument-to-controller information interchange level. This document was developed so that, an instrument designed in accordance with it, shall be able to conform to *IEEE Std. 488.1-1987 Standard Digital Interface for Programmable Instrumentation*, and *IEEE Std. 488.2-1987 Codes, Formats and Common Commands For Use With IEEE Std. 488.1-1987*. Conformance to *IEEE 488.1-1987* and *IEEE 488.2-1987* is not required by this document, recognizing that some instruments implement physical interfaces other than the *IEEE 488.1-1987*. However, SCPI is based upon the concepts and terminology used within these standards.

This document may impact the related person-to-instrument and the instrument-to-person information interchange levels. Though the main purpose of this document is to define the interface as it relates to remote control, implementation of these codes and formats may affect the “front panel” of the instrument involved.

## 1.3 SCPI Goals

The goal of *Standard Commands for Programmable Instruments* (SCPI) is to reduce Automatic Test Equipment (ATE) program development time. SCPI accomplishes this goal by providing a consistent programming environment for instrument control and data usage. This consistent programming environment is achieved by the use of defined program

## 1999 SCPI Syntax & Style

messages, instrument responses, and data formats across all SCPI instruments, regardless of manufacturer.

A consistent program environment uses the same commands and parameters to control instruments that have the same functionality. These program commands and parameters are sent from a controller to an instrument using *IEEE 488.1*, VXIbus, RS-232C, etc., interfaces. SCPI instruments are very flexible in accepting a range of command and parameter formats, which makes the instrument easier to program. The instrument responses sent back to the controller can be either data or status information. SCPI instrument response format of a particular query is well-defined and reduces the programming effort to understand instrument data and status information. Data information can be formatted so that it is device- and measurement-independent.

SCPI programming consistency is both vertical and horizontal. Vertical programming consistency defines program messages within an instrument class. An example of vertical consistency is using the same command for reading DC voltage from several different multimeters. Horizontal consistency is using the same command to control similar functions across instrument classes. For example, the trigger command would be the same for an identical trigger function found among counters, oscilloscopes, function generators, etc.

A key to consistent programming is the reduction of multiple ways to control similar instrument functions. The philosophy of SCPI is for the same instrument functions to be controlled by the same SCPI commands. To simplify learning, SCPI uses industry-standard names and terms that are manufacturer and customer supported.

SCPI provides several different levels of instrument control. Simple Measure commands provide users easy and quick control of SCPI instrumentation, while more detailed commands provide traditional instrument control.

SCPI is designed to be expanded with new defined commands in the future without causing programming problems. As new instruments are introduced, the intent is to maintain program compatibility with existing SCPI instruments. SCPI ATE test programs designed to run with new instruments may not be compatible with existing instruments. In other words, the test programs are upward-compatible, but not downward-compatible.

To promote wide industry acceptance, SCPI is publicly available for implementation by anyone, whether or not they are a member of the SCPI Consortium.

The Consortium does not release working documents or provisional documentation. Only approved standards are released to the public.

### 1.4

#### SCPI Usage

The advantage of SCPI for the ATE system programmer is reducing the time learning how to program new SCPI instruments after programming their first SCPI instrument.

Programmers who use programming languages such as BASIC, C, FORTRAN, etc., to send instrument commands to instruments will benefit from SCPI. Also, programmers who implement instrument device drivers for ATE program generators and/or software instrument front panels will benefit by SCPI's advantages. SCPI defines instrument

commands, parameters, data, and status. It is not an application package, programming language, or software intended for instrument front panel control.

SCPI is designed to be layered on top of the hardware-independent portion of *IEEE 488.2*. SCPI can be used with controller-to-instrument interfaces such as *IEEE 488.1*, VXIbus, RS-232C, etc.

### 1.5

#### Instrument Interchangeability

By providing a consistent programming environment, replacing one SCPI instrument with another SCPI instrument in an ATE system will usually require less effort than with non-SCPI instruments. SCPI is not a standard which completely provides for interchangeable instrumentation. SCPI helps move toward interchangeability by defining instrument commands and responses, but it does not define instrument functionality, accuracy, resolution, connectors, etc., which is needed to provide true instrument interchangeability without affecting the ATE system hardware and software.

## **1999 SCPI Syntax & Style**

---

## 2 References

- “ANSI S1.4”
- *ANSI X3.4-1977*, American National Standard Code for Information Interchange; ISO Std.646-1983, ISO 7 bit Coded Character Set for Information Interchange
- *ANSI X3.42-1975*, American National Standard Representation of Numeric Values in Character Strings for Information Interchange; ISO Std. 6093-1985, Representation of Numeric Values in Character Strings for Information Interchange
- *ANSI/EIA/TIA-562-1989*, Electrical Characteristics for an Unbalanced Digital Interface
- *ANSI/IEEE Std 181-1977*, IEEE Standard on Pulse Measurement and Analysis by Objective Techniques
- *ANSI/IEEE Std 194-1977*, IEEE Standard Pulse Terms and Definitions
- *ANSI/IEEE Std. 260-1978*, An American National Standard IEEE Standard Letter Symbols for Units of Measurement (SI Units, Customary Inch-Pound Units, and Certain Other Units); ISO Std.1000-1981, SI Units and Recommendations for the Use of Their Multiples and Certain Other Units
- *ANSI/IEEE Std 488.1-1987*, IEEE Standard Digital Interface for Programmable Instrumentation
- *ANSI/IEEE Std 488.2-1992*, IEEE Standard Codes, Formats, Protocols, and Common Commands for use with ANSI/IEEE Std 488.1-1987
- *ANSI/IEEE Std 754-1985*, IEEE Standard for Binary Floating-Point Arithmetic
- *Bell Telephone “Per BTSM 41004”*
- *“CCIR Recommendation 468-2”*
- *“CCITT Recommendation P53”*
- *CCITT Recommendation V.42, Fascile V111.1 (Blue book, 1988)*, Error Correcting Procedures for DCE’s Using Asynchronous-to-Synchronous Conversion
- *“Dolby Labs Bulletin No 19/4”*
- *EIA RS-232-D*, Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange
- *EIA RS-422*, Electrical Characteristics of Balanced Voltage Digital Interface Circuits
- *“Fields and Waves in Communication Electronics,”* Ramo, Whinnery, and Van Duzer
- *“IEC Recommendation 179”*

## 1999 SCPI Syntax & Style

- *IEEE Micro, Volume 8, Number 4, August, 1988, pp 62-76*
- *ISO Std. 2955-1983, Information processing—Representation of SI and other units in systems with limited character sets*
- *“On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform,”* F.J. Harris, Proc. of the IEEE, Vol 66-1, January, 1978, pp 51-83
- *SAE J2264 1995-04 Chassis Dynamometer Simulation of Road Load Using Coast Down Techniques.*
- *“Tuning a Control Loop Performance,”* Gregory K. McMillan, Instrument Society of America, ISBN 0-87664-694-1
- *VXI Consortium INC, VMEbus Extensions for Instrumentation Systems Specification, Revision 2.0*

## 3 Life Cycle

SCPI is a “living” standard. Additional commands will always be needed to meet the needs of new technologies and new instruments. The SCPI Consortium meets regularly to propose and review extensions to the command set. New commands may be proposed by members of the Consortium, and by other interested parties.

Proposals accepted by the Consortium are published and distributed to member companies and are available for immediate use. Approved proposals are reviewed annually. After the annual review, a new revision of the SCPI Standard is published, and the commands become a permanent part of the standard.

All copies of the SCPI Standard contain a version year of issue. All SCPI instruments can be queried to determine the version-year to which they conform.

### 3.1

#### **Adding a Capability**

In general, a command proposed as an extension to SCPI will be approved if it is well formed according to the rules set forth in “Syntax and Style”, if it conforms in style to commands in the subsystem where it is placed, and if there is not already a command to control the same functionality. The Consortium rejects commands only on the basis of objective criteria.

SCPI is designed to grow in an upwardly compatible way. For a control program, this means that the additions to SCPI will not change the meanings of existing commands, and standard programs will not need to be rewritten except to add new functionality. For an instrument, upward compatibility means that the instrument functionality will not become inaccessible due to additions to the language; existing instruments will not become obsolete by the extension of SCPI.

### 3.2

#### **Obsoleting a Capability**

It is possible that a command may someday have to be altered or deleted in order to permit important new functionality to be implemented. Such an event represents the failure of the extension process because it breaks upward compatibility. Proposals for changes that break upward compatibility will only be accepted if there is overwhelming evidence that the benefits to users and the member companies of the Consortium outweigh the cost to existing users.

The following commands have been deleted from the SCPI Volume II, Command Reference as a part of the SCPI life cycle process. Reuse is discouraged. Any reinstatement is to be accompanied by a careful review of compatibility issues with all previous versions of the SCPI standard.

KEYWORD	PARAMETER FORM	NOTES	DELETION DATE
STATus			1996
:QUEue			1996
:ENABLE	<event_queue_list>		1996
[:NEXT]?		[query only]	1996

## 1999 SCPI Syntax & Style

### 3.3

#### Device Dependent Commands

From time to time manufacturers may elect to build instruments with capabilities that are not covered by a current version of this standard. Presumably the manufacturer will submit these new commands into the SCPI Life Cycle process. If the consortium adopts commands that are incompatible those originally used by the manufacturer, the manufacturer is permitted to provide his original commands in subsequent products for compatibility.

## 4 SCPI Compliance Criteria

This section indicates mandatory and optional capabilities of SCPI. These capabilities are fully specified elsewhere, in “Syntax and Style”, “Command Reference”, “Data Interchange Format” and *IEEE 488.2*, and an instrument designer shall fully comply with the referenced sections. Only designs conforming to these requirements shall be able to claim conformance to the SCPI specification.

### 4.1 IEEE 488.2 Requirements

All SCPI instruments shall conform to the specifications for **devices** in *IEEE 488.2*, except that section *4.1 IEEE 488.1 Requirements* shall be omitted where an instrument does not implement an *IEEE 488.1* interface.

Additionally, a SCPI instrument shall be able to parse <compound command program header> and <compound query program header>, to handle the tree structured commands in SCPI.

#### 4.1.1 IEEE Mandated Commands

All SCPI instruments shall implement all the common commands declared mandatory by *IEEE 488.2*.

Mnemonic	Name	488.2 Section
*CLS	Clear Status Command	10.3
*ESE	Standard Event Status Enable Command	10.10
*ESE?	Standard Event Status Enable Query	10.11
*ESR?	Standard Event Status Register Query	10.12
*IDN?	Identification Query	10.14
*OPC	Operation Complete Command	10.18
*OPC?	Operation Complete Query	10.19
*RST	Reset Command	10.32
*SRE	Service Request Enable Command	10.34
*SRE?	Service Request Enable Query	10.35
*STB?	Read Status Byte Query	10.36
*TST?	Self-Test Query	10.38
*WAI	Wait-to-Continue Command	10.39

#### 4.1.2 IEEE Optional Common Commands

*IEEE Std. 488.2* describes several optional commands which may be implemented in a device. None of these commands are required by SCPI.

#### 4.1.3 IEEE Common Command Implications

Proper implementation of some of the *IEEE 488.2* common commands carry some implications which are not immediately obvious. Primarily this has to do with the synchronization commands.

#### 4.1.3.1 Overlapped and Sequential Commands

IEEE 488.2 defines a distinction between overlapped and sequential commands. As defined in IEEE 488.2, a sequential command is one which finishes executing before the next command starts executing. An overlapped command is one which does not finish executing before the next command starts executing. These types of commands are described in IEEE 488.2, section 12. Examples are given in IEEE 488.2, appendix B.

IEEE 488.2 defines three common commands (\*OPC, \*WAI, \*OPC?) which a device controller can use to synchronize its operation to the execution of overlapped commands. The definitions for these common commands appear in sections 10.18, 10.19, and 10.39 of IEEE 488.2. All three are required by IEEE 488.2, even if none of the device's commands are overlapped.

Implementing any device commands as overlapped will increase the complexity of the implementation of the three synchronization common commands.

Each overlapped command has associated with it a Pending Operation flag. The device sets this flag TRUE when it passes the corresponding command from the Execution Control block to the Device Action block. The device sets the flag false when the device operation is finished, or has been aborted.

IEEE 488.2 defines a No Operation Pending flag. This flag is FALSE whenever any Pending Operation flag is TRUE, and is TRUE whenever all Pending Operation flags are FALSE. IEEE 488.2 also permits device designers to decide which overlapped commands are included in the No Operation Pending flag. A device may implement commands which select which overlapped commands are included in the No Operation Pending flag, although the SCPI standard does not include such a command.

IEEE 488.2 requires user documentation to clearly indicate which commands, if any, are overlapped, and which overlapped commands are included and which are excluded from the No Operation Pending flag.

#### 4.1.3.2 \*CLS

This command clears all status data structures in a device. For a device which minimally complies with SCPI, these registers are:

SESR	(IEEE 488.2)
OPERation Status Register	(SCPI)
QUEStionable Status Register	(SCPI)
Error/Event Queue	(SCPI)

Execution of \*CLS shall also clear any additional status data structures implemented in the device. The corresponding enable registers are unaffected. See the table in Command Reference, 20.7.

\*CLS forces the device into OCIS and OQIS (see 4.1.3.3 and 4.1.3.4) without setting the No Operation Pending flag TRUE and without setting the OPC bit of the SESR TRUE and without placing a “1” into the Output Queue.

For example, suppose a device implements INITiate[:IMMediate] as an overlapped command. Assuming that the trigger model is programmed so that it will eventually return to the IDLE state, and that INITiate[:IMMediate] takes longer to execute than \*OPC, sending these commands to this device:

```
INITiate;*OPC
```

results in initiating the trigger model and, after some time, setting the OPC bit in the SESR. However, sending these commands:

```
INITiate;*OPC;*CLS
```

still initiates the trigger model. Since the operation is still pending when the device executes \*CLS, the device does not set the OPC bit until it executes another \*OPC command.

#### 4.1.3.3 \*OPC and \*WAI

A device is in the Operation Complete Command Active State (OCAS) after it has executed \*OPC. The device returns to the Operation Complete Command Idle State (OCIS) whenever the No Operation Pending flag is TRUE, at the same time setting the OPC bit of the SESR TRUE. See IEEE 488.2 Fig 12-4. The following events force the device into OCIS without setting the No Operation Pending flag TRUE and without setting the OPC bit of the SESR:

- power on
- receipt of the dcas message (device clear)
- execution of \*CLS
- execution of \*RST

Implementation of the \*OPC and \*WAI commands is straightforward in devices which implement only sequential commands. When executing \*OPC the device simply sets the OPC bit of SESR. Executing \*WAI is a no-operation. The device is, in effect, always in OCIS.

In devices which implement overlapped commands the implementation of \*OPC and \*WAI is a little more complicated. After executing \*OPC the device must not set the OPC bit of SESR until the device returns to OCIS, even though it continues to parse and execute commands. After executing \*WAI the device must execute no further commands or queries until the No Operation Pending flag is TRUE, or receipt of a dcas message, or a power on.

#### 4.1.3.4 \*OPC?

SCPI adds the requirement that \*OPC? is implemented as a sequential command.

A device is in the Operation Complete Query Active State (OQAS) after it has executed \*OPC?. The device returns to the Operation Complete Query Idle State (OQIS) whenever the No Operation Pending flag is TRUE, at the same time placing a "1" in the Output Queue. See IEEE 488.2 Fig 12-6. The following events force the device into OQIS without setting the No Operation Pending flag TRUE and without placing a "1" in the Output Queue:

- power on
- receipt of the dcas message (device clear)

## 1999 SCPI Syntax & Style

Implementation of the \*OPC? query is straightforward in devices which implement only sequential commands. When executing \*OPC? the device simply places a "1" in the Output Queue.

The implementation of overlapped commands in a device complicates the implementation of \*OPC? and places some restrictions on the implementation of the Message Exchange Protocol (MEP). IEEE 488.2 dictates that devices shall send query responses in the order that they receive the corresponding queries (IEEE 488.2 6.4.5.4). Although IEEE 488.2 recommends that \*OPC? be the last query in a program message, there is nothing to prevent a controller program from ignoring this suggestion. This is why \*OPC? must be sequential.

4

### 4.1.3.5 \*RST

The “\*RST Conditions” chapter of Volume 1 gives pretty a lucid description of the SCPI requirements of \*RST. There are, however, some implications.

#### 4.1.3.5.1 Interaction With the Synchronization Commands

\*RST stops the execution of any overlapped commands. The natural outcome of this is that the No Operation Pending flag will go TRUE. Execution of \*RST must place the device in OCIS before setting the No Operation Pending flag TRUE, so that the effect of any pending \*OPC command is nullified (that is, the OPC bit of the SESR does NOT get set).

#### 4.1.3.5.2 Implications For \*SAV and \*RCL

In devices which implement the optional \*SAV and \*RCL common commands, the scope of instrument settings affected by these commands shall be the same as that affected by \*RST. This means that if \*RST has any effect upon a command, then so must \*RCL. Conversely, if the device designer wants \*RCL to have an effect upon a command, then \*RST must also have an effect upon that command. Inversely, and more importantly, if the device designer wants a command to be EXCLUDED from the scope of \*SAV and \*RCL, then it must also be excluded from the scope of \*RST.

#### 4.1.3.5.3 \*RST and \*RCL as Overlapped Commands

While \*RST stops the execution of running overlapped commands, the instrument designer may find it useful to make the \*RST command, itself, overlapped. For example, if resetting requires the return to an initial position of a slow-reacting, electro-mechanical device. In this case, performance may be better served with \*RST overlapped as subsequent commands may be parsed and executed while the resetting is completing. Even while overlapped, the \*RST command shall always complete the resetting of device state variables before subsequent commands are processed to insure the integrity of \*RST exception programming. For similar reasons, \*RCL may be an overlapped command.

If \*RST or \*RCL is overlapped, its Pending-Operation flag shall be reported in the No-Operation-Pending flag.

If execution of \*RST or \*RCL causes a device operation which is equivalent to executing an overlapped command, the device shall set TRUE the Pending-Operation flag associated with that command.

## 1999 SCPI Syntax & Style

### 4.1.3.6 \*IDN?

IEEE 488.2 is purposefully vague about the content of each of the four fields in the response syntax. SCPI adds no further requirement, but here are some suggestions:

- All devices produced by a company should implement the \*IDN? response consistently.
- Field 1, the Manufacturer field, should be identical for all devices produced by a single company.
- Field 2, the Model field, should NOT contain the word “MODEL”.
- Field 4, the Firmware level field, should contain information about all separately revisable subsystems. This information can be contained in single or multiple revision codes.

## 4.2

### SCPI Requirements

IEEE 488.2 describes the syntax of programming and device behavior to a certain level. Further refinement of the syntax and addition rules are imposed by SCPI in order to give instruments a common “look and feel”. The “Syntax and Style” volume of this standard describes the concepts behind SCPI and sets guidelines for originating new commands. A SCPI device shall follow these guidelines and style requirements.

### 4.2.1 Required Commands

The following commands are required in all SCPI instruments:

Mnemonic	Command Reference Section	Syntax and Style Section
:SYSTem		
:ERRor	21.8	
[:NEXT]?	21.8.3e (see Note 1)	1996
:VERSion?	19.16 (see Note 2)	1991
:STATus	18	5
:OPERation		
[:EVENT]?		
:CONDITION?		
:ENABLE		
:ENABLE?		
:QUESTIONable		
[:EVENT]?		
:CONDITION?		
:ENABLE		
:ENABLE?		
:PRESet		

Note 1: The requirement changed from SYSTem:ERRor? to SYSTem:ERRor[:NEXT]? in version 1995.1.

Note 2: Requirement applies for all versions greater than 1990.0

### 4.2.2 Optional Commands

All other commands in the “Command Reference” are considered optional, depending on the capabilities of the instrument. That is, the control of any instrument capability that is described in SCPI shall be implemented exactly as specified by using the appropriate SCPI defined commands. For example, an instrument dedicated to voltage measurement would not implement commands for sensing frequency. Certain commands, if implemented, require that other commands also be implemented.

4

When a device does not support all alternative parameter values that are allowed for a SCPI command, it may implement a subset of these values unless otherwise stated. For example, if an instrument implements only RECTangular and UNIForm data shaping for its CALCulate:TRANSform:WINDow command, it may generate an error on receipt of any other SCPI defined parameter value (FLATtop,HAMMing, etc). However, a device must implement all parameters of a multi-parameter SCPI command.

Commands required to implement a SCPI-described capability may be omitted under special conditions. The special condition exists when a command to be implemented affects a part of the instrument in which the configuration is fixed, and that the fixed configuration for that command corresponds to the value it would take on when an *IEEE 488.2 \*RST* reset command is issued. For example, an instrument with a display that is configured to be permanently on does not need to implement the command that turns the display ON or OFF, since at *\*RST* it is required by SCPI to be ON. If an instrument has a fixed configuration in which a setting does not correspond to the *\*RST* value, then the instrument shall implement the command for that setting even though it has only a single legal value.

### 4.2.3 Documentation Requirements

The documentation for a SCPI instrument shall list the version number for which the instrument complies. This information shall appear on instrument specification sheets and related documents, as well as the programming manual.

The manual for a SCPI instrument shall have a separate section titled “*SCPI Conformance Information*”. This section shall list separately:

- The SCPI version to which the instrument complies
- The syntax of all SCPI confirmed commands implemented by the instrument. Confirmed commands are those commands which are published in the latest version of SCPI to which the instrument conforms. Commands in the DIAGnostic subsystem, and other commands which are not intended for end-user use need not be documented in this section.
- The syntax of all SCPI approved commands implemented by the instrument. Approved commands are those commands which have been approved by the SCPI Consortium, but are not contained in the version of SCPI to which the instrument conforms.
- The syntax of all commands implemented by the instrument, which are not part of

## **1999 SCPI Syntax & Style**

the SCPI definition.

## 5 Notation

Extensive use is made of syntax flow diagrams and tables throughout this document. Associated with these are notational styles, which are described in this chapter.

### 5.1 Interpreting Command Tables

Command tables are used to define a set of SCPI commands. A table shows the commands, their hierarchical relationships, related parameters (if any), and associated notes. The table is broken into 3 columns; the KEYWORD, the PARAMETER FORM, and any NOTES.

The KEYWORD column provides the name of the command. The actual name of the command consists of one or more keywords since SCPI commands are based on a hierarchical structure, also known as a **tree system**. In such a system, associated commands are grouped together under a common node in the hierarchy, analogous to the way leaves at a same level are connected at a common branch. This and similar branches are connected to fewer and thicker branches, until they meet at the root of the tree. The closer to the root, the higher a node is considered in the hierarchy. To obtain a particular leaf or a particular command, the full path to it must be specified. This path is represented in the tables by placing the highest node in the hierarchy in the left-most position. Lower nodes in the hierarchy are indented one position to the right, below the parent node.

Square brackets ([ ]) are used to enclose a keyword that is optional when programming the command; that is, the instrument shall process the command to have the same effect whether the option node is omitted by the programmer or not. Such a node is called a **default node**. Letter case in tables is used to differentiate between the accepted short form (the uppercase characters) and the long form (the whole keyword). The significance of the short and long form keywords is explained in the “Program Headers” chapter.

The PARAMETER FORM column indicates the number and order of parameters in a command and their legal values. A command may allow the use of a SCPI-defined parameter type, a literal, or a combination of the two. The SCPI-defined parameter types are described in the “Parameters” chapter, and are distinguished by enclosing the type name in angle brackets (<>). A literal is typically a word that enumerates a setting that cannot be described using the SCPI-defined parameter types.

In the PARAMETER FORM column, a number of characters have special significance. Square brackets ([ ]) are used to enclose one or more parameters that are optional when controlling the instrument. Braces ({ }), or curly brackets, are used to enclose one or more parameters that may be included zero or more times. The vertical bar (|) can be read as “or” and is used to separate alternative parameter options.

The query form of a command is generated by appending a question mark to the last keyword. However, not all commands have a query form, and some commands exist only in the query form. The NOTES column is used to indicate this.

As an example, suppose the existence of the following table.

## 1999 SCPI Syntax & Style

KEYWORD	PARAMETER FORM	COMMENTS
:FREQuency		
[:CW]	<numeric_value>	
:AUTO	<Boolean>	
:CENTer	<numeric_value>	
:SPAN	<numeric_value>	

To set the frequency value for a Continuous Wave signal, the following command would be sent:

```
FREQuency:CW 2000000
```

5

Alternatively, since the CW node is optional the following command is also valid:

```
FREQuency 2000000
```

To set the value of AUTO, either of the following commands are allowed, again due to the optional CW node:

```
FREQuency:CW:AUTO OFF
```

```
FREQuency:AUTO OFF
```

The query form of the AUTO command is either of the following:

```
FREQuency:CW:AUTO?
```

```
FREQuency:AUTO?
```

In the “Command Descriptions,” the detailed descriptions of the individual commands are provided immediately after the command table, in the order in which the commands appear in the table, reading from top to bottom.

### 5.2

### Interpreting Syntax Flow Diagrams

Syntax flow diagrams are used extensively throughout the SCPI document and *IEEE 488.2* to provide pictorial representations of syntax. These representations are also known as **railroad diagrams**.

The flow through a diagram is given by lines and arrows. These link together the various objects used to form a command. Objects exist in the diagram as either a circle or a box. Circles indicate literal characters and the boxes represent a syntactic structure that is defined elsewhere in this document or in *IEEE 488.2*. Flow through the diagrams generally proceeds left-to-right. Diagrams are entered on the left, and the syntax is satisfied when the diagram is exited on the right. When an element or group of elements in the diagram is repeatable, a reverse, right-to-left path will be shown around and above the element(s), and is marked with a left-facing arrow. When an element or group of elements in the diagram are optional, a left-to-right bypass path will be shown around and below the element(s). A branch in the path indicates a choice of elements.

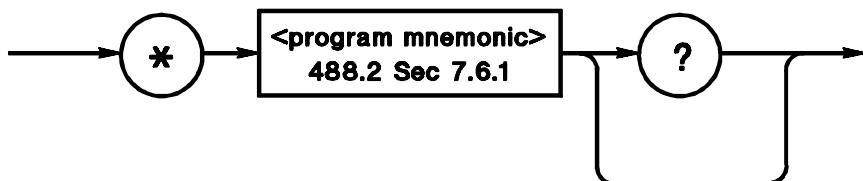
Lowercase and uppercase letters are considered equivalent; however, no attempt has been made to specify both alternatives in every instance. Whitespace characters (as defined in *IEEE 488.2*, section 7.4.1.2) are optional in many places, and no attempt has been made to specify “optional whitespace” everywhere it may exist.

## 6 Program Headers

Program headers are keywords that identify the command. The program headers follow the syntax described in section 7.6 of *IEEE 488.2*. Instruments shall accept both upper and lowercase characters without distinguishing between the cases. Program headers consist of two distinct types, common command headers and instrument-control headers.

### 6.1 Common Command and Query Headers

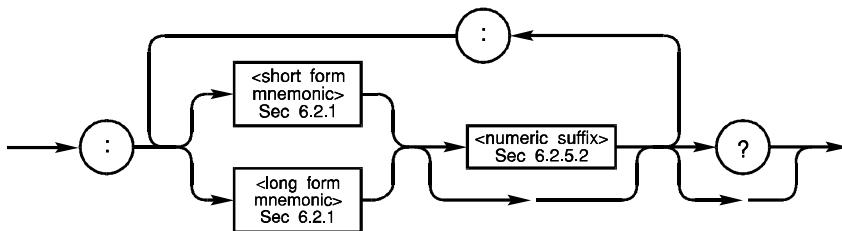
The common command and query program header syntax is specified in *IEEE 488.2* for use with the *IEEE 488.2*-defined common commands and queries. Their syntax is:



6

### 6.2 Instrument-Control Headers

Instrument-control headers are used for all other instrument commands, typically those related to source and measurement control. The syntax for instrument-control headers is:



The definition of the effect of the colons in an instrument-control header is covered later in “Traversal of the Header Tree.”

#### 6.2.1 Mnemonic Generation Rules

Each instrument-control header or keyword has both a long and a short form. A SCPI instrument shall accept only the exact short and the exact long forms. Sending a header that is not the short form, nor the complete long form to a SCPI instrument shall cause it to generate an error. *IEEE 488.2* limits the length of a header to 12 characters, including any numeric suffix that may appear. The long form header is either a single word or an abbreviation of a phrase. The short form header is an abbreviation of the long form header. In order to maintain a consistent set of mnemonics, SCPI defines the rules to generate a mnemonic.

The long form mnemonic is generated from either a single word or a phrase. If a single word is used, then that word becomes the mnemonic. If a phrase is used, then the mnemonic is the

## 1999 SCPI Syntax & Style

first letter of each word and the entire last word. For example the phrase “relative velocity” would generate RVELOCITY as the long form command header.

The short form mnemonic is usually the first four characters of the long form command header. The exception to this is when the long form consists of more than four characters and the fourth character is a vowel. In such cases, the vowel is dropped and the short form becomes the first three characters of the long form. For example, the short form of FREE is FREE, however, the short form of SWEEP is SWE. Note that elsewhere in this document a special notation is employed to differentiate the short form keyword from the longform of the same keyword. The long form of the keyword is shown, with the short form portion shown in uppercase characters, and the rest of the keyword is shown in lowercase characters. Thus Relative VELOCITY keyword would be shown as RVELocY.

6

The short form generation rules imply that phrases such as “Jump Start” and “Jump Stop” are not allowed. Although the long forms JSTART and JSTOP are unique, the short form is the same in both cases, “JST.” This can be overcome by changing the phrases to “Jump Begin” and “Jump End”, thus creating unique long and short forms. Alternatively, the mnemonic JUMP can become an additional level in the tree, allowing STARt and STOP to become their own mnemonics, giving JUMP:STARt and JUMP:STOP.

The mnemonic generation rules allow keywords such as “TIME” and “TImEr” at the same tree level, since these are unique in both forms. This is not recommended since an unfamiliar user may select the wrong function unintentionally. If the two functions have the same number and type of parameters, the instrument may not signal an error, further exacerbating the situation.

All instrument command headers are allowed a numeric suffix to differentiate multiple instances of the same structure, such as multi-channel instruments. The numeric suffix is applied to both the long and short forms. For example, TRIG1 is the short form of TRIGger1. A numeric suffix of 1 is implied on all instrument command headers that do not explicitly define a suffix; thus, TRIG is equivalent to TRIG1.

As a general rule, a mnemonic should contain no digits. Ending a short or long form mnemonic in a digit creates an ambiguity in separating the mnemonic from any added numeric suffix.

### 6.2.2

#### Building the Command Tree

SCPI commands are based on a hierarchical structure. This allows the same instrument-control header (keyword) to be used several times for different purposes, providing that the mnemonic occurs in a unique position in the hierarchy. That is, the mnemonic does not collide with any other mnemonics at the same level, or one level different where a default node exists.

Using hierarchical commands should eliminate the need for most multiword mnemonics. The use of multiword mnemonics is discouraged and they should only be used when:

- A). The use of the hierarchical scheme would add several additional layers to the tree, and

- B). No further growth at the level of those potentially intermediate nodes is anticipated.

These style guidelines should act as a starting point for creating new commands.

1. The lowest nodes should have the broadest base possible. For example, the center frequency command should be FREQuency:CENTER rather than CENTER:FREQuency. Since center is an adjective to the noun frequency, the tree will build with fewer duplicate nodes if FREQuency:CENTER is chosen.
2. If a function is broad-based and thought of as a separate subsystem, make it a separate subsystem even if it could properly be classified under another subsystem. For example, bandwidth is a separate subsystem, even though it could be classified under frequency. Bandwidth is generally thought of as a separate subsystem and not classified with other frequency commands by customers.
3. Keep the tree as shallow as possible, usually three levels or less. If the tree for a given implementation is considerably deeper than it is wide, something may be wrong with the command structure.
4. Some nodes may be made optional. When choosing to define a node as the default at a particular level, be sure that it is the most common choice at that level. Also, check carefully for conflicting keywords, since all nodes below a default are effectively promoted by one level when the default node is assumed.
5. Some root keywords are optional, depending on the instrument capabilities. In an instrument which is primarily a source, the root mnemonic [SOURce] is optional. [SENSe] is optional for instruments which are primarily sensors, and [ROUTE] is optional for scanners. This is at the discretion of the device designer. For horizontal compatibility purposes, the instrument should accept this node if it is sent as part of a program message.
6. A complete tree path shall be unique within SCPI. Thus, if a path is assigned a particular function in any instrument, it must perform the same function in all instruments which implement that path.
7. SCPI also discourages the use of different keywords performing the same function. This problem is more difficult because of industry-standard terminology. A good example is setting the output level on a source. Microwave sources call this power level, RF sources use amplitude level, and power supplies use voltage level, yet the function is the same in all three. One of the tools available is to provide aliases for the function in order to provide both sets of terminology. Whenever possible, SCPI shall implement as aliases those instances where duplicate names are necessary. Where aliases are used, one term shall be defined which is the primary keyword. This keyword shall appear in all instruments.
8. The tree structure shall follow the general form of the instrument model described in the “Command Reference.”
9. Implementations shall use the language constructs described in the “Command Reference,” if a construct exists for the desired feature. If the feature does not exist, the

## 1999 SCPI Syntax & Style

designer is responsible for following the life cycle described in an earlier chapter in order to integrate the capability into SCPI.

10. The value or syntax of a parameter shall not change the type or number of other parameters in that command. If such a condition is encountered, additional nodes should be added to the tree.

For example, SCPI shall not include the structures such as:

```
:SYSTem
  :BEEPer    1,<frequency>
  :BEEPer    2,<time>
```

Rather, SCPI uses a structure such as:

```
:SYSTem
  :BEEPer
    :FREQuency <numeric_value>
    :TIME      <numeric_value>
```

11. In general, parameters should only appear at the leaf nodes of the tree. In the undesirable example below, the command to set a single frequency is placed at a different level to the commands that are used for setting a range (span) of frequencies. Conceptually equivalent level commands are being placed at different levels, leading to possible misconceptions in use. “Optional” nodes should be created as required to accomplish the same task. For example, suppose an instrument superficially wants the following set of commands:

```
FREQuency    <numeric_value>
  :CENTer     <numeric_value>
  :SPAN       <Boolean>
```

The SCPI implementation shall overcome this by creating a node under frequency that represents a command to set a single frequency, such as CW (Continuous Wave). Causing all the command parameters to be associated with leaf nodes. To retain simple operation as in the case of setting a single frequency, the CW node can be made optional, providing it does not create a keyword conflict. Thus our example becomes:

```
FREQuency
  [:CW]      <numeric_value>
  :CENTer    <numeric_value>
  :SPAN      <Boolean>
```

Exceptions to this guideline are the AUTO and STEP constructs defined in this document. The AUTO and STEP commands are formed from the leaf node for the value. Other exceptions may be defined. In these cases, the exception shall be noted in the command structure of the language document.

Nodes should not be added to a command tree simply to make the tree have a uniform depth. If the only usable names for a terminal seem not to have any meaning (for instance, VALue), the addition may not be helpful and the tree should be implemented unbalanced. An example

of this is the SOURce:FREQuency:SPAN node, which has the commands HOLD, LINK, and FULL beneath it.

12. If a particular device function is not alterable or does not exist, then the associated SCPI command need not be implemented unless the value of the function is different than the defined \*RST value for that function. If the \*RST value is device-dependent, then the function need not be implemented. For example:

- A. A source that has no amplitude modulation does NOT have to implement AM:STATe OFF because AM:STATe OFF is specified at \*RST.
- B. A device that only sweeps in frequency (and has no CW function) must implement FREQ:MODE SWEep because FREQ:MODE is specified as CW at \*RST. However, it is permissible to allow only SWEep as a parameter to FREQ:MODE.
- C. A device that produces a single, nonalterable CW frequency of 100 MHZ need not accept FREQ[:CW] 100 MHZ since the \*RST value of FREQ:CW is device-dependent.

The intent of this rule is to allow common features with varying complexities to be accessed. This rule specifies the minimum conditions of including a node. However, designers are encouraged to include additional degenerate nodes which are important to their market. In example A above, a designer may include AM:STATe OFF as a command, and is encouraged to do so if the instrument is being targeted into markets where AM modulation is a typical feature of instruments in this market.

13. A new command may not be added if there is already a command to control the same functionality. This rule insures two instruments will not be needlessly incompatible when designers choose arbitrarily among multiple commands available to control a particular function. (Notice that SCPI assures compatibility when aliased commands are implemented by making one path required). A new command may not be added to the standard unless it can be shown to control functions not controlled by any existing commands. The following guidelines may help in determining whether a command controls new functionality or represents illegal duplication, and guide selection of one command over another.

*Does the setting of the new command affect the measurement result returned?*

Signal preconditioning commands (like ATTenuator, FILTer, GAIN) appear in the INPut subsystem and in SENSe. These commands are not aliases; The setting of INPut:ATTenuator changes the reported signal level, while SENSe:ATTenuator would be compensated for so the reported signal level would not be affected. Since these commands affect the returned value differently, they control different functions and are not duplicate controls.

*Does the new command control a function with a different purpose than any existing command?*

Numerical post-processing commands (like WINDOW, AVERage, SMOothing) appear in the sensor and also in CALC. The function properly belongs in the sensor when its purpose is to

## 1999 SCPI Syntax & Style

enhance accuracy or correct for artifacts introduced by the sensor. The function belongs in CALCulate when its purpose is to change the presentation of the data or provide analytical tools not directly related to measurement.

*Does the command control measurement hardware?*

Some functions (like SMOothing) may be performed directly on the signal in hardware or in software after A/D conversion. While this rule can be difficult to apply, the fact that a command controls measurement hardware often indicates that it belongs in the sensor rather than CALCulate.

*Does it make sense if the same operation may be done twice?*

Another way to tell that a command controls new functionality is if it makes sense for the controlled function to appear twice in a single instrument. The AVERage subsystem is a good example; it is not unreasonable to build an instrument that averages several readings to improve the accuracy of the measurement, and also provides averaging as a statistical function in the CALCulate subsystem.

14. In general, SCPI commands which are only events without a query form do not have any parameters. Including a parameter might mislead a user into believing the value could be queried as most commands in SCPI have a query form, see 6.2.3.

An exception to this guideline is when the action is performed on a named object within the instrument and these objects cannot be enumerated. For example, <file\_name>, <trace\_name>, and <data\_handle> are objects whose values cannot be entirely listed within SCPI. The user can create any number of <file\_names> or <trace\_names>. The signal routing concepts in SCPI allow a large number of possible <data\_handles>.

### 6.2.3 Queries

All commands, unless otherwise noted, have an additional query form. As defined in IEEE 488.2, a query is a command header with a question mark symbol appended (for example, FREQ:CENT?). When a query form of a command is received, the current setting associated with the command is placed in the output buffer. Query responses do not include the command header. Execution of a query form has no side effects. It shall not cause any settings or couplings within the instrument to change. An exception is in MEASurement subsystem, where instrument settings may change as a result of the measurement.

Both the command and query forms shall be implemented unless a comment or note indicating otherwise appears in the keyword summary. Even when a command accepts only a single selection, both the command and query forms shall still be implemented. A note of the form “[event; no query]” or “[no query]” indicates that only the command form and not the query form shall be implemented. A note of the form “[query only]” indicates that only the query form and not the command form shall be implemented.

When character data is used for a parameter, both longform and shortform values shall be recognized. If the command has a query form with character response data, the shortform value is always returned.

When numeric parameters are queried, the result shall be returned in fundamental units unless otherwise specified or requested. When several different units may be considered fundamental (for example, dBuV or dBm), the units of the returned result shall be documented in the query response description for the command. A query may have a related :UNITs node to change the default units of the accepted or returned value, where this exists the coupling shall be clearly indicated in the command description.

## 6.2.4

### Traversal of the Header Tree

*IEEE 488.2* allows the designer some discretion on how compound headers are handled within the rules of section 7.6.1.5. SCPI imposes additional requirements regarding how a compound command is parsed.

Multiple <PROGRAM MESSAGE UNIT> elements may be sent in a <PROGRAM MESSAGE>. The first command is always referenced to the root node. Subsequent commands, however, are referenced to the same tree level as the previous command in a message unit. SCPI parsers shall follow the example presented in *IEEE 488.2*, section A.1.1. A SCPI parser shall NOT implement the enhanced tree walking implementation described in section A.1.2.

For example, if a hypothetical instrument had the command tree:

```

FREQuency
  :STARt      <numeric_value>
  :STOP       <numeric_value>
  :SLEW       <numeric_value>
    :AUTO      <Boolean> | ONCE
  :BANDwidth  <numeric_value>
POWER
  :STARt      <numeric_value>
  :STOP       <numeric_value>
  BAND        A|B|C|D

```

Then the following commands shall behave as described:

- FREQ:STAR 3 MHZ;STOP 5 MHZ<nl> shall set the start frequency to 3 MHz and the stop frequency to 5 MHz.
- FREQ:STAR 3 MHZ;:FREQ:STOP 5 MHZ<nl> shall set the start frequency to 3 MHz and the stop frequency to 5 MHz.
- FREQ:STAR 3 MHZ;POW:STOP 5 DBM<nl> may set the start frequency to 3 MHz and shall generate an error because POW is not a node at the current parser level.
- FREQ:STAR 3 MHZ;SLEW:AUTO ON<nl> shall set the start frequency to 3 MHZ and couple the frequency slew rate.
- FREQ:SLEW:AUTO ON;STOP 5 MHZ<nl> may couple the slew rate and shall generate a command error since the coupling command is not at the same level as the stop command.

## 1999 SCPI Syntax & Style

- FREQ:SLEW 3 MHZ/S;AUTO ON<nl> may set the frequency slew rate to 3 MHZ/S and shall generate a command error because :AUTO is not at the same level.
- FREQ:START 3 MHZ;BAND 1 MHZ<nl> shall set the start frequency to 3 MHz and the bandwidth to 1 MHz.
- FREQ:START 3 MHz;:BAND A<nl> shall set the start frequency to 3 MHz and select band A.
- FREQ:SLEW:AUTO ON;3 MHZ/S<nl> may couple the frequency slew rate and shall generate a syntax error, because IEEE 488.2 specifies that headers must be sent with each command.

6

Default nodes in the tree shall not alter the header path of the parser in order to follow the IEEE 488.2 rules for compound headers. IEEE 488.2, section 7.6.1.5 makes no mention of default nodes, so adding something to the header path is not allowed. For example, if a hypothetical instrument had the command tree:

```
DISPlay
[:STATE]      <Boolean>
:DATA         <string>
```

Then the following situations would have these results:

- DISP:STAT ON;DATA “Hello, world!”<nl> shall turn the display on and display “Hello, world!”.
- DISP ON;DATA “Hello, world!”<nl> may turn the display on and shall flag an error (assuming that DATA is not a root mnemonic).

### 6.2.5 Multiple Capabilities and Numeric Keyword Suffixes

Many instruments provide multiple capabilities, by duplicating internal functional blocks. One example is an instrument that can display more than one measurement at a time. Another example is an instrument that can make a particular measurement on one of several input channels. SCPI provides a mechanism to address this duplicate functionality while retaining the same hierarchy for each duplication.

#### 6.2.5.1 Single Instrument with Many Electrical Ports

When only electrical ports (front panel terminals) are involved, the <channel\_list> notation is sufficient to describe the measurement connection. An instrument which needs to select one or more of several input terminals can be logically modeled as a simple instrument with one set of inputs connected to a scanner (signal multiplexor).

#### 6.2.5.2 Multiple Identical Capabilities

When an instrument has several independent measurement channels (or other capabilities, such as displays or sources), the selection of which to use is designated by a numeric suffix attached to a program mnemonic. If the suffix is absent, the command is treated as being designated for capability number one. In this way, channel one of a multiple channel instrument can accept the same commands as a single channel instrument to allow for

upward compatibility. Further, adding multiple channels or capabilities to an instrument class where multiple channels were not anticipated becomes possible.

The node that contains the numeric suffix is the one where the multiple capability occurs. This may be at any level of the tree: root, intermediate or terminal node. As an example, OUTP5:MOD3:FM2 would specify the second FM signal component of the third modulation signal on the fifth output channel.

Devices which implement multiple channels/capabilities shall recognize nodes which do not have numeric suffixes and treat them as channel/capability 1. Instruments with a single channel/capability are not required to accept a numeric suffix. This allows for upward compatibility.

### 6.2.5.3

#### Logical Instruments

A complex instrument such as a VXI card cage can be logically modeled as separate instruments, each with its own (secondary) bus address. For example, common functions for all instruments can be located at secondary address 00, and otherwise each instrument responds individually to *IEEE 488.2* commands. This requires that each logical instrument (card) implement its own status model and I/O buffers.

## **1999 SCPI Syntax & Style**

## 7 Parameters

SCPI uses the parameter forms described in *IEEE 488.2*, section 7.7, with some additional restrictions. Also note that SCPI specifies the values for all commands upon receipt of \*RST. In some cases, these reset values are device-dependent, but all measurement parameters must be set to some deterministic value for that particular instrument, which in turn must be documented in the instrument manual.

### 7.1 Character Program Data

Where possible, the same truncation rules are used for parameters as for headers. However, in many cases industry standards take precedence. For example, IDC is a better choice for DC current than anything SCPI rules would define, simply because it is an existing standard with wide acceptance. In addition, several character program data forms are predefined as extensions of <DECIMAL NUMERIC PROGRAM DATA>.

### 7.2 Decimal Numeric Program Data

Numeric elements shall be used only for representing numeric quantities. They shall not be used for selecting functions on a “One of N” position switch.

Implementations shall accept numeric data as described in *IEEE 488.2*, section 7.7.2.4. However, any number which exceeds +9.9 E 37 shall generate an execution error (-222, “Data out of range”). Numbers shall be rounded to the closest “correct” value that the instrument accepts without error. This document does not define the value a parameter is set to if an out-of-range value is received.

#### 7.2.1 <numeric\_value> Definition

The decimal numeric element is abbreviated as <numeric\_value> throughout this document. This is different from the <NRF> described in section 7.7.2.1 of *IEEE 488.2* in several ways.

Several forms of <CHARACTER PROGRAM DATA> are defined as special forms of numbers. These are: MINimum, MAXimum, DEFault, UP, DOWN, Not A Number (NAN), INFinity, and Negative INFinity (NINF). Individual commands are required to accept MIN and MAX. DEFault, UP, DOWN, NAN, INFinity, and NINFinity may be implemented at the discretion of the designer, in which case it shall be noted in the instrument documentation. Where an optional form is accepted, it will be noted in the command description.

A <non-decimal numeric> (*IEEE 488.2*, section 7.7.4), a <numeric\_expression>, or a <label> is part of <numeric\_value> if an instrument implements these features.

#### 7.2.1.1 DEFault

The special <numeric\_value> parameter DEFault may be provided to allow the instrument to select a value for a parameter. When DEFault is sent, the instrument shall select a value which is deemed to be convenient to the customer. The setting may be device-dependent, or it may be defined as part of this standard. The use of DEFault is optional on a command-by-command basis. Individual commands shall document where DEFault is required.

For example, to use the SYST:TIME command to set a device’s clock ahead one hour (Daylight Savings Time), a program might send SYST:TIME UP,DEF,DEF<nl>.

## 1999 SCPI Syntax & Style

Another example is found in the MEASure commands. The syntax of the command which measures DC voltage is:

```
MEASure:VOLTage:DC [<expected value>[,<resolution>]]
```

The MEASure command specifies that parameters are defaulted from the right and that any parameter may be defaulted by using DEFault in place of the parameter. The following command would measure DC voltage, defaulting the range to an instrument dependent value (possibly autorange), but specifying the resolution at 0.001 Volt:

```
MEASure:VOLTage:DC DEFault,0.001V
```

### 7.2.1.2 MINimum|MAXimum

The special form numeric parameters MINimum and MAXimum shall be provided which assume the limit values for the parameter. The maximum and minimum shall be queryable by sending <header>? MAXimum|MINimum. The MAXimum value refers to the largest value that the function can currently be set to, and MINimum refers to the value closest to negative infinity that the function can be currently set to.

Some commands have multiple parameters. The query form of these commands returns a list of values representing the current value of each of the parameters, in the order of their normal occurrence in a program message. If a MINimum/MAXimum query of multiple parameters is allowed, the keywords MINimum and MAXimum must occur as many times in the query as there are parameters. MINimum requests that the instrument return the legal value which is closest to negative infinity for the parameter; MAXimum requests the legal value which is closest to positive infinity.

For example, suppose an instrument implements the SYST:TIME command, which requires three parameters, and allows MIN/MAX queries on this command. The following queries shall have these results:

- SYST:TIME?<nl> shall return the current setting of the time-of-day clock in the instrument.
- SYST:TIME? MAX,MAX,MAX<nl> could return 23,59,59.
- SYST:TIME? MAX<nl> shall set an error (-109, “Missing parameter”), since three parameters are required and only one was sent.

### 7.2.1.3 UP/DOWN

An instrument may optionally allow the use of steps for some or all of its numeric entry. If steps are used, the keywords UP and DOWN shall be used as numeric parameters which perform stepping. Steps may be adjustable through the step node for each individual parameter.

The instrument may step a parameter when UP or DOWN is received in lieu of a numeric value. This capability is optional. However, if the capability is implemented, the device shall include a node for each command which accepts step parameters. This node will specify the step. The step may either be a fixed linear size or a logarithmic number representing number of decades/step.

**7.2.1.3.1        STEP Subsystem Command Syntax**

The form of the step subsystem is as follows:

```
:STEP
[:INCrement]      <numeric_value>
:PDECade          <numeric_value>
:MODE              LINear|LOGarithmic|L125|L13
:AUTO              <Boolean>|ONCE
```

**7.2.1.3.2        [:INCrement] <numeric\_value>**

This command controls the step size in absolute units when STEP:MODE LINear is selected.

At \*RST, this value is device-dependent.

**7.2.1.3.3        :PDECade <numeric\_value>**

This command controls the number of steps per decade when STEP:MODE LOGarithmic is selected.

At \*RST, this value is device-dependent.

**7.2.1.3.4        :MODE LINear|LOGarithmic|L125|L13**

This command controls the linearity of steps. The various parameters have the following meanings:

- LINear: Steps are a fixed value added to or subtracted from the current value of the parameter.
- LOGarithmic: Steps are placed at points spaced logarithmically. If the current value of the function is  $x$ , the value of the function after executing an up shall be determined by the following formula:

$$10^{\left\lceil \frac{\lceil \log_{10}(x) * \text{STEP:PDECade} \rceil + 1}{\text{STEP:PDECade}} \right\rceil}$$

and the value of the function after executing a DOWN shall be determined by:

$$10^{\left\lfloor \frac{\lceil \log_{10}(x) * \text{STEP:PDECade} \rceil - 1}{\text{STEP:PDECade}} \right\rfloor}$$

where  $\lfloor$  is the symbol for *floor*, and  $\lceil$  is the symbol for *ceiling*.

- L125 — Steps are determined by the sequence  
... 0.1,0.2,0.5,1,2,5,10,20,50,100 ...

Executing UP will set the value of the function to the next higher value in the sequence.  
Executing DOWN will set the value of the function to the next lower value in the sequence.

## 1999 SCPI Syntax & Style

- L13 — Steps are determined by the sequence  
... 0.1,0.3,1,3,10,30,100 ...

Executing UP will set the value of the function to the next higher value in the sequence.  
Executing DOWN will set the value of the function to the next lower value in the sequence.

At \*RST, this value is device-dependent.

### 7.2.1.3.5 :AUTO <Boolean>|ONCE

The step mode and increment is coupled to other instrument settings and physical inputs when AUTO is ON.

### 7.2.1.3.6 STEP Subsystem Examples

```
FREQ:CENT:STEP 5 MHZ<n1>
FREQ:CENT UP<n1>
```

sets the center frequency step to 5 MHz, and then increments the current value by 5 MHz.

```
BAND:RES 1MHZ<n1>
BAND:RES:STEP:MODE LOG;PDEC 3<n1>
BAND:RES UP<n1>
```

The above sequence would specify a logarithmic step, set three steps/decade, and then increment the resolution bandwidth by 1/3 decade (logarithmic) to a final value of 2.154 MHz.

```
BAND:RES 1MHZ<n1>
BAND:RES:STEP:MODE L125<n1>
BAND:RES UP<n1>
```

The above sequence would specify a 125 logarithmic sequence, and then increment the resolution bandwidth to the next step in the sequence to a final value of 2.0 MHz.

### 7.2.1.4 INFINITY and Negative INFINITY (NINF)

A special case is made for infinity. Positive infinity is represented as 9.9 E 37. Negative infinity is -9.9 E 37. Instruments shall accept numbers within this range as valid numeric data. They shall also limit response data to this range. If a valid instrument setting is infinity or negative infinity, the values 9.9 E 37 and -9.9 E 37 shall be accepted and returned to represent positive and negative infinity respectively, and on input, the implementation shall accept INFinity as an alias for positive infinity, and NINFinity as an alias for negative infinity.

The numeric values for positive and negative infinity were chosen so that they fit into a 32-bit IEEE 754 floating point number. This allows easy implementation using standard tools in all popular languages and operating systems. Note that this does not limit the numeric resolution. These values are larger than any quantities used or anticipated in instruments.

### 7.2.1.5 Not A Number (NAN)

Not a number is represented as 9.91 E 37. Not a number is defined in IEEE 754. Typical applications are dividing zero by zero or subtracting infinity from infinity. Not a number is

also used to represent missing data. A typical application is when a portion of a trace has not been acquired yet. On input, devices shall accept NAN as an alias for not a number.

The numeric value for NAN was chosen so that it can be represented as a 32-bit *IEEE 754* floating point number. This allows easy implementation using standard tools in all popular languages and operating systems. Note that this does not limit the numeric resolution. This value is larger than any quantities used or anticipated in instruments.

## 7.2.2

### **Unit Suffixes**

The <DECIMAL NUMERIC PROGRAM DATA> may be followed by an optional suffix. All parameters which have associated units shall accept a suffix. Only suffixes appropriate for the command should be accepted. All suffixes must include the associated unit. See *IEEE 488.2*, 7.7.3, <SUFFIX PROGRAM DATA>, for a complete discussion of this topic.

Suffixes must accept multipliers except in cases where the multiplier is illogical, such as dBm or PCT. Table 7-2 in *IEEE 488.2* lists the allowed multipliers. If any of the multipliers are allowed with a suffix, then all the multipliers shall be interpreted properly. Compound suffixes are allowed.

If a suffix is included, the suffix and associated multiplier, if implemented, are applied to the parameter. If the suffix is omitted, default units are used. The default unit is determined either from the :UNIT subsystem as described in 7.5 of Syntax & Style and in Chapter 23 of the Command Reference, or it is the fundamental unit associated with the parameter.

Fundamental units are either described in the appropriate subsystem or in the one shown in *IEEE 488.2*, table 7-1.

## 7.3

### **Boolean Program Data**

The form <Boolean> is used throughout this document as a shorthand for the form ON|OFF|<NRf>. Boolean parameters have a value of 0 or 1 and are unitless.

On input, an <NRf> is rounded to an integer. A nonzero result is interpreted as 1. This algorithm is the same as the one described in *IEEE 488.2*, section 10.25.3.

The <CHARACTER PROGRAM DATA> elements ON and OFF shall be accepted on input for increased readability. ON corresponds to 1 and OFF corresponds to 0.

Queries shall return 1 or 0, never ON or OFF.

## 7.4

### **Coupling of Functions**

There are two forms of coupling: functional and value. A coupling occurs when sending a command changes the value associated with another command. In general, functional couplings are discouraged except in the MEASure subsystem. Value couplings are allowed if the coupling equations are well specified.

Some functions may have the ability to be decoupled. If functions can be decoupled, the ability to recouple them shall also exist. If a function cannot be decoupled, then a node to set its value shall not exist, or shall be implemented as query-only. If a node is settable under some conditions, then the node may exist.

## 1999 SCPI Syntax & Style

Another level in the tree is used to control coupling. The keyword AUTO shall be used to control coupling. For example, the command to couple resolution bandwidth would be:

```
BAND:RES:AUTO ON
```

Setting a value explicitly for a function shall cause the function to be decoupled (:AUTO OFF), if decoupling is possible. Otherwise an error shall be generated.

Selecting :AUTO ONCE shall cause the function to select the most appropriate value for current conditions (range, signal level, etc), and then be decoupled. A subsequent query of an :AUTO? will always return 0, since ONCE is an event which leaves the function in AUTO OFF mode.

### 7.4.1 Functional Coupling

**Functional coupling** occurs when a command causes side effects in the basic operating mode of the device. An example is a command which sets the start frequency and also starts a sweep.

If a command which invokes a functional coupling is necessary, the instrument shall also implement primitive commands which do not have functional couplings. The instrument may then implement a complex command which is described as a combination of these primitive commands. This complex command may contain functional couplings, and must document the sequence of primitive commands used in its programming documentation.

For example, if a source needs a command which sets the AM modulation level, and which has the side effect of turning the modulation ON, the following could be implemented:

Primitives:

AM

```
:STATE      <Boolean>  
[:DEPTh]    <numeric_value>
```

Complex command added:

AM

```
:SDEPth     <numeric_value>
```

Where AM:SDEPth is defined in the manual as:

AM

```
:DEPth      <numeric_value>;  
:STATe     ON
```

### 7.4.2 Value Coupling

The other form of coupling is called **value coupling**, which is allowed in SCPI. Values are coupled when a command changes the value of other numeric settings in the instrument. The coupling relation would follow device-dependent equations. For example, bandwidth in a receiver may be coupled to the frequency span. The individual command descriptions define these coupling equations.

Groups of functions may be coupled in complex ways which affect the values for each other. For example START, STOP, CENTER, and SPAN are all value-coupled. CENTER is the arithmetic average of START and STOP. SPAN is the difference between START and STOP. Changing any one affects the value of two others.

### 7.4.3 Automatic Coupling

A special type of coupling is the **automatic coupling**. This is when the device provides an algorithm to select the value of a parameter. This algorithm could be based upon physical inputs or other settings. When it is desirable to turn this algorithm on and off, the keyword AUTO is added. The AUTO command may accept parameters of ON|OFF|ONCE. ON enables the instrument algorithm. OFF disables the instrument algorithm. ONCE causes the algorithm to be employed once, changing the associated parameter, and then reverting to AUTO OFF.

### 7.5 Units of Measure and Suffixes

Most of the information on units and suffixes is specified in *IEEE 488.2*, section 7.7.3. However, further clarification is needed for units of power and amplitude, and for unitless quantities.

#### 7.5.1 Units of Amplitude and Power

- The fundamental unit of linear power is the Watt (W).
- The fundamental unit of linear amplitude is the Volt (V).
- The fundamental unit of logarithmic power is the dBm.
- The fundamental unit of logarithmic amplitude is the dBV.

Industry practices have caused other units to become widely used in various disciplines, particularly the units of uV, dBuV, dBuW and dBmV. Furthermore, many instruments cover a broad spectrum of applications where different units are considered the standard. Therefore, instruments which implement commands whose parameters might be expressed in several different amplitude or power units shall provide the following means of setting and measuring amplitude and power functions:

If the unit is not specified, the default unit shall be that specified with a command in the UNIT subsystem, or a UNIT command in the appropriate sub-tree. The \*RST values of settings in the UNIT subsystem shall be device-dependent. However, if a \*RST unit is different than the specified default unit, or is a logarithmic unit, then the command in the UNIT subsystem for that unit must be implemented.

If a suffix is sent with an <NRF>, the unit corresponding to that suffix shall override the default unit for that parameter.

If an instrument accepts units from one of the classifications described above, it must accept all suffixes of that classification. For example, if an instrument accepts Watts, it must also accept uW, mW, etc. This includes all suffix multipliers described for that unit in *IEEE 488.2*. However, it may, but is not required, to accept Volts as a unit. An instrument which accepts both requires a defined impedance for the conversion

$$(Power = \frac{Voltage^2}{Impedance})$$

either explicitly through an input or output impedance command or implicitly if the impedance of the instrument is known and fixed. A further restriction, is that if an instrument accepts logarithmic units, it shall also accept the comparable linear unit. For example, an instrument which accepts dBm must also accept Watts, MW, UW, etc. For example:

```
POW:UNIT DBUV<n1>
POW:LEV 50<n1>
```

shall set the amplitude level to 50 dBuV.

7

```
POW:UNIT DBM<n1>
POW:LEV 5V<n1>
```

would set the power level to 5 Volts.

Since some instruments are used in applications where both power and voltage units are appropriate, the system's impedance may be important to the user. This impedance is needed to do the conversion between power and voltage (and maybe even current). All instruments are encouraged to provide a node for at least querying this impedance. The node shall be called :IMPedance, and will generally appear under the INPut or OUTPut subsystem. Open circuits shall be expressed by setting IMPedance INFinity. It is therefore possible to express output power in Volts:

```
POW:UNIT V<n1>
OUTP:IMP 50 OHM<n1>POW:LEV 5<n1>
```

would set the power level to 5 Volts into a 50 Ohm load, or 0.5 Watt.

It is also sometimes necessary to determine which amplitude measurement is being made. Therefore, a measurement qualification may be appended to the suffix. The suffix appendices are described as:

- PK: Peak amplitude
- PP: Peak-to-peak amplitude
- RMS: RMS amplitude

If no suffix appendix is specified, the default is device-dependent. However, all instruments shall accept the suffix appendix for all types they support.

For example, a voltmeter which measures RMS voltage shall accept the suffixes V and VRMS. A function generator which outputs peak voltage would accept the suffixes V and VPK. An RF signal source might accept dBUV, dBUVRMS and dBUVPK.

### 7.5.2

#### Expressing Unitless Quantities

The ratio of two linear quantities shall default to a unitless ratio (A/B). For example, if the input power for a system is 5 Watts, and the output power is 20 Watts, the power gain =  $20W/5W = 4$ .

The difference between two logarithmic quantities shall be expressed in dB. For example if the input power of a system is 3 dBm and the output power is 5 dBm then the gain of the system is  $5 \text{ dBm} - 3 \text{ dBm} = 2 \text{ dB}$ .

Under exceptional conditions other units may be used for the ratio of two quantities. A valid exception would be if a standards-setting organization (Bell, CCITT, military, IEEE, IEC, etc.) specifies that a measurement shall be made using a special unit like PCT (percent).

## 1999 SCPI Syntax & Style

## 8 Expressions

### 8.1 Function

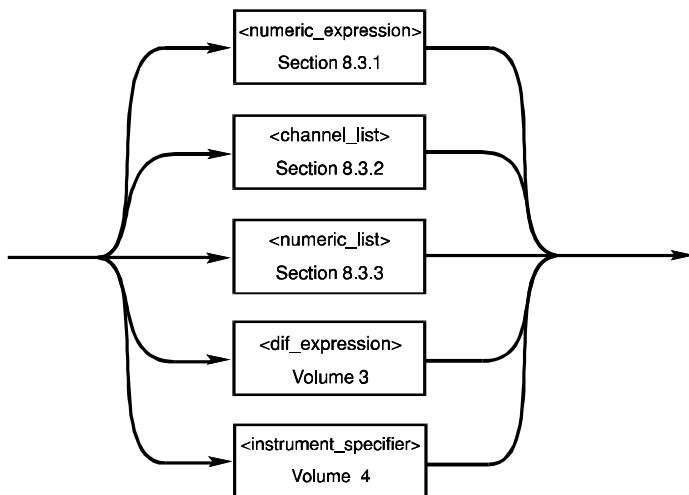
The <EXPRESSION PROGRAM DATA> described in *IEEE 488.2* is more tightly defined in this section. SCPI defines five types of expressions, numeric expressions, channel lists, numeric lists, Data Interchange Format (dif) expressions, and instrument specifier expressions. The dif expressions are defined in Volume 3, “Data Interchange Format.” Instrument specifiers are defined in Volume 4, “Instrument Class Applications.”

Other expression types may be added in future releases of this document, including but not limited to logical expressions, trigger expressions, and sequence expressions. Other uses of *IEEE 488.2* expressions are legal, and may be documented in the individual command descriptions.

### 8.2 Usage

The use of expressions is optional in SCPI. For expression types defined in this section, it is permissible to use any subset. Only expression types which are required by a particular command may be recognized as parameters of that command.

8



### 8.3 Syntax

Inside expressions, white space as defined in *IEEE 488.2*, section 7.4.1.2 is allowed between any lexical elements. For expressions defined in this section, a **lexical element** is defined as any operator, punctuation, or *IEEE 488.2* syntactic element. For a <dif\_expression>, spaces may appear around block names, block modifiers, keywords, or value types. See *Data Interchange Format*, Section 3.2. Note that explicit white space may be required to separate lexical elements. No semantic meaning should be attached to the case of alpha characters

used in operators. For example, the following are instances of the same operator:

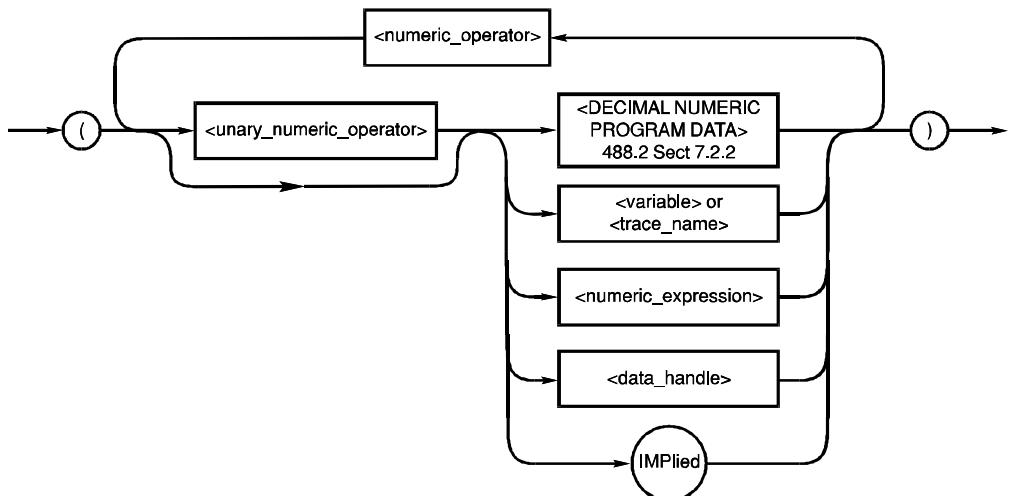
AND

and

And

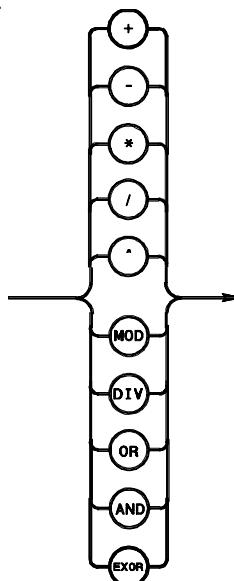
### 8.3.1 Numeric Expression

A numeric expression is a collection of terms which evaluates to a trace, number, array, or other data element.

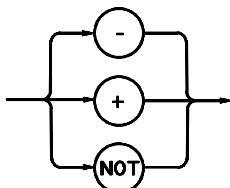


#### 8.3.1.1 Syntax

Where *<numeric\_operator>* is defined as:

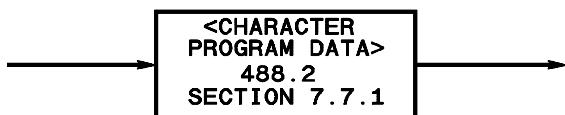


Where <unary\_numeric\_operator> is defined as:



Unary operators should not be used with signed numbers.

Where <variable> or <trace\_name> is defined as the diagram shows:



### 8.3.1.2 Precedence Rules

Expressions shall be evaluated according to the following precedence rules:

1. Enclosed by parentheses
2. Unary operators (+ and -)
3. ^(exponentiation)
4. \* (multiplication), / (division), MOD and DIV
5. + (addition) and - (subtraction)
6. NOT
7. AND
8. OR and EXOR
9. Left to right

### 8.3.1.3 Semantics

As stated in the syntax diagram, expressions may contain terms which are numbers, traces, variables, or expressions.

Elements in a numeric expression are promoted to the size and type of the most complex element, and the result of the expression is of that type. That is, an expression containing a <trace\_name> will be evaluated according to the rules for arithmetic on traces, and the result will be a trace. If an expression contains both a <trace\_name> and scalar data, a trace is created with all elements set to the value of the scalar data before arithmetic is performed. For example, if TREF is a trace\_name, the expression (TREF-3) results in a trace the same size as TREF, with each data element lower by three.

### 8.3.2 Channel Lists

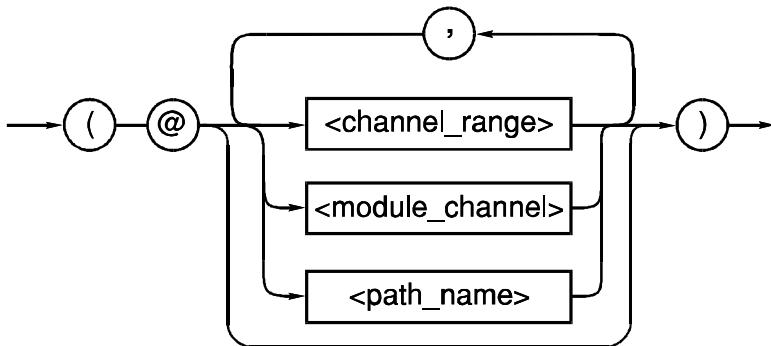
Channel lists are used to specify electrical ports on an instrument. They are typically used for signal routing either in a standalone switch box or in an instrument with multiple input channels. An instrument with multiple channels may or may not do any signal switching as

## 1999 SCPI Syntax & Style

the result of a channel list. Completely separate sensing channels are allowed, but may appear to the language the same as channels which are switched.

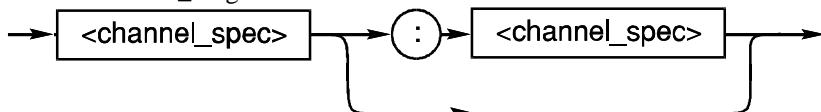
A channel list may appear in measurement, configuration, and other such commands. For example, MEAS:VOLT? (@1,3,4:6) says measure the voltage on channels 1, 3, and 4 through 6. Whether the measurements are performed simultaneously or in the order in the list is unspecified. Channel lists are also used by the ROUTe subsystem, “Command Reference,” 15.1.

The syntax for a <channel\_list> is:



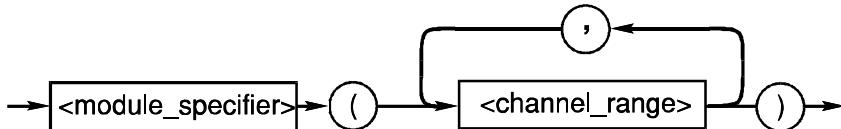
8

Where <channel\_range> is defined as:



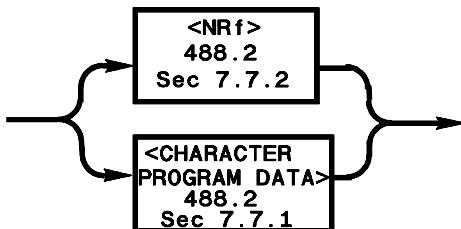
Where both instances of <channel\_spec> must contain the same number of dimensions.

Where <module\_channel> is defined as:

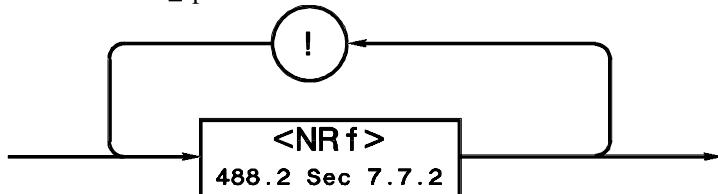


## 1999 SCPI Syntax & Style

Where `<channel_range>` indicates all the channels from the first number through the second number and where `<moduleSpecifier>` is defined as:



Where `<channelSpec>` is defined as:



8

The number of dimensions in a `<channelSpec>` is one greater than the number of occurrences of the ‘!’ character in the `<channelSpec>`

When a `<channelRange>` of dimension greater than one is scanned, the right-most index of the `<channelSpec>` varies most rapidly.

The multidimensional channel range is to be considered as a list of single dimension channel ranges. For example, (@!1!2:3!) means dimension 1 (row) ranges from 1 to 2 and dimension 2 (column) ranges from 1 to 3. If the size of the matrix in this example is 2X8, then the range (@!1!2:3!) does *not* mean: row 1, column 1 through 8 and subsequently row 2, column 1 through 3, but row 1, column 1 through 3 and next row 2 column 1 through 3. This method provides a functional compatibility between commands sent to matrix switches with different row lengths.

Channel lists are order sensitive. For example:

SCAN (@5:3) means close 5, 4, 3 in order.

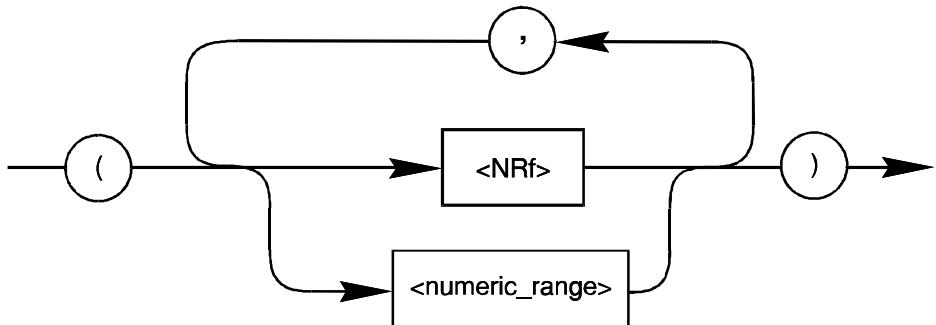
SCAN (@!1!2:3) means close 1!1, 1!2, 1!3, 2!1, 2!2, 2!3 in order.

SCAN (@!1!3:2!1) means close 1!3, 1!2, 1!1, 2!3, 2!2, 2!1 in order.

### 8.3.3 Numeric Lists

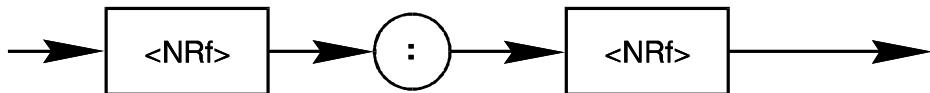
A numeric list is a an expression format for compactly expressing numbers and ranges of numbers in a single parameter.

The syntax for <numeric\_list> is:



8

Where <numeric\_range> is defined as:



The range is inclusive of the specified numbers.

## 9 Status Reporting

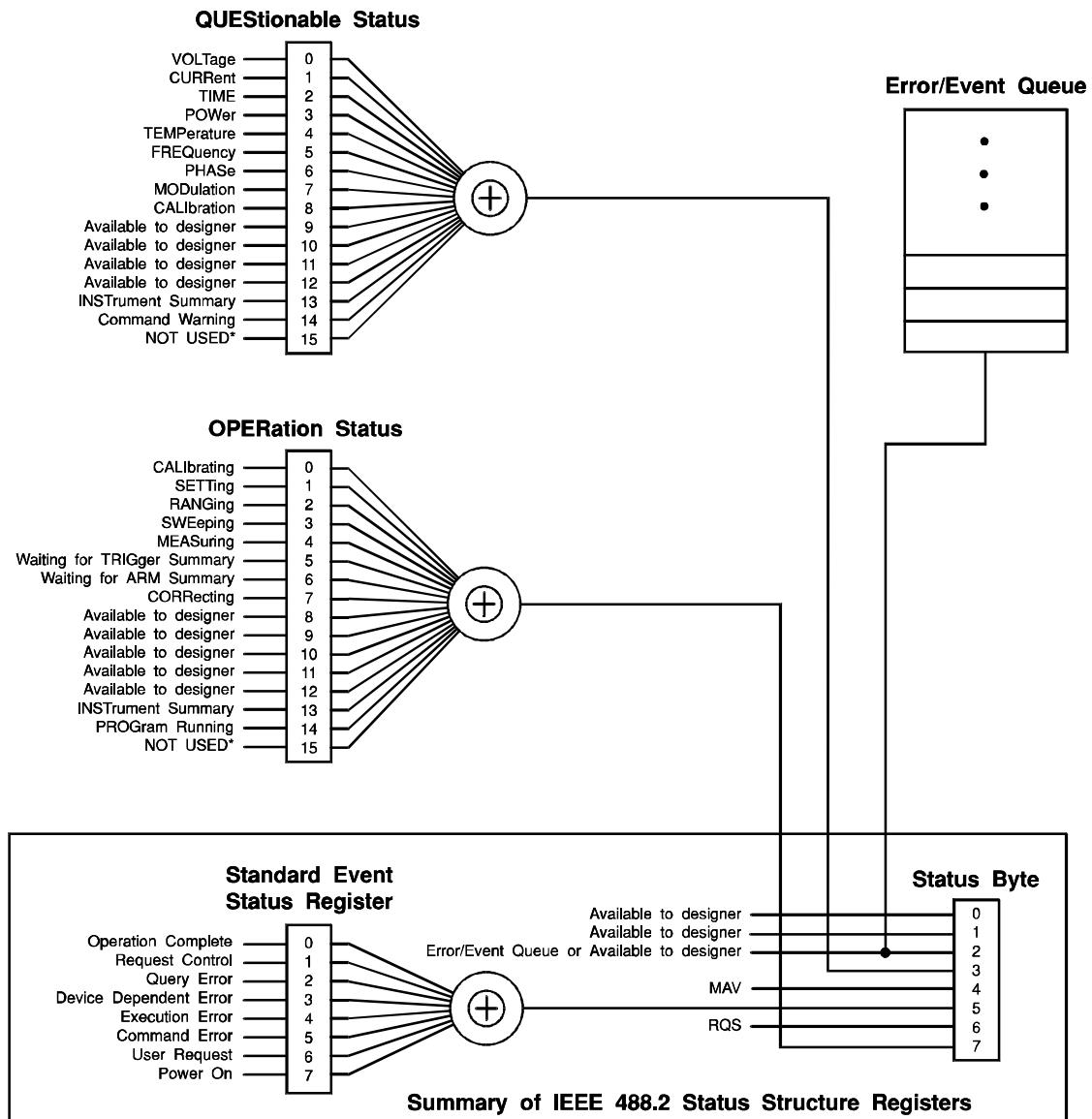
SCPI requires the status mechanism described in chapter 11 of *IEEE 488.2*, including full implementation of the Event Status register structure.

A SCPI device shall include the SCPI-defined OPERation status register and QUESTionable data/signal status register with the associated condition, event, and enable commands. In Figure 9-1 on the following page, “Minimum Status Reporting Structure Required by SCPI”, a pictorial representation is given that shows the core of the SCPI-required status reporting capability. Additional requirements are established for instruments that support either multiple logical INSTRuments or the expanded capability TRIGger model.

In general, a status register should fit into a 16-bit integer with the most-significant bit always zero (positive logic).

In the figures in this chapter, an elongated box is used to represent the “Status Data Structure-Register Model,” which is defined in *IEEE 488.2*. SCPI consists of condition, event, enable and optional transition registers. Two concentric circles with a cross through the center one indicates a logical summing.

## 1999 SCPI Syntax & Style



\* The use of Bit 15 is not allowed since some controllers may have difficulty reading a 16 bit unsigned integer. The value of this bit shall always be 0.

**Figure 9-1 Minimum Status Reporting Structure Required by SCPI**

## 9.1

### The Device-Dependent Register Model

The Device-Dependent Register model follows the structure described in *IEEE 488.2*, section 11.4.2. The transition filter described in figure 11-6 is actually a pair of programmable transition filters which are described below.

The commands which access these registers are described in the “SCPI Language Description.”

If the Error/Event Queue Summary is reported in the Status Model, then bit 2 of the Status Byte shall be used to reflect the Empty/Non-Empty status of the queue. A bit value of 1 indicates the queue is not empty. See the SYSTem:ERRor subsystem, Command Reference, 21.8.

## 9.2

### Transition Filters

Transition filters are described in *IEEE 488.2*, section 11.4.2.2.1. SCPI allows the use of programmable transition filters. When transition filters are used, SCPI requires the use of separate positive and negative transition filters. A positive transition filter allows an event to be reported when a condition changes from false to true. A negative filter allows an event to be reported when a condition changes from true to false. Setting both positive and negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disables event reporting.

The contents of transition filters are unchanged by \*CLS and \*RST.

## 9.3

### Operation Status Register

The OPERation status register contains conditions which are part of the instrument’s normal operation.

The definition of each of these bits (condition register) is as follows:

- **0-CALibrating** — The instrument is currently performing a calibration.
- **1-SETTling** — The instrument is waiting for signals it controls to stabilize enough to begin measurements.
- **2-RANGing** — The instrument is currently changing its range.
- **3-SWEeping** — A sweep is in progress.
- **4-MEASuring** — The instrument is actively measuring.
- **5-Waiting for TRIG** — The instrument is in a “wait for trigger” state of the trigger model.
- **6-Waiting for ARM** — The instrument is in a “wait for arm” state of the trigger model.
- **7-CORRecting** — The instrument is currently performing a correction.

## 1999 SCPI Syntax & Style

- **8-12 available to designer**
- **13-INSTRument Summary Bit** One of n multiple logical instruments is reporting OPERational status.
- **14-PROGram running** — A user-defined programming is currently in the run state.
- **15 always zero**

### 9.4

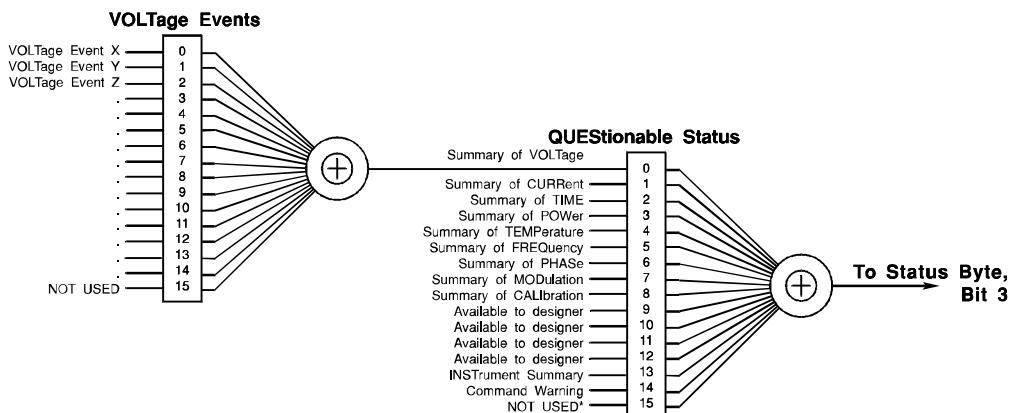
#### QUESTIONable Data/Signal Status Register

The QUESTIONable status register set contains bits which give an indication of the quality of various aspects of the signal.

A bit set in the condition register indicates that the data currently being acquired or generated is of questionable quality due to some condition affecting the parameter associated with that bit. For example, if the FREQ bit were set, this would mean that the frequency accuracy of the signal was of questionable quality.

9

The frequency bit might, in turn, have a register set associated with it, further refining the error into device-dependent conditions such as loop unlocked, oven cold, or reference signal missing. This layering of registers is called “fan-out”. See Figure 9-2 for an illustration of this technique. The device designer should be aware that adding registers to the status model increases complexity. At the same time, it may be the only way to communicate time sensitive status information to the instrument controller.



**Figure 9-2 Hierarchical Expansion of the QUESTIONable Status Register**

Bit 14 is defined as the Command Warning bit. This bit indicates a non-fatal warning that relates to the instrument's interpretation of a command, query, or one or more parameters of a specific command or query. Setting this bit is a warning to the application that the resultant instrument state or action is probably what was expected but may deviate in some manner.

For example, the Command Warning bit is set whenever a parameter in one of the Measurement Instruction commands or queries is ignored during execution. Such a parameter may be ignored because it cannot be specified by a particular instrument.

Bit 13, INSTRument summary, is described later in this chapter in association with multiple logical instruments.

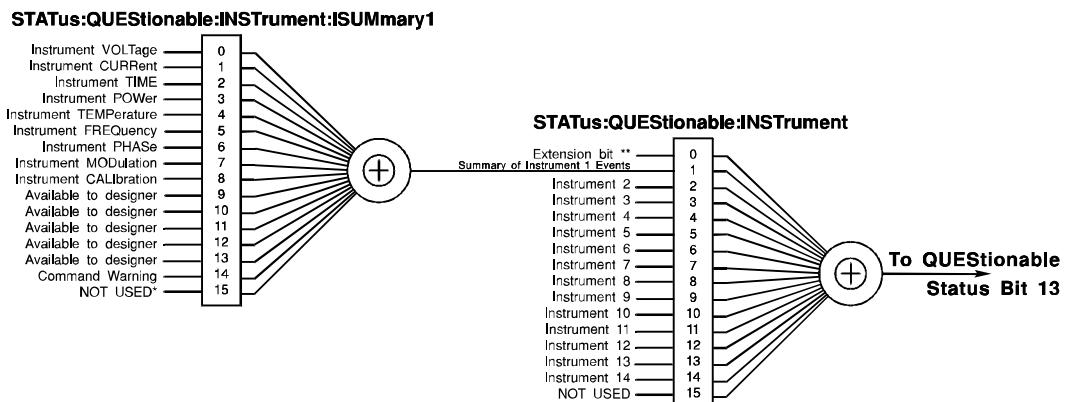
Bits in both OPERation and QUESTionable status registers may be redefined by the application programmer. This optional feature allows an application program to configure two registers in whatever form it likes. Any error or event number from the entire pool of errors/events the instrument can generate may be mapped into any bit of the OPERation or QUESTionable register.

For example, an application program controlling a voltmeter may choose to map VOLTage event X, VOLTage event Y and VOLTage event Z directly into the QUESTionable register so that it has more direct access to them. Another purpose for this feature is to minimize the need for adding complex hierarchical registers to the status model. The mapping is controlled by the STATUS:OPERATION:MAP and STATUS:QUESTIONABLE:MAP commands.

## 9.5

### Multiple Logical Instruments

A SCPI device that supports multiple logical instruments may include an INSTRument summary status register and an individual instrument ISUMmary for each logical instrument.



\* The use of Bit 15 is not allowed since some controllers may have difficulty reading a 16 bit unsigned integer. The value of this bit shall always be 0.

\*\* The extension bit is used, where required, to summarize the events in Instrument15 and up. The extension of the status structure shall conform to IEEE 488.2

Note that the OPERation Register is expanded in the same manner for multiple logical instruments.

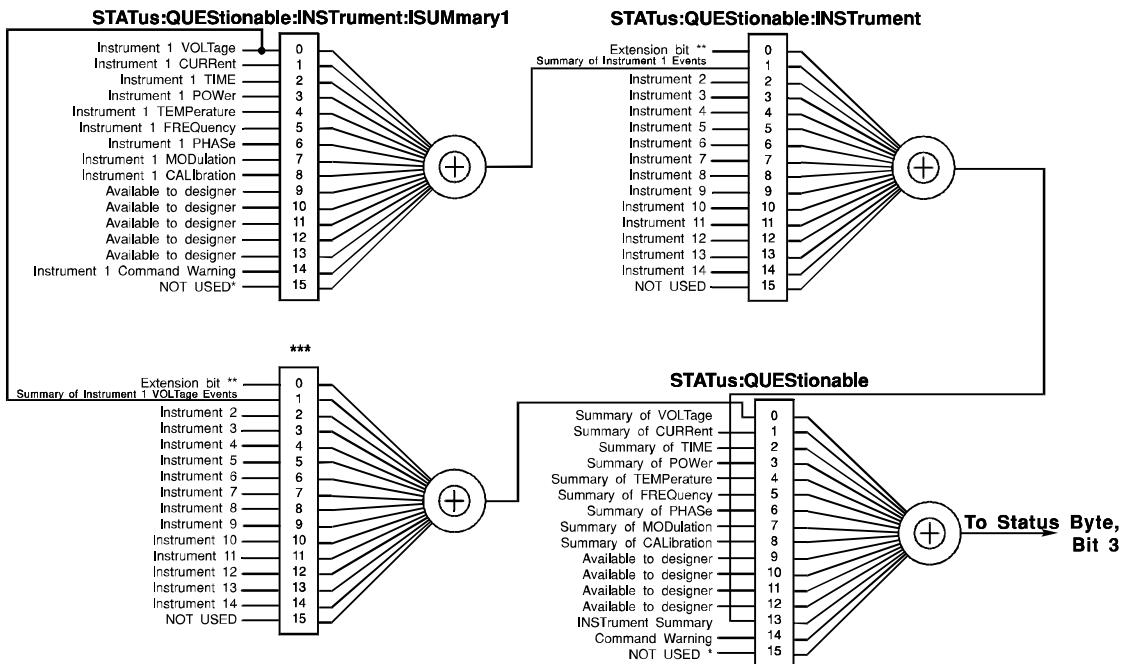
**Figure 9-3 Expansion of the INSTRument Summary Bit for Multiple Logical Instruments**

## 1999 SCPI Syntax & Style

The ISUMmary registers shall report to the INSTRument register, which in turn shall report to bit 13 of the QUESTIONable or OPERation status register. This is shown pictorially in Figure 9-3, "Expansion of the INSTRument Summary Bit for Multiple Logical Instruments." Such a multilogical instrument may also expand the QUESTIONable and OPERATION register in the manner shown in Figure 9-4, "Expansion of QUESTIONable Register for Multiple Logical Instruments."

Using such a status register configuration allows a status event to be cross-referenced by instrument and type of event. Further, when using a single logical instrument, the status structure is seen to behave in a manner that is directly compatible with a single physical instrument of the same capability. This affords upward compatibility.

For multiple logical instruments, the INSTRument register indicates which instrument(s) have generated an event. The ISUMmary register is a pseudo-questionable status register for



\* The use of Bit 15 is not allowed since some controllers may have difficulty reading a 16 bit unsigned integer. The value of this bit shall always be 0.

\*\* The extension bit is used, where required, to summarize the events in Instrument15 and up. The extension of the status structure shall conform to IEEE 488.2

\*\*\* The name of the register will be specified in a later version of SCPI

Note that the OPERATION Register is expanded in the same manner for multiple logical instruments.

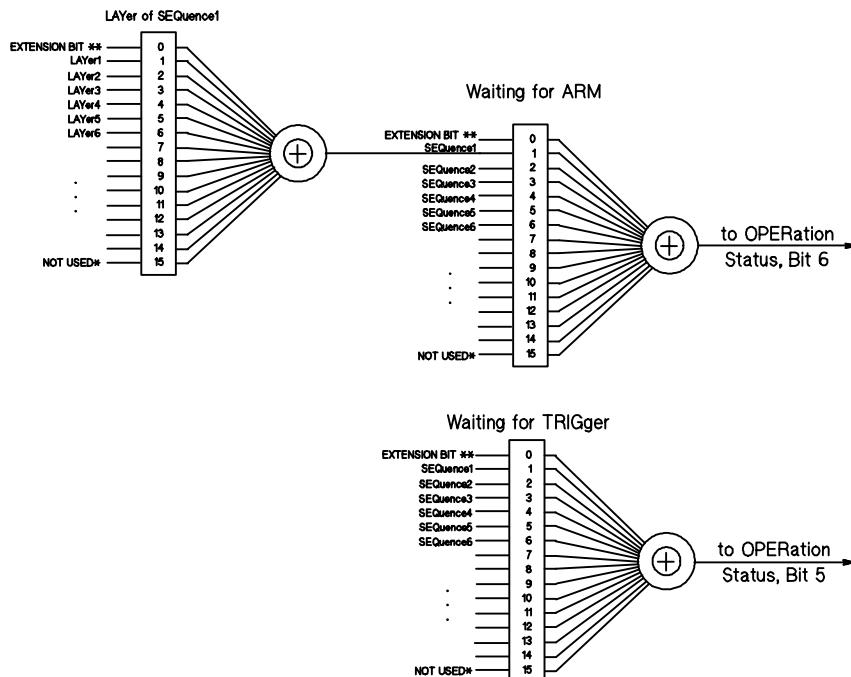
**Figure 9-4 Expansion of QUESTIONable Register for Multiple Logical Instruments**

a particular logical instrument. There may be two INSTRument registers for each logical instrument in the device. For each event type, such as VOLTage, all the events may be ORed together from each logical instrument to provide a summary by event type to the QUEstionable status register.

## 9.6

### Status Structure for the Expanded Capability Trigger Model

For instruments that implement the expanded capability trigger model, those instruments may implement status structures to indicate the exact point in which the wait for ARM or TRIGger is occurring. If multiple SEQuences are employed, then the status structure shall exist to indicate in which SEQuences the wait for ARM or TRIGger is occurring. If multiple ARM LAYers are employed, then the status structure shall exist to indicate in which LAYER the wait for ARM or TRIGger is occurring. Figure 9-5, “Expansion of the OPERational Register for the Expanded Capability Trigger Model,” shows the structure for an instrument that employs more than one TRIGger sequence, more than one ARM sequence, and multiple ARM LAYers.



\* The use of Bit 15 is not allowed since some controllers may have difficulty reading a 16 bit unsigned integer. The value of this bit shall always be 0.

\*\* The extension bit is used, where required, to summarize the events in INSTRument15 and up. The extension of the status structure shall conform to IEEE 488.2

**Figure 9-5** Expansion of OPERATION Register for the Expanded Capability Trigger Model

## **1999 SCPI Syntax & Style**

## 10 \*RST Conditions

All instruments shall return to a known configuration when the *IEEE 488.2* \*RST command is received. The “Command Reference” specifies the \*RST condition for every command. In some cases, the condition may be instrument-dependent. **Instrument-dependent** means that the designer may determine the setting at \*RST. However, the setting shall be known for the instrument. Leaving a setting undefined at \*RST is not allowed, unless specifically permitted in the command description.

Consideration shall be given to safety if the instrument is capable of producing hazardous conditions.

Operation after a \*RST is optimized for remote operation.

A \*RST command returns the instrument to a state where it is waiting for a command to initiate measurements or other instrument actions.

All instruments shall place themselves in the trigger idle state. Source instruments should not be sourcing power at any output port. This condition may be reached by lowering the power/voltage level or by turning the output state to OFF. Input ranges should normally be set to AUTORANGE or minimum sensitivity. Settings which are normally coupled should be coupled. Special modes, complex modulations, postprocessing, or similar functions which are generally application-dependent should be disabled leaving the instrument in its most fundamental mode of operation.

Finally, thought should be given to implementing \*RST conditions so that the incremental programming necessary after \*RST for the applications be minimized.

The SYSTem:PRESet command performs the same action as the front panel preset key. This typically sets all instrument parameters to values for good local/human interaction. In manual operation, it is often desirable to have SYSTem:PRESet enable continuous measurements. This may be different from the \*RST state and the power-on state.

Status structures are not affected by \*RST. The status event bits are cleared by \*CLS and by reading the event register. The error/event queue is also cleared by \*CLS. Device-dependent and SCPI status registers and queues are preset with the SCPI required STATus:PRESet command.

If a device is unable to implement the \*RST condition of a function, it may choose a different \*RST condition. However, under these circumstances, the command for this function must be implemented, even if only one selection is available. If the device implements the stated \*RST setting, the function must assume that value at \*RST.

## 1999 SCPI Syntax & Style

10

## 11 Naming Conventions

In many SCPI subsystems there is the need for assigning names to structures or expressions or arbitrary data memory.

This includes naming of windows in the DISPlay subsystem, traces in TRACe, expressions to specify events and sequences in TRIGger, expressions to specify cards and channel lists in ROUTe, and various items elsewhere. If possible, these naming operations should all have the same form so that these operations in the various subsystems can be consistent. A user can therefore transfer knowledge gained in one area to a similar task in another subsystem.

For this proposed general-purpose naming facility, there is no “enable” command; all currently defined names will be associated with their data at define/assign time and exist until deleted. There are two forms of the “purge” command: one which operates on individual names as well as one for the whole name space. Names have implicit types and memory is allocated based on the subsystem in which the name is created. Names must be unique within the system to allow a designer to implement these commands using one global name space.

The same keywords for naming will be used in all subsystems which allow names. Even though TRACe:DEFine and DISPlay:WINDOW:DEFine require different parameters, the operations are similar and the instrument programmer can expect similar keywords.

The name space is not affected by \*RST, and names remain associated with their data after this operation.

11

KEYWORD	PARAMETER FORM	NOTES
<b>Naming Sub-subsystem:</b>		
:DEFine	<name>,<data>	
:DEFine?	<name>	
:DElete		
[:NAME]	<name>	
:ALL		[event; no query]
:CATalog?		[query only]

### Command Descriptions

Differences among these commands and differences from the macro definition facility are noted in the descriptions.

The name is sent as <character data> rather than as a string, because that is the simplest IEEE 488.2 type that meets the requirements. Macro definitions needed the <string> type because macro names could contain colons and queries, and their total length could be more than 12 characters. Names do not have those extensions. When a name is returned (for example, in a “CATalog?” query), it must be sent as <string response data> so that a null string (“”) can indicate that nothing is defined.

## 1999 SCPI Syntax & Style

Names defined using these commands are not purged at \*RST. The only way of deleting them is through a DELETED or DELETED:ALL command.

### 11.1 :DEFIne <name>,<data>

This command associates a user-specified <name> with data <data>. The type of <data> is dependent on the subsystem in which the command occurs, and must be specified in the syntax for that tree. The <data> field may be optional for some subsystems.

### 11.2 :DEFIne? <name>

This query requests the instrument to return the definition of <name>. The type of data returned depends on the specific subsystem in which the command occurs. If possible (that is, if IEEE 488.2 allows), it should be returned in the same form it was sent.

### 11.3 :DELETED[:NAME] <name>

This command undefines the name, disassociates it from any data, and frees the name and its data memory for use by other definitions. There is no query associated with this action.

### 11.4 :DELETED:ALL

This command undefines all names in this subsystem, and frees any associated data memory. There is no query associated with this action.

### 11.5 :CATalog?

This query requests a list of defined names in this subsystem. The instrument must return a list of one or more strings, each containing one name and separated by commas. If no names are defined, a single null string is returned.

## A Programming Tips

By following a set of simple guidelines, a programmer can reduce the number of errors encountered when using SCPI instruments. All references in this appendix are to Volume I: Syntax and Style, unless otherwise specified.

1. Avoid sending default nodes. See 5.1 and 6.2.2 item 5. Default nodes are used within SCPI to allow the language to grow. Thus, older instruments will not have implemented the default node. A command is more likely to work with more instruments if the default nodes are omitted. For example, use:

```
INP:FILT ON
```

rather than:

```
INP:FILT:LPAS:STAT ON
```

2. Avoid sending a numeric suffix of 1 in applications that only use the default capability. See 6.2.5.2. An instrument with multiple capabilities is required to interpret a header without the numeric suffix as if a numeric suffix of one had been used. Leaving off the numeric suffix means the same commands will work with an instrument that has multiple capabilities and an instrument that does not. For example, use:

```
OUTP ON
```

rather than

```
OUTP1 ON
```

3. Be careful when sending coupled commands. See 7.4.2. Many instruments adhere to the suggestion in IEEE 488.2 section 6.4.5.3. If the coupled commands are contiguous in the same program message, the instrument is more likely to resolve any conflict between the current settings and the new settings. For example, use:

```
:FREQ:STAR 100;SPAN 100
```

which sets START to 100, STOP to 200, CENTER to 150, and SPAN to 100. Assume that START was 200 and STOP was 500 before sending;

```
:FREQ:STAR 100
```

Now, START is 100, STOP is 500, CENTER is 300, and SPAN is 400. Then send:

```
:FREQ:SPAN 100
```

which sets START to 250, STOP to 350, SPAN to 100, and leaves CENTER at 300 . These states are very different and the second is probably not the intended one.

4. Verify instrument settings when knowing their exact value is required. If the actual value of a setting is important, query the setting after programming it. An instrument is required to round parameters. See 7.2. For example, sending:

## 1999 SCPI Syntax & Style

```
:VOLT 1.68  
:VOLT?
```

may return 1.68, 1.7, or 2 depending on the capability of the instrument.

### 5. Send commands and queries in different program messages.

The response from a query combined in a program message with commands that affect the queried value is not predictable. Sending:

```
:FREQ:STAR 100;SPAN 100  
:FREQ:STAR?
```

always returns 100. When:

```
:FREQ:STAR 100;STAR?;SPAN 100
```

is sent, however, the result is not specified by SCPI. The result could be the value of START before the command was sent since the instrument might defer executing the individual commands until a program message terminator is received. The result could also be 100 if the instrument executes commands as they are received.

### 6. Use the highest level commands possible for compatibility across instruments. Whenever feasible, use the MEASure, CONFigure, READ?, FETCh?, and INITiate commands. Use the lower level commands only when a special characteristic of an instrument must be manipulated.

A

# Index

---

◇ 5-1  
[] 5-1  
{ } 5-1  
| 5-1

## A

aliases 6-3  
ANSI  
    X3.4-1977 2-1  
    X3.42-1975 2-1  
ANSI/EIA  
    TIA-562-1989 2-1  
ARM 9-3

## B

Bell Telephone BTSM 41004 2-1  
<Boolean> 7-5  
BTSM 41004 2-1  
building command trees 6-2

## C

CALibrating 9-3  
capability, multiple 6-8  
CATalog 11-2  
CCIR Recommendation  
    468-2 2-1  
CCITT Recommendation  
    P53 2-1  
    V.42 2-1  
channel  
    lists 8-3  
channel list 8-1  
channel\_list 6-8, 8-4  
<channel\_range> 8-4 - 8-5  
<channel\_spec> 8-5  
character  
    case 5-2  
character case 6-1  
<character data> 11-1  
CHARACTER PROGRAM DATA 7-5  
command table notation 5-1  
common command header 6-1  
common commands  
    effects of \*RST 10-1

mandatory 4-1  
optional 4-1  
compliance criteria 4-1  
CORRecting 9-3  
coupling 7-5  
    functional 7-6  
value 7-6

## D

data interchange format 8-1  
<DECIMAL NUMERIC PROGRAM DATA>  
    7-1, 7-5  
DEFault 7-1  
default node 5-1  
DEFine 11-2  
    query 11-2  
definition, macro 11-1  
DELETED 11-2  
device dependence 3-2  
devices 4-1  
dif 8-1  
display window naming 11-1  
Dolby Labs Bulletin No 19/4 2-1  
DOWN 7-1 - 7-2  
    STEP 7-3

## E

EIA  
    RS-232-D 2-1  
    RS-422 2-1  
element  
    lexical 8-1  
    syntactic 8-1  
expression  
    channel list 8-1  
    data interchange format 8-1  
    dif 8-1  
    instrument specifier 8-1  
    numeric 8-1 - 8-2  
    numeric list 8-1  
    precedence rules 8-3  
<EXPRESSION PROGRAM DATA> 8-1

# 1999 SCPI Syntax & Style

## F

flow diagram notation 5-2  
form  
  long 5-1, 6-1  
  short 5-1, 6-1  
front panel 1-1

## H

header  
  command command 6-1  
  command tree generation 6-2  
  instrument control 6-1  
  keyword generation 6-1  
  longform 6-1  
  numeric suffix 6-2, 6-8  
  queries 6-6  
  query 6-1  
  short form 6-2  
  tree traversal 6-7  
hierarchical structure 5-1

## I

IEC Recommendation  
  179 2-1  
IEEE  
  Std. 181-1977 2-1  
  Std. 194-1977 2-1  
  Std. 260-1978 2-1  
  Std. 488.1-1987 1-1, 2-1, 4-1  
  Std. 488.2-1987 1-1, 1-3, 4-1, 6-1, 6-7  
  Std. 488.2-1992 2-1  
  Std. 754-1985 2-1  
INFinity 7-1, 7-4  
INSTrument 9-6  
instrument control header 6-1  
instrument dependent 10-1  
instrument specifier expression 8-1  
INSTrument Summary Bit 9-4  
ISO  
  Std. 2955-1983 2-2  
ISUMmary 9-6

## K

KEYWORD 5-1  
keywords 6-1

## L

label 7-1  
lexical element 8-1  
life cycle 3-1 - 3-2  
living standard 3-1  
logical instruments 6-9  
long form 5-1, 6-1

## M

macro definition 11-1  
mandated commands 4-1  
MAXimum 7-1  
MEASuring 9-3  
MINimum 7-1 - 7-2  
<module\_channel> 8-4  
<moduleSpecifier> 8-5  
multiple  
  capabilities 6-8  
  electrical ports 6-8  
  identical capabilities 6-8  
  logical instruments status 9-5

## N

naming conventions 11-1  
  CATalog 11-2  
  DEFine 11-2  
  DELete 11-2  
NAN (not a number) 7-4  
nomenclature  
  see notation 5-1  
<non-decimal numeric> 7-1  
Not A Number (NAN) 7-1  
notation 5-1  
  command tables 5-1  
  query 5-1  
  status structure diagrams 9-1  
  syntax flow diagrams 5-2  
NOTES 5-1  
<NRf> 7-1  
numeric expression 8-1 - 8-2  
numeric list 8-1  
<numeric\_expression> 7-1  
<numeric\_list> 8-6  
<numeric\_operator> 8-2  
<numeric\_range> 8-6

## 1999 SCPI Syntax & Style

### O

obsolescence 3-1  
OFF 7-5  
ON 7-5  
OPERation 9-1  
OPERation Status Register 9-3  
optional  
    use of expressions 8-1  
    commands 4-6  
    common commands 4-1  
    IEEE 488.1 interface 4-1  
    nodes 6-4  
    SCPI commands 4-6

### P

PARAMETER FORM 5-1  
parameters 7-1  
    <CHARACTER PROGRAM DATA> 7-1  
    <NRf> 7-1  
    <numeric\_value> 7-1  
    Boolean program data 7-5  
    character program data 7-1  
    decimal numeric program data 7-1  
    DEFault 7-1  
    DOWN 7-2  
    INFinity 7-4  
    MAXimum 7-2  
    NAN (not a number) 7-4  
    STEP UP/DOWN 7-3  
    unit suffixes 7-5  
    UP 7-2  
precedence rules 8-3  
primary keyword 6-3  
program headers 6-1  
    queries 6-6  
    <PROGRAM MESSAGE> 6-7  
    <PROGRAM MESSAGE UNIT> 6-7  
PROGram running 9-4

### Q

queries 6-6  
QUEStionable 9-1  
QUEStionable Data 9-4

### R

railroad diagrams 5-2  
RANGing 9-3  
References 2-1

requirements  
    documentation 4-6  
    IEEE 488.1 4-1  
    IEEE 488.2 4-1  
    IEEE mandated commands 4-1  
    minimum status structure 9-1  
    multiple capabilities 6-9  
    safety 10-1  
    SCPI 4-5  
    SCPI commands 4-5  
rounding  
    numeric program data 7-1

### S

SAE J2264 2-2  
safety requirements 10-1  
SCPI  
    requirements 4-5  
    semantics 8-3  
    SETTling 9-3  
    short form 5-1, 6-1  
    Signal Status Register 9-4  
    square brackets 5-1  
    status reporting 9-1  
        device dependent registers 9-3  
        diagram notation 9-1  
        effect of \*CLS 9-3  
        effect of \*RST 9-3  
        enhanced TRIGger model 9-7  
        minimum requirement 9-1  
        multiple logical instruments 9-5  
        OPERation 9-1, 9-3  
        QUEStionable 9-1, 9-4  
        transition filters 9-3  
        TRIGger model 9-1  
STEP 7-3  
    AUTO 7-4  
    INCReement 7-3  
    MODE 7-3  
    PDECade 7-3  
    <string> 11-1  
    <SUFFIX PROGRAM DATA> 7-5  
SWEeping 9-3  
Syntax and Style 1-1  
syntax flow diagram notation 5-2  
systems instruments 1-1

### T

trace naming 11-1  
    <trace\_name> 8-3

## 1999 SCPI Syntax & Style

transition filters 9-3  
    negative 9-3  
    positive 9-3  
Traversal of header tree 6-7  
tree system 5-1  
tree walking 6-7  
    enhanced 6-7  
TRIGger 9-3  
LAYers 9-7  
SEQuences 9-7

### U

<unary\_numeric\_operator> 8-3  
unit  
    amplitude and power 7-7  
    suffixes 7-5  
    unitless quantities 7-9  
    units of measure 7-7  
UP 7-1 - 7-2  
    STEP 7-3

### V

<variable or trace\_name> 8-3  
VXI 2-2, 6-9

**Standard  
Commands for  
Programmable  
Instruments  
(SCPI)**

**Volume 2: Command Reference**

---

VERSION 1999.0  
May, 1999  
Printed in U.S.A.

# Table of Contents

---

Chapter 1	Introduction . . . . .	1-1
Chapter 2	Instrument Model . . . . .	2-1
Chapter 3	Measurement Instructions . . . . .	3-1
Chapter 4	CALCulate Subsystem . . . . .	4-1
Chapter 5	CALibration Subsystem . . . . .	5-1
Chapter 6	CONTrol Subsystem . . . . .	6-1
Chapter 7	DIAGnostic Subsystem . . . . .	7-1
Chapter 8	DISPlay Subsystem . . . . .	8-1
Chapter 9	FORMAT Subsystem . . . . .	9-1
Chapter 10	HCOPy . . . . .	10-1
Chapter 11	INPut Subsystem . . . . .	11-1
Chapter 12	INSTrument Subsystem . . . . .	12-1
Chapter 13	MEMORY Subsystem . . . . .	13-1
Chapter 14	MMEMorySubsystem . . . . .	14-1
Chapter 15	OUTPut Subsystem . . . . .	15-1
Chapter 16	PROGram Subsystem . . . . .	16-1
Chapter 17	ROUTe Subsystem . . . . .	17-1
Chapter 18	SENSe Subsystem . . . . .	18-1
Chapter 19	SOURce Subsystem . . . . .	19-1
Chapter 20	STATus Subsystem . . . . .	20-1
Chapter 21	SYSTem Subsystem . . . . .	21-1
Chapter 22	TEST Subsystem . . . . .	22-1
Chapter 23	TRACe   DATA . . . . .	23-1
Chapter 24	TRIGger Subsystem . . . . .	24-1
Chapter 25	UNIT Subsystem . . . . .	25-1
Chapter 26	VXI Subsystem . . . . .	26-1

## **1999 SCPI Command Reference**

# Table of Contents

---

## Chapter 1 Introduction

1.1	Requirements .....	1-1
1.2	Organization .....	1-1

## Chapter 2 Instrument Model

2.1	Signal Routing .....	2-2
2.1.1	Setting the Destination for Data Flow .....	2-2
2.2	Measurement Function .....	2-3
2.2.1	INPut .....	2-3
2.2.2	SENSe .....	2-4
2.2.3	CALCulate .....	2-4
2.3	Signal Generation .....	2-4
2.3.1	OUTPut .....	2-4
2.3.2	SOURce .....	2-5
2.3.3	CALCulate .....	2-5
2.4	TRIGger .....	2-5
2.5	MEMory .....	2-5
2.6	FORMAT .....	2-5
2.7	Internal Routing .....	2-5
2.7.1	Data and Control Flow .....	2-6
2.7.2	Numeric Suffixes .....	2-7
2.7.3	Lamina and Cloned Models .....	2-7
2.7.3.1	FEED Exceptions .....	2-8
2.7.3.2	Amorphous SENSe Model .....	2-8
2.7.3.3	Other Subsystems .....	2-9
2.7.4	FEED <data_handle> .....	2-9
2.7.4.1	Sensor Function Selection .....	2-9
2.7.4.2	FEEDs from CALCULATE Sub-Blocks .....	2-12
2.7.5	COMBINE .....	2-12
2.7.6	Memory Associated with Data Flow .....	2-12
2.7.7	Querying the Data Flow .....	2-14
2.7.8	<event_handle> .....	2-14

## Chapter 3 Measurement Instructions

3.1	CONFigure:<function> <parameters>[,<source list>] .....	3-2
3.2	FETCh[:<function>]? <parameters>[,<source list>] .....	3-3
3.3	READ[:<function>]? <parameters> [,<source list>] .....	3-4
3.4	MEASure:<function>? <parameters>[,<source list>] .....	3-5
3.5	<function> .....	3-5

## 1999 SCPI Command Reference

3.6	Presentation Layer .....	3-7
3.6.1	Presentation Command Summary .....	3-7
3.6.2	[:SCALar] .....	3-7
3.6.3	:ARRay <size> .....	3-7
3.7	Fundamental Measurement Layer .....	3-7
3.7.1	Fundamental Measurement Command Summary .....	3-7
3.7.2	:VOLTage [<expected_value>[,<resolution>]] .....	3-8
3.7.3	:CURRent [<expected_value>[,<resolution>]] .....	3-8
3.7.4	:POWER [<expected_value>[,<resolution>]] .....	3-8
3.7.5	:RESistance [<expected_value>[,<resolution>]] .....	3-8
3.7.6	:FRESistance [<expected_value>[,<resolution>]] .....	3-8
3.7.7	:TEMPerature [<transducer>[,<type> [,<expected_value>[,<resolution>]]]] .....	3-8
3.8	Measurement Function Layer .....	3-9
3.8.1	Simple Measurements .....	3-9
3.8.1.1	Simple Measurements Command Summary .....	3-9
3.8.1.2	:AC .....	3-9
3.8.1.3	[:DC] .....	3-9
3.8.1.4	:FREQuency [<expected_value>[,<resolution>]] .....	3-9
3.8.1.4.1	:BURSt [<expected_value>[,<resolution>]] .....	3-9
3.8.1.4.2	:PRF [<expected_value>[,<resolution>]] .....	3-10
3.8.1.5	:PERiod [<expected_value>[,<resolution>]] .....	3-10
3.8.1.6	:PHASe [<expected value>[,<resolution>]] .....	3-10
3.8.2	Time Domain Waveform Measurements .....	3-10
3.8.2.1	Waveform Measurements Command Summary .....	3-10
3.8.2.2	:AMPLitude .....	3-11
3.8.2.3	:LOW .....	3-12
3.8.2.4	:HIGH .....	3-12
3.8.2.5	:RISE .....	3-12
3.8.2.5.1	:TIME [<low reference>[,<high reference> [,<expected_value>[,<resolution>]]]] .....	3-12
3.8.2.5.2	:OVERshoot .....	3-12
3.8.2.5.3	:PRESHoot .....	3-12
3.8.2.6	:RTIMe [<low reference>[,<high reference> [,<expected_value>[,<resolution>]]]] .....	3-12
3.8.2.7	:FALL .....	3-13
3.8.2.7.1	:TIME [<low reference>[,<high reference> [,<expected_value>[,<resolution>]]]] .....	3-13
3.8.2.7.2	:OVERshoot .....	3-13
3.8.2.7.3	:PRESHoot .....	3-13
3.8.2.8	:FTIMe [<low reference>[,<high reference> [,<expected_value>[,<resolution>]]]] .....	3-13
3.8.2.9	:PWIDth [<reference>] .....	3-13
3.8.2.10	:NWIDth [<reference>] .....	3-14
3.8.2.11	:PDUtYcycle :DCYCle [<reference>] .....	3-14
3.8.2.12	:NDUTyCycle [<reference>] .....	3-14

## 1999 SCPI Command Reference

3.8.2.13	:TMAXimum Layer:Time Domain Waveform Measurements .....	3-14
3.8.2.14	:TMINimum .....	3-14
3.8.2.15	:MINimum .....	3-14
3.8.2.16	:MAXimum .....	3-14
3.8.2.17	:PTPeak .....	3-14

## Chapter 4 CALCulate Subsystem

4.1	:AVERage subsystem .....	4-6
4.1.1	:CLEar .....	4-6
4.1.2	:COUNt <numeric_value> .....	4-7
4.1.2.1	:AUTO <Boolean>   ONCE .....	4-7
4.1.3	[::STATe] <Boolean> .....	4-7
4.1.4	:TCONtrol EXPonential   MOVing   NORMal   REPeat .....	4-7
4.1.5	:TYPE COMplex   ENVelope   MAXimum   MINimum   RMS   SCALar .....	4-8
4.2	:CLIMits .....	4-9
4.2.1	:FAIL? .....	4-9
4.2.2	:FLIMits .....	4-9
4.2.2.1	[::DATA]? .....	4-9
4.2.2.2	:POINts? .....	4-9
4.3	:DATA? .....	4-10
4.3.1	:PREamble? .....	4-10
4.4	:DERivative .....	4-10
4.4.1	:STATe <Boolean> .....	4-10
4.4.2	:POINts <numeric_value> .....	4-10
4.5	:FEED <data_handle> .....	4-10
4.6	:FILTer .....	4-11
4.6.1	[::GATE] .....	4-11
4.6.1.1	:TIME .....	4-11
4.6.1.1.1	:STATe <Boolean> .....	4-11
4.6.1.1.2	[::TYPE ] BPASs NOTCh .....	4-11
4.6.1.1.3	:STARt <numeric_value> .....	4-11
4.6.1.1.4	:STOP <numeric_value> .....	4-11
4.6.1.1.5	:SPAN <numeric_value> .....	4-12
4.6.1.1.6	:CENTer <numeric_value> .....	4-12
4.6.1.1.7	:POINts <numeric_value> .....	4-12
4.6.1.1.7.1	:AUTO <Boolean> ONCE .....	4-12
4.6.1.1.8	:WINDOW RECTangular UNIForm FLATtop HAMMING  HANNing KBESsel FORCe EXPonential .....	4-12
4.6.1.1.9	:KBESsel <numeric_value> .....	4-12
4.6.1.1.10	:EXPonential <numeric_value> .....	4-12
4.6.1.1.11	:FORCe <numeric_value> .....	4-13
4.6.1.2	:FREQuency .....	4-13
4.6.1.2.1	:STATe <Boolean> .....	4-13
4.6.1.2.2	[::TYPE ] BPASs NOTCh .....	4-13

## 1999 SCPI Command Reference

4.6.1.2.3	:STARt <numeric_value> .....	4-13
4.6.1.2.4	:STOP <numeric_value> .....	4-13
4.6.1.2.5	:SPAN <numeric_value> .....	4-14
4.6.1.2.6	:CENTer <numeric_value> .....	4-14
4.6.1.2.7	:POINts <numeric_value> .....	4-14
4.6.1.2.7.1	:AUTO <Boolean> ONCE .....	4-14
4.6.1.2.8	:WINDow RECTangular UNIForm FLATtop HAMMing  HANNing KBESsel FORCe EXPonential .....	4-14
4.6.1.2.9	:KBESsel <numeric_value> .....	4-14
4.6.1.2.10	:EXPonential <numeric_value> .....	4-14
4.6.1.2.11	:FORCe <numeric_value> .....	4-15
4.7	:FORMat NONE MLINear MLOGarithmic PHASe REAL  IMAGinary SWR GDELay  COMPlex NYQuist NICHols POLar  UPHase .....	4-15
4.7.1	:UPHase .....	4-16
4.7.1.1	:CREFerence <numeric_value> .....	4-17
4.7.1.2	:PREFerence <numeric value> .....	4-17
4.8	:GDAPerture .....	4-17
4.8.1	:SPAN <numeric_value> .....	4-17
4.8.2	:APERture <numeric_value> .....	4-17
4.9	:IMMediate .....	4-18
4.9.1	:AUTO <Boolean> .....	4-18
4.10	:INTegral .....	4-18
4.10.1	:STATe <Boolean> .....	4-18
4.10.2	:TYPE SCALar   MOVing .....	4-18
4.11	:LIMit .....	4-18
4.11.1	:STATe <Boolean> .....	4-19
4.11.2	:CONTrol .....	4-19
4.11.2.1	[:DATA] <numeric_value>{,<numeric_value>} .....	4-19
4.11.2.2	:POINts? .....	4-19
4.11.3	:UPPer .....	4-19
4.11.3.1	[:DATA] <numeric_value>{,<numeric_value>} .....	4-19
4.11.3.2	:POINts? .....	4-20
4.11.3.3	:STATe <Boolean> .....	4-20
4.11.4	:LOWer .....	4-20
4.11.4.1	[:DATA] <numeric_value>{,<numeric_value>} .....	4-20
4.11.4.2	:POINts? .....	4-20
4.11.4.3	:STATe <Boolean> .....	4-20
4.11.5	:FAIL? .....	4-20
4.11.6	:FCOut? .....	4-20
4.11.7	:REPort .....	4-20
4.11.7.1	[:DATA]? .....	4-20
4.11.7.2	:POINts? .....	4-21
4.11.8	:CLEar .....	4-21

## 1999 SCPI Command Reference

4.11.8.1	:AUTO <Boolean>   ONCE .....	4-21
4.11.8.2	[:IMMEDIATE] .....	4-21
4.11.9	:INTerpolate <Boolean> .....	4-21
4.12	:MATH .....	4-21
4.12.1	[:EXPRESSION] <numeric_expression> .....	4-21
4.12.1.1	:CATalog? .....	4-21
4.12.1.2	[:DEFINE]<numeric_expression> .....	4-22
4.12.1.3	:DELetE .....	4-22
4.12.1.3.1	[:SElected]<expression_name> .....	4-22
4.12.1.3.2	:ALL .....	4-22
4.12.1.4	:NAME <expression_name> .....	4-22
4.12.2	:STATe <Boolean> .....	4-23
4.13	:SMOothing .....	4-23
4.13.1	[:STATe] <Boolean> .....	4-23
4.13.2	:APERture <numeric_value> .....	4-23
4.13.3	:POINts <numeric_value> .....	4-23
4.14	:STATe <Boolean> .....	4-23
4.15	:TRANSform .....	4-23
4.15.1	:HISTogram .....	4-24
4.15.1.1	:COUNt <numeric_value> .....	4-24
4.15.1.2	:ORDinate RATio   PERCent   PCT   COUNt .....	4-24
4.15.1.3	:POINts <numeric_value> .....	4-24
4.15.1.4	:RANGE .....	4-24
4.15.1.4.1	:AUTO <Boolean> .....	4-25
4.15.1.5	:STATe <Boolean> .....	4-25
4.15.2	:TIME .....	4-25
4.15.2.1	:STATe <Boolean> .....	4-25
4.15.2.2	[:TYPE ] LPASS BPASS .....	4-25
4.15.2.3	:STIMulus STEP IMPulse .....	4-25
4.15.2.4	:STARt <numeric_value> .....	4-25
4.15.2.5	:STOP <numeric_value> .....	4-26
4.15.2.6	:SPAN <numeric_value> .....	4-26
4.15.2.7	:CENTer <numeric_value> .....	4-26
4.15.2.8	:POINts <numeric_value> .....	4-26
4.15.2.8.1	:AUTO <Boolean> ONCE .....	4-26
4.15.2.9	:WINDOW RECTangular UNIForm FLATtop HAMMING  HANNing KBESsel FORCe EXPonential .....	4-26
4.15.2.10	:KBESsel <numeric_value> .....	4-27
4.15.2.11	:EXPonential <numeric_value> .....	4-27
4.15.2.12	:FORCe <numeric_value> .....	4-27
4.15.3	:DISTance .....	4-27
4.15.3.1	:STATe <Boolean> .....	4-27
4.15.3.2	[:TYPE ] LPASS BPASS .....	4-27
4.15.3.3	:STIMulus STEP IMPulse .....	4-27
4.15.3.4	:STARt <numeric_value> .....	4-28

## 1999 SCPI Command Reference

4.15.3.5	:STOP <numeric_value> .....	4-28
4.15.3.6	:SPAN <numeric_value> .....	4-28
4.15.3.7	:CENTer <numeric_value> .....	4-28
4.15.3.8	:POINts <numeric_value> .....	4-28
4.15.3.8.1	:AUTO <Boolean> ONCE .....	4-28
4.15.3.9	:WINDow RECTangular UNIForm FLATtop HAMMING  HANNing KBESsel FORCe EXPonential .....	4-29
4.15.3.10	:KBESsel <numeric_value> .....	4-29
4.15.3.11	:EXPonential <numeric_value> .....	4-29
4.15.3.12	:FORCe <numeric_value> .....	4-29
4.15.4	:FREQuency .....	4-29
4.15.4.1	:STATE <Boolean> .....	4-29
4.15.4.2	[:TYPE ] LPASS BPASS .....	4-29
4.15.4.3	:STIMulus STEP IMPulse .....	4-30
4.15.4.4	:STARt <numeric_value> .....	4-30
4.15.4.5	:STOP <numeric_value> .....	4-30
4.15.4.6	:SPAN <numeric_value> .....	4-30
4.15.4.7	:CENTer <numeric_value> .....	4-30
4.15.4.8	:POINts <numeric_value> .....	4-31
4.15.4.8.1	:AUTO <Boolean> ONCE .....	4-31
4.15.4.9	:WINDow RECTangular UNIForm FLATtop HAMMING  HANNing KBESsel FORCe EXPonential .....	4-31
4.15.4.10	:KBESsel <numeric_value> .....	4-31
4.15.4.11	:EXPonential <numeric_value> .....	4-31
4.15.4.12	:FORCe <numeric_value> .....	4-31
4.16	:PATH(MATH TRANSform FILTER SMOothing FORMAT LIMIT AVERage) {,(MATH TRANSform FILTER SMOothing FORMAT LIMIT AVERage)} .....	4-31

## Chapter 5 CALibration Subsystem

5.1	[:ALL] .....	5-2
5.2	[:ALL]? .....	5-2
5.3	:AUTO <Boolean> ONCE .....	5-2
5.4	:BINertia .....	5-2
5.4.1	:AVERage? .....	5-2
5.4.2	:HSPeed <numeric_value> .....	5-3
5.4.3	:INITiate .....	5-3
5.4.4	:LSPeed <numeric_value> .....	5-3
5.4.5	:NRUNs <numeric_value> .....	5-3
5.4.6	:SDEVIation? .....	5-4
5.4.7	:UPDate .....	5-4
5.5	:DATA <arbitrary block program data> .....	5-4
5.6	:PLOSS .....	5-4
5.6.1	:APCoeff<numeric_value>,<numeric_value>,<numeric_value>[,<numeric_value>] .....	5-4
5.6.2	:INITiate .....	5-4

## 1999 SCPI Command Reference

5.6.3	:LATime .....	5-5
5.6.4	:STIMe <numeric_value> .....	5-5
5.6.5	:UPDate .....	5-5
5.7	:SOURce INTernal EXTernal .....	5-6
5.8	:STATe <Boolean> .....	5-6
5.9	:VALue <numeric_value> .....	5-6
5.10	:WARMup .....	5-6
5.10.1	:INITiate .....	5-6
5.10.2	:SPEEd <numeric_value> .....	5-7
5.10.3	:TIMEout <numeric_value> .....	5-7
5.11	:ZERO .....	5-7
5.11.1	:AUTO <Boolean> ONCE .....	5-7
5.11.2	:FSENsor .....	5-7
5.11.2.1	:INITiate .....	5-8
5.11.2.2	:LATime <numeric_value> .....	5-8
5.11.2.3	:LEVel? .....	5-8
5.11.2.4	:SPEEd <numeric_value> .....	5-8
5.11.2.5	:STIMe <numeric_value> .....	5-8
5.11.2.6	:UPDate .....	5-9

## Chapter 6 CONTROL Subsystem

6.1	:APOWer .....	6-1
6.1.1	[::STATe] <Boolean> .....	6-1
6.2	:BLOWer .....	6-2
6.2.1	[::STATe] <Boolean> .....	6-2
6.3	:BRAKE .....	6-2
6.3.1	[::STATe] <Boolean> .....	6-2
6.4	:COMPressor .....	6-2
6.4.1	[::STATe] <Boolean> .....	6-2
6.5	:COVer .....	6-2
6.5.1	[::ADJust] OPEN CLOSE SOPEN SCLOSE .....	6-2
6.5.2	:POStion? .....	6-3
6.6	:EBENch .....	6-3
6.6.1	:CLEan .....	6-3
6.6.1.1	:INITiate .....	6-3
6.6.1.2	:DURation <numeric_value> .....	6-3
6.7	:IDLE .....	6-3
6.7.1	:INITiate .....	6-3
6.8	:LIFT .....	6-3
6.8.1	[::ADJust] UP DOWN .....	6-4
6.8.2	:POStion? .....	6-4
6.9	:MCOnrol .....	6-4
6.9.1	[::STATe] <Boolean> .....	6-4
6.10	:ROTation .....	6-4

## 1999 SCPI Command Reference

6.10.1	[:DIRection] . . . . .	6-4
6.11	:VCDevice . . . . .	6-4
6.11.1	[:STATE] <Boolean> . . . . .	6-4
6.11.2	:TDIameter <numeric_value> . . . . .	6-5

## Chapter 7 DIAGnostic Subsystem

## Chapter 8 DISPlay Subsystem

8.1	:ANNAnnotation . . . . .	8-4
8.1.1	[:ALL] <Boolean> . . . . .	8-4
8.1.2	:AMPLitude <Boolean> . . . . .	8-4
8.1.3	:FREQuency <Boolean> . . . . .	8-4
8.2	:BRIGhtness <numeric_value> . . . . .	8-4
8.3	:CMAP . . . . .	8-4
8.3.1	:DEFault . . . . .	8-4
8.3.2	:COLor . . . . .	8-4
8.3.2.1	:HSL <hue>,<sat>,<lum> . . . . .	8-5
8.3.2.2	:RGB <red>,<green>,<blue> . . . . .	8-5
8.4	:CONTrast <numeric_value> . . . . .	8-5
8.5	:ENABLE <Boolean> . . . . .	8-5
8.6	:MENU . . . . .	8-6
8.6.1	[:NAME] <menu_name> . . . . .	8-6
8.6.2	:STATe <Boolean> . . . . .	8-6
8.6.3	:KEY <string> . . . . .	8-6
8.7	[:WINDOW] . . . . .	8-6
8.7.1	:BACKground . . . . .	8-6
8.7.1.1	:COLor <numeric_value> . . . . .	8-6
8.7.2	:GEOMetry . . . . .	8-6
8.7.2.1	:LLEFt <numeric_value>,<numeric_value> . . . . .	8-7
8.7.2.2	:SIZE <numeric_value>,<numeric_value> . . . . .	8-7
8.7.2.3	:URIGht <numeric_value>,<numeric_value> . . . . .	8-7
8.7.3	:GRAPHics . . . . .	8-7
8.7.3.1	:CLEAR . . . . .	8-7
8.7.3.2	:COLor <numeric_value> . . . . .	8-7
8.7.3.3	:CSIZE <numeric_value>[,<numeric_value>] . . . . .	8-7
8.7.3.4	[:DRAW] <numeric_value>,<numeric_value> . . . . .	8-8
8.7.3.5	:PCL <block> . . . . .	8-8
8.7.3.6	:HPGL <block> . . . . .	8-8
8.7.3.7	:IDRaw <numeric_value>,<numeric_value> . . . . .	8-8
8.7.3.8	:IMOVe <numeric_value>,<numeric_value> . . . . .	8-8
8.7.3.9	:LABel <string> . . . . .	8-8
8.7.3.10	:LDIRection <numeric_value> . . . . .	8-8
8.7.3.11	:LTYPe <numeric_value>[,<numeric_value>] . . . . .	8-8
8.7.3.12	:MOVE <numeric_value>,<numeric_value> . . . . .	8-9

## 1999 SCPI Command Reference

8.7.3.13	:STATe <Boolean>	.....	8-9
8.7.4	[:STATe] <Boolean>	.....	8-9
8.7.5	:TEXT	.....	8-9
8.7.5.1	:ATTRibutes <Boolean>	.....	8-9
8.7.5.2	:CLEar	.....	8-9
8.7.5.3	:COLOr <numeric_value>	.....	8-9
8.7.5.4	:CSIZe <numeric_value>[,<numeric_value>]	.....	8-10
8.7.5.5	:FEED <data_handle>	.....	8-10
8.7.5.6	[:DATA] <string>   <block>	.....	8-10
8.7.5.7	:LOCate <numeric_value>[,<numeric_value>]	.....	8-10
8.7.5.8	:PAGE <numeric_value>	.....	8-10
8.7.5.9	:STATe <Boolean>	.....	8-10
8.7.6	:TRACe	.....	8-10
8.7.6.1	:COLOr <numeric_value>	.....	8-10
8.7.6.2	:FEED <data_handle>	.....	8-10
8.7.6.3	:GRATicule	.....	8-11
8.7.6.3.1	:AXIS	.....	8-11
8.7.6.3.1.1	[:STATe] <Boolean>	.....	8-11
8.7.6.3.2	:FRAMe	.....	8-11
8.7.6.3.2.1	[:STATe] <Boolean>	.....	8-11
8.7.6.3.3	:GRID	.....	8-11
8.7.6.3.3.1	:AUTO <Boolean>	.....	8-11
8.7.6.3.3.2	[:STATe] <Boolean>	.....	8-11
8.7.6.4	:PERSistence <numeric_value>	.....	8-11
8.7.6.4.1	:AUTO <Boolean> ONCE	.....	8-11
8.7.6.5	:STATe <Boolean>	.....	8-11
8.7.6.6	:X	.....	8-12
8.7.6.6.1	:LABel <string>	.....	8-12
8.7.6.6.2	[:SCALe]	.....	8-12
8.7.6.6.2.1	:AUTO <Boolean>  ONCE	.....	8-12
8.7.6.6.2.2	:CENTer <numeric_value>	.....	8-12
8.7.6.6.2.3	:LEFT <numeric_value>	.....	8-12
8.7.6.6.2.4	:PDIVision <numeric_value>	.....	8-13
8.7.6.6.2.5	:RIGHT <numeric_value>	.....	8-13
8.7.6.7	:Y	.....	8-13
8.7.6.7.1	:LABel <string>	.....	8-13
8.7.6.7.2	:RLINe <Boolean>	.....	8-13
8.7.6.7.3	[:SCALe]	.....	8-13
8.7.6.7.3.1	:AUTO <Boolean>  ONCE	.....	8-14
8.7.6.7.3.2	:BOTTom <numeric_value>	.....	8-14
8.7.6.7.3.3	:PDIVision <numeric_value>	.....	8-14
8.7.6.7.3.4	:RLEVel <numeric_value>	.....	8-14
8.7.6.7.3.5	:RPOSition <numeric_value>	.....	8-15
8.7.6.7.3.6	:TOP <numeric_value>	.....	8-15
8.7.6.7.4	:SPACing LOGarithmic   LINear	.....	8-15

## 1999 SCPI Command Reference

8.7.6.8	:R .....	8-15
8.7.6.8.1	:LABel <string> .....	8-15
8.7.6.8.2	[.SCALE] .....	8-15
8.7.6.8.2.1	:AUTO <Boolean>   ONCE .....	8-15
8.7.6.8.2.2	:CPOInt <numeric_value> .....	8-15
8.7.6.8.2.3	:OEDGe <numeric_value> .....	8-16
8.7.6.8.3	:SPACing LOGarithmic   LINear .....	8-16

## Chapter 9 FORMat Subsystem

9.1	:BORDer NORMAL SWAPPed .....	9-1
9.2	[:DATA] <type>[,<length>] .....	9-1
9.3	:DINTerchange <Boolean> .....	9-2
9.4	:SREGister ASCII   BINary   HEXadecimal   OCTal .....	9-3

## Chapter 10 HCOPy

10.1	:ABORT .....	10-4
10.2	:DATA? .....	10-4
10.3	:DESTination<data_handle> .....	10-4
10.4	:DEVICE .....	10-5
10.4.1	:CMAP .....	10-5
10.4.1.1	:COLor .....	10-5
10.4.1.1.1	:HSL <hue>,<sat>,<lum> .....	10-5
10.4.1.1.2	:RGB <red>,<green>,<blue> .....	10-5
10.4.1.2	:DEFault .....	10-6
10.4.2	:COLor <Boolean> .....	10-6
10.4.3	:LANGuage PCL[<n>]   HPGL[<n>]   POSTscript[<n>] .....	10-6
10.4.4	:MODE TABLE   GRAPh .....	10-7
10.4.5	:RESolution <numeric_value> .....	10-7
10.4.5.1	:UNIT <SUFFIX PROGRAM DATA> .....	10-7
10.4.6	:SPEEd <numeric_value> .....	10-7
10.4.6.1	:UNIT <SUFFIX PROGRAM DATA> .....	10-7
10.5	:FEED <data_handle> .....	10-7
10.6	[:IMMEDIATE] .....	10-7
10.7	:ITEM .....	10-7
10.7.1	:ALL .....	10-8
10.7.1.1	:DATA? .....	10-8
10.7.1.2	[:IMMEDIATE] .....	10-8
10.7.2	:ANNotation .....	10-8
10.7.2.1	:COLor <numeric_value> .....	10-8
10.7.2.2	:DATA? .....	10-8
10.7.2.3	[:IMMEDIATE] .....	10-8
10.7.2.4	:STATe <Boolean> .....	10-8
10.7.3	:CUT .....	10-8
10.7.3.1	:DATA? .....	10-9

## 1999 SCPI Command Reference

10.7.3.2	[ <b>:IMMEDIATE</b> ] . . . . .	10-9
10.7.3.3	<b>:STATe</b> <Boolean> . . . . .	10-9
10.7.4	<b>:FFEed</b> . . . . .	10-9
10.7.4.1	<b>:DATA?</b> . . . . .	10-9
10.7.4.2	[ <b>:IMMEDIATE</b> ] . . . . .	10-9
10.7.4.3	<b>:STATe</b> <Boolean> . . . . .	10-9
10.7.5	<b>:LABel</b> . . . . .	10-9
10.7.5.1	<b>:COLor</b> <numeric_value> . . . . .	10-9
10.7.5.2	<b>:DATA?</b> . . . . .	10-10
10.7.5.3	[ <b>:IMMEDIATE</b> ] . . . . .	10-10
10.7.5.4	<b>:STATe</b> <Boolean> . . . . .	10-10
10.7.5.5	<b>:TEXT</b> <string> . . . . .	10-10
10.7.6	<b>:MENU</b> . . . . .	10-10
10.7.6.1	<b>:COLor</b> <numeric_value> . . . . .	10-10
10.7.6.2	<b>:DATA?</b> . . . . .	10-10
10.7.6.3	[ <b>:IMMEDIATE</b> ] . . . . .	10-10
10.7.6.4	<b>:STATe</b> <Boolean> . . . . .	10-10
10.7.7	<b>:TDSTamp</b> . . . . .	10-11
10.7.7.1	<b>:COLor</b> <numeric_value> . . . . .	10-11
10.7.7.2	<b>:DATA?</b> . . . . .	10-11
10.7.7.3	[ <b>:IMMEDIATE</b> ] . . . . .	10-11
10.7.7.4	<b>:STATe</b> <Boolean> . . . . .	10-11
10.7.8	[ <b>:WINDOW</b> ] . . . . .	10-11
10.7.8.1	<b>:DATA?</b> . . . . .	10-11
10.7.8.2	[ <b>:IMMEDIATE</b> ] . . . . .	10-11
10.7.8.3	<b>:STATe</b> <Boolean> . . . . .	10-11
10.7.8.4	<b>:TEXT</b> . . . . .	10-11
10.7.8.4.1	<b>:COLor</b> <numeric_value> . . . . .	10-12
10.7.8.4.2	<b>:DATA?</b> . . . . .	10-12
10.7.8.4.3	[ <b>:IMMEDIATE</b> ] . . . . .	10-12
10.7.8.4.4	<b>:STATe</b> <Boolean> . . . . .	10-12
10.7.8.5	<b>:TRACe</b> . . . . .	10-12
10.7.8.5.1	<b>:COLor</b> <numeric_value> . . . . .	10-12
10.7.8.5.2	<b>:DATA?</b> . . . . .	10-12
10.7.8.5.3	<b>:GRATicule</b> . . . . .	10-12
10.7.8.5.3.1	<b>:COLor</b> <numeric_value> . . . . .	10-12
10.7.8.5.3.2	<b>:DATA?</b> . . . . .	10-12
10.7.8.5.3.3	[ <b>:IMMEDIATE</b> ] . . . . .	10-13
10.7.8.5.3.4	<b>:STATe</b> <Boolean> . . . . .	10-13
10.7.8.5.4	[ <b>:IMMEDIATE</b> ] . . . . .	10-13
10.7.8.5.5	<b>:LTYPe SOLid   DOTted   DASHed   STYLLe&lt;n&gt;</b> . . . . .	10-13
10.7.8.5.6	<b>:STATe</b> <Boolean> . . . . .	10-13
10.8	<b>:PAGE</b> . . . . .	10-13
10.8.1	<b>:DIMensions</b> . . . . .	10-13
10.8.1.1	<b>:AUTO</b> <Boolean> . . . . .	10-13

## 1999 SCPI Command Reference

10.8.1.2	:LLEFt <numeric_value>,<numeric_value> .....	10-14
10.8.1.3	:QUADrant[<n>] .....	10-14
10.8.1.4	:URIGht <numeric_value>,<numeric_value> .....	10-14
10.8.2	:LENGth <numeric_value> .....	10-14
10.8.3	:ORIentation LANDscape   PORTrait .....	10-14
10.8.4	:SCALe <numeric_value> .....	10-14
10.8.5	:SIZE CUSTom A B C D E A0 A1 A2 A3 A4 B0 B1 B2 B3 B4 B5 .....	10-15
10.8.6	:UNIT <SUFFIX PROGRAM DATA> .....	10-15
10.8.7	:WIDTh <numeric_value> .....	10-15
10.9	:SDUMp .....	10-15
10.9.1	:DATA? .....	10-15
10.9.2	[:IMMEDIATE] .....	10-15

## Chapter 11 INPut Subsystem

11.1	:ATTenuation <numeric_value> .....	11-3
11.1.1	:AUTO <Boolean> ONCE .....	11-3
11.1.2	:STATe <Boolean> .....	11-3
11.2	:BIAS .....	11-3
11.2.1	:CURRent .....	11-3
11.2.1.1	:AC <numeric_value> .....	11-4
11.2.1.2	[:DC] <numeric_value> .....	11-4
11.2.2	[:STATE] <Boolean> .....	11-4
11.2.3	:TYPE CURRent   VOLTage .....	11-4
11.2.4	:VOLTage .....	11-4
11.2.4.1	:AC <numeric_value> .....	11-4
11.2.4.2	[:DC] <numeric_value> .....	11-4
11.3	:COUPLing AC DC GROund .....	11-4
11.4	:FILTer .....	11-5
11.4.1	:AWEighting .....	11-5
11.4.1.1	[:STATE] <Boolean> .....	11-5
11.4.2	:HPASs .....	11-5
11.4.2.1	:FREQuency <numeric_value> .....	11-5
11.4.2.2	[:STATE] <Boolean> .....	11-5
11.4.3	[:LPASs] .....	11-5
11.4.3.1	:FREQuency <numeric_value> .....	11-5
11.4.3.2	[:STATE] <Boolean> .....	11-5
11.5	:GAIN <numeric_value> .....	11-6
11.5.1	:AUTO <Boolean> ONCE .....	11-6
11.5.2	:STATe <Boolean> .....	11-6
11.6	:GUARD LOW FLOat .....	11-6
11.7	:IMPedance <numeric_value> .....	11-6
11.8	:LOW FLOat GROund .....	11-6
11.9	:OFFSet <numeric_value> .....	11-7
11.9.1	[:STATE] <Boolean> .....	11-7

## 1999 SCPI Command Reference

11.10	:POLarity NORMAl INVerted .....	11-7
11.11	:POLarization <numeric_value> .....	11-7
11.11.1	:HORizontal .....	11-7
11.11.2	:VERTical .....	11-7
11.12	:POSIon .....	11-7
11.12.1	[X] .....	11-8
11.12.1.1	:ANGLE .....	11-8
11.12.1.1.1	:DIRection UP DOWN .....	11-8
11.12.1.1.2	[:IMMediate] <numeric_value> .....	11-8
11.12.1.1.3	:LIMit .....	11-8
11.12.1.1.3.1	:HIGH <numeric_value> .....	11-8
11.12.1.1.3.2	:LOW <numeric_value> .....	11-8
11.12.1.1.3.3	:STATe <Boolean> .....	11-8
11.12.1.1.4	:OFFSet <numeric_value> .....	11-8
11.12.1.1.5	:VELOCITY <numeric_value> .....	11-9
11.12.1.2	[:DISTance] .....	11-9
11.12.1.2.1	:DIRection UP DOWN .....	11-9
11.12.1.2.2	[:IMMediate] <numeric_value> .....	11-9
11.12.1.2.3	:LIMit .....	11-9
11.12.1.2.3.1	:HIGH <numeric_value> .....	11-9
11.12.1.2.3.2	:LOW <numeric_value> .....	11-9
11.12.1.2.3.3	:STATe <Boolean> .....	11-9
11.12.1.2.4	:OFFSet <numeric_value> .....	11-9
11.12.1.2.5	:VELOCITY <numeric_value> .....	11-10
11.12.2	:Y .....	11-10
11.12.2.1	:ANGLE .....	11-10
11.12.2.1.1	:DIRection UP DOWN .....	11-10
11.12.2.1.2	[:IMMediate] <numeric_value> .....	11-10
11.12.2.1.3	:LIMit .....	11-10
11.12.2.1.3.1	:HIGH <numeric_value> .....	11-10
11.12.2.1.3.2	:LOW <numeric_value> .....	11-10
11.12.2.1.3.3	:STATe <Boolean> .....	11-10
11.12.2.1.4	:OFFSet <numeric_value> .....	11-10
11.12.2.1.5	:VELOCITY <numeric_value> .....	11-11
11.12.2.2	[:DISTance] .....	11-11
11.12.2.2.1	:DIRection UP DOWN .....	11-11
11.12.2.2.2	[:IMMediate] <numeric_value> .....	11-11
11.12.2.2.3	:LIMit .....	11-11
11.12.2.2.3.1	:HIGH <numeric_value> .....	11-11
11.12.2.2.3.2	:LOW <numeric_value> .....	11-11
11.12.2.2.3.3	:STATe <Boolean> .....	11-11
11.12.2.2.4	:OFFSet <numeric_value> .....	11-11
11.12.2.2.5	:VELOCITY <numeric_value> .....	11-12
11.12.3	:Z .....	11-12
11.12.3.1	:ANGLE .....	11-12

## 1999 SCPI Command Reference

11.12.3.1.1	:DIRection UP DOWN .....	11-12
11.12.3.1.2	[::IMMediate] <numeric_value> .....	11-12
11.12.3.1.3	:LIMit .....	11-12
11.12.3.1.3.1	:HIGH <numeric_value> .....	11-12
11.12.3.1.3.2	:LOW <numeric_value> .....	11-12
11.12.3.1.3.3	:STATe <Boolean> .....	11-12
11.12.3.1.4	:OFFSet <numeric_value> .....	11-12
11.12.3.1.5	:VELOCITY <numeric_value> .....	11-13
11.12.3.2	[::DISTance] .....	11-13
11.12.3.2.1	:DIRection UP DOWN .....	11-13
11.12.3.2.2	[::IMMediate] <numeric_value> .....	11-13
11.12.3.2.3	:LIMit .....	11-13
11.12.3.2.3.1	:HIGH <numeric_value> .....	11-13
11.12.3.2.3.2	:LOW <numeric_value> .....	11-13
11.12.3.2.3.3	:STATe <Boolean> .....	11-13
11.12.3.2.4	:OFFSet <numeric_value> .....	11-13
11.12.3.2.5	:VELOCITY <numeric_value> .....	11-14
11.13	[::STATe] <Boolean> .....	11-14
11.14	:TYPE <character data> .....	11-14

## Chapter 12 INSTRument Subsystem

12.1	:CATalog? .....	12-1
12.1.1	:FULL? .....	12-1
12.2	:COUPle[:<subsystem>] ALL NONE <list> .....	12-2
12.3	:DEFine .....	12-2
12.3.1	:GROup <identifier>,<identifier_list> .....	12-2
12.3.2	[::NAME] <identifier>,<numeric_value> .....	12-3
12.4	:DElete .....	12-3
12.4.1	:ALL .....	12-3
12.4.2	[::NAME] <identifier> .....	12-3
12.5	:NSElect <numeric_value> .....	12-4
12.6	[::SElect] <identifier> .....	12-4
12.7	:STATe <Boolean> .....	12-4

## Chapter 13 MEMory Subsystem

13.1	:CATalog .....	13-4
13.1.1	[::ALL]? .....	13-5
13.1.2	:ASCii? .....	13-5
13.1.3	:BINary? .....	13-5
13.1.4	:MACRo? .....	13-5
13.1.5	:STATe? .....	13-5
13.1.6	:TABLE? .....	13-6
13.2	:CLEar .....	13-6
13.2.1	[::NAME] <name> .....	13-6

## 1999 SCPI Command Reference

13.2.2	:TABLE .....	13-6
13.3	:COPY .....	13-6
13.3.1	[:NAME] <name>,<name> .....	13-6
13.3.2	:TABLE <name> .....	13-7
13.4	:DATA<name>,<data> .....	13-7
13.5	:DELetE .....	13-7
13.5.1	:ALL .....	13-7
13.5.2	[:NAME] <name> .....	13-7
13.6	:EXCHange .....	13-7
13.6.1	[:NAME] <name>,<name> .....	13-7
13.6.2	:TABLE <name> .....	13-8
13.7	:FREE .....	13-8
13.7.1	[:ALL]? .....	13-8
13.7.2	:ASCii? .....	13-8
13.7.3	:BINary? .....	13-8
13.7.4	:MACRo? .....	13-8
13.7.5	:STATe? .....	13-9
13.7.6	:TABLE? .....	13-9
13.8	:NSTates? .....	13-9
13.9	:STATe .....	13-9
13.9.1	:CATalog? .....	13-9
13.9.2	:DEFIne <name> ,<register_number> .....	13-9
13.10	:TYPE? <name> .....	13-9
13.11	:TABLE .....	13-10
13.11.1	:BNUMber <numeric_value> {,<numeric_value>} .....	13-10
13.11.1.1	:POINts? .....	13-10
13.11.2	:CCURve <numeric_value> {,<numeric_value>} .....	13-10
13.11.2.1	:POINts? .....	13-10
13.11.3	:CONCetration <numeric_value> {,<numeric_value>} .....	13-10
13.11.3.1	:POINts? .....	13-10
13.11.4	:CONDition .....	13-10
13.11.4.1	[:MAGNitude] <Boolean>{,<Boolean>} .....	13-11
13.11.4.1.1	:POINts? .....	13-11
13.11.5	:CPOint <numeric_value> {,<numeric_value>} .....	13-11
13.11.5.1	:POINts? .....	13-11
13.11.6	:CURRent .....	13-11
13.11.6.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	13-11
13.11.6.1.1	:POINts? .....	13-11
13.11.6.2	:PHASe <numeric_value>{,<numeric_value>} .....	13-11
13.11.6.2.1	:POINts? .....	13-12
13.11.7	:DFACtory <numeric_value> {,<numeric_value>} .....	13-12
13.11.7.1	:POINts? .....	13-12
13.11.8	:DLASt <numeric_value> {,<numeric_value>} .....	13-12
13.11.8.1	:POINts? .....	13-13
13.11.9	:DLINearize <numeric_value> {,<numeric_value>} .....	13-13

## 1999 SCPI Command Reference

13.11.9.1	:POINts?	13-13
13.11.10	:EXPected <numeric_value> {,<numeric_value>}	13-13
13.11.10.1	:POINts?	13-14
13.11.11	:DEFIne <structure_string>[,<numeric_value>]	13-14
13.11.12	:FORCe	13-14
13.11.12.1	[:MAGNitude] <numeric_value>{,<numeric_value>}	13-14
13.11.12.1.1	:POINts?	13-14
13.11.13	:FREQuency <numeric_value>{,<numeric_value>}	13-15
13.11.13.1	:POINts?	13-15
13.11.14	:LABel <string> {,<string>}	13-15
13.11.14.1	:POINts?	13-15
13.11.15	:LLIMit <numeric_value> {,<numeric_value>}	13-15
13.11.15.1	:POINts?	13-15
13.11.16	:LOG <string> {,<string>}	13-15
13.11.16.1	:POINts?	13-16
13.11.17	:LOSS	13-16
13.11.17.1	[:MAGNitude] <numeric_value>{,<numeric_value>}	13-16
13.11.17.1.1	:POINts?	13-16
13.11.17.2	:PHASe <numeric_value>{,<numeric_value>}	13-16
13.11.17.2.1	:POINts?	13-16
13.11.18	:NCURve <numeric_value> {,<numeric_value>}	13-16
13.11.18.1	:POINts?	13-16
13.11.19	:POWER	13-17
13.11.19.1	[:MAGNitude] <numeric_value>{,<numeric_value>}	13-17
13.11.19.1.1	:POINts?	13-17
13.11.20	:RAW <numeric_value> {,<numeric_value>}	13-17
13.11.20.1	:POINts?	13-17
13.11.21	:RESistance	13-17
13.11.21.1	[:MAGNitude] <numeric_value>{,<numeric_value>}	13-17
13.11.21.1.1	:POINts?	13-17
13.11.22	:SElect <name>	13-18
13.11.23	:SPEEd	13-18
13.11.23.1	[:MAGNitude] <numeric_value>{,<numeric_value>}	13-18
13.11.23.1.1	:POINts?	13-18
13.11.24	:TIME	13-18
13.11.24.1	[:MAGNitude] <numeric_value>{,<numeric_value>}	13-18
13.11.24.1.1	:POINts?	13-18
13.11.25	:TOLERance <numeric_value> {,<numeric_value>}	13-18
13.11.25.1	:POINts?	13-19
13.11.26	:ULIMit <numeric_value> {,<numeric_value>}	13-19
13.11.26.1	:POINts?	13-19
13.11.27	:VOLTage	13-19
13.11.27.1	[:MAGNitude] <numeric_value>{,<numeric_value>}	13-19
13.11.27.1.1	:POINts?	13-19
13.11.27.2	:PHASe <numeric_value>{,<numeric_value>}	13-19

## 1999 SCPI Command Reference

13.11.27.2.1	:POINTs?	.....	13-20
13.11.28	:WFACtor <numeric_value> {,<numeric_value>}	.....	13-20
13.11.28.1	:POINTs?	.....	13-20

## Chapter 14 MMEMory Subsystem

14.1	:CATalog? [<msus>]	.....	14-2
14.2	:CDIRectory [<directory_name>]	.....	14-3
14.3	:CLOSE	.....	14-3
14.4	:COPY <file_source>,<file_destination>	.....	14-3
14.5	:DATA <file_name>,<data>	.....	14-3
14.6	:DElete <file_name>[,<msus>]	.....	14-3
14.7	:FEED <data_handle>	.....	14-4
14.8	:INITialize [<msus>[, (LIF DOS HFS)[,<numeric_value>]]]	.....	14-4
14.9	:LOAD and :STORe	.....	14-4
14.9.1	:DINTerchange <label>,<file_name>[,<msus>]	.....	14-4
14.9.1.1	:TRACe <label>,<file_name>[,<msus>]	.....	14-4
14.9.2	:MACRo <label>,<file_name>[,<msus>]	.....	14-4
14.9.3	:STATE <numeric_value>,<file_name>[,<msus>]	.....	14-4
14.9.4	:TABLE <label>,<file_name>[,<msus>]	.....	14-5
14.9.5	:TRACe <label>,<file_name>[,<msus>]	.....	14-5
14.10	:MSIS [<msus>]	.....	14-5
14.11	:MOVE (<src_file>,<dest_file>)  (<src_file>,<src_msus>,<dest_file>,<dest_msus>)	.....	14-5
14.12	:NAME <file_name>[,<msus>]	.....	14-5
14.13	:OPEN	.....	14-6
14.14	:PACK [<msus>]	.....	14-6

## Chapter 15 OUTPut Subsystem

15.1	:ATTenuation <numeric_value>	.....	15-3
15.2	:COUpling AC DC	.....	15-3
15.3	:FILTer	.....	15-3
15.3.1	:AUTO <Boolean>   ONCE	.....	15-3
15.3.2	:EXTernal	.....	15-3
15.3.2.1	[:STATE] <Boolean>	.....	15-4
15.3.3	:HPASs	.....	15-4
15.3.3.1	:FREQuency <numeric_value>	.....	15-4
15.3.3.2	[:STATE] <Boolean>	.....	15-4
15.3.3.3	:TYPE BESSel   CHEByShev	.....	15-4
15.3.4	[:LPASs]	.....	15-4
15.3.4.1	:FREQuency <numeric_value>	.....	15-4
15.3.4.2	[:STATE] <Boolean>	.....	15-4
15.3.4.3	:TYPE BESSel   CHEByShev	.....	15-4
15.4	:IMPedance <numeric_value>	.....	15-5
15.5	:LOW FLOat GROund	.....	15-5

## 1999 SCPI Command Reference

15.6	:POLarity NORMAl   INVerted .....	15-5
15.7	:POLArization <numeric_value> .....	15-5
15.7.1	:HORizontal .....	15-5
15.7.2	:VERTical .....	15-5
15.8	:POSIon .....	15-5
15.8.1	[:X] .....	15-5
15.8.1.1	:ANGLE .....	15-6
15.8.1.1.1	:DIRection UP DOWN .....	15-6
15.8.1.1.2	[:IMMEDIATE] <numeric_value> .....	15-6
15.8.1.1.3	:LIMit .....	15-6
15.8.1.1.3.1	:HIGH <numeric_value> .....	15-6
15.8.1.1.3.2	:LOW <numeric_value> .....	15-6
15.8.1.1.3.3	:STATE <Boolean> .....	15-6
15.8.1.1.4	:OFFSet <numeric_value> .....	15-6
15.8.1.1.5	:VELOCITY <numeric_value> .....	15-6
15.8.1.2	[:DISTance] .....	15-7
15.8.1.2.1	:DIRection UP DOWN .....	15-7
15.8.1.2.2	[:IMMEDIATE] <numeric_value> .....	15-7
15.8.1.2.3	:LIMit .....	15-7
15.8.1.2.3.1	:HIGH <numeric_value> .....	15-7
15.8.1.2.3.2	:LOW <numeric_value> .....	15-7
15.8.1.2.3.3	:STATE <Boolean> .....	15-7
15.8.1.2.4	:OFFSet <numeric_value> .....	15-7
15.8.1.2.5	:VELOCITY <numeric_value> .....	15-7
15.8.2	:Y .....	15-8
15.8.2.1	:ANGLE .....	15-8
15.8.2.1.1	:DIRection UP DOWN .....	15-8
15.8.2.1.2	[:IMMEDIATE] <numeric_value> .....	15-8
15.8.2.1.3	:LIMit .....	15-8
15.8.2.1.3.1	:HIGH <numeric_value> .....	15-8
15.8.2.1.3.2	:LOW <numeric_value> .....	15-8
15.8.2.1.3.3	:STATE <Boolean> .....	15-8
15.8.2.1.4	:OFFSet <numeric_value> .....	15-8
15.8.2.1.5	:VELOCITY <numeric_value> .....	15-8
15.8.2.2	[:DISTance] .....	15-9
15.8.2.2.1	:DIRection UP DOWN .....	15-9
15.8.2.2.2	[:IMMEDIATE] <numeric_value> .....	15-9
15.8.2.2.3	:LIMit .....	15-9
15.8.2.2.3.1	:HIGH <numeric_value> .....	15-9
15.8.2.2.3.2	:LOW <numeric_value> .....	15-9
15.8.2.2.3.3	:STATE <Boolean> .....	15-9
15.8.2.2.4	:OFFSet <numeric_value> .....	15-9
15.8.2.2.5	:VELOCITY <numeric_value> .....	15-9
15.8.3	:Z .....	15-10
15.8.3.1	:ANGLE .....	15-10

## 1999 SCPI Command Reference

15.8.3.1.1	:DIRection UP DOWN .....	15-10
15.8.3.1.2	[::IMMediate] <numeric_value> .....	15-10
15.8.3.1.3	:LIMit .....	15-10
15.8.3.1.3.1	:HIGH <numeric_value> .....	15-10
15.8.3.1.3.2	:LOW <numeric_value> .....	15-10
15.8.3.1.3.3	:STATe <Boolean> .....	15-10
15.8.3.1.4	:OFFSet <numeric_value> .....	15-10
15.8.3.1.5	:VELOCITY <numeric_value> .....	15-10
15.8.3.2	[::DISTance] .....	15-11
15.8.3.2.1	:DIRection UP DOWN .....	15-11
15.8.3.2.2	[::IMMediate] <numeric_value> .....	15-11
15.8.3.2.3	:LIMit .....	15-11
15.8.3.2.3.1	:HIGH <numeric_value> .....	15-11
15.8.3.2.3.2	:LOW <numeric_value> .....	15-11
15.8.3.2.3.3	:STATe <Boolean> .....	15-11
15.8.3.2.4	:OFFSet <numeric_value> .....	15-11
15.8.3.2.5	:VELOCITY <numeric_value> .....	15-11
15.9	::PROTection .....	15-12
15.9.1	:DELay <numeric_value> .....	15-12
15.9.2	[::STATe] <Boolean> .....	15-12
15.9.3	:TRIPped? .....	15-12
15.9.4	:CLEar .....	15-12
15.10	::ROSCillator .....	15-12
15.10.1	[::STATe] <Boolean> .....	15-12
15.11	::TTLTrg<n> ::ECLTrg<n> .....	15-12
15.11.1	::IMMediate .....	15-12
15.11.2	:LEVel <Boolean> .....	15-13
15.11.3	:POLarity NORMAL INVerted .....	15-13
15.11.4	:PROTocol SYNChronous SSYNchronous ASYNchronous .....	15-13
15.11.5	:WIDTh <numeric_value> .....	15-13
15.11.6	[::STATe] <Boolean> .....	15-13
15.11.7	:SOURce <character data> .....	15-13
15.12	[::STATe] <Boolean> .....	15-14
15.13	::TYPE <character data> .....	15-14

## Chapter 16 PROGram Subsystem

16.1	::CATalog? .....	16-2
16.2	[::SELected] .....	16-2
16.2.1	::DEFine <program> .....	16-2
16.2.2	::DELetE .....	16-2
16.2.2.1	[::SELected] .....	16-2
16.2.2.2	::ALL .....	16-2
16.2.3	::EXECute <program_command> .....	16-2
16.2.4	::MALLOCate <nbytes> DEFault .....	16-3

## 1999 SCPI Command Reference

16.2.5	:NAME <progname> .....	16-3
16.2.6	:NUMBER <varname>{,<nvalues>} .....	16-3
16.2.7	:STATe RUN PAUSE STOP CONTinue .....	16-3
16.2.8	:STRing <varname>{,<svalues>} .....	16-4
16.2.9	:WAIT .....	16-4
16.3	:EXPLicit .....	16-4
16.3.1	:DEFIne <progname>,<program> .....	16-4
16.3.2	:DELete <progname> .....	16-5
16.3.3	:EXECute <progname>,<program_command> .....	16-5
16.3.4	:MALLocate <progname>,(<nbytes> DEFault) .....	16-5
16.3.5	:NUMBER <progname>,<varname>{,<nvalues>} .....	16-5
16.3.6	:STATe <progname>,(RUN PAUSE STOP CONTinue) .....	16-5
16.3.7	:STRing <progname>,<varname>{,<svalues>} .....	16-6
16.3.8	:WAIT <progname> .....	16-6

## Chapter 17 ROUTe Subsystem

17.1	:CLOSe <channel_list> .....	17-1
17.1.1	:STATe? .....	17-2
17.2	:MODule .....	17-2
17.2.1	:CATalog? .....	17-2
17.2.2	[:DEFIne] <module_name>,<module_address> .....	17-2
17.2.3	:DELete .....	17-2
17.2.3.1	:ALL .....	17-2
17.2.3.2	[:NAME] <module_name> .....	17-3
17.3	:OPEN <channel_list> .....	17-3
17.3.1	:ALL .....	17-3
17.4	:PATH .....	17-3
17.4.1	:CATalog? .....	17-3
17.4.2	[:DEFIne] <path_name>,<channel_list> .....	17-3
17.4.3	:DELete .....	17-4
17.4.3.1	:ALL .....	17-4
17.4.3.2	[:NAME] <path_name> .....	17-4
17.5	:SAMPlE .....	17-4
17.5.1	:CATalog? .....	17-4
17.5.2	[:OPEN] BAG DILute PRE POST MID CEFFiciency  NONE ZERO SPAN VERify MANifold .....	17-4
17.6	:SCAN <channel_list> .....	17-6
17.7	TERMinals FRONt REAR BOTH NONE .....	17-6

## Chapter 18 SENSe Subsystem

18.1	AM Subsystem .....	18-4
18.1.1	[:DEPTH] .....	18-4
18.1.1.1	:RANGe .....	18-4
18.1.1.1.1	:AUTO <Boolean>   ONCE .....	18-4

## 1999 SCPI Command Reference

18.1.1.1.2	[ <b>:UPPer</b> ] <numeric_value>	18-4
18.1.1.3	<b>:LOWer</b> <numeric_value>	18-4
18.1.2	<b>:TYPE</b> LINear LOGarithmic	18-4
18.2	<b>AVERage</b> Subsystem	18-6
18.2.1	A typical device action for SENS: <b>AVERage</b>	18-6
18.2.2	<b>:COUNt</b> <numeric_value>	18-7
18.2.2.1	<b>:AUTO</b> <Boolean> ONCE	18-7
18.2.3	[ <b>:STATE</b> ] <Boolean>	18-7
18.2.4	<b>:TCOntrol</b> EXPonential   MOVing   NORMal   REPeat	18-7
18.2.5	<b>:TYPE</b> COMPlex   ENVelope   MAXimum   MINimum   RMS   SCALar	18-8
18.3	<b>BANDwidth BWIDth</b> Subsystem	18-10
18.3.1	[ <b>:RESolution</b> ] <numeric_value>	18-10
18.3.1.1	<b>:AUTO</b> <Boolean> ONCE	18-10
18.3.1.2	<b>:RATio</b> <numeric_value>	18-10
18.3.1.3	<b>:TRACK</b> <Boolean>	18-10
18.3.2	<b>:VIDeo</b> <numeric_value>	18-10
18.3.2.1	<b>:AUTO</b> <Boolean> ONCE	18-11
18.3.2.2	<b>:RATio</b> <numeric_value>	18-11
18.4	<b>CONCentration</b> Subsystem	18-12
18.4.1	<b>:CSET</b> <numeric_value>,<numeric_value>	18-12
18.4.2	<b>:LOWer</b> <numeric_value>	18-12
18.4.3	<b>:LSET</b> POLYnomial<n>   SRATional<n>,<numeric_value>{,<numeric_value>}	18-12
18.4.4	<b>:RANGe</b>	18-13
18.4.4.1	<b>:AUTO</b>	18-13
18.4.4.1.1	<b>:LOWer</b> <numeric_value>	18-13
18.4.4.1.2	[ <b>:STATE</b> ] <Boolean>	18-13
18.4.4.1.3	<b>:UPPer</b> <numeric_value>	18-13
18.4.4.2	[ <b>:FIXed</b> ] <numeric_value>	18-13
18.4.5	<b>:TALign</b> <numeric_value>	18-13
18.4.6	<b>:UPPer</b> <numeric_value>	18-14
18.5	<b>CONDition</b> Subsystem	18-15
18.5.1	<b>:LEVel</b> <numeric_value>   TTL ECL	18-15
18.6	<b>CORRection</b> Subsystem	18-16
18.6.1	<b>:AUTO</b>	18-17
18.6.2	<b>:CALCulate</b>	18-17
18.6.3	<b>:COLLect</b>	18-17
18.6.3.1	[ <b>:ACQuire</b> ] STANdard	18-17
18.6.3.2	<b>:METHod</b> TPORt	18-18
18.6.3.3	<b>:SAVE</b> [<trace_name>]	18-18
18.6.4	<b>:CSET</b>	18-18
18.6.4.1	[ <b>:SElect</b> ] <name>	18-18
18.6.4.2	<b>:STATE</b> <Boolean>	18-18
18.6.5	<b>:EDELay</b>	18-19
18.6.5.1	<b>:DISTance</b> <numeric_value>	18-19
18.6.5.2	<b>:STATE</b> <Boolean>	18-19

## 1999 SCPI Command Reference

18.6.5.3	[TIME] <numeric_value>	18-19
18.6.6	:IMPedance	18-19
18.6.6.1	[INPut]:OUTPut	18-19
18.6.6.1.1	[MAGNitude] <numeric_value>	18-20
18.6.6.2	:STATe <Boolean>	18-20
18.6.7	:LOSS :GAIN :SLOPe	18-20
18.6.7.1	[INPut]:OUTPut	18-20
18.6.7.1.1	:AUTO ON OFF	18-20
18.6.7.1.2	[MAGNitude] <numeric_value>	18-21
18.6.7.1.3	:PHASe <numeric_value>	18-21
18.6.7.2	:STATe <Boolean>	18-21
18.6.8	:OFFSet	18-21
18.6.8.1	[MAGNitude] <numeric_value>	18-21
18.6.8.2	:PHASe <numeric_value>	18-21
18.6.8.3	:STATe <Boolean>	18-22
18.6.9	:RVELOCITY	18-22
18.6.9.1	:COAX <numeric_value>	18-22
18.6.9.2	:MEDIUM COAX WAVeguide	18-22
18.6.9.3	:STATe <Boolean>	18-22
18.6.9.4	:WAVeguide <numeric_value>	18-22
18.6.9.4.1	:FCUToff <numeric_value>	18-22
18.6.10	:SPOint	18-23
18.6.10.1	:ACQuire	18-23
18.6.10.2	:DTOLerance <numeric_value>	18-23
18.6.11	[STATe] <Boolean>	18-23
18.6.12	:ZERO	18-23
18.6.12.1	:ACQuire	18-23
18.6.12.2	:DTOLerance <numeric_value>	18-23
18.7	CURRent Subsystem	18-25
18.7.1	:AC[:DC]	18-25
18.7.1.1	:APERture <numeric_value>	18-25
18.7.1.2	:NPLCycles <numeric_value>	18-26
18.7.1.3	:ATTenuation <numeric_value>	18-26
18.7.1.3.1	:AUTO <Boolean>	18-26
18.7.1.4	:PROTection	18-26
18.7.1.4.1	[LEVel] <numeric_value>	18-26
18.7.1.4.2	:STATe <Boolean>	18-26
18.7.1.4.3	:TRIPped?	18-26
18.7.1.4.4	:CLEar	18-26
18.7.1.5	:RANGe	18-27
18.7.1.5.1	[UPPer] <numeric_value>	18-27
18.7.1.5.2	:LOWer <numeric_value>	18-27
18.7.1.5.3	:AUTO <Boolean> ONCE	18-27
18.7.1.5.3.1	:DIRection UP DOWN EITHER	18-27
18.7.1.5.3.2	:LLIMit <numeric_value>	18-27

## 1999 SCPI Command Reference

18.7.1.5.3.3	:ULIMit <numeric_value>	18-27
18.7.1.5.4	:OFFSet <numeric_value>	18-28
18.7.1.5.5	:PTPeak <numeric_value>	18-28
18.7.1.6	:REference <numeric_value>	18-28
18.7.1.6.1	:STATe <Boolean>	18-28
18.7.1.7	:RESolution <numeric_value>	18-28
18.7.1.7.1	:AUTO <Boolean> ONCE	18-28
18.7.2	:DETector INTernal   EXTernal	18-29
18.8	DETector Subsystem	18-30
18.8.1	:BANDwidth   BWIDth	18-30
18.8.2	[:FUNCTION] <detector function>	18-30
18.8.2.1	:AUTO <Boolean> ONCE	18-31
18.8.3	:SHAPe LINear LOGarithmic	18-31
18.9	DISTance Subsystem	18-32
18.9.1	:RESET	18-32
18.10	FILTer Subsystem	18-33
18.10.1	[:LPASs]	18-33
18.10.1.1	[:STATe] <Boolean>	18-33
18.10.1.2	:FREQuency <numeric_value>	18-33
18.10.2	:HPASs	18-33
18.10.2.1	[:STATe] <Boolean>	18-34
18.10.2.2	:FREQuency <numeric_value>	18-34
18.10.3	:DEMPhasis	18-34
18.10.3.1	[:STATe] <Boolean>	18-34
18.10.3.2	:TCONstant <numeric_value>	18-34
18.10.4	:CCITt	18-34
18.10.4.1	[:STATe] <Boolean>	18-34
18.10.5	:CMESsage	18-34
18.10.5.1	[:STATe] <Boolean>	18-34
18.10.6	:CCIR	18-35
18.10.6.1	[:STATe] <Boolean>	18-35
18.10.7	:CARM	18-35
18.10.7.1	[:STATe] <Boolean>	18-35
18.10.8	:AWEighting	18-35
18.10.8.1	[:STATe] <Boolean>	18-35
18.11	FM Subsystem	18-36
18.11.1	[:DEViation]	18-36
18.11.1.1	:RANGE	18-36
18.11.1.1.1	:AUTO <Boolean>   ONCE	18-36
18.11.1.1.2	[:UPPer] <numeric_value>	18-36
18.11.1.1.3	:LOWer <numeric_value>	18-36
18.12	FREQuency Subsystem	18-37
18.12.1	:APERture <numeric_value>	18-37
18.12.2	:CENTer <numeric_value>	18-37
18.12.3	[:CW]:FIXed] <numeric_value>	18-38

## 1999 SCPI Command Reference

18.12.3.1	:AFC <Boolean> ONCE . . . . .	18-38
18.12.3.2	:AUTO <Boolean> ONCE . . . . .	18-38
18.12.4	:MANual <numeric_value> . . . . .	18-38
18.12.5	:MODE CW FIXed SWEep LIST SOURce . . . . .	18-38
18.12.6	:MULTiplier <numeric_value> . . . . .	18-39
18.12.7	:OFFSet <numeric_value> . . . . .	18-39
18.12.8	:RANGE . . . . .	18-39
18.12.8.1	[:UPPer] <numeric_value> . . . . .	18-39
18.12.8.2	[:LOWER] <numeric_value> . . . . .	18-39
18.12.8.3	:AUTO <Boolean> ONCE . . . . .	18-39
18.12.9	:RESolution <numeric_value> . . . . .	18-40
18.12.9.1	:AUTO <Boolean> ONCE . . . . .	18-40
18.12.10	:SPAN <numeric_value> . . . . .	18-40
18.12.10.1	:HOLD <Boolean> . . . . .	18-40
18.12.10.2	:LINK CENTer STARt STOP . . . . .	18-40
18.12.10.3	:FULL . . . . .	18-40
18.12.11	:STARt <numeric_value> . . . . .	18-41
18.12.12	:STOP <numeric_value> . . . . .	18-41
18.13	FUNCTION & DATA Subsystem . . . . .	18-42
18.13.1	DATA? [<data_handle>] . . . . .	18-42
18.13.1.1	SENSe <data_handle>s . . . . .	18-42
18.13.1.2	:PREamble? [<data_handle>] . . . . .	18-43
18.13.2	:FUNCTION . . . . .	18-43
18.13.2.1	:CONCurrent <Boolean> . . . . .	18-43
18.13.2.2	:OFF <sensor_function>{,<sensor_function>} . . . . .	18-44
18.13.2.2.1	:ALL . . . . .	18-44
18.13.2.2.2	:COUNt? . . . . .	18-44
18.13.2.3	[:ON] <sensor_function>{,<sensor_function>} . . . . .	18-44
18.13.2.3.1	:ALL . . . . .	18-45
18.13.2.3.2	:COUNt? . . . . .	18-45
18.13.2.4	:STATE? <sensor_function> . . . . .	18-45
18.13.2.5	<sensor_function> . . . . .	18-45
18.13.2.6	<presentation_layer> . . . . .	18-47
18.13.2.6.1	[XNONE:] . . . . .	18-47
18.13.2.6.2	XTIMe: . . . . .	18-47
18.13.2.6.3	XFREquency: . . . . .	18-47
18.13.2.6.4	XPOWER: . . . . .	18-47
18.13.2.6.5	XVOLTage: . . . . .	18-47
18.13.2.6.6	XCURrent: . . . . .	18-47
18.13.2.7	<function_name> . . . . .	18-48
18.13.2.8	<function> . . . . .	18-49
18.13.2.8.1	ACCeleration . . . . .	18-50
18.13.2.8.2	AM . . . . .	18-50
18.13.2.8.2.1	[:DEPTH] . . . . .	18-50
18.13.2.8.2.2	:DISTortion . . . . .	18-50

## 1999 SCPI Command Reference

18.13.2.8.2.3	:FREQuency .....	18-51
18.13.2.8.2.4	:SNDRatio .....	18-51
18.13.2.8.2.5	:SNR .....	18-51
18.13.2.8.2.6	:THD .....	18-51
18.13.2.8.3	CONCcentration .....	18-51
18.13.2.8.3.1	:RAW .....	18-51
18.13.2.8.3.2	:SDEViation .....	18-51
18.13.2.8.3.3	:TALign .....	18-51
18.13.2.8.4	CONDITION .....	18-51
18.13.2.8.5	CURRent .....	18-51
18.13.2.8.5.1	[:DC] .....	18-51
18.13.2.8.5.2	:AC .....	18-51
18.13.2.8.6	DISTance .....	18-52
18.13.2.8.7	FM .....	18-52
18.13.2.8.7.1	:[DEViation] .....	18-52
18.13.2.8.7.2	:DISTortion .....	18-52
18.13.2.8.7.3	:FREQuency .....	18-52
18.13.2.8.7.4	:SNDRatio .....	18-52
18.13.2.8.7.5	:SNR .....	18-52
18.13.2.8.7.6	:THD .....	18-52
18.13.2.8.8	FERRor .....	18-52
18.13.2.8.9	FORCe .....	18-52
18.13.2.8.10	FREQuency .....	18-52
18.13.2.8.11	FRESistance .....	18-53
18.13.2.8.12	PERiod .....	18-53
18.13.2.8.13	PHASe .....	18-53
18.13.2.8.14	PM .....	18-53
18.13.2.8.14.1	:[DEViation] .....	18-53
18.13.2.8.14.2	:DISTortion .....	18-53
18.13.2.8.14.3	:FREQuency .....	18-53
18.13.2.8.14.4	:SNDRatio .....	18-53
18.13.2.8.14.5	:SNR .....	18-53
18.13.2.8.14.6	:THD .....	18-53
18.13.2.8.15	POWer .....	18-53
18.13.2.8.15.1	:AC .....	18-54
18.13.2.8.15.2	:ACHannel .....	18-54
18.13.2.8.15.3	:COHerence .....	18-54
18.13.2.8.15.4	:CROSSs .....	18-54
18.13.2.8.15.5	[:DC] .....	18-54
18.13.2.8.15.6	:DISTortion .....	18-54
18.13.2.8.15.7	:PSDensity .....	18-54
18.13.2.8.15.8	:S11;S12;S22;S21 .....	18-54
18.13.2.8.15.9	:SNDRatio .....	18-55
18.13.2.8.15.10	:SNR .....	18-55
18.13.2.8.15.11	:THD .....	18-55

## 1999 SCPI Command Reference

18.13.2.8.16	PULM . . . . .	18-55
18.13.2.8.17	RESistance . . . . .	18-55
18.13.2.8.18	SPEed . . . . .	18-55
18.13.2.8.18.1	:FRONt . . . . .	18-55
18.13.2.8.18.2	[:REAR] . . . . .	18-55
18.13.2.8.19	SSB . . . . .	18-56
18.13.2.8.20	TEMPerature . . . . .	18-56
18.13.2.8.21	TIMer . . . . .	18-56
18.13.2.8.21.1	COUNt . . . . .	18-56
18.13.2.8.22	TINTerval . . . . .	18-56
18.13.2.8.23	TOTalize . . . . .	18-56
18.13.2.8.24	TPLoss . . . . .	18-56
18.13.2.8.25	VOLTage . . . . .	18-56
18.13.2.8.25.1	:AC . . . . .	18-56
18.13.2.8.25.2	:CDFunction . . . . .	18-56
18.13.2.8.25.3	[:DC] . . . . .	18-56
18.13.2.8.25.4	:HISTogram . . . . .	18-56
18.13.2.8.25.5	:PDFunction . . . . .	18-57
18.14	LIST Subsystem . . . . .	18-58
18.14.1	:COUNt <numeric_value> . . . . .	18-58
18.14.2	:DIRection UP DOWN . . . . .	18-58
18.14.3	:DWEI<numeric_value>{,<numeric_value>} . . . . .	18-58
18.14.3.1	:POINts? . . . . .	18-58
18.14.4	:FREQuency <numeric_value>{,<numeric_value>} . . . . .	18-59
18.14.4.1	:POINts? . . . . .	18-59
18.14.5	:SEQUence <numeric_value>{,<numeric_value>} . . . . .	18-59
18.14.5.1	:AUTO <Boolean> ONCE . . . . .	18-59
18.14.5.2	:POINts? . . . . .	18-59
18.15	MIXer Subsystem . . . . .	18-60
18.15.1	:BIAS <numeric_value> . . . . .	18-60
18.15.1.1	:AUTO <Boolean> ONCE . . . . .	18-60
18.15.1.2	:LIMit <numeric_value> . . . . .	18-60
18.15.2	:HARMonic <numeric_value> . . . . .	18-60
18.15.2.1	:AUTO <Boolean> ONCE . . . . .	18-60
18.15.3	:LOSS <numeric_value> . . . . .	18-61
18.15.3.1	:AUTO <Boolean> . . . . .	18-61
18.16	PM Subsystem . . . . .	18-62
18.16.1	[:DEViation] . . . . .	18-62
18.16.1.1	:RANGe . . . . .	18-62
18.16.1.1.1	:AUTO <Boolean>   ONCE . . . . .	18-62
18.16.1.1.2	[:UPPer]<numeric_value> . . . . .	18-62
18.16.1.1.3	:LOWER<numeric_value> . . . . .	18-62
18.17	POWER Subsystem . . . . .	18-63
18.17.1	:ACHannel . . . . .	18-63
18.17.1.1	:SPACing . . . . .	18-63

## 1999 SCPI Command Reference

18.17.1.1.1	:LOWER <numeric_value>	18-64
18.17.1.1.1.1	:AUTO <Boolean>	18-64
18.17.1.1.2	[:UPPer] <numeric_value>	18-64
18.17.2	:AC[:DC]	18-64
18.17.2.1	:APERture <numeric_value>	18-64
18.17.2.2	:NPLCycles <numeric_value>	18-64
18.17.2.3	:ATTenuation <numeric_value>	18-64
18.17.2.3.1	:AUTO <Boolean>	18-65
18.17.2.4	:PROTection	18-65
18.17.2.4.1	[:LEVel] <numeric_value>	18-65
18.17.2.4.2	:STATe <Boolean>	18-65
18.17.2.4.3	:TRIPped?	18-65
18.17.2.4.4	:CLEar	18-65
18.17.2.5	:RANGE	18-65
18.17.2.5.1	[:UPPer] <numeric_value>	18-65
18.17.2.5.2	:LOWER <numeric_value>	18-66
18.17.2.5.3	:AUTO <Boolean> ONCE	18-66
18.17.2.5.3.1	:DIRection UP DOWN EITHER	18-66
18.17.2.5.3.2	:LLIMit <numeric_value>	18-66
18.17.2.5.3.3	:ULIMit <numeric_value>	18-66
18.17.2.5.4	:OFFSet <numeric_value>	18-66
18.17.2.5.5	:PTPeak <numeric_value>	18-66
18.17.2.6	:REFerence <numeric_value>	18-67
18.17.2.6.1	:STATe <Boolean>	18-67
18.17.2.7	:RESolution <numeric_value>	18-67
18.17.2.7.1	:AUTO <Boolean> ONCE	18-67
18.17.3	:DETector INTERNAL   EXTERNAL	18-67
18.18	RESistance FRESistance Subsystem	18-68
18.18.1	:APERture <numeric_value>	18-68
18.18.2	:NPLCycles <numeric_value>	18-68
18.18.3	:OCOMPensated <Boolean>	18-68
18.18.4	:RANGE	18-69
18.18.4.1	[:UPPer] <numeric_value>	18-69
18.18.4.2	:LOWER <numeric_value>	18-69
18.18.4.3	:AUTO <Boolean> ONCE	18-69
18.18.4.3.1	:DIRection UP DOWN EITHER	18-69
18.18.4.3.2	:LLIMit <numeric_value>	18-69
18.18.4.3.3	:ULIMit <numeric_value>	18-69
18.18.5	:REFERENCE <numeric_value>	18-70
18.18.5.1	:STATe <Boolean>	18-70
18.18.6	:RESolution <numeric_value>	18-70
18.18.6.1	:AUTO <Boolean> ONCE	18-70
18.19	ROSCillator Subsystem	18-71
18.19.1	[:INTERNAL]	18-71
18.19.1.1	:FREQuency <numeric_value>	18-71

## 1999 SCPI Command Reference

18.19.2	:EXTernal .....	18-71
18.19.2.1	:FREQuency <numeric_value> .....	18-71
18.19.3	:SOURce INTernal EXTernal NONE CLK10 CLK100 .....	18-71
18.19.3.1	:AUTO <Boolean> ONCE .....	18-72
18.20	SMOoothing Subsystem .....	18-73
18.20.1	[:STATE] <Boolean> .....	18-73
18.20.2	:APERture <numeric_value> .....	18-73
18.20.3	:POINts <numeric_value> .....	18-73
18.21	SSB Subsystem .....	18-74
18.21.1	:TYPE USB LSB A1 .....	18-74
18.22	STABilize Subsystem .....	18-75
18.22.1	:NTOLerance <numeric_value> .....	18-75
18.22.2	[:STATE] <Boolean> .....	18-75
18.22.3	:TIME<n> <numeric_value> .....	18-76
18.23	SWEep Subsystem .....	18-77
18.23.1	:COUNt <numeric_value> .....	18-78
18.23.2	:DIRection UP DOWN .....	18-78
18.23.3	:DWELl <numeric_value> .....	18-78
18.23.3.1	:AUTO <Boolean> ONCE .....	18-79
18.23.4	:GENeration STEPped ANALog .....	18-79
18.23.5	:MODE AUTO MANual .....	18-79
18.23.6	:OFFSet .....	18-79
18.23.6.1	:POINts <numeric_value> .....	18-79
18.23.6.2	:TIME <numeric_value> .....	18-79
18.23.7	:OREference .....	18-80
18.23.7.1	:LOCation <numeric_value> .....	18-80
18.23.7.2	:POINts <numeric_value> .....	18-80
18.23.8	:POINts <numeric_value> .....	18-80
18.23.9	:REALtime .....	18-81
18.23.9.1	[:STATE] <Boolean> .....	18-81
18.23.10	:SPACing LINear LOGarithmic .....	18-81
18.23.11	:STEP <numeric_value> .....	18-82
18.23.12	:TIME <numeric_value> .....	18-82
18.23.12.1	:AUTO <Boolean> ONCE .....	18-82
18.23.12.2	:LLIMit <numeric_value> .....	18-83
18.23.13	:TINTerval <numeric_value> .....	18-83
18.24	VOLTage Subsystem .....	18-84
18.24.1	:AC[:DC] .....	18-84
18.24.1.1	:APERture <numeric_value> .....	18-84
18.24.1.2	:NPLCycles <numeric_value> .....	18-85
18.24.1.3	:ATTenuation <numeric_value> .....	18-85
18.24.1.3.1	:AUTO <Boolean> .....	18-85
18.24.1.4	:PROTection .....	18-85
18.24.1.4.1	[:LEVel] <numeric_value> .....	18-85
18.24.1.4.2	:STATE <Boolean> .....	18-85

## 1999 SCPI Command Reference

18.24.1.4.3	:TRIPped?	18-85
18.24.1.4.4	:CLEar	18-85
18.24.1.5	:RANGE	18-86
18.24.1.5.1	[:UPPer] <numeric_value>	18-86
18.24.1.5.2	:LOWER <numeric_value>	18-86
18.24.1.5.3	:AUTO <Boolean> ONCE	18-86
18.24.1.5.3.1	:DIRECTION UP DOWN EITHER	18-86
18.24.1.5.3.2	:LLIMit <numeric_value>	18-86
18.24.1.5.3.3	:ULIMit <numeric_value>	18-86
18.24.1.5.4	:OFFSet <numeric_value>	18-87
18.24.1.5.5	:PTPeak <numeric_value>	18-87
18.24.1.6	:REFERENCE <numeric_value>	18-87
18.24.1.6.1	:STATE <Boolean>	18-87
18.24.1.7	:RESolution <numeric_value>	18-87
18.24.1.7.1	:AUTO <Boolean> ONCE	18-87
18.24.2	:DETEctor INTernal   EXTERNAL	18-88
18.25	WINDOW Subsystem	18-89
18.25.1	[:TYPE] RECTangular UNIFORM FLATtop HAMMING HANNING  KBESsel FORCe EXPonential	18-89
18.25.1.1	:KBESsel <numeric_value>	18-89
18.25.1.2	:EXPonential <numeric_value>	18-89
18.25.1.3	:FORCe <numeric_value>	18-89

## Chapter 19 SOURce Subsystem

19.1	ACCeleration Subsystem	19-5
19.1.1	[:LEVel] <numeric_value>	19-5
19.2	AM Subsystem	19-6
19.2.1	:COUpling AC DC GROund	19-6
19.2.2	[:DEPTH] <numeric_value>	19-6
19.2.3	:EXTERNAL	19-6
19.2.3.1	:COUpling AC DC GROund	19-7
19.2.3.2	:IMPedance <numeric_value>	19-7
19.2.3.3	:POLarity NORMal INVerted	19-7
19.2.4	:INTernal	19-7
19.2.4.1	:FREQuency <numeric_value>	19-7
19.2.5	:MODE	19-7
19.2.6	:POLarity NORMal INVerted	19-8
19.2.7	:SENSitivity <numeric_value>	19-8
19.2.8	:SOURce EXTERNAL INTERNAL{,EXTERNAL ,INTERNAL}	19-8
19.2.9	:STATe <Boolean>	19-8
19.2.10	:TYPE	19-9
19.3	COMBine Subsystem	19-10
19.3.1	:FEED <data_handle>	19-10
19.4	CORRection Subsystem	19-11

## 1999 SCPI Command Reference

19.4.1	<code>[:STATe] &lt;Boolean&gt;</code>	19-11
19.4.2	<code>:COLLect</code>	19-12
19.4.2.1	<code>[:ACQUire]</code>	19-12
19.4.2.2	<code>:METHod PMETer</code>	19-12
19.4.2.3	<code>:SAVE [&lt;name&gt;]</code>	19-12
19.4.3	<code>:CSET</code>	19-12
19.4.3.1	<code>[:SELECT] &lt;name&gt;</code>	19-12
19.4.3.2	<code>STATe &lt;Boolean&gt;</code>	19-12
19.4.4	<code>:OFFSet</code>	19-13
19.4.4.1	<code>[:MAGNitude] &lt;numeric_value&gt;</code>	19-13
19.4.4.2	<code>:PHASe &lt;numeric_value&gt;</code>	19-13
19.4.4.3	<code>:STATe &lt;Boolean&gt;</code>	19-13
19.4.5	<code>:LOSS :GAIN :SLOPe</code>	19-13
19.4.5.1	<code>:STATe &lt;Boolean&gt;</code>	19-14
19.4.5.2	<code>[:OUTPut]</code>	19-14
19.4.5.2.1	<code>[:MAGNitude] &lt;numeric_value&gt;</code>	19-14
19.4.5.2.2	<code>:PHASe &lt;numeric_value&gt;</code>	19-14
19.4.6	<code>:EDELay</code>	19-14
19.4.6.1	<code>[:TIME] &lt;numeric_value&gt;</code>	19-14
19.4.6.2	<code>:DISTance &lt;numeric_value&gt;</code>	19-14
19.4.6.3	<code>:STATe &lt;Boolean&gt;</code>	19-15
19.4.7	<code>:RVELOCITY</code>	19-15
19.4.7.1	<code>:MEDIum COAX WAVeguide</code>	19-15
19.4.7.2	<code>:COAX &lt;numeric_value&gt;</code>	19-15
19.4.7.3	<code>:WAVeguide &lt;numeric_value&gt;</code>	19-15
19.4.7.3.1	<code>:FCUTOff &lt;numeric_value&gt;</code>	19-15
19.4.7.4	<code>:STATe &lt;Boolean&gt;</code>	19-15
19.5	<code>CURRent Subsystem</code>	19-17
19.5.1	<code>:ATTenuation &lt;numeric_value&gt;</code>	19-18
19.5.1.1	<code>:AUTO &lt;Boolean&gt;</code>	19-18
19.5.2	<code>:ALC</code>	19-18
19.5.2.1	<code>[:STATe] &lt;Boolean&gt;</code>	19-18
19.5.2.2	<code>:SEARch &lt;Boolean&gt; ONCE</code>	19-18
19.5.2.3	<code>:SOURce INTernal DIODE PMETer MMHead</code>	19-19
19.5.2.4	<code>:BANDwidth :BWIDth &lt;numeric_value&gt;</code>	19-19
19.5.2.4.1	<code>:AUTO &lt;Boolean&gt; ONCE</code>	19-19
19.5.3	<code>:CENTer &lt;numeric_value&gt;</code>	19-19
19.5.4	<code>[:LEVel]</code>	19-19
19.5.4.1	<code>[:IMMediate]</code>	19-19
19.5.4.1.1	<code>[:AMPLitude] &lt;numeric_value&gt;</code>	19-20
19.5.4.1.1.1	<code>:AUTO &lt;Boolean&gt; ONCE</code>	19-20
19.5.4.1.2	<code>:OFFSet &lt;numeric_value&gt;</code>	19-20
19.5.4.1.3	<code>:HIGH &lt;numeric_value&gt;</code>	19-20
19.5.4.1.4	<code>:LOW &lt;numeric_value&gt;</code>	19-20
19.5.4.2	<code>:TRIGgered</code>	19-21

## 1999 SCPI Command Reference

19.5.4.2.1	[ <b>:AMPLitude</b> ] <numeric_value>	19-21
19.5.4.2.2	<b>:OFFSet</b> <numeric_value>	19-21
19.5.4.2.3	<b>:HIGH</b> <numeric_value>	19-21
19.5.4.2.4	<b>:LOW</b> <numeric_value>	19-21
19.5.5	<b>:LIMit</b>	19-21
19.5.5.1	[ <b>:AMPLitude</b> ] <numeric_value>	19-21
19.5.5.2	<b>:OFFSet</b> <numeric_value>	19-22
19.5.5.3	<b>:HIGH</b> <numeric_value>	19-22
19.5.5.4	<b>:LOW</b> <numeric_value>	19-22
19.5.5.5	<b>:STATe</b> <Boolean>	19-22
19.5.6	<b>:MANual</b> <numeric_value>	19-22
19.5.7	<b>:MODE</b> FIXed SWEep LIST	19-22
19.5.8	<b>:PROTection</b>	19-23
19.5.8.1	[ <b>:LEVel</b> ] <numeric_value>	19-23
19.5.8.2	<b>:STATe</b> <Boolean>	19-23
19.5.8.3	<b>:TRIPped?</b>	19-23
19.5.8.4	<b>:CLEar</b>	19-23
19.5.9	<b>:RANGE</b> <numeric_value>	19-23
19.5.9.1	<b>:AUTO</b> <Boolean> ONCE	19-23
19.5.10	<b>:REFERENCE</b> <numeric_value>	19-23
19.5.10.1	<b>:STATe</b> <Boolean>	19-24
19.5.11	<b>:SLEW</b> <numeric_value>	19-24
19.5.12	<b>:SPAN</b> <numeric_value>	19-24
19.5.12.1	<b>:FULL</b>	19-24
19.5.12.2	<b>:HOLD</b> <Boolean>	19-24
19.5.12.3	<b>:LINK</b> CENTer STARt STOP	19-24
19.5.13	<b>:STARt</b> <numeric_value>	19-24
19.5.14	<b>:STOP</b> <numeric_value>	19-25
19.6	<b>DM Subsystem</b>	19-26
19.6.1	<b>:FORMat</b> <modulation format>	19-27
19.6.2	<b>:STATe</b> <Boolean>	19-27
19.6.3	<b>:SOURce</b> EXTERNAL PRBS CALibrate	19-27
19.6.4	<b>:FILTer</b>	19-27
19.6.4.1	[ <b>:SOURce</b> ] INTERNAL EXTERNAL	19-27
19.6.4.2	<b>:ICORrection</b> <numeric_value>	19-27
19.6.4.3	<b>:QCORrection</b> <numeric_value>	19-27
19.6.5	<b>:IQRatio</b>	19-28
19.6.5.1	<b>:STATe</b> <Boolean>	19-28
19.6.5.2	[ <b>:MAGNitude</b> ] <numeric_value>	19-28
19.6.6	<b>:LEAKage</b>	19-28
19.6.6.1	<b>:STATe</b> <Boolean>	19-28
19.6.6.2	[ <b>:MAGNitude</b> ] <numeric_value>	19-28
19.6.6.3	<b>:ANGLE</b> <numeric_value>	19-28
19.6.7	<b>:QUADrature</b>	19-28
19.6.7.1	<b>:STATe</b> <Boolean>	19-29

## 1999 SCPI Command Reference

19.6.7.2	:ANGLE <numeric_value>	19-29
19.6.8	:COUPling	19-29
19.6.8.1	[:ALL] AC DC GROund	19-29
19.6.8.2	:DATA AC DC GROund	19-29
19.6.8.3	:CLOCk AC DC GROund	19-29
19.6.9	:THRehold	19-29
19.6.9.1	[:ALL] <numeric_value>	19-29
19.6.9.2	:DATA <numeric_value>	19-29
19.6.9.3	:CLOCk <numeric_value>	19-29
19.6.10	:DMODe SERial PARallel	19-30
19.6.11	:FRAMe	19-30
19.6.11.1	:SOURce INTernal EXTernal	19-30
19.6.12	:POLarity [:ALL] NORMAL INVerted	19-30
19.6.12.1	:I<n> NORMAL INVerted	19-30
19.6.12.2	:Q<n> NORMAL INVerted	19-30
19.6.12.3	:IClock NORMAL INVerted	19-30
19.6.12.4	:QClock NORMAL INVerted	19-30
19.6.13	:CLOCK	19-30
19.6.13.1	:SOURce NONE INTernal EXTernal	19-31
19.7	FM Subsystem	19-32
19.7.1	:COUPling AC DC GROund	19-32
19.7.2	[:DEViation] <numeric_value>	19-32
19.7.3	:EXTernal	19-32
19.7.3.1	:COUPling AC DC GROund	19-33
19.7.3.2	:IMPedance <numeric_value>	19-33
19.7.3.3	:POLarity NORMAL INVerted	19-33
19.7.4	:INTernal	19-33
19.7.4.1	:FREQuency <numeric_value>	19-33
19.7.5	:MODE LOCKed UNLocked	19-33
19.7.6	:POLarity NORMAL INVerted	19-34
19.7.7	:SENSitivity <numeric_value>	19-34
19.7.8	:SOURce EXTernal INTernal{,EXTernal ,INTernal}	19-34
19.7.9	:STATE <Boolean>	19-34
19.8	FORCe Subsystem	19-35
19.8.1	:CDOWn	19-35
19.8.1.1	:INITiate	19-36
19.8.1.2	:SOFFset <numeric_value>	19-36
19.8.1.3	:NRUNs <numeric_value>	19-37
19.8.1.4	:RLDerivation	19-37
19.8.1.4.1	:FACCeptance <numeric_value>	19-37
19.8.1.4.2	:INITiate	19-37
19.8.1.4.3	:RMAXimum <numeric_value>	19-38
19.8.1.4.4	:RVERify <numeric_value>	19-38
19.8.2	:CONFigure	19-39
19.8.2.1	:ABRake	19-39

## 1999 SCPI Command Reference

19.8.2.1.1	:GAIN <numeric_value> . . . . .	19-39
19.8.2.1.2	[::STATe] <Boolean> . . . . .	19-39
19.8.2.1.3	:THReShold <numeric_value> . . . . .	19-39
19.8.2.2	:GRADE . . . . .	19-39
19.8.2.2.1	:LEVel <numeric_value> . . . . .	19-39
19.8.2.2.2	:SOURce <INTERNAL EXTERNAL> . . . . .	19-39
19.8.2.2.3	[::STATe] <Boolean> . . . . .	19-40
19.8.2.3	[:VEHicle] . . . . .	19-40
19.8.2.3.1	:DCoefficient <numeric_value>,<numeric_value>,<numeric_value> [,<numeric_value>] . . . . .	19-40
19.8.2.3.2	:DINertia <numeric_value> . . . . .	19-40
19.8.2.3.3	[::STATe] <Boolean> . . . . .	19-40
19.8.2.3.4	:TCoefficient <numeric_value>,<numeric_value>,<numeric_value> [,<numeric_value>] . . . . .	19-41
19.8.2.3.5	:TINertia <numeric_value> . . . . .	19-41
19.8.2.3.6	:WEIGht <numeric_value> . . . . .	19-41
19.8.3	:INITiate . . . . .	19-41
19.8.4	[:LEVel] <numeric_value> . . . . .	19-42
19.8.5	:RLSimulation . . . . .	19-42
19.8.5.1	:INITiate . . . . .	19-42
19.9	FREQuency Subsystem . . . . .	19-43
19.9.1	:CENTer <numeric_value> . . . . .	19-43
19.9.2	[::CW FIXed] <numeric_value> . . . . .	19-43
19.9.2.1	.:AUTO <Boolean> ONCE . . . . .	19-43
19.9.3	:MANual <numeric_value> . . . . .	19-44
19.9.4	:MODE CW FIXed SWEep LIST SENSe . . . . .	19-44
19.9.5	:MULTiplier <numeric_value> . . . . .	19-44
19.9.6	:OFFSet <numeric_value> . . . . .	19-44
19.9.7	:RESolution <numeric_value> . . . . .	19-45
19.9.7.1	AUTO <Boolean>   ONCE . . . . .	19-45
19.9.8	:SPAN <numeric_value> . . . . .	19-45
19.9.8.1	:FULL . . . . .	19-45
19.9.8.2	:HOLD <Boolean> . . . . .	19-45
19.9.8.3	:LINK CENTer STARt STOP . . . . .	19-45
19.9.9	:STARt <numeric_value> . . . . .	19-46
19.9.10	:STOP <numeric_value> . . . . .	19-46
19.10	FUNCTION Subsystem . . . . .	19-47
19.10.1	[::SHAPe] <source_shape> . . . . .	19-47
19.10.2	:MODE <source_mode> . . . . .	19-47
19.11	LIST Subsystem . . . . .	19-48
19.11.1	:AM . . . . .	19-49
19.11.1.1	:DEPTh <numeric_value>{,<numeric_value>} . . . . .	19-49
19.11.1.1.1	:POINts? . . . . .	19-49
19.11.2	:APRobe <numeric_list>{,<numeric_list>} . . . . .	19-49
19.11.2.1	:POINts? . . . . .	19-49

## 1999 SCPI Command Reference

19.11.3	:CONCurrent <numeric_value>{,<numeric_value>} .....	19-49
19.11.3.1	:AUTO <Boolean> ONCE .....	19-50
19.11.3.2	:POINTs? .....	19-50
19.11.4	:CONTrol .....	19-50
19.11.4.1	:APOWer <Boolean>{,<Boolean>} .....	19-50
19.11.4.1.1	:POINTs? .....	19-50
19.11.4.2	:BLOWer <Boolean>{,<Boolean>} .....	19-50
19.11.4.2.1	:POINTs? .....	19-50
19.11.4.3	:COMPressor <Boolean>{,<Boolean>} .....	19-50
19.11.4.3.1	:POINTs? .....	19-50
19.11.5	:COUNt <numeric_value> .....	19-50
19.11.6	:CURRent <numeric_value>{,<numeric_value>} .....	19-50
19.11.6.1	:POINTs? .....	19-50
19.11.7	:DIRection UP DOWN .....	19-51
19.11.8	:DWELl <numeric_value>{,<numeric_value>} .....	19-51
19.11.8.1	:POINTs? .....	19-51
19.11.9	:FREQuency <numeric_value>{,<numeric_value>} .....	19-51
19.11.9.1	:POINTs? .....	19-51
19.11.10	:GENeration DSEQUence SEQUence DCONcurrent CONCurrent .....	19-51
19.11.11	:PULM .....	19-51
19.11.11.1	:STATe <Boolean>{,<Boolean>} .....	19-51
19.11.11.1.1	:POINTs? .....	19-51
19.11.12	:POWer <numeric_value>{,<numeric_value>} .....	19-52
19.11.12.1	:POINTs? .....	19-52
19.11.13	:RESistance <numeric_value>{,<numeric_value>} .....	19-52
19.11.13.1	:POINTs? .....	19-52
19.11.14	:RTIMe <numeric_value>{,<numeric_value>} .....	19-52
19.11.14.1	:POINTs? .....	19-52
19.11.15	:SEQUence <numeric_value>{,<numeric_value>} .....	19-52
19.11.15.1	:AUTO <Boolean> ONCE .....	19-52
19.11.15.2	:POINTs? .....	19-52
19.11.16	:TEMPerature <numeric_value>{,<numeric_value>} .....	19-52
19.11.16.1	:POINTs? .....	19-52
19.11.17	:VOLTage <numeric_value>{,<numeric_value>} .....	19-53
19.11.17.1	:POINTs? .....	19-53
19.12	MARKer Subsystem .....	19-54
19.12.1	:AMPLitude <Boolean> .....	19-54
19.12.2	:AOFF .....	19-54
19.12.3	:FREQuency <numeric_value> .....	19-54
19.12.4	:MODE FREQuency POSiition DELTa .....	19-54
19.12.5	:POINT <numeric_value> .....	19-54
19.12.6	:REFerence <numeric_value> .....	19-55
19.12.7	[:STATe] <Boolean> .....	19-55
19.13	PHASe Subsystem .....	19-56
19.13.1	[:ADJust] <numeric_value> .....	19-56

## 1999 SCPI Command Reference

19.13.1.1	:STEP <numeric_value>	19-56
19.13.2	:SOURce INTernal EXTernal	19-56
19.13.3	:REFerence	19-56
19.14	PM Subsystem	19-57
19.14.1	[:DEViation] <numeric_value>	19-57
19.14.2	:SENSitivity <numeric_value>	19-57
19.14.3	:MODE LOCKed UNLocked	19-57
19.14.4	:STATE <Boolean>	19-58
19.14.5	:SOURce EXTernal INTERNAL{,EXTernal ,INTERNAL}	19-58
19.14.6	:COUpling AC DC GROund	19-58
19.14.7	:POLarity NORMAL INVerted	19-58
19.14.8	:INTERNAL	19-59
19.14.8.1	:FREQuency <numeric_value>	19-59
19.14.9	:EXTernal	19-59
19.14.9.1	:IMPedance <numeric_value>	19-59
19.14.9.2	:COUpling AC DC GROund	19-59
19.14.9.3	:POLarity NORMAL INVerted	19-59
19.15	POWer Subsystem	19-60
19.15.1	:ATTenuation <numeric_value>	19-61
19.15.1.1	:AUTO <Boolean>	19-61
19.15.2	:ALC	19-61
19.15.2.1	[:STATE] <Boolean>	19-61
19.15.2.2	:SEARch <Boolean> ONCE	19-61
19.15.2.3	:SOURce INTERNAL DIODe PMETer MMHead	19-62
19.15.2.4	:BANDwidth BWIDth <numeric_value>	19-62
19.15.2.4.1	:AUTO <Boolean> ONCE	19-62
19.15.3	:CENTer <numeric_value>	19-62
19.15.4	[:LEVel]	19-62
19.15.4.1	[:IMMediate]	19-62
19.15.4.1.1	[:AMPLitude] <numeric_value>	19-63
19.15.4.1.2	:OFFSet <numeric_value>	19-63
19.15.4.1.3	:HIGH <numeric_value>	19-63
19.15.4.1.4	:LOW <numeric_value>	19-63
19.15.4.2	:TRIGgered	19-63
19.15.4.2.1	[:AMPLitude] <numeric_value>	19-64
19.15.4.2.2	:OFFSet <numeric_value>	19-64
19.15.4.2.3	:HIGH <numeric_value>	19-64
19.15.4.2.4	:LOW <numeric_value>	19-64
19.15.5	:LIMit	19-64
19.15.5.1	[:AMPLitude] <numeric_value>	19-64
19.15.5.2	:OFFSet <numeric_value>	19-64
19.15.5.3	:HIGH <numeric_value>	19-65
19.15.5.4	:LOW <numeric_value>	19-65
19.15.5.5	:STATE <Boolean>	19-65
19.15.6	:MANual <numeric_value>	19-65

## 1999 SCPI Command Reference

19.15.7	:MODE FIXed SWEep LIST .....	19-65
19.15.8	:PROtection .....	19-65
19.15.8.1	[:LEVel] <numeric_value> .....	19-65
19.15.8.2	:STATE <Boolean> .....	19-66
19.15.8.3	:TRIPPed? .....	19-66
19.15.8.4	:CLEar .....	19-66
19.15.9	:RANGE <numeric_value> .....	19-66
19.15.9.1	:AUTO <Boolean> ONCE .....	19-66
19.15.10	:REFerence <numeric_value> .....	19-66
19.15.10.1	:STATE <Boolean> .....	19-66
19.15.11	:SLEW <numeric_value> .....	19-66
19.15.12	:SPAN <numeric_value> .....	19-67
19.15.12.1	:HOLD <Boolean> .....	19-67
19.15.12.2	:LINK CENTER STARt STOP .....	19-67
19.15.12.3	:FULL .....	19-67
19.15.13	:STARt <numeric_value> .....	19-67
19.15.14	:STOP <numeric_value> .....	19-67
19.16	PULse Modulation Subsystem .....	19-68
19.16.1	:EXTernal .....	19-68
19.16.1.1	:HYSTeresis <numeric_value> .....	19-68
19.16.1.2	:IMPedance <numeric_value> .....	19-68
19.16.1.3	:LEVel <numeric_value> .....	19-68
19.16.1.4	:POLarity NORMAL INVerted .....	19-69
19.16.2	:INTernal .....	19-69
19.16.2.1	:FREQency <numeric_value> .....	19-69
19.16.3	MODE .....	19-69
19.16.4	:POLarity NORMAL INVerted .....	19-69
19.16.5	:SOURce EXTernal INTernal{,EXTernal ,INTernal} .....	19-70
19.16.6	:STATE <Boolean> .....	19-70
19.17	PULSe Subsystem .....	19-71
19.17.1	:PERiod <numeric_value> .....	19-71
19.17.2	:WIDTH <numeric_value> .....	19-71
19.17.3	:DCYCle <numeric_value> .....	19-71
19.17.4	:HOLD WIDTH DCYCle .....	19-71
19.17.5	:DELy <numeric_value> .....	19-72
19.17.6	:DOUBLE .....	19-72
19.17.6.1	[:STATE] <Boolean> .....	19-72
19.17.6.2	:DELy <numeric_value> .....	19-72
19.17.7	:TRANSition .....	19-72
19.17.7.1	:STATE <Boolean> .....	19-72
19.17.7.2	[:LEADING] <numeric_value> .....	19-72
19.17.7.3	:TRAiling <numeric_value> .....	19-73
19.17.7.3.1	:AUTO <Boolean> ONCE .....	19-73
19.17.8	:COUNt <numeric_value> .....	19-73
19.17.9	:POLarity NORMAL COMplement INVerted .....	19-73

## 1999 SCPI Command Reference

19.18	RESistance Subsystem	19-74
19.18.1	[:LEVel]	19-75
19.18.1.1	[:IMMEDIATE]	19-75
19.18.1.1.1	[:AMPLitude] <numeric_value>	19-75
19.18.1.2	:OFFSet <numeric_value>	19-75
19.18.1.3	:HIGH <numeric_value>	19-75
19.18.1.4	:LOW <numeric_value>	19-75
19.18.1.2	:TRIGgered	19-75
19.18.1.2.1	[:AMPLitude] <numeric_value>	19-76
19.18.1.2.2	:OFFSet <numeric_value>	19-76
19.18.1.2.3	:HIGH <numeric_value>	19-76
19.18.1.2.4	:LOW <numeric_value>	19-76
19.18.2	:LIMit	19-76
19.18.2.1	[:AMPLitude] <numeric_value>	19-76
19.18.2.2	:OFFSet <numeric_value>	19-76
19.18.2.3	:HIGH <numeric_value>	19-76
19.18.2.4	:LOW <numeric_value>	19-77
19.18.3	:PROtection	19-77
19.18.3.1	[:LEVel] <numeric_value>	19-77
19.18.3.2	:STATe <Boolean>	19-77
19.18.3.3	:TRIPped?	19-77
19.18.3.4	:CLEar	19-77
19.18.4	:SLEW <numeric_value>	19-77
19.18.5	:CENTer <numeric_value>	19-77
19.18.6	:SPAN <numeric_value>	19-77
19.18.6.1	:HOLD <Boolean>	19-78
19.18.6.2	:LINK CENTer STARt STOP	19-78
19.18.6.3	:FULL	19-78
19.18.7	:STARt <numeric_value>	19-78
19.18.8	:STOP <numeric_value>	19-78
19.18.9	:MANual <numeric_value>	19-78
19.18.10	:MODE FIXed SWEep LIST	19-79
19.18.11	:REFerence <numeric_value>	19-79
19.18.11.1	:STATe <Boolean>	19-79
19.18.12	:RANGe <numeric_value>	19-79
19.18.12.1	:AUTO <Boolean> ONCE	19-79
19.19	ROSCillator Subsystem	19-80
19.19.1	[:INTERNAL]	19-80
19.19.1.1	:FREQuency <numeric_value>	19-80
19.19.2	:EXTernal	19-80
19.19.2.1	:FREQuency <numeric_value>	19-80
19.19.3	:SOURce INTERNAL EXTERNAL NONE	19-80
19.19.3.1	:AUTO <Boolean> ONCE	19-80
19.20	SPEEd Subsystem	19-82
19.20.1	:INITiate	19-82

## 1999 SCPI Command Reference

19.20.2	[:LEVel] <numeric_value> . . . . .	19-82
19.20.3	:SSDLoss . . . . .	19-82
19.20.3.1	:INITiate . . . . .	19-83
19.20.3.2	:LATime <numeric_value> . . . . .	19-83
19.20.3.3	:STIMe <numeric_value> . . . . .	19-84
19.21	SWEep Subsystem . . . . .	19-85
19.21.1	:TIME <numeric_value> . . . . .	19-85
19.21.1.1	:AUTO <Boolean> ONCE . . . . .	19-85
19.21.1.2	:LLIMit <numeric_value> . . . . .	19-85
19.21.2	:DWELl <numeric_value> . . . . .	19-85
19.21.2.1	:AUTO <Boolean> ONCE . . . . .	19-86
19.21.3	:DIRection UP DOWN . . . . .	19-86
19.21.4	:MODE AUTO MANual . . . . .	19-86
19.21.5	:SPACing LINear LOGarithmic . . . . .	19-86
19.21.6	:GENeration STEPped ANALog . . . . .	19-86
19.21.7	:STEP <numeric_value> . . . . .	19-87
19.21.8	:POINts <numeric_value> . . . . .	19-87
19.21.9	:COUNt <numeric_value> . . . . .	19-87
19.22	TEMPeratureSubsystem . . . . .	19-88
19.22.1	:APRobe <numeric_list> . . . . .	19-88
19.22.2	:DWELl <numeric_value> . . . . .	19-88
19.22.3	:LCONstants . . . . .	19-89
19.22.3.1	:DERivative <numeric_value> . . . . .	19-89
19.22.3.2	[:GAIN] <numeric_value> . . . . .	19-89
19.22.3.3	:INTEGRal <numeric_value> . . . . .	19-89
19.22.4	:MODE FIXed LIST PROGram . . . . .	19-90
19.22.5	:PROTection . . . . .	19-90
19.22.5.1	[:HIGH] . . . . .	19-90
19.22.5.1.1	:CLEar . . . . .	19-90
19.22.5.1.2	[:LEVel] <numeric_value> . . . . .	19-90
19.22.5.1.3	:STATe <Boolean> . . . . .	19-91
19.22.5.1.4	:TOUT <numeric_value> . . . . .	19-91
19.22.5.1.5	:TRIPped? . . . . .	19-91
19.22.5.2	:LOW . . . . .	19-91
19.22.5.2.1	:CLEar . . . . .	19-91
19.22.5.2.2	[:LEVel] <numeric_value> . . . . .	19-91
19.22.5.2.3	:STATe <Boolean> . . . . .	19-91
19.22.5.2.4	:TOUT <numeric_value> . . . . .	19-92
19.22.5.2.5	:TRIPped? . . . . .	19-92
19.22.6	:RTIMe <numeric_value> . . . . .	19-92
19.22.7	[:SPOint] <numeric_value> . . . . .	19-92
19.23	VOLTage Subsystem . . . . .	19-93
19.23.1	:ATTenuation <numeric_value> . . . . .	19-94
19.23.1.1	:AUTO <Boolean> . . . . .	19-94
19.23.2	:ALC . . . . .	19-94

## 1999 SCPI Command Reference

19.23.2.1	[::STATe] <Boolean>	19-94
19.23.2.2	::SEARch <Boolean> ONCE	19-94
19.23.2.3	::SOURce INTernal DIODe PMETer MMHead	19-95
19.23.2.4	::BANDwidth::BWIDth <numeric_value>	19-95
19.23.2.4.1	::AUTO <Boolean> ONCE	19-95
19.23.3	::CENTer <numeric_value>	19-95
19.23.4	[::LEVel]	19-95
19.23.4.1	[::IMMEDIATE]	19-95
19.23.4.1.1	[::AMPLitude] <numeric_value>	19-96
19.23.4.1.1.1	::AUTO <Boolean> ONCE	19-96
19.23.4.1.2	::OFFSet <numeric_value>	19-96
19.23.4.1.3	::HIGH <numeric_value>	19-96
19.23.4.1.4	::LOW <numeric_value>	19-96
19.23.4.2	::TRIGgered	19-97
19.23.4.2.1	[::AMPLitude] <numeric_value>	19-97
19.23.4.2.2	::OFFSet <numeric_value>	19-97
19.23.4.2.3	::HIGH <numeric_value>	19-97
19.23.4.2.4	::LOW <numeric_value>	19-97
19.23.5	::LIMit	19-97
19.23.5.1	[::AMPLitude] <numeric_value>	19-97
19.23.5.2	::OFFSet <numeric_value>	19-98
19.23.5.3	::HIGH <numeric_value>	19-98
19.23.5.4	::LOW <numeric_value>	19-98
19.23.5.5	::STATe <Boolean>	19-98
19.23.6	::MANual <numeric_value>	19-98
19.23.7	::MODE FIXed SWEep LIST	19-98
19.23.8	::PROTection	19-99
19.23.8.1	[::LEVel] <numeric_value>	19-99
19.23.8.2	::STATe <Boolean>	19-99
19.23.8.3	::TRIPped?	19-99
19.23.8.4	::CLEAR	19-99
19.23.9	::RANGe <numeric_value>	19-99
19.23.9.1	::AUTO <Boolean> ONCE	19-99
19.23.10	::REFerence <numeric_value>	19-99
19.23.10.1	::STATe <Boolean>	19-100
19.23.11	::SLEW <numeric_value>	19-100
19.23.12	::SPAN <numeric_value>	19-100
19.23.12.1	::HOLD <Boolean>	19-100
19.23.12.2	::LINK CENTer STARt STOP	19-100
19.23.12.3	::FULL	19-100
19.23.13	::STARt <numeric_value>	19-100
19.23.14	::STOP <numeric_value>	19-101

## Chapter 20 STATus Subsystem

20.1	:OPERation .....	20-3
20.1.1	:BIT<n> .....	20-3
20.1.2	:CONDITION? .....	20-3
20.1.3	:ENABLE <NRf>   <non-decimal numeric> .....	20-3
20.1.4	[:EVENT]? .....	20-4
20.1.5	:MAP <NRf>,<NRf> .....	20-4
20.1.6	:NTRansition <NRf>   <non-decimal numeric> .....	20-4
20.1.7	:PTRansition <NRf>   <non-decimal numeric> .....	20-4
20.2	:PRESet .....	20-4
20.3	:QUEstionable .....	20-7
20.3.1	:BIT<n> .....	20-7
20.3.2	:CONDITION? .....	20-7
20.3.3	:ENABLE <NRf>   <non-decimal numeric> .....	20-7
20.3.4	[:EVENT]? .....	20-7
20.3.5	:MAP <NRf>,<NRf> .....	20-7
20.3.6	:NTRansition <NRf>   <non-decimal numeric> .....	20-7
20.3.7	:PTRansition <NRf>   <non-decimal numeric> .....	20-7

## Chapter 21 SYSTem Subsystem

21.1	:ALTernative <numeric_value> .....	21-3
21.1.1	:STATe <Boolean> .....	21-3
21.2	:BEEPer .....	21-3
21.2.1	:FREQuency <numeric_value> .....	21-3
21.2.2	[:IMMEDIATE][<frequency>[,<time>[,<volume>]]] .....	21-4
21.2.3	:STATe <Boolean> .....	21-4
21.2.4	:TIME <numeric_value> .....	21-4
21.2.5	:VOLume <numeric_value> .....	21-4
21.3	:CAPability? .....	21-4
21.4	:COMMUnicate .....	21-4
21.4.1	:CENTronics .....	21-4
21.4.1.1	:FEED <data_handle> .....	21-5
21.4.2	:GPIB .....	21-5
21.4.2.1	:RDEVice .....	21-5
21.4.2.1.1	:ADDRes <numeric_value>[,<numeric_value>] .....	21-5
21.4.2.1.2	:FEED <data_handle> .....	21-5
21.4.2.2	[:SELF] .....	21-5
21.4.2.2.1	:ADDRes <numeric_value>[,<numeric_value>] .....	21-5
21.4.3	:SERial .....	21-6
21.4.3.1	:CONTrol .....	21-6
21.4.3.1.1	:DTR ON OFF STANDARD IBFull .....	21-6
21.4.3.1.2	:RTS ON OFF STANDARD IBFull RFR .....	21-6
21.4.3.2	:FEED <data_handle> .....	21-7
21.4.3.3	[:RECeive] .....	21-7

## 1999 SCPI Command Reference

21.4.3.3.1	:BAUD <numeric_value>	21-7
21.4.3.3.2	:BITS <numeric_value>	21-7
21.4.3.3.3	:PACE XON ACK NONE	21-7
21.4.3.3.3.1	:THreshold	21-8
21.4.3.3.4	:PARity	21-8
21.4.3.3.4.1	:CHECK <Boolean>	21-8
21.4.3.3.4.2	[:TYPE] EVEN ODD ZERO ONE NONE IGNore	21-8
21.4.3.3.5	:SBITS <numeric_value>	21-8
21.4.3.4	:TRANSmit	21-8
21.4.3.4.1	:AUTO <Boolean>	21-9
21.4.3.4.2	:BAUD <numeric_value>	21-9
21.4.3.4.3	:BITS <numeric_value>	21-9
21.4.3.4.4	:DELay <numeric_value>	21-9
21.4.3.4.5	:PACE XON ACK NONE	21-9
21.4.3.4.6	:PARity	21-9
21.4.3.4.6.1	[:TYPE] EVEN ODD ZERO ONE NONE	21-9
21.4.3.4.7	:SBITS <numeric_value>	21-9
21.4.4	:SOCKEt <n>	21-10
21.4.4.1	:ADDReSS <string>	21-10
21.4.4.2	:CONNECT	21-10
21.4.4.3	:DISConnect	21-10
21.4.4.4	:FEED <n> <data_handle>{,<data_handle>}	21-10
21.4.4.4.1	:OCOndition <event_handle>	21-10
21.4.4.4.2	:SCOndition <event_handle>	21-10
21.4.4.5	:LISTen	21-10
21.4.4.6	:PORT <numeric_value>	21-10
21.4.4.7	:TYPE TCP UDP	21-11
21.5	:CPON <card_destination> ALL	21-11
21.6	:CTYPe? <card_destination>	21-11
21.7	:DATE <year>,<month>,<day>	21-11
21.8	:ERRor Subsystem	21-11
21.8.1	The Error/Event Queue	21-13
21.8.2	Error/Event numbers	21-14
21.8.3	No Error	21-14
21.8.4	ALL?	21-14
21.8.5	CODE	21-14
21.8.5.1	ALL?	21-14
21.8.5.2	[NEXT]?	21-14
21.8.6	COUNT?	21-14
21.8.7	:ENABLE	21-15
21.8.7.1	:ADD <numeric list>	21-15
21.8.7.2	:DELETE <numeric list>	21-15
21.8.7.3	[:LIST] <numeric list>	21-15
21.8.8	[NEXT]?	21-15
21.8.9	Command Error	21-15

## 1999 SCPI Command Reference

21.8.10	Execution Error .....	21-19
21.8.11	Device-Specific Error .....	21-25
21.8.12	Query Error .....	21-26
21.8.13	Power On Event .....	21-27
21.8.14	User Request Event .....	21-27
21.8.15	Request Control Event .....	21-28
21.8.16	Operation Complete Event .....	21-28
21.9	:HELP .....	21-28
21.9.1	:HEADers? .....	21-28
21.9.2	:SYNTax? <command_header> .....	21-31
21.10	:KEY <numeric_value> .....	21-32
21.10.1	:CATalog .....	21-32
21.10.2	:DEFIne <numeric_value>,<block>[,<string>] .....	21-32
21.10.3	:DELete <numeric_value> .....	21-32
21.11	:KLOCk <Boolean> .....	21-33
21.12	:LANGuage <string> .....	21-33
21.13	:LFREquency<numeric_value> .....	21-33
21.13.1	:AUTO <Boolean>   ONCE .....	21-33
21.14	:LOCK .....	21-34
21.14.1	:OWNer? .....	21-34
21.14.2	:RELEASE .....	21-34
21.14.3	:REQuest? .....	21-34
21.15	:PASSword .....	21-35
21.15.1	:CDISable <password> .....	21-35
21.15.2	[:CENable] <password> .....	21-35
21.15.2.1	:STATE? .....	21-35
21.15.3	:NEW <current password>,<new password> .....	21-35
21.16	:PRESet .....	21-35
21.17	:SECurity .....	21-36
21.17.1	:IMMEDIATE .....	21-36
21.17.2	[:STATE] <Boolean> .....	21-36
21.18	:SET <block data> .....	21-36
21.19	:TIME <hour>,<minute>,<second> .....	21-37
21.19.1	:TIme .....	21-38
21.19.1.1	:COUNt <numeric_value> .....	21-38
21.19.1.2	[:STATE] <Boolean> .....	21-38
21.20	:TZONe <hour> [,<minute>] .....	21-38
21.21	:VERSion? .....	21-38

## Chapter 22 TEST Subsystem

## Chapter 23 TRACe | DATA

23.1	:CATalog? .....	23-2
23.2	:COPY <trace_name>, ( <trace_name>   <data_handle> ) .....	23-2

## 1999 SCPI Command Reference

23.3	[:DATA]<trace_name>,(<block> <dif_expression> <numeric_value>{,<numeric_value>}))	23-2
23.3.1	:LINE <trace_name>,<numeric_value>,<numeric_value>,<numeric_value>,<numeric_value>	23-3
23.3.2	:PREamble? <trace_name>	23-3
23.3.3	:VALue <trace_name>,<numeric_value>,<numeric_value>	23-3
23.4	:DEFIne <trace_name>[,<numeric_value> <trace_name>)]	23-3
23.5	:DELete .....	23-4
23.5.1	[:NAME] <trace_name>	23-4
23.5.2	:ALL .....	23-4
23.6	:FEED <trace_name>, ( <data_handle>   NONE )	23-4
23.6.1	:CONTrol <trace_name>, ALWays   OCONDition   NEXT   NEVer	23-4
23.6.2	:OCONDition <trace_name>, <condition_expr>	23-5
23.7	:FREE?	23-5
23.8	:POINts <trace_name>[,<numeric_value>]	23-5
23.8.1	:AUTO <trace_name>,(<Boolean> ONCE)	23-5

## Chapter 24 TRIGger Subsystem

24.1	ARM-TRIGger Model .....	24-1
24.2	Model Layers .....	24-2
24.2.1	IDLE State .....	24-2
24.2.2	Initiated .....	24-2
24.2.3	Event Detection Layer .....	24-3
24.3	Sequence Event Use .....	24-4
24.4	Expanded Capability Trigger Model .....	24-5
24.4.1	LAYer Nomenclature .....	24-6
24.4.2	Standard SEQuences .....	24-7
24.4.3	Subservient Sequences .....	24-7
24.5	ABORt .....	24-12
24.6	ARM .....	24-12
24.6.1	[:SEQUence] .....	24-12
24.6.1.1	:DEFIne <sequence_name> .....	24-12
24.6.1.1.1	MGRules <Boolean> .....	24-13
24.6.1.2	[:LAYer] .....	24-14
24.6.1.2.1	:COUNt <numeric_value> .....	24-14
24.6.1.2.2	:COUpling AC DC .....	24-14
24.6.1.2.3	:DELay <numeric_value> .....	24-14
24.6.1.2.3.1	:AUTO <Boolean> ONCE .....	24-14
24.6.1.2.4	:ECL .....	24-15
24.6.1.2.5	:ECOunt <numeric_value> .....	24-15
24.6.1.2.6	:FILTer .....	24-15
24.6.1.2.6.1	:HPASSs .....	24-15
24.6.1.2.6.2	[:LPASSs] .....	24-15
24.6.1.2.7	:HYSTeresis <numeric_value> .....	24-16

## 1999 SCPI Command Reference

24.6.1.2.8	[:IMMEDIATE] .....	24-16
24.6.1.2.9	:LEVEL <numeric_value> .....	24-16
24.6.1.2.9.1	:AUTO <Boolean>   ONCE .....	24-16
24.6.1.2.10	:LINK <event_handle> .....	24-17
24.6.1.2.11	PROTocol .....	24-17
24.6.1.2.11.1	VXI SYNChronous SSYNchronous ASYNchronous .....	24-17
24.6.1.2.12	:SIGNAl .....	24-17
24.6.1.2.13	:SLOPe POSitive NEGative EITHER .....	24-18
24.6.1.2.14	:SOURce <parameter> .....	24-18
24.6.1.2.15	:TImer <numeric_value> .....	24-19
24.6.1.2.16	:TTL .....	24-19
24.6.1.2.17	:TYPE EDGE   VIDeo .....	24-19
24.6.1.2.18	:VIDeo .....	24-19
24.6.1.2.18.1	:FIELD .....	24-19
24.6.1.2.18.2	:FORMAT .....	24-20
24.6.1.2.18.3	:LINE .....	24-20
24.6.1.2.18.4	:SSIGNAL .....	24-20
24.7 INITiate .....	.....	24-21
24.7.1	:CONTinuous <Boolean> .....	24-21
24.7.1.1	[:ALL] <Boolean> .....	24-22
24.7.1.2	:NAME <sequence_name>,<Boolean> .....	24-22
24.7.1.3	:SEQUence <Boolean> .....	24-22
24.7.2	[:IMMEDIATE] .....	24-22
24.7.2.1	[:ALL] .....	24-22
24.7.2.2	:NAME <sequence_name> .....	24-23
24.7.2.3	:SEQUence .....	24-23
24.7.3	:POFLag INCLUde   EXCLUde .....	24-23
24.8 TRIGger .....	.....	24-23
24.8.1	[:SEQUENCE] .....	24-23
24.8.1.1	:ATRigger .....	24-23
24.8.1.1.1	[:STATe] <Boolean> .....	24-24
24.8.1.2	:COUNt <numeric_value> .....	24-24
24.8.1.3	:COUPling AC DC .....	24-24
24.8.1.4	:DEFine <sequence_name> .....	24-24
24.8.1.4.1	:MGRules <Boolean> .....	24-25
24.8.1.5	:DELay <numeric_value> .....	24-25
24.8.1.5.1	:AUTO <Boolean> ONCE .....	24-26
24.8.1.6	:ECL .....	24-26
24.8.1.7	:ECOUNT <numeric_value> .....	24-26
24.8.1.8	:FILTter .....	24-26
24.8.1.8.1	:HPASs .....	24-26
24.8.1.8.1.1	:FREQuency <numeric_value> .....	24-26
24.8.1.8.1.2	[:STATe] <Boolean> .....	24-26
24.8.1.8.2	:[LPASs] .....	24-26
24.8.1.8.2.1	:FREQuency <numeric_value> .....	24-27

## 1999 SCPI Command Reference

24.8.1.8.2.2	[::STATe] <Boolean>	24-27
24.8.1.9	::HOLDOff <numeric_value>	24-27
24.8.1.10	::HYSTeresis <numeric_value>	24-27
24.8.1.11	[::IMMEDIATE]	24-27
24.8.1.12	::LEVel <numeric_value>	24-28
24.8.1.12.1	::AUTO <Boolean>   ONCE	24-28
24.8.1.13	::LINK <event_handle>	24-28
24.8.1.14	::PROTocol	24-28
24.8.1.14.1	::VXI SYNChronous SSYNchronous ASYNchronous	24-29
24.8.1.15	::SIGNal	24-29
24.8.1.16	::SLOPe POSitive NEGative EITHER	24-29
24.8.1.17	::SOURce <parameter>	24-29
24.8.1.18	::TIMER <numeric_value>	24-30
24.8.1.19	::TTL	24-30
24.8.1.20	::TYPE EDGE   VIDeo	24-31
24.8.1.21	::VIDeo	24-31
24.8.1.21.1	::FIELd	24-31
24.8.1.21.1.1	[::NUMBER] <numeric_value>	24-31
24.8.1.21.1.2	::SElect ODD   EVEN   ALL   NUMBER	24-31
24.8.1.21.2	::FORMAT	24-31
24.8.1.21.2.1	::LPFRame <numeric_variable>	24-31
24.8.1.21.3	::LINE	24-32
24.8.1.21.3.1	[::NUMBER] <numeric_value>	24-32
24.8.1.21.3.2	::SElect ALL   NUMBER	24-32
24.8.1.21.4	::SSIGnal	24-32
24.8.1.21.4.1	::POLarity POSitive   NEGative	24-32

## Chapter 25 UNIT Subsystem

25.1	::ANGLE DEG RAD	25-1
25.2	::CURRent, ::POWER, and ::VOLTage	25-1
25.3	::TEMPerature C CEL F FAR K	25-2
25.4	::TIME HOUR MINute SECond	25-2

## Chapter 26 VXI Subsystem

26.1	::CONFIGure	26-3
26.1.1	::DNUMber?	26-3
26.1.2	::HIERarchy?	26-3
26.1.2.1	::ALL?	26-4
26.1.2.2	::VERBOSE?	26-4
26.1.2.2.1	::ALL?	26-4
26.1.3	::INFormation?	26-4
26.1.3.1	::ALL?	26-6
26.1.3.2	::VERBOSE?	26-6
26.1.3.3	::ALL?	26-6

## 1999 SCPI Command Reference

26.1.4	:LADDress?	.....	26-6
26.1.5	:NUMBER?	.....	26-7
26.2	REGister	.....	26-7
26.2.1	:READ? <register>	.....	26-7
26.2.1.1	:VERBose? <register>	.....	26-8
26.2.2	:WRITe (<numeric_value>   <register>), <data>	.....	26-8
26.3	:RESET?	.....	26-8
26.3.1	:VERBose?	.....	26-9
26.4	:SElect <logical_address>	.....	26-9
26.5	:WSPRotocol	.....	26-9
26.5.1	:COMMAND	.....	26-9
26.5.1.1	[:ANY] <data>	.....	26-9
26.5.1.2	:AHLIne <hand_id>, <line_number>	.....	26-9
26.5.1.3	:AILINE <int_id>, <line_number>	.....	26-9
26.5.1.4	:AMControl <response_mask>	.....	26-10
26.5.1.5	:ANO	.....	26-10
26.5.1.6	:BAvailable <Boolean>, <byte>	.....	26-10
26.5.1.7	:BNO <Boolean>	.....	26-10
26.5.1.8	:BRQ	.....	26-10
26.5.1.9	:CEVent <Boolean>, <event_number>	.....	26-10
26.5.1.10	:CLR	.....	26-10
26.5.1.11	:CLOCk	.....	26-11
26.5.1.12	:CRESpone <response_mask>	.....	26-11
26.5.1.13	:ENO	.....	26-11
26.5.1.14	:GDEVice <logical_address>	.....	26-11
26.5.1.15	:ICOMmander <logical_address>	.....	26-11
26.5.1.16	:RDEVice <logical_address>	.....	26-11
26.5.1.17	:RHANdlers	.....	26-11
26.5.1.18	:RHLIne <hand_id>	.....	26-12
26.5.1.19	:RILINE <int_id>	.....	26-12
26.5.1.20	:RINTerrupter	.....	26-12
26.5.1.21	:RMODid	.....	26-12
26.5.1.22	:RPERror	.....	26-12
26.5.1.23	:RPRotocol	.....	26-12
26.5.1.24	:RSTB	.....	26-12
26.5.1.25	:RSARea	.....	26-13
26.5.1.26	:SLModid <Boolean>, <MODID 6-0>	.....	26-13
26.5.1.27	:SLOCk	.....	26-13
26.5.1.28	:SUModid <Boolean>, <MODID 12-7>	.....	26-13
26.5.1.29	:TRIGger	.....	26-13
26.5.2	:MESSAGE	.....	26-13
26.5.2.1	:RECeive? <count>   <terminator>	.....	26-13
26.5.2.2	:SEND <message_string> [, (END NEND)]	.....	26-13
26.5.3	:QUERy	.....	26-14
26.5.3.1	[:ANY]? <data>	.....	26-14

## 1999 SCPI Command Reference

26.5.3.2	:AHLIne? <hand_id>,<line_number> .....	26-14
26.5.3.3	:AILIne? <int_id>,<line_number> .....	26-14
26.5.3.4	:AMControl? <response_mask> .....	26-14
26.5.3.5	:ANO? .....	26-14
26.5.3.6	:BNO? <Boolean> .....	26-14
26.5.3.7	:BRQ? .....	26-15
26.5.3.8	:CEVent? <Boolean>,<event_number> .....	26-15
26.5.3.9	:CRESpone? <response_mask> .....	26-15
26.5.3.10	:ENO? .....	26-15
26.5.3.11	:RDEvice? <logical_address> .....	26-15
26.5.3.12	:RHANdlers? .....	26-15
26.5.3.13	:RHLIne? <hand_id> .....	26-15
26.5.3.14	:RILIne? <int_id> .....	26-16
26.5.3.15	:RINTerrupter? .....	26-16
26.5.3.16	:RMODid? .....	26-16
26.5.3.17	:RPERror? .....	26-16
26.5.3.18	:RPRotocol? .....	26-16
26.5.3.19	:RSTB? .....	26-16
26.5.3.20	:RSARea? .....	26-16
26.5.3.21	:SLModid? <Boolean>,<MODID 6-0> .....	26-17
26.5.3.22	:SUModid? <Boolean>,<MODID 12-7> .....	26-17
26.5.4	:RESPonse? .....	26-17

## **1999 SCPI Command Reference**

## **1      Introduction**

### **1.1   Requirements**

This volume of the SCPI standard, “Command Reference,” shall be used in conjunction with the “Syntax and Style” volume.

### **1.2   Organization**

The first chapter in this volume of the SCPI standard describes the SCPI model that is used in defining the functional areas of SCPI. The subsequent chapters describe each functional area, known as a subsystem, and are in alphabetic order by subsystem name.

## **1999 SCPI Command Reference**

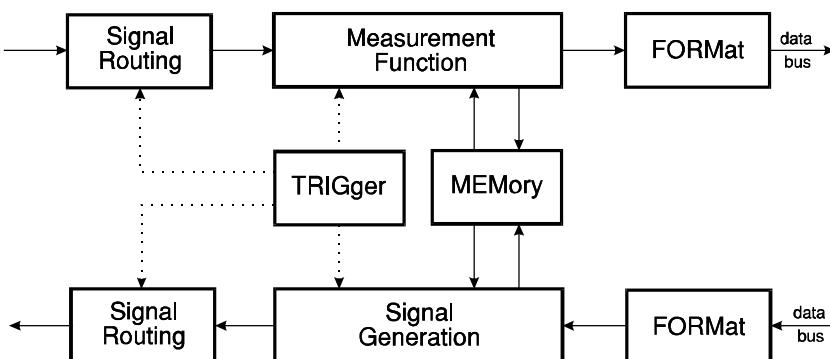
## 2 Instrument Model

A model is used within SCPI as a means of achieving compatibility. SCPI concerns itself with three types of compatibility. The first form of compatibility is called vertical compatibility. Vertical compatibility is where two instruments of the same type have identical controls (e.g. two oscilloscopes both have the same controls for their timebases, triggers, and voltage settings).

2

The second form of compatibility is called horizontal compatibility. Horizontal compatibility is where two instruments can make the same measurement, regardless of the actual measurement techniques used. To be horizontally compatible, both instruments would use the same commands to make this measurement. For example, both an oscilloscope and a counter can perform a risetime measurement on a pulse. The two instruments are said to be horizontally compatible if the same command is used in both instruments.

The third form of compatibility is called functional compatibility. Functional compatibility is where two instruments which perform the same function do so with the same commands. For example, a spectrum analyzer and an rf source may both sweep in frequency. If the same frequency and sweep commands are used in both instruments, they would be functionally compatible in this area.



**Figure 2-1 Model of a Programmable Instrument**

Figure 2-1, “Model of a Programmable Instrument,” represents the way in which instrument functionality is viewed and categorized by SCPI. The purpose of this categorization is to provide organization and consistency between the various commands available in SCPI for all the different types of instrumentation. The model defines where elements of the language must be assigned in the SCPI hierarchy. Major areas of signal functionality are shown broken into blocks; each of these blocks are major command subtrees in SCPI. In this volume of the SCPI standard, each subtree is discussed in its own chapter.

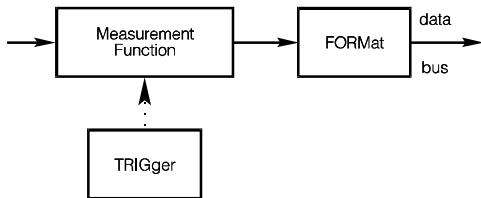
The model describes the flow of measurement and applied signal data through the instrument. The administrative data flows associated with commands, queries, performing calibrations, mass memory accesses, and other related functions are not included in this model. The model does not define how an instrument handles or formats data internally. In

## 1999 SCPI Command Reference

Figure 2-1, data flow is represented by solid arrows and control flow is represented by dashed arrows.

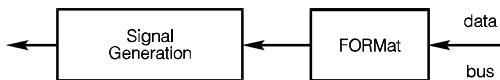
Individual instruments shall only implement those blocks which apply to their implementation. For example, a voltmeter which has a single input and no user addressable memory might include the measurement function, TRIGger, and FORMat blocks.

2



**Figure 2-2 Simplified Model for a Sensor Instrument**

Similarly, a non-triggered, single output power supply might include the signal generation and FORMat blocks.



**Figure 2-3 Simplified Model for a Source Instrument**

An instrument with a particular block is not always required to implement a subtree or any commands associated with that block. This occurs when the block exists in a fixed configuration that does not conflict with related commands, and that have the \*RST default values. For example, an instrument might have a fixed TTL level external trigger, with no adjustable TRIGger parameters. In this case no part of the TRIGger subsystem need be implemented. However, if trigger controls exist, those controls shall be implemented as defined in the TRIGger subsystem.

### 2.1

#### Signal Routing

The purpose of the signal routing block is to control the routing of signals between an instrument's signal ports and its internal signal functionality. Signal routing also controls the connection from signal port to signal port, where such capability exists. The commands which control this block are described in the SCPI tree under the ROUTe subsystem. The implementation of this subsystem is optional for those instruments that have fixed connections to the measurement function block or the signal generation block.

#### 2.1.1

##### Setting the Destination for Data Flow

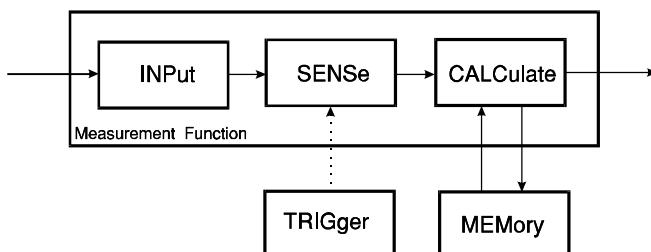
The FEED model works well when the user is focused on setting the source for the data flow into a block. Occasionally, the user's focus is on the destination of data flow from a block. When this view is appropriate, a DESTination <data\_handle> command is used.

DESTination is an event; it has no query form. The command sets one or more FEED settings to known values. Assume the subsystem where the DESTination command is located could be accessed with a <data\_handle> of <source\_data\_handle>. When DESTination <data\_handle> is executed, any FEED command which is currently set to <source\_data\_handle> shall be set to “”. The FEED command under the <data\_handle> subsystem shall be set to <source\_data\_handle>. Any subsystem which may appear in the <data\_handle> shall contain a FEED command. An allowed value of the <data\_handle> for that FEED command shall be <source\_data\_handle>.

## 2.2

### Measurement Function

The measurement function block converts a physical signal into an internal data form that is available for formatting into bus data. It may perform the additional tasks of signal conditioning and post conversion calculation. The measurement function box is subdivided into three distinct parts: INPut, SENSe, and CALCulate. These are shown as solid boxes in Figure 2-4, “Expanded Measurement Function Model.” The dotted boxes shown represent blocks from Figure 2-1, “Model of a Programmable Instrument.”



**Figure 2-4 Expanded Measurement Function Model**

The subdivisions INPut, SENSe, and CALCulate are not detailed in the “Model of a Programmable Instrument” figure because at this lower level of division, functionality across different instruments can no longer be considered horizontally compatible. For example, measuring the frequency of a signal is done in SENSe block on a counter, but is done in the CALCulate block in a spectrum analyzer. The two instruments are considered compatible since they both can perform the function. However, the horizontal compatibility only occurs at the high level MEASurement commands.

#### 2.2.1 INPut

The purpose of the INPut block is to condition the incoming signal before it is converted into data by the SENSe block. INPut block functions include filtering, biasing, frequency conversion (such as a mixer or prescaler function), and attenuation. The INPut block appears in the SCPI tree under the INPut subsystem. The implementation of this subsystem is optional for those instruments that have no INPut block characteristics.

**2.2.2****SENSe**

The purpose of the SENSe block is to convert signal(s) into internal data that can be manipulated by normal computer techniques. The commands associated with the SENSe block control the various characteristics of the conversion process. Examples are range, resolution, gate time, normal mode rejection, etc. This block does not include any mathematical manipulation of the data after it has been converted.

2

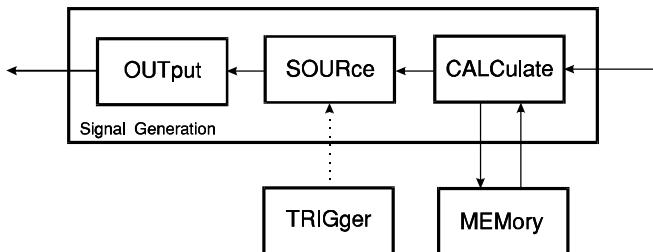
The SENSe commands appear in the SCPI tree under the SENSe subsystem. The SENSe block is required for all sensor instruments, since the nature of a bus programmable instrument implies a conversion from an electrical signal to a “number.” However, the SENSe subsystem contains a large number of subservient subsystems, subdividing the different signal characteristics being sensed. An instrument is required to only implement the appropriate commands from those subservient subsystems that apply.

**2.2.3****CALCulate**

The purpose of the CALCulate block is to convert or derive sensed data into a form more useful to the application. Typical calculations include converting units, and postprocessing calculations (for example, calculation of rise time from a time domain waveform). The CALCulate commands are described in the CALCulate subsystem.

**2.3****Signal Generation**

The signal generation block is responsible for the conversion of data into physical signals. It may perform additional tasks of preconversion calculation and signal conditioning. The signal generation box is subdivided into three distinct parts: CALCulate, SOURce, and OUTPut. These are shown as solid boxes in Figure 2-5, “Expanded Signal Generation Model.” The dotted boxes shown represent blocks from Figure 2-1, “Model of a Programmable Instrument.”



**Figure 2-5 Expanded Signal Generation Model**

The subdivisions CALCulate, SOURce, and OUTPut are not detailed in the “Model of a Programmable Instrument,” because at this lower level of division, functionality across different instruments can no longer be considered horizontally compatible.

**2.3.1****OUTPut**

The purpose of the OUTPut block is to condition the outgoing signal after it has been generated. The OUTPut block functions include filtering, biasing, frequency conversion (such as a mixer function), and attenuation. The OUTPut block appears in the SCPI tree

under the OUTPut subsystem. As with the INPut block in sensor-type instruments, this block appears in all source-type instruments, though the commands may not.

### 2.3.2 **SOURce**

The purpose of the SOURce block is to generate a signal based on specified characteristics and/or supplied data. The commands associated with this block describe the characteristics of the generated signal.

The SOURce commands are described in the SOURce subsystem. The SOURce block is required for all source instruments. The SOURce subsystem contains a large number of subservient subsystems, sub-dividing the different signal characteristics being sourced. An instrument is only required to implement the appropriate commands from those subservient subsystems that apply.

### 2.3.3 **CALCulate**

The purpose of the CALCulate block is to convert application data to account for anomalies in generating a signal, such as correcting for external effects, converting units and changing domains. The commands associated with the CALCulate block are defined in the CALCulate subsystem.

### 2.4 **TRIGger**

The purpose of the TRIGger block is to provide an instrument with synchronization capability with external events. The TRIGger block is described in chapter 22, and appears in the SCPI tree as TRIGger, ARM, INITiate, and ABORT subsystems. Two models exist in the TRIGger block, one complex and the other simple. Both models may be reduced to a level that corresponds to the sophistication of an instrument's trigger capabilities.

### 2.5 **MEMory**

The purpose of the MEMory block is to hold data inside the instrument. The memory may be implicit and inaccessible to the user (for example, internal calibration data), and may be fixed (for example, current measurement data) or may be allocated and user-addressed.

The allocation of memory appears in SCPI under the MEMory subsystem. The manipulation of memory is performed by commands throughout the SENSe, SOURce, FORMat, DISPlay, and other subsystems. While every programmable instrument contains memory, not all such instruments provide the user with explicit control of this memory. In such cases there is no requirement to implement the MEMory subsystem.

### 2.6 **FORMAT**

The purpose of the format block is to convert between data representations, especially on the data that is transferred over the external interface. An example is conversion of internal data formats to ASCII.

Formatting appears in SCPI under the FORMAT subsystem. All instruments are required to use the formats described in *IEEE 488.2*.

### 2.7 **Internal Routing**

The instrument model presented so far introduced the fundamental elements or blocks from which an instrument is constructed, and defined a simple or lamina arrangement of the connections between those blocks, known as the "Lamina Model." This model is adequate

for many instruments. In some instruments the connections may not be as shown, as multiple instances of certain individual blocks from the Lamina Model may exist in the instrument. In such instruments the connections between blocks may be programmable. Each instance of a block is differentiated by means of the numeric suffix used with the block name.

### 2.7.1 Data and Control Flow

All of the solid lines in the Lamina Model represent data flow between the blocks. The external signal (often electrical) can be considered as data, since it contains information. In a sensor type instrument using the Lamina Model, this data is routed through the ROUTe block into the INPut block, the data then flows into the SENSe block, then on to the CALCulate block, and finally into the DISPLAY block or to the system controller over the bus. In a source type instrument using the Lamina Model, data from the controller flows over the bus, through the CALCulate block, then through the SOURCE block. Finally, the data in the form of an external signal (often electrical) flows into the OUTPut block, and is then distributed through the ROUTe block.

Obviously, each block has points where data flow is absorbed and emitted. Many blocks have only one point where the data is absorbed and one point where data is emitted, however there is no restriction on how many of either of these points a block must have, further the number of points where data is absorbed and the number of points where data is emitted do not have to appear in equal numbers.

When describing the data flow, the SCPI model specifies for each point where data is absorbed where that data is obtained from. This selection of data flow into a block is made with the FEED command.

There are certain restrictions that apply to the data FEED for each block; these restrictions are enumerated under each block's command description for its FEED. That is, a particular block's FEED is only allowed access to certain points in the data stream, generally the FEED must select a block which is of the same type as the block that immediately precedes it in the instrument model described earlier. For example a SENSe block which deals with physical signals, is not allowed to FEED data from a CALCulate block, which is in an internal format, typically numeric data. Further, SENSe blocks are only allowed to FEED data from INPut blocks. Instruments that effectively have no INPut functionality, that is no form of signal conditioning, still are considered to have INPut blocks, so an INPut is available to be specified as the FEED of the SENSe block. A block, such as the INPut block just described, is called a null block.

Associated with the data flow is a control mechanism to ensure the coherent processing of the data. Each process has points where control flow is absorbed and emitted. There is no restriction on how many of either of these control flow points a block must have. A control flow carries messages or events. These messages or events are absorbed from a control flow by a process, to control the progress and direction of the process. In turn, each process emits messages or events that are used by other processes. The action of moving from one process to another is an event. Each process may be sub-divided into a series of steps, the action from moving from one step to another is also an event.

In a particular instrument, there will be a number of control flows established as part of the instrument's configuration. A typical example is found in the TRIGger block, where the final layer in a trigger sequence generates an event that causes the "device action" to be performed, and a subsequent event is returned by the block(s) performing the device action to indicate its completion. Another example, is one where the completion of a data acquistion by the SENSe block is an event that is used by the CALCulate block to automatically start the calculation process.

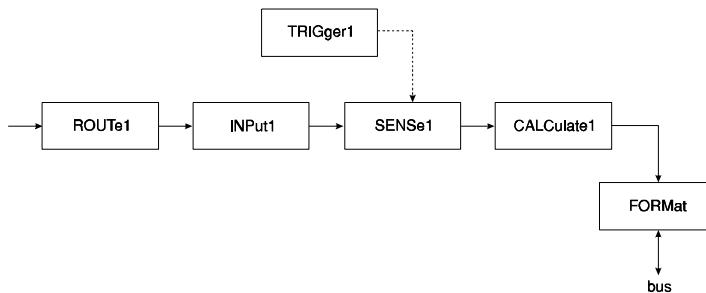
In certain cases some of the control flows may be programmable, allowing a user to select the controlling events. When describing the control flow, the SCPI model specifies for each point where control information flows in, where it obtains its control flow from. The selection of a control flow into a block is made with the LINK command.

## 2.7.2 Numeric Suffixes

Numeric suffixes are associated with each block to provide them with a unique identifier. The way in which a block with a particular suffix is related to other blocks in an instrument depends upon the model employed. The next section describes the Lamina Model and the closely related Cloned Model, also described is the Amorphous SENSe Model.

## 2.7.3 Lamina and Cloned Models

The Model of a Programmable instrument shows one instance of each of the major blocks (or subsystems) that are available in SCPI. In this case each block is given a numeric suffix of 1. This numeric suffix is not usually shown, as it may be defaulted. The relationship between the blocks is fixed in the order shown. For example, Figure 2-6 shows the Lamina Model for a sensor instrument where the name of each block is shown in its explicit form.

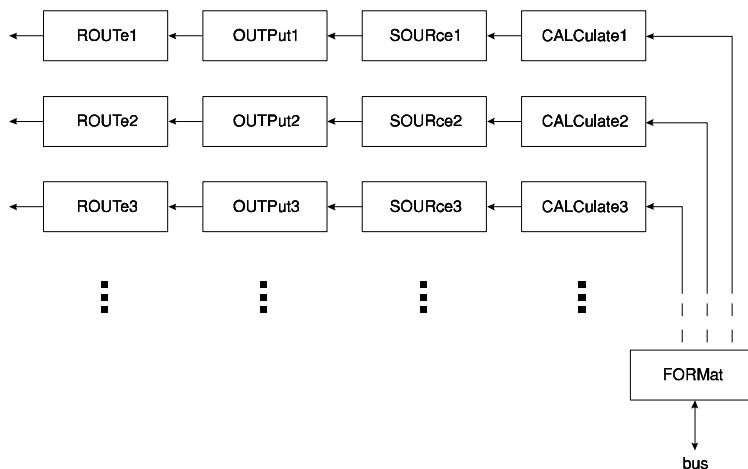


**Figure 2-6 Lamina Model for a Sensor Instrument**

The Cloned Model repeats the Lamina Model as many times as required. Each repetition of the Lamina Model is accompanied with a different numeric suffix, incremented by 1 from the last. The cloned model is suitable for instruments that have multiple independent capabilities. For example, shown in Figure 2-7 is the Cloned Model that could be used to represent a multiple independent capability source instrument.

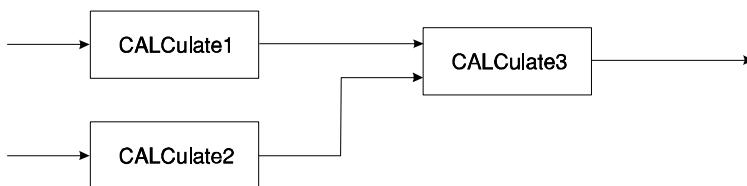
### 2.7.3.1 FEED Exceptions

As mentioned earlier, a particular block's FEED is only allowed access to certain points in the data stream, generally the FEED must select a block which is of the same type as the block that immediately precedes it, in the Instrument Model.



**Figure 2-7 Cloned Model for a Multi-Channel Instrument**

CALCulate typically follows this rule, obtaining data from the preceding block, which in the case of a sensor instrument would be the SENSe block. However, where an implementation permits, CALCULATE blocks may be cascaded to provide complex functionality, it is permissible for a CALCULATE block to use another CALCULATE block as its FEED, as shown in Figure 2-8:



**Figure 2-8 Cascaded CALCULATE Blocks**

DISPlay is also a special case, it may obtain data from any point where the data is in an internal format, typically numeric data. Examples include a CALCULATE block or a SENSe block.

### 2.7.3.2 Amorphous SENSe Model

In certain sensor instruments there are SENSe blocks that produce parallel independent results, where certain parts of the SENSe subsystem are shared. A distinguishing feature of the Amorphous SENSe Model is that the SENSe block has more than one INPut block

attached to it. In this model the numeric suffix associated with the INPut block is associated with the subsystems of the SENSe block, not the SENSe itself. Thus INPut<n>, where <n> is a number, would relate to the subsystems of SENSe such as, SENSe:VOLTage<n>, SENSe:CORRection<n> or SENSe:DETector<n>. This is typical, unless a particular subsystem of SENSe has been explicitly configured differently, to provide different or additional combinations of functionality.

One example, is where a SENSe block with two power sensors, provides two POWER results and a cross-product result such as a POWER:RATio. Another example, is where a SENSe block with two voltage sensors and two independent time sweep capabilities can be arranged to provide four different results.

### 2.7.3.3 Other Subsystems

Only one instance of certain blocks (or subsystems) is permitted in an instrument, these are FORMAT, INSTRument, MEMORY, MMEMORY, STATUs, SYSTem and TRACe. In most instruments there is only one DISPLAY, since it is rare to find instruments with two or more displays that are entirely separate and distinct.

### 2.7.4 FEED <data\_handle>

The parameter to a FEED command is <data\_handle>. All <data\_handle>s are defined to be IEEE 488.2 STRING PROGRAM DATA. Along a data stream there are discrete points where data may be picked up. These points have names that are called <data\_handle>s. For blocks with a single point where data is emitted, the block name with its numeric suffix is the <data\_handle>. For example, the <data\_handle> for CALCulate3 is the string “CALCulate3”.

Note that the <data\_handle> in the FEED command may consist of either long or short forms mnemonics. However the query shall return the short form and shall omit any default nodes. Thus, the query for the above example would return the string “CALC3”.

The <data\_handle>s for the different SENSe block found in each of the model, is described in the following sections.

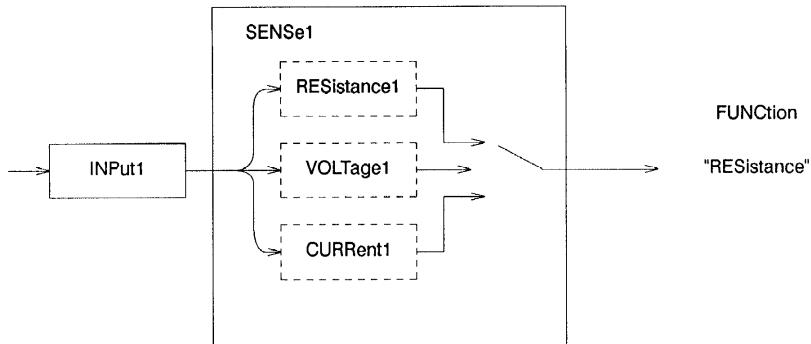
### 2.7.4.1 Sensor Function Selection

Selecting a sensor’s function effects the data flow inside of the SENSe block. Selecting a particular <sensor\_function> enables the data flow associated with it. If a <sensor\_function> is enabled then there must be a <data\_handle> associated with the data flow generated by that function. A SENSe block that has only a single point where data is emitted has a single SENSe:FUNCTION selected. The selections are mutually exclusive of one another, that is enabling one function would disable another. In the case shown in Figure 2-10, the SENSe:FUNCTION selected is “RESistance”, since it is the only data flow emitted from the block, SENSe1 is all that is needed to create a unique <data\_handle> for this case.

In instruments which have amorphous SENSe blocks one or more functions can be enabled simultaneously within the same block, that is selecting a function adds to the list of currently enabled functions. Such a SENSe block has one or more points where data is emitted, thus the SENSe block and its suffix is no longer sufficient to distinguish the different data flows.

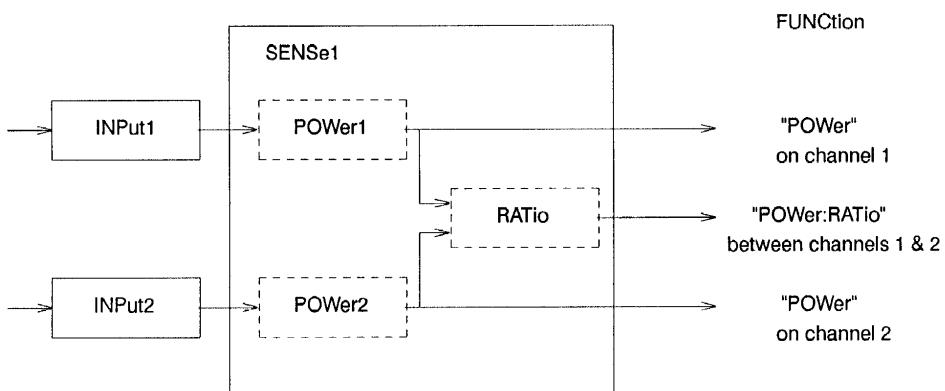
## 1999 SCPI Command Reference

It is necessary for the <data\_handle> for these data flows to include the SENSe:FUNCTION that generated the flow, along with the SENSe block and its numeric suffix.



**Figure 2-10 Single Function Active SENSe Block**

In Figure 2-9, the sensed characteristic alone is not sufficient, to uniquely specify a function, and therefore cannot be used to uniquely indentify a data flow. By including in the <sensor\_function> the INPut block number(s) associated with a particular data flow, within the SENSe block, a unique specification for the <sensor\_function> and thus a unique <data\_handle> can be formed. Where the <sensor\_function> is included in a <data\_handle>, the SENSe node with the optional numeric suffix of 1, may be omitted altogether, since many instruments that employ an amorphous SENSe block, will have only one SENSe block, that is SENSe1.

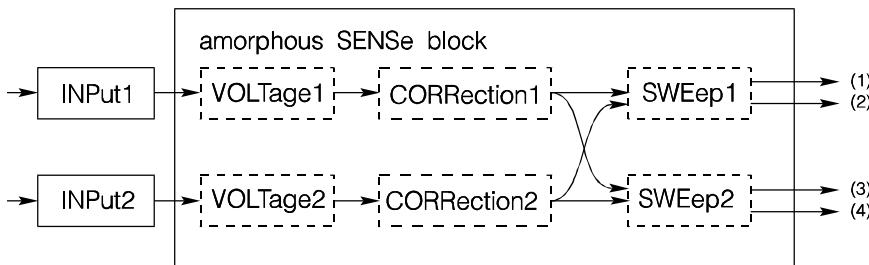


**Figure 2-9 Amorphous SENSe Block with Cross-Product**

<b>DESCRIPTION</b>	<b>&lt;sensor_function&gt;</b>	<b>&lt;data_handle&gt;</b>
The power function on INPut1	"POWer 1"	"SENSe:POWer 1"
The power ratio on INPut1 & INPut2	"POWer:RATio 1,2"	"SENSe:POWer:RATio 1,2"
The power function on INPut2	"POWer 2"	"SENSe:POWer 2"

The amorphous SENSe block has an additional level of complexity, where SENSe subsystems (or sub-blocks) are used in combination as shown in Figure 2-11.

Figure 2-11 shows an amorphous SENSe block with shared SWEep subsystems, and the corresponding <sensor\_function>s. It should be noted that the function part of the specification indicates the physical characteristic being sensed, and not which subsystem of SENSe is used. The subsystems used are specified by association with the INPut block, in the example above VOLTage1 and CORRection1 are tied to the data flow associated with INPut1. Where the association is not as specified above, the subsystem with its numeric suffix is appended to the specification after the literal token 'ON'. Note how SWEep1 associated with INPut1 and SWEep2 associated with INPut2 can be defaulted, as they follow the default relationship of an amorphous SENSe block.



- (1) "VOLTage:AC 1 ON SWEep1" or "VOLTage:AC 1"
- (2) "VOLTage:AC 2 ON SWEep1"
- (3) "VOLTage:AC 1 ON SWEep2"
- (4) "VOLTage:AC 2 ON SWEep2" or "VOLTage:AC 2"

**Figure 2-11 Amorphous SENSe Block with Shared SWEep**

The <data\_handle> is then the <sensor\_function> preceded by the SENSe block and its suffix. Thus a FEED for a display trace from an amorphous SENSe block, as shown above, might look like:

```
DISPLAY:WINDOW:TRACe:FEED "SENSe1:VOLTage:AC 2 ON SWEEP1"
```

### 2.7.4.2 FEEDs from CALCulate Sub-Blocks

The CALCulate block has only one data stream through it. The CALCulate block is divided into smaller sub-blocks, which correspond to the subsystems found in the CALCulate subsystem.

Due to the discrete way in which each active sub-block exists, the data stream is accessible after each sub-block, where permitted by the implementation. The name of the CALCulate block and its numeric suffix is not sufficient to select the intermediate points. Therefore the <data\_handle> to access these intermediate points is comprised of the CALCulate block name and numeric suffix followed by a colon (:) and the name of a CALCulate sub-block. For example, if a CALCulate consists of MATH1, TRANSform, followed by MATH2, then the <data\_handle> to select the data flow after the TRANSform, but before MATH2, would be the string “CALCulate:TRANSform”

### 2.7.5 COMBine

The COMBine block acts to combine several data streams or signals. Each separate input to a COMBine block has a FEED command.

The output is the sum of the inputs. For example if the inputs are complex vectors representing waveforms with certain phases, then the ouput is the linear combination of the inputs which is also a complex vector.

Several different FEED commands might possibly specify the same <data\_handle>, in effect routing a single data stream into multiple subsystems.

### 2.7.6 Memory Associated with Data Flow

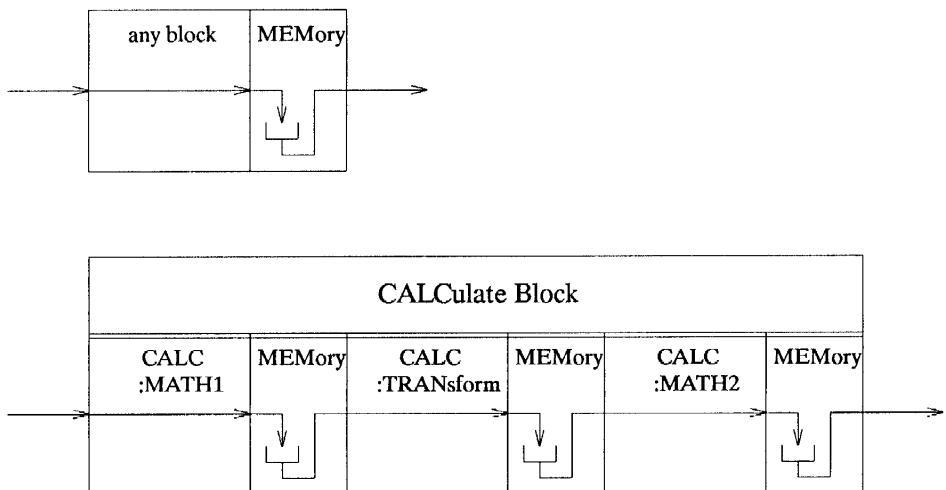
In the Model of a Programmable Instrument a MEMory block is shown adjacent to the other blocks in the model. For the user area memory this appears to be the correct model, however, for “instrument data” this proves to be cumbersome. A better model for “instrument data” is obtained by imagining that memory is something that pervades the whole model, any point along the data flow potentially may have memory associated with it. In general terms memory exists at all points along this flow of data.

The examples in Figure 2-12 show how memory is associated with the simplest of blocks, it then shows how memory is associated with a CALCulate block consisting of three sub-blocks. This is a very simple model to present to the user; at every point along the data flow memory exists. The fact that it exists, does not mean that a particular implementation has to allow the user access to it, as the data may not be in a form useful to the user. If access is provided to these intermediate memory stores, certain restrictions may apply depending on the memory usage. Within this model there are two types of memory:

- TRANSIENT - memory and data are retained only long enough to be used by next step in data flow

- STATIC - memory is permanently allocated and data is retained, until next “acquisition”

If the memory is transient then it is not possible to obtain data from it after the data has flowed past this point. Thus, querying such a point will not return anything, until new data flows into the newly created transient memory. This would occur if such a query were sent in the same program message as an INITiate command.



**Figure 2-12 MEMory Association with a Generic and a CALCulate Block**

Static memory is permanently allocated to the specified point in the data flow. Any data flowing past this point will fill this memory, and the data at that point may be queried at any time. However, should a new data flow start, for example with an INITiate, then the contents of the memory shall be replaced with the new data.

The names used in querying memory should reflect where they occur in the data flow. To best achieve this the “handles” given to these points for connecting blocks together are also appropriate in data query.

There is a third type of memory, user memory which is already defined in the TRACe|DATA subsystem.

Given the FEED mechanism it is possible for a TRACe name, such as MY-TRACE, to tap the data flow. Every time that new data flows past the “tapped” point, MY-TRACE will be updated with the new data. While this may be useful, a significant benefit can be achieved by providing a control mechanism that selectively allows data into the TRACe, upon a specified condition of events. For example, it should be possible to set up the control so that MY-TRACE will only receive new data when a limit test fails.

It can be seen that a user can create a TRACe, such as MY- TRACE, that may track the “tapped” point, but MY-TRACE and the memory at the tapped point are distinct and separate. Note that the memory at that point will always (while it exists) reflect the latest data flow, while MY-TRACE may not since the data flow into it may be gated. It is not permitted to create TRACe | DATA names identical to any data handles.

### 2 2.7.7

#### **Querying the Data Flow**

SCPI allows the data flow to be queried, so that a system controller can obtain intermediate results. This is only possible where the data flow is in an internal format, typically numeric data. That is, if the data flow is some physical characteristic such as an electrical signal, then the data flow at such a point is not queryable.

The form of the query is generally the block name followed by :DATA?. In certain cases, this either does not provide a unique data flow, or more resolution is required than just the block level. The SENSe:DATA? query has an optional function name as a parameter to define the particular data flow required out of the SENSe block. An implementation may allow access to the data flow after each sub-block subsystem in the CALCulate block. Thus, the query becomes the heirarchical name of the sub-block followed by :DATA?, for example the data flow after MATH1 could be obtained with CALCulate:MATH1:DATA?. The returned data is affected by the format subsystem.

### 2.7.8

#### **<event\_handle>**

The <event\_handle> is the way events are described that are used by the LINK commands throughout SCPI. Events may exist in all blocks of the Instrument Model, although a particular implementation will dictate what events are available to be used with each LINK command implemented.

The general form of <event\_handle> is:

**to be determined**

Events that are commonplace in instruments, are those found in the TRIGger subsystem; the action of exiting any layer generates an event. The <event\_handle>s to reference these TRIGger generated events are of the form:

“ARM[ :SEQUence[ :LAYER ] ]” or “TRIGger[ :SEQUence ]”

Other common examples include completion of a process in a block or an instrument state transition.

## Measurement Instructions

The purpose of the MEASure group of instructions is to acquire data using a set of high-level instructions. The MEASure group of instructions have a duality, in that they exhibit command and query characteristics. The exception to this is CONFigure, which has distinct query and command forms. These instructions are independent of the block diagram and refer to the characteristics of the signal being measured. These instructions are intended to be used with the measurement <functions> referenced later in this chapter.

The MEASure group of commands are structured to allow the user to trade off interchangeability with fine control of the measurement process. MEASure? provides a complete capability where the instrument is configured, a measurement taken, and results returned in one operation. Often, more precise control of the measurement is required. Therefore, MEASure? is complemented by providing two commands, CONFigure and READ?. CONFigure performs the configuration portion of the measurement and READ? performs the data acquisition, postprocessing, and data output portions of the measurement. This allows the user to perform a generic configuration of the measurement through CONFigure and then customize the measurement by changing particular instrument-dependent functions. The READ? then completes the measurement process.

READ?, in turn, is broken down into two additional commands INITiate[:IMMEDIATE] and FETCh?. INITiate[:IMMEDIATE] performs the data acquisition. This command is described in chapter 22. FETCh? performs the postprocessing function and returns the data. This allows the user to perform several different FETCh? functions on a single set of acquired data. For example, an oscilloscope can acquire measurement data that can yield many different signal characteristics such as frequency or AC and DC voltages. Thus, a transient signal may be captured once using a MEASure?, READ? or INITiate. A FETCh? may then be used to obtain each of the different signal characteristics without reacquiring a new measurement.

MEASure? provides the best compatibility between instruments because no knowledge of the instrument is required to perform the operation. CONFigure/READ? is less compatible if instrument reconfiguration is performed between the CONFigure and READ? operations. This is because the reconfiguration is, by necessity, instrument-specific. FETCh? is also less compatible because knowledge of the instrument is necessary to determine whether the necessary information has been captured by the instrument. For example, an oscilloscope can capture both rise time and pulse width in a single acquisition. Therefore, if pulse width was acquired through a MEASure? command, it would be possible to FETCh? rise time. However, if pulse width were measured with a counter, the rise time information might not be available without performing another data acquisition. Therefore FETCh? could not be used.

Changing certain parts of an instrument's configuration shall cause existing measurement data to be invalidated. Specifically, in figure 2-1, "Generalized Model of a Programmable Instrument" in chapter 2, "Instrument Model," any reconfiguration of signal routing, measurement function, signal generation and/or trigger shall cause existing readings to be invalidated. For example, the sequence:

```
INITiate; CONFIGure : VOLTage ; FETCh : VOLTage;
```

## 1999 SCPI Command Reference

would cause an error to be generated as the data was invalidated by the CONFigure command. Reconfiguring the display or format blocks shall not have this effect.

KEYWORD	PARAMETER FORM	NOTES
	CONFigure:<function> <parameters>[,<source list>]	
	FETCh[:<function>]? <parameters>[,<source list>]	[query only]
	READ[:<function>]? <parameters>[,<source list>]	[query only]
	MEASure:<function>?<parameters>[,<source list>]	[query only]

### 3 3.1

#### **CONFigure:<function> <parameters>[,<source list>]**

Sets up the instrument in order to perform the measurement specified by the function. Individual functions are described in <functions> later in this chapter. The parameters of this command are described in the individual function descriptions. The execution of the CONFigure command may affect the value of any other setting in the instrument. The state of the instrument is not specified after the execution of this command, except that a subsequent READ? QUERY operation would perform the specified function.

If parameters are omitted, they shall assume the values specified by the particular function description. Parameters may be defaulted from the right by omitting them, or anywhere by substituting the keyword DEFault. DEFault must be accepted as a parameter for this command.

The query form of the command, CONFigure?, returns the setup configured by the last CONFigure or MEASure? query. If the instrument state has changed through receiving commands other than CONFigure or MEASure?, the information returned by a CONFigure? may not reflect the actual measurement conditions. However, it is permissible for an instrument to attempt to track these changes and construct an accurate CONFigure query response. The format of the response data is <STRING RESPONSE DATA> which is of the form:

" <function> <parameters>"

For example, if the previous CONFigure command:

CONFIGURE:VOLTage:AC 5,.001

had been executed, the query CONFigure? would return:

"VOLT:AC 5.0,0.001"

A special parameter type is the <source list>. The <source list> is syntactically a <channel list> as described in "Syntax and Style." The purpose of the <source list> is to specify the physical port which is the source of the measurement. For example, (@2) would specify that the measurement should occur on channel 2. Multiple channels can be specified by using multiple channels in the channel list. For example, (@1,3:5,9) specifies that measurements are to be taken on channels 1,3,4,5, and 9. SCPI does not specify if the acquisitions are simultaneous or sequential. A function which requires multiple channels, such as time interval, would specify each channel as a separate list. For example, (@1),(@2) would specify a two-channel measurement between channels 1 and 2. A multiple channel, multiple measurement is specified by using multiple channels in each of the lists. For example,

(@1,2),(@3,4) specifies a pair of two-channel measurements, the first between channels 1 and 3 and the second between channels 2 and 4.

At \*RST, the configuration is set to a device-dependent value.

### 3.2

#### **FETCh[:<function>]? <parameters>[,<source list>]**

Retrieves the measurements taken by the INITiate command and places them into the device's output buffer. By specifying a function as part of a compound header, the device will retrieve the value of the function requested derived from the data taken by the latest INITiate command. For instance, even if a WAVEform was the original function when the measurement was taken, functions that can be calculated from the stored data can be FETChed. When <function> is omitted, the last <function> fetched, read, or measured shall be used.

The FETCh? query will return data any time that the last reading is valid. Data becomes invalid under the following conditions:

- When \*RST is executed,
- When an INITiate is executed.
- When there is any reconfiguration of signal routing, measurement function, signal generation and/or trigger blocks.
- When the sensor begins acquisition of a new reading.

If data is invalid, the FETCh? query shall not be complete until all data is valid. The exceptions to this are, if the instrument is in the trigger IDLE state and the data is invalid, or the instrument has been reconfigured as defined above and no new measurement has been initiated. In such cases, the FETCh? routine shall generate an error -230 and no result shall be returned. A common cause for this error is receiving a FETCh? after a \*RST.

Parameters are allowed in the FETCh? query. They may be defaulted from the right by omitting them, or in any other position by substituting the keyword DEFault. DEFault must be accepted as a parameter for this query. If a parameter is specified, it is used in the calculation if applicable. If the parameter is invalid because it conflicts with the acquired data, an error -221 shall be generated. If the instrument receives a parameter which is unexpected (too many parameters), it shall process the query and return the data. The unexpected parameter shall be ignored, and the "command warning" bit of the "data QUESTIONable" status register shall be set.

A special parameter type is the <source list>. The <source list> is syntactically a <channel list> as described in "Syntax and Style." The purpose of the <source list> is to specify the physical port which is the source of the measurement. For example, (@2) would specify that the measurement should occur on channel 2. Multiple channels can be specified by using multiple channels in the channel list. For example, (@1,3:5,9) specifies that measurements are to be taken on channels 1,3,4,5, and 9. SCPI does not specify if the acquisitions are simultaneous or sequential. A function which requires multiple channels, such as time interval, would specify each channel as a separate list. For example, (@1),(@2) would specify a two-channel measurement between channels 1 and 2. A multiple channel, multiple

measurement is specified by using multiple channels in each of the lists. For example, (@1,2),(@3,4) specifies a pair of two-channel measurements, the first between channels 1 and 3 and the second between channels 2 and 4.

The format of the returned data is determined by the FORMat command.

### 3.3

#### **READ[:<function>]? <parameters> [,<source list>]**

The READ? query is identical to:

```
ABORT;  
INITiate;  
FETCH[ :<function>]? <parameters>[,<source list>]
```

The READ? function provides a method of performing a FETCH? operation on fresh data.

A common application is to use the READ? in conjunction with a CONFigure command in order to provide a MEASure? capability in which the application programmer is allowed to provide fine adjustments to the instrument's block diagram. For example,

```
CONFigure:VOLTage:RISE:TIME 10 PCT, 90 PCT, 0.001 S  
SWEEp:TIME .05 S  
VOLTage:AC:RANGe 5 V  
READ:VOLTage:RISE:TIME?
```

Allows the applications programmer to specify the amplitude sensitivity and sweep time in order to provide the exact measurement needed.

If parameters are omitted, they are assumed to be those currently in use. Parameters may be defaulted from the right by omitting them, or anywhere by substituting the keyword DEFault. DEFault shall be accepted as a parameter for this query. If the instrument receives a parameter which is unexpected (too many parameters), it shall process the query and return the data, ignoring the unexpected parameter and shall also set the "command warning" bit of the "data QUEstionable" status register.

A special parameter set is the <source list>. The <source list> is syntactically a <channel list> as described in "Syntax and Style." The purpose of the <source list> is to specify the physical port which is the source of the measurement. For example (@2) would specify that the measurement should occur on channel 2. Multiple channels can be specified by using multiple channels in the channel list. For example, (@1,3:5,9) specifies that measurements are to be taken on channels 1,3,4,5, and 9. SCPI does not specify if the acquisitions are simultaneous or sequential. A function which requires multiple channels, such as time interval, would specify each channel as a separate list. For example, (@1),(@2) would specify a two-channel measurement between channels 1 and 2. A multiple channel, multiple measurement is specified by using multiple channels in each of the lists. For example, (@1,2),(@3,4) specifies a pair of two channel measurements, the first between channels 1 and 3 and the second between channels 2 and 4.

The format of the returned data is determined by the FORMat command.

If TRIGger:SOURce BUS is selected, execution of this query shall cause an error -214 to be generated and no data shall be returned. If ARM:SOURce BUS is selected, the same action shall occur, except that the error returned shall be -215.

### 3.4

#### **MEASure:<function>? <parameters>[,<source list>]**

The MEASure? query is a query identical to:

```
ABORT;
CONFIGure:<function> <parameters>;
READ:<function>? <parameters>[,<source list>];
```

The MEASure? query provides a complete measurement sequence, including configuration and reading of the data. MEASure? is used when the generic measurement is acceptable and fine adjustment of the instrument block diagram is unnecessary.

If parameters are omitted, they shall assume the values specified by the particular function description. Parameters may be defaulted from the right by omitting them, or anywhere by substituting the keyword DEFault. DEFault must be accepted as a parameter for this query. If the instrument receives a parameter which is unexpected (too many parameters), it shall process the query and return the data, ignoring the unexpected parameter and shall also set the “command warning” bit of the “Data questionable” status register.

A special parameter set is the <source list>. The <source list> is syntactically a <channel list> as described in “Syntax and Style.” The purpose of the <source list> is to specify the physical port which is the source of the measurement. For example (@2) would specify that the measurement should occur on channel 2. Multiple channels can be specified by using multiple channels in the channel list. For example, (@1,3:5,9) specifies that measurements are to be taken on channels 1,3,4,5, and 9. SCPI does not specify if the acquisitions are simultaneous or sequential. A function which requires multiple channels, such as time interval, would specify each channel as a separate list. For example, (@1),(@2) would specify a two-channel measurement between channels 1 and 2. A multiple channel, multiple measurement is specified by using multiple channels in each of the lists. For example, (@1,2),(@3,4) specifies a pair of two-channel measurements, the first between channels 1 and 3 and the second between channels 2 and 4.

The format of the returned data is determined by the FORMat command.

If TRIGger:SOURce BUS is selected, execution of this query shall cause an error -214 to be generated and no data shall be returned. If ARM:SOURce BUS is selected, the same action shall occur, except that the error returned shall be -215.

### 3.5

#### **<function>**

Functions define the measurement operation to be used by the MEASure?, CONFIGure, READ?, and FETCh? instructions and are used directly in combination with these instructions. The functions used in conjunction with the MEASure? instructions are different than those described in the SENSe chapter, in that the MEASure? functions are signal-oriented and potentially independent of the hardware.

## 1999 SCPI Command Reference

A function is described as a subtree under the CONFigure, READ?, FETCH?, and MEASure? instructions. The functions are layered into presentation, fundamental measurement, and measurement function layers as follows:

```
MEASure[:<presentation layer>]
    [:<fundamental measurement layer>]
        :<measurement function>?
```

For example, to measure an array of DC Voltage measurements, we specify the following:

```
MEASure:ARRay:VOLTage:DC?
```

The presentation layer defaults to a simple scalar measurement. The fundamental measurement layer has a device dependent default. For example, an oscilloscope would default VOLTage as its <fundamental measurement layer>. Therefore to perform a rise time measurement on an oscilloscope, the instrument shall accept any and all of the following commands:

```
MEASure:SCALAR:VOLTage:RISE:TIME?
MEASure:VOLTage:RISE:TIME?
MEASure:SCALAR:RISE:TIME?
MEASure:RISE:TIME?
```

Each layer of the function can have a set of parameters associated with it. Within each set, parameters default from the right. Any parameter in a set may be optional unless the command description states that the parameter is required. Further, any parameter which is allowed to be defaulted may be defaulted by substituting the keyword DEFault for the parameter.

Each set of parameters is separated from the others by enclosing them in parentheses. The exception is that the rightmost parameter set may omit the parentheses. Therefore the syntax is as follows:

```
MEASure:<presentation layer>
    :<fundamental measurement layer>
        :<measurement function>?
            [ [ [ (<presentation parameters>) ]
                , (<fundamental measurement parameters>) ]
                , <measurement function parameters> ]
            [ , <source list> ]
```

For example, if we wished to measure an array of voltage rise times, where we wished to acquire 100 readings, and the approximate voltage value was 5 V, and the expected 10-90 rise time was 5 ms, we could write the following query:

```
MEASure:ARRay:VOLTage:RISE:TIME? (100),(5 V),10,90,5 MS
```

The parameter group is only allowed when the corresponding header is part of the command. For example, the command:

```
MEASure:RISE:TIME? (5 V),10,90,5 MS
```

would generate an error because the fundamental measurement parameters have been specified without specifying the fundamental measurement layer in the header. However, as

the syntax shows, it is permissible to default groups of parameters from the left. Therefore it is possible to default the fundamental measurement parameters while specifying the fundamental measurement layer as follows:

```
MEASure:VOLTage:RISE:TIME? 10,90,5 MS
```

A majority of the functions share two parameters, expected value and resolution. **Expected value** is a guess as to the actual value of the measurement. It is used instead of range because range is a device-dependent concept, describing the actual sensor characteristics, while expected value is signal-oriented, describing the expected signal.

**Resolution** describes the absolute resolution of the measurement. If the parameter is omitted, the instrument selects a resolution. If the instrument cannot provide a measurement at the desired resolution, an error -231 is generated, but an answer is still returned if possible.

### 3.6

#### Presentation Layer

The purpose of the presentation layer is to handle groups of data and present them in the way the user specifies. This area is still under discussion, and therefore most of the presentation types (for example, histograms, time and frequency domain waveforms, and so on) are not included in this release.

##### 3.6.1

#### Presentation Command Summary

##### PRESENTATION KEYWORD

##### PARAMETER

[:SCALar]

:ARRay

<size>

##### 3.6.2

#### [:SCALar]

A SCALar or single value of the desired measurement is taken. SCALar has no associated parameters, and is a default node at this level.

##### 3.6.3

#### :ARRay <size>

An array of readings is specified. No domain information (time or frequency spacing) is implied in the record. There is one associated parameter, size. This specifies the number of readings to be taken. This parameter cannot be defaulted.

### 3.7

#### Fundamental Measurement Layer

The purpose of the fundamental measurement layer is to specify the fundamental signal characteristic being measured.

##### 3.7.1

#### Fundamental Measurement Command Summary

##### KEYWORD

##### PARAMETER

VOLTage

[<expected\_value>[,<resolution>]]

CURRent

[<expected\_value>[,<resolution>]]

POWER

[<expected\_value>[,<resolution>]]

RESistance

[<expected\_value>[,<resolution>]]

FRESistance

[<expected\_value>[,<resolution>]]

TEMPerature

[<transducer>[,<type>[,<expected value>[,<resolution>]]]]

- 3.7.2 :VOLTage[<expected\_value>[,<resolution>]]**  
Measures the voltage characteristics of the signal. If resolution is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified, either autoranging or a default range is selected.
- 3.7.3 :CURRent[<expected\_value>[,<resolution>]]**  
Measures the current characteristics of the signal. If resolution is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified, either autoranging or a default range is selected.
- 3.7.4 :POWer [<expected\_value>[,<resolution>]]**  
Measures the power characteristics of the signal. If resolution is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified, either autoranging or a default range is selected.
- 3.7.5 :RESistance[<expected\_value>[,<resolution>]]**  
Measures the resistance at the input. If resolution is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified, either autoranging or a default range is selected.
- 3.7.6 :FRESistance[<expected\_value>[,<resolution>]]**  
Measures the four-wire resistance at the input. If resolution is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified, either autoranging or a default range is selected.
- 3.7.7 :TEMPerature [<transducer>[,<type> [,<expected\_value>[,<resolution>]]]]**  
Measures the temperature at the sensor. The <transducer> parameter is character data which can take on the following values:

KEYWORD	DESCRIPTION
TCouple	Thermocouple
THERMistor	Thermistor
FTHermistor	Four-wire thermistor
RTD	Resistive temperature device
FRTD	Four-wire resistive temperature device

If a transducer is not specified, the instrument-dependent default is used.

The <type> parameter is either character data or <NRF> format data, depending on which transducer is specified. If TCouple is selected, the parameter is character data specifying the thermocouple type. Examples of valid values for thermocouple type are: J,K,R,S,T,B,E, and N14.

Thermistors and four-wire thermistors use the nominal resistance value for the type, as a numeric parameter. Typical values are 2250, 5000, and 10000.

RTDs and FRTDs use a <numeric\_value> which is specified by the manufacturer. Typical values are 85 and 92.

If a type is not specified, an instrument-dependent default is used.

If resolution is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected value> is not specified, either autoranging or a default range is selected.

### 3.8

## Measurement Function Layer

This layer presents the specific functions to be measured. This layer has been broken down into groups of measurements to provide clarity. This is only done for purposes of clarifying the presentation and there is no grouping provided in the tree.

A trailing :RATio node may be added to the Measurement Function layer, to request for a RATio measurement between the characteristics of two signals. The RATio taken, relates to the signal characteristics specified by the preceding nodes.

The <source\_list> consists of two <channel\_list> parameters if the RATio node is employed in the command. The first <channel\_list> parameter specifies the numerator and the second one is the denominator. For example, MEASure:FREQuency:RATio? (@1),(@2) measures the ratio of the signal frequency of channel 1 to the signal frequency on channel 2.

The parameters, specified with a ratio measurement command, relate to the signal characteristics of the first <channel\_list>; this is the signal which determines the numerator.

### 3.8.1

## Simple Measurements

The simple measurements set the fundamental characteristics of steady state signals.

### 3.8.1.1

## Simple Measurements Command Summary

KEYWORD	PARAMETER	
:AC		
[:DC]		
:FREQuency	[<expected_value>[,<resolution>]]	
:BURSt	[<expected_value>[,<resolution>]]	1992
:PRF	[<expected_value>[,<resolution>]]	1992
:PERiod	[<expected_value>[,resolution>]]	
:PHAse	[<expected_value>[,resolution>]]	1992

### 3.8.1.2

## :AC

Measures the AC RMS characteristic of the signal.

### 3.8.1.3

## [:DC]

Measures the DC level of the signal.

### 3.8.1.4

## :FREQuency [<expected\_value>[,<resolution>]]

Measures the frequency of the signal. If resolution is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified, either autoranging or a default range is selected.

### 3.8.1.4.1

## :BURSt [<expected\_value>[,<resolution>]]

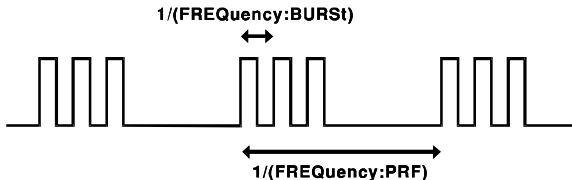
Measures the frequency of an ON-OFF modulated signal during its ON period. Such signals are often referred to as burst signals. The characteristic to be measured is the frequency inside the burst as is shown in Figure 3-1. If <resolution> is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified either auto ranging or a device dependent, particular range is selected.

**3.8.1.4.2 :PRF [<expected\_value>[,<resolution>]]**

Measures the repetition frequency of the ON-OFF modulation of a signal that is periodically switched ON and OFF. This characteristic is often referred to as Pulsed Repetition Frequency (PRF) as shown in Figure 3-1. If <resolution> is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified either autoranging or a default range is selected.

**3.8.1.5 :PERiod [<expected\_value>[,<resolution>]]**

The period (that is, 1/frequency) of a signal. If resolution is not specified, the instrument

**Figure 3-1 Burst Signal**

chooses a resolution based on its own tradeoffs. If <expected\_value> is not specified, either autoranging or a default range is selected.

**3.8.1.6 :PHASe [<expected value>[,<resolution>]]**

Measures the PHASe. If <resolution> is not specified, the instrument chooses a resolution based on its own tradeoffs. If <expected value> is not specified either auto ranging or a default range is selected.

If two channel lists are specified, the second shall be the reference.

**3.8.2 Time Domain Waveform Measurements**

These functions measure the time domain characteristics of pulsed signals.

Figure 3-2 is used to illustrate the terms being used to describe pulse signal characteristics.

**3.8.2.1 Waveform Measurements Command Summary****KEYWORD                    PARAMETER FORM**

:AMPLitude

:LOW

:HIGH

:RISE

:TIME                        [<lower reference>[,<upper reference>  
[,<expected\_value>[,<resolution>]]]]]

:OVERshoot

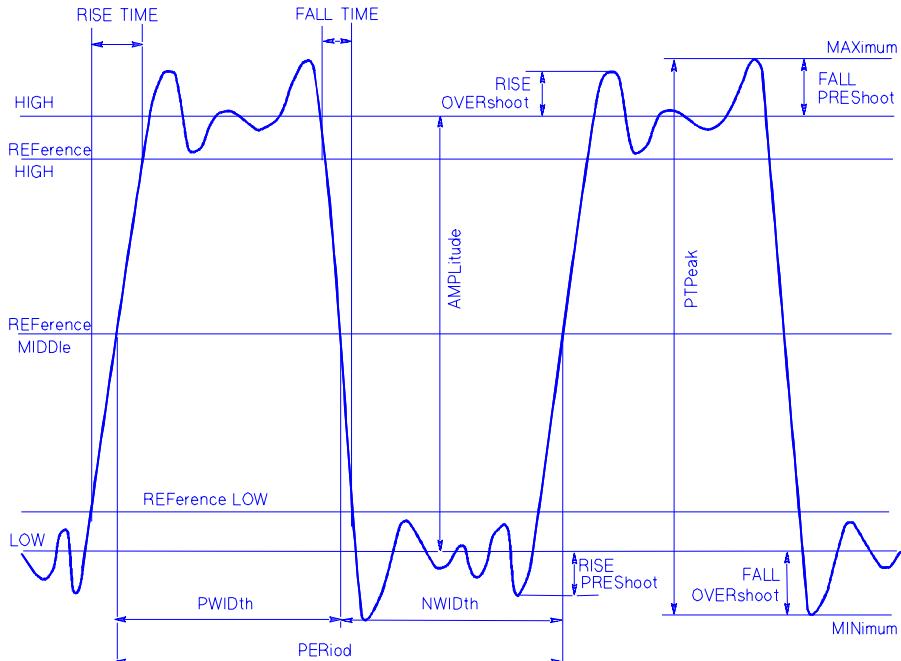
:PREShoot

<b>KEYWORD</b>	<b>PARAMETER FORM</b>
:RTIME	[<lower reference>[,<upper reference> [,<expected_value>[,<resolution>]]]]
:FALL	[<lower reference>[,<upper reference> [,<expected_value>[,<resolution>]]]]
:TIME	[<lower reference>[,<upper reference> [,<expected_value>[,<resolution>]]]]
:OVERshoot	[<lower reference>[,<upper reference> [,<expected_value>[,<resolution>]]]]
:PREShoot	[<lower reference>[,<upper reference> [,<expected_value>[,<resolution>]]]]
:FTIME	[<lower reference>[,<upper reference> [,<expected_value>[,<resolution>]]]]
:NWIDth	[<reference>]
:PWIDth	[<reference>]
:PDU <sub>T</sub> cycle :DCY <sub>C</sub> le	[<reference>]
:NDU <sub>T</sub> cycle	[<reference>]
:TMAXimum	
:TMINimum	
:MINimum	
:MAXimum	
:PTPeak	

1992

**3.8.2.2 :AMPLitude**

Amplitude is the signal amplitude calculated by taking HIGH - LOW.

**Figure 3-2 Pulse Measurement Terminology**

**3.8.2.3 :LOW**

This command specifies the LOW signal level. LOW is defined as the less positive of the BASE and TOP signal levels as defined in IEEE Std 194 sections 3.2.1 and 3.2.2.

**3.8.2.4 :HIGH**

The HIGH signal level. HIGH is defined as the more positive of the BASE and TOP signal levels as defined in IEEE Std 194 sections 3.2.1 and 3.2.2.

**3.8.2.5 :RISE**

This node specifies measurements on a rising edge of a waveform. This node has no associated parameters. This node specifies the first rising edge of the waveform.

**3.8.2.5.1 :TIME [<low reference>,<high reference> [<expected\_value>[,<resolution>]]]**

The time interval during which the instantaneous amplitude of a pulse increases from the low reference to high reference of the normal pulse amplitude.

This function is an alias to RTIME. If RISE:TIME is implemented, RTIME shall also be implemented. It is recommended that RISE:TIME also be implemented.

Default low and high references are 10% and 90% of normal high-signal level. These defaults may be changed via the parameters. The reference parameters have a default unit of percent. An implementation may allow the references to be entered in absolute sensor units. For example, an implementation may choose to allow the following syntax:

```
MEASure:VOLTage:RISE:TIME 0.2V,4.5V,.005S
```

to measure the 0.2 to 4.5 Volt rise time of a voltage signal.

**3.8.2.5.2 :OVERshoot**

The difference between the HIGH and the peak amplitude to which the instantaneous pulse waveform initially rises. This function uses the first edge of the waveform, expressed as a percentage of the waveform amplitude.

**3.8.2.5.3 :PRESHoot**

The difference between the LOW signal level and the negative peak signal value to which the waveform initially falls expressed as a percentage of waveform amplitude.

**3.8.2.6 :RTIME [<low reference>,<high reference> [<expected\_value>[,<resolution>]]]**

The time interval during which the instantaneous amplitude of a pulse increases from the low reference to high reference of the normal pulse amplitude.

RISE:TIME is an alias to this function. If RISE:TIME is implemented, RTIME shall also be implemented.

Default low and high references are 10% and 90% of normal high-signal level. These defaults may be changed via the parameters. The reference parameters have a default unit of percent. An implementation may allow the references to be entered in absolute sensor units. For example, an implementation may choose to allow the following syntax:

```
MEASure:VOLTage:RISE:TIME 0.2V,4.5V,.005S
```

to measure the 0.2 to 4.5 Volt rise time of a voltage signal.

### 3.8.2.7 :FALL

This node specifies measurements on a falling edge of a waveform. This node has no associated parameters. This node assumes the first falling edge of the waveform.

#### 3.8.2.7.1 :TIME [<low reference>,<high reference>]

[,<expected\_value>[,<resolution>]]]

The time interval during which the instantaneous amplitude of a pulse decreases from the high reference to low reference of the normal pulse amplitude.

This function is an alias to FTIMe. If FALL:TIME is implemented, FTIMe shall also be implemented. It is recommended that FALL:TIME also be implemented.

Default low and high references are 10% and 90% of normal high-signal level. These defaults may be changed via the parameters. The reference parameters have a default unit of percent. An implementation may allow the references to be entered in absolute sensor units. For example, an implementation may choose to allow the following syntax:

```
MEASure:VOLTage:FALL:TIME 0.2V,4.5V,.005S
```

to measure the 4.5 to 0.2 Volt fall time of a voltage signal.

### 3.8.2.7.2 :OVERshoot

The difference between the LOW and the negative peak amplitude to which the instantaneous pulse waveform initially falls, expressed as a percentage of the waveform amplitude.

### 3.8.2.7.3 :PRESHoot

The difference between the HIGH signal level and the positive peak signal value to which the waveform initially rises expressed as a percentage of waveform amplitude.

#### 3.8.2.8 :FTIMe [<low reference>,<high reference>]

[,<expected\_value>[,<resolution>]]]

The time interval during which the instantaneous amplitude of a pulse decreases from the high reference to low reference of the normal pulse amplitude.

FALL:TIME is an alias to this function. If FALL:TIME is implemented, FTIMe shall also be implemented.

Default low and high references are 10% and 90% of normal high-signal level. These defaults may be changed via the parameters. The reference parameters have a default unit of percent. An implementation may allow the references to be entered in absolute sensor units. For example, an implementation may choose to allow the following syntax:

```
MEASure:VOLTage:FALL:TIME 0.2V,4.5V,.005S
```

to measure the 4.5 to 0.2 Volt fall time of a voltage signal.

### 3.8.2.9 :PWIDth [<reference>]

The positive width expressed in seconds from the first rising edge reference to the next falling edge reference. The same reference is used for the rising and falling edges. The

default reference is defined as midway between HIGH and LOW levels. Other positive references may be specified by the parameter.

### 3.8.2.10 :NWIDth [<reference>]

The negative width expressed in seconds from the first falling edge reference to the next rising edge reference. The same reference is used for the rising and falling edges. The default reference is midway between HIGH and LOW levels. Other references may be specified by the parameter.

3

### 3.8.2.11 :PDUtcycle|:DCYCle [<reference>]

Positive dutycycle. The ratio of PWIDth to PERiod. The reference specifies the point on the pulse where the duty cycle is determined. This parameter defaults to 50%.

### 3.8.2.12 :NDUTcycle [<reference>]

Negative dutycycle. The ratio of NWIDth to PERiod. The reference specifies the point on the pulse where the duty cycle is determined. This parameter defaults to 50%.

### 3.8.2.13 :TMAXimum Layer:Time Domain Waveform Measurements

The time at which the first occurrence of the maximum voltage occurs.

### 3.8.2.14 :TMINimum

The time at which the first occurrence of the minimum voltage occurs.

### 3.8.2.15 :MINimum

The MINimum command calculates the minimum value of the waveform.

### 3.8.2.16 :MAXimum

The MAXimum command calculates the maximum value of the waveform.

### 3.8.2.17 :PTPeak

This command measures the Peak To Peak value of the signal which is (MAXimum - MINimum).

## 4 CALCulate Subsystem

The CALCulate subsystem exists to perform postacquisition data processing. Functions in the SENSe subsystem are related to data acquisition, while the CALCulate subsystem operates on the data acquired by a SENSe function.

The CALCulate subsystem is logically between the SENSe subsystem and data output to either the bus or the display. When a measurement is triggered by a MEASure command, an INITiate command, or meeting the prevailing TRIGger conditions, the SENSe subsystem collects data. This data is transformed by CALCulate, as specified, and then passed on to the selected output. In effect, the collection of new data triggers the CALCulate subsystem. The CALCulate subsystem may also be directed by command to perform a transform, making it possible to change the configuration of CALCulate and consequently derive a different set of results from the same SENSe data set without reacquiring sense data.

The CALCulate subsystem consists of a number of independent subsystems. Each of the subsystems is a sub-block of the CALCulate block. The data flows through the sub-blocks in a serial fashion, through the first sub-block, then onto the second sub-block and so on. The manner in which these sub-blocks are arranged is specified in the PATH command. It is permissible for a CALCulate block to have more than one instance of any of the sub-blocks. Instances of the same sub-block are differentiated by a numeric suffix. For example, two independent filters would exist as the :CALCulate:FILTter1 and the :CALCulate:FILTter2 subsystem sub-blocks.

Note: The notation  $X_n^*$  indicates the complex conjugate of a number. So:

$$XX^* = \text{MAG}^2(X) = |X|^2$$

KEYWORD	PARAMETER FORM	NOTES
CALCulate		
:AVERage		1992
:CLEAR		[no query] 1992
:COUNT	<numeric_value>	1992
:AUTO	<Boolean>   ONCE	1992
[:STATe]	<Boolean>	1992
:TCONtrol	EXPonential   MOVing   NORMal   REPeat	1992
:TYPE	COMPlex   ENVelope   MAXimum   MINimum   RMS   SCALar	1992
:CLIMits		1991
:FAIL?		[query only] 1991
:FLIMits		1991
[:DATA]?		[query only] 1991
:POINts?		[query only] 1991
:DATA?		[query only] 1995
:DATA		1995
:PREamble?		[query only] 1994
:DERivative		1993
:STATe	<Boolean>	1993

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:POINts	<numeric_value>	1993
:FEED	<data_handle>	1991
:FILTER		
[:GATE]		
:TIME		
:STATE	<Boolean>	1992
[:TYPE ]	BPASs NOTCh	
:STARt	<numeric_value>	
:STOP	<numeric_value>	
:SPAN	<numeric_value>	
:CENTer	<numeric_value>	
:POINts	<numeric_value>	
:AUTO	<Boolean> ONCE	
:WINDOW	RECTangular UNIForm FLATtop  HAMMing HANNing  KBESsel FORCe EXPonential	
:KBESsel	<numeric_value>	
:EXPonential	<numeric_value>	
:FORCe	<numeric_value>	
:FREQuency		
:STATE	<Boolean>	1992
[:TYPE ]	BPASs NOTCh	
:STARt	<numeric_value>	
:STOP	<numeric_value>	
:SPAN	<numeric_value>	
:CENTer	<numeric_value>	
:POINts	<numeric_value>	
:AUTO	<Boolean> ONCE	
:WINDOW	RECTangular UNIForm FLATtop  HAMMing HANNing  KBESsel FORCe EXPonential	
:KBESsel	<numeric_value>	
:EXPonential	<numeric_value>	
:FORCe	<numeric_value>	
:FORMat	NONE MLINear MLOGarithmic PHASe  REAL IMAGinary SWR GDELay  COMPlex NYQuist NICHols POLar  UPHase	1991 1991
:UPHase		1991
:CREference	<numeric_value>	1991
:PREference	<numeric_value>	1991
:GDAperture		
:SPAN	<numeric_value>	
:APERture	<numeric_value>	

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:IMMEDIATE		
:AUTO	<Boolean>	1992
:INTEGRAL		
:STATE	<Boolean>	1993
:TYPE	SCALAR   MOVING	1993
:LIMIT		
:STATE	<Boolean>	1993
:CONTROL		1991
[:DATA]	<numeric_value>{,<numeric_value>}	1991
:POINTS?		[query only] 1991
:UPPER		1991
[:DATA]	<numeric_value>{,<numeric_value>}	1991
:POINTS?		[query only] 1991
:STATE	<Boolean>	1991
:LOWER		1991
[:DATA]	<numeric_value>{,<numeric_value>}	1991
:POINTS?		[query only] 1991
:STATE	<Boolean>	1991
:FAIL?		[query only] 1991
:FCOUNT?		[query only] 1991
:REPORT		1991
[:DATA]?		[query only] 1991
:POINTS?		[query only] 1991
:CLEAR		1991
:AUTO	<Boolean>	1991
[:IMMEDIATE]		[no query] 1991
:INTERPOLATE	<Boolean>	1991
:MATH		
[:EXPRESSION]	<numeric_expression>	1991
:CATALOG?		<query only> 1993
[:DEFINE]	<numeric_expression>	1993
:DELETE		1993
[:SELECTED]	<expression_name>	1993
:ALL		1993
:NAME	<expression_name>	1993
:STATE	<Boolean>	1992
:SMOOTHING		
:STATE	<Boolean>	1992
:APERTURE	<numeric_value>	
:POINTS	<numeric_value>	
:STATE	<Boolean>	1992
:TRANSFORM		
:HISTOGRAM		1993
:COUNT	<numeric_value>	1993

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:ORDinate	RATio   PERCent   PCT   COUNT	1993
:POINts	<numeric_value>	1993
:RANGE		1993
:AUTO	<Boolean>	1993
:STATe	<Boolean>	1993
:TIME		
:STATe	<Boolean>	1992
[:TYPE ]	LPASs BPASs	
:STIMulus	STEP IMPulse	
:STARt	<numeric_value>	
:STOP	<numeric_value>	
:SPAN	<numeric_value>	
:CENTER	<numeric_value>	
:POINts	<numeric_value>	
:AUTO	<Boolean> ONCE	
:WINDOW	RECTangular UNIFORM FLATtop  HAMMING HANNing  FORCe EXPonential	
:KBESsel	<numeric_value>	
:EXPonential	<numeric_value>	
:FORCe	<numeric_value>	
:DISTance		
:STATe	<Boolean>	1992
[:TYPE ]	LPASs BPASs	
:STIMulus	STEP IMPulse	
:STARt	<numeric_value>	
:STOP	<numeric_value>	
:SPAN	<numeric_value>	
:CENTER	<numeric_value>	
:POINts	<numeric_value>	
:AUTO	<Boolean> ONCE	
:WINDOW	RECTangular UNIFORM FLATtop  HAMMING HANNing  FORCe EXPonential	
:KBESsel	<numeric_value>	
:EXPonential	<numeric_value>	
:FORCe	<numeric_value>	
:FREQency		
:STATe	<Boolean>	1992
[:TYPE ]	LPASs BPASs	
:STIMulus	STEP IMPulse	
:STARt	<numeric_value>	
:STOP	<numeric_value>	
:SPAN	<numeric_value>	

## 1999 SCPI Command Reference

KEYWORD	PARAMETER FORM	NOTES
:CENTer	<numeric_value>	
:POINTs	<numeric_value>	
:AUTO	<Boolean> ONCE	
:WINDOW	RECTangular UNIFORM FLATtop  HAMMING HANNING  KBESSel FORCe EXPonential	
:KBESSel	<numeric_value>	
:EXPonential	<numeric_value>	
:FORCe	<numeric_value>	
:PATH	(MATH TRANSform FILTER  SMOothing FORMAT  LIMIT  AVERage) {,(MATH TRANSform FILTER  SMOothing FORMAT  LIMIT  AVERage)}	1991 1992 [query only] 1991 1992

## 4.1

**:AVERage subsystem**

CALCulate:AVERage

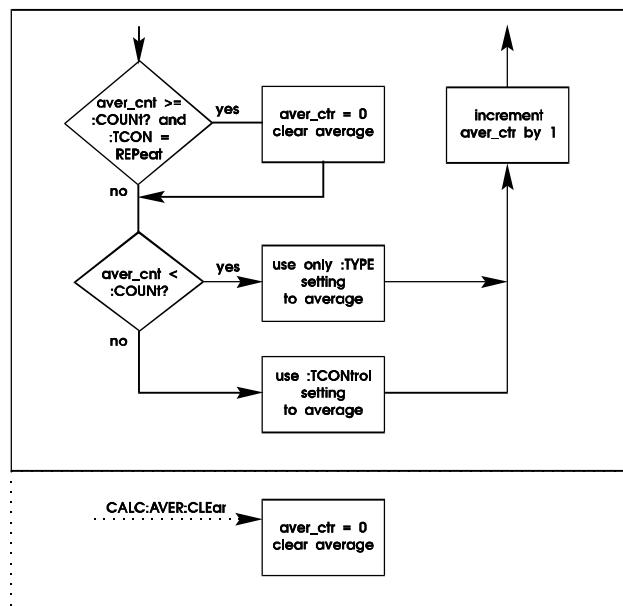
This subsystem provides an analytical post-processing function. The AVERage subsystem combines successive measurements to produce a new composite result. The new result has the same number of points and control axis as the original measurement.

A :COUNt command is provided to specify the number of measurements to combine. When :COUNT averages are reached, the TerminalCONtrol switch specifies the operation of the average subsystem. The TerminalCONtrol switch can be configured to continue replacing old values with new values (MOVing), to exponentially weight new measurements

(EXPonential), to automatically repeat (REPeat), or to continue adding new measurements in the NORMal fashon, as specified by :TYPE.

The average operation differs from CALC:SMOOthing in the fact that AVERage combines the jth point of each measurement with the jth point of preceeding measurements, whereas in SMOOthing the jth point is processed with adjacent points (...., j-2, j-1, j, j+1, j+2, ...) of the current measurement set.

A typical device action for CALC:AVERage



**Figure 4-1 AVERage device action**

## 4.1.1

**:CLEar**

CALCulate:AVERage:CLEar

This command causes the average data to be cleared and the average counter reset to zero.

**4.1.2 :COUNt <numeric\_value>**

CALCulate:AVERage:COUNt

Specifies the number of measurements to combine using the :TYPE setting. After :COUNt measurements have been averaged, the operation of the AVERage subsystem is controlled by the setting of the :TCONtrol node.

The count for array based results is independent of the number of points in the result array. For example, an instrument which normally provides a measurement result which is a 401 point array would specify an average count of two to average two results, not 802.

When averaging is ON, some devices may automatically set :COUNt values in the TRIGger subsystem based on the AVER:COUNt value, such that the TRIGger subsystem provides enough triggers for the average.

At \*RST, this value is device dependent.

**4.1.2.1 :AUTO <Boolean> | ONCE**

CALCulate:AVERage:COUNt:AUTO

AUTO ON causes the device to select a value for :COUNt which is appropriate for the current measurement. Selecting AUTO ONCE will have the effect of setting AUTO ON and then OFF.

At \*RST, this value is set to OFF.

**4.1.3 [:STATe] <Boolean>**

CALCulate:AVERage:STATE

The STATe command is used to turn averaging ON and OFF.

At \*RST, this value is set to OFF.

**4.1.4 :TCONtrol EXPonential | MOVing | NORMal | REPeat**

CALCulate:AVERage:TCONtrol

TerminalCONtrol specifies the action of the AVERage subsystem when more than AVERage:COUNt measurement results are generated.

The parameter has the following meanings:

**EXPonential** Continue the average with an exponential weighting applied to old values.  
The additional averages are weighted using:

For :TYPE SCALar when  $X_n$  is real and :TYPE COMPlex when  $X_n$  is either real or complex

$$AVG_n = \frac{1}{k} X_n + \frac{k-1}{k} AVG_{n-1}$$

For :TYPE SCALar when  $X_n$  is complex

$$AVG_n = \frac{1}{k} MAG(X_n) + \frac{k-1}{k} AVG_{n-1}$$

## 1999 SCPI Command Reference

For :TYPE RMS when X<sub>n</sub> is real

$$AVG_n = \sqrt{\frac{1}{k}X_n^2 + \frac{k-1}{k}AVG_{n-1}^2}$$

For :TYPE RMS when X<sub>n</sub> is complex

$$AVG_n = \sqrt{\frac{1}{k}X_nX_n^* + \frac{k-1}{k}AVG_{n-1}^2}$$

The exponential factor k is equal to the AVERage:COUNt value. Some instruments may use an approximation to true exponential.

Operation with EXPonential :TCON is undefined with a :TYPE setting of MINimum, MAXimum, or ENVelope.

MOVing As new data is added the oldest data is discarded.

REPeat Clear average data and counter and restart the average process.

NORMal Additional averages continue to be accumulated according to the :TYPE selection.

At \*RST, this value is device dependent.

### 4.1.5 :TYPE COMplex | ENVelope | MAXimum | MINimum | RMS | SCALar

CALCulate:AVERage:TYPE

This command selects the type of averaging, as follows:

COMplex The points in the summation are treated as real/imaginary pairs.

$$AVG(n) = \frac{1}{n} \sum_{i=1}^n X_i$$

ENVelope Both the MIN and MAX values are retained.

If X<sub>n</sub> is real

MAXimum AVG(n) = MAX(X<sub>1</sub>... X<sub>n</sub>)

MINimum AVG(n) = MIN(X<sub>1</sub>...X<sub>n</sub>)

RMS

$$AVG(n) = \sqrt{\frac{1}{n} \sum_{i=1}^n X_i^2}$$

SCALar The scalar magnitude of each point is averaged.

$$AVG(n) = \frac{1}{n} \sum_{i=1}^n X_i$$

If X<sub>n</sub> is complex

MAXimum AVG(n) = MAX( MAG(X<sub>1</sub>)... MAG(X<sub>n</sub>) )

MINimum       $\text{AVG}(n) = \text{MIN}(\text{MAG}(X_1) \dots \text{MAG}(X_n))$

RMS

$$\text{AVG}(n) = \sqrt{\frac{1}{n} \sum_{i=1}^n X_i X_i^*}$$

SCALar      The scalar magnitude of each point is averaged.

$$\text{AVG}(n) = \frac{1}{n} \sum_{i=1}^n |X_i|$$

At \*RST, this value is device dependent.

## 4.2 :CLIMits

CALCulate:CLIMits

This node defines commands to obtain Composite LIMit results within the specified CALCulate. It is permissible for CALCulate to contain two or more LIMit tests. These tests may be active simultaneously. The commands under CLIMits provide a mechanism to determine a summary of the PASS/FAIL information, and also a way to directly ascertain which particular LIMit tests are failing.

### 4.2.1 :FAIL?

CALCulate:CLIMits:FAIL?

FAIL is a summary of all the individual LIMit:FAILs. If any of the individual LIMit:FAILs have a non-zero value, indicating a failure, the composite FAIL query shall return a 1. If there are no failures then a 0 shall be returned. When all the failed LIMits have been CLEared, then the composite FAIL shall also be cleared.

### 4.2.2 :FLIMits

CALCulate:CLIMits:FLIMits

This node provides the information to determine the Failed LIMits, that is, identify how many and which LIMit tests have failed. When all the failed LIMits have been CLEared, then FLIMits shall also be cleared.

#### 4.2.2.1 [:DATA?]

CALCulate:CLIMits:FLIMits:DATA?

This query returns a number or list of numbers, each number corresponds to an instance of the LIMit subsystem within the CALCulate block. For example, if CALC:LIMit1, CALC:LIMit2 and CALC:LIMit3 existed, the query shall return 1,3, if LIMit1 and LIMit3 had failed. If there are no failures then NAN shall be returned.

#### 4.2.2.2 :POINts?

CALCulate:CLIMits:FLIMits:POINts?

This query shall return the number of failed LIMits in DATA. If there are no failures then a 0 shall be returned.

### 4.3

#### **:DATA?**

CALCulate:DATA?

Provides access to the result of the CALCulate subsystem. The query form outputs the present CALCulate results.

This command is query-only and therefore has no \*RST condition.

### 4.3.1

#### **:PREamble?**

CALCulate:DATA:PREamble?

PREamble? returns the preamble information supporting the DATA(CURVe(VALUes)) but omits the following data:

- DIF blocks of waveform, measurement, and delta.
- DIF curve block keywords VALUes and CSUM and their parameters.

The DIF difid block must contain the keyword SCOPe with the parameter of PREamble.

While the PREamble? query is designed to efficiently transmit data relating directly to the curve, the various notes and identification parameters may be sent if absolutely required.

### 4.4

#### **:DERivative**

CALCulate:DERivative

This subsystem calculates the derivative of the points in a data set. Each point in the differentiated data set is the derivative at the corresponding point in the original data set (note this requires some approximation at the end points). The units of the result of this process is the quotient of the currently selected amplitude units of the differentiated data and the units of the X-axis.

### 4.4.1

#### **:STATe <Boolean>**

CALCulate:DERivative:STATe

Determines whether the derivative function is enabled.

At \*RST this function is OFF

### 4.4.2

#### **:POINts <numeric\_value>**

CALCulate:DERivative:POINts

Specifies the number of points provided to the differential algorithm. They will be centered about the current data point.

At \*RST this value is device dependent.

### 4.5

#### **:FEED <data\_handle>**

CALCulate:FEED

Sets or queries the data flow to be fed into the CALCulate block.

At \*RST, FEED shall be set to a device dependent value.

**4.6 :FILT<sub>r</sub>**CALCulate:FILT<sub>r</sub>

This subsystem defines a filtering process to be performed on the SENSe data. A filter can operate in domains, which may or may not be the domain of the acquisition data.

**4.6.1 [:GATE]**CALCulate:FILT<sub>r</sub>:GATE

Specifies the or modification of the data set according to the selected parameters in a specific domain.

**4.6.1.1 :TIME**CALCulate:FILT<sub>r</sub>:GATE:TIME

This filter selection will modify the acquisition data to remove or modify information in a specified time region. Units of time are seconds.

**4.6.1.1.1 :STATe <Boolean>**CALCulate:FILT<sub>r</sub>:GATE:TIME:STATe

Determines whether the time filter is enabled.

At \*RST, this function is OFF.

**4.6.1.1.2 [:TYPE ] BPASs|NOTCh**CALCulate:FILT<sub>r</sub>:GATE:TIME:TYPE

Specifies what occurs to the data in the specific time region. The parameters have the following meanings:

- BPASs — Pass all information in specified time region and reject all else.
- NOTCh — Reject all information in specified time region and pass all else.

At \*RST, this value is device-dependent.

**4.6.1.1.3 :STARt <numeric\_value>**CALCulate:FILT<sub>r</sub>:GATE:TIME:STARt

Specifies the start time of the filter. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STARt is set to MIN.

**4.6.1.1.4 :STOP <numeric\_value>**CALCulate:FILT<sub>r</sub>:GATE:TIME:STOP

Specifies the stop time of the filter. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STOP is set to MIN.

**4.6.1.1.5 :SPAN <numeric\_value>**

CALCulate:FILTer:GATE:TIME:SPAN

Specifies the time span of the filter. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, SPAN is set to MIN.

**4.6.1.1.6 :CENTer <numeric\_value>**

CALCulate:FILTer:GATE:TIME:CENTER

Specifies the center time of the filter. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, CENTER is set to MIN.

**4.6.1.1.7 :POINts <numeric\_value>**

CALCulate:FILTer:GATE:TIME:POINTS

Specifies the number of points output by the filter subsystem.

At \*RST, this value is device-dependent.

**4.6.1.1.7.1 :AUTO <Boolean>|ONCE**

CALCulate:FILTer:GATE:TIME:POINTS:AUTO

Setting AUTO ON allows POINTs to be set by device-dependent parameters (for example, the size of incoming SENSe data).

At \*RST, AUTO is ON.

**4.6.1.1.8 :WINDow RECTangular|UNIFORM|FLATtop|HAMMING****|HANNing|KBESsel|FORCe|EXPonential**

CALCulate:FILTer:GATE:TIME:WINDow

Specifies the type and parameter of data windowing (shaping) done prior to the filter.

Individual windowing algorithms are defined in “*On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform*,” F. J. Harris, Proc. of the IEEE, Vol. 66-1, January 1978, pp 51-83.

At \*RST, this value is device-dependent.

**4.6.1.1.9 :KBESsel <numeric\_value>**

CALCulate:FILTer:GATE:TIME:KBESsel

The Kaiser BESsel command sets the parametric window parameter for the Kaiser Bessel window.

At \*RST, this value is device-dependent.

**4.6.1.1.10 :EXPonential <numeric\_value>**

CALCulate:FILTer:GATE:TIME:EXPonential

Enters the exponential decay time constant which characterizes the EXPonential window.

At \*RST, this value is device-dependent.

#### 4.6.1.1.11 :FORCe <numeric\_value>

CALCulate:FILTer:GATE:TIME:FORCe

Enters the time value parameter corresponding to the width of the gated portion of the input time record for FORCe windows.

At \*RST, this value is device-dependent.

#### 4.6.1.2 :FREQuency

CALCulate:FILTer:GATE:FREQuency

This filter selection will modify the acquisition data to remove or modify information in a specified frequency region.

Units are Hertz.

#### 4.6.1.2.1 :STATE <Boolean>

CALCulate:FILTer:GATE:FREQuency:STATE

Determines whether the frequency filter is enabled.

At \*RST, this function is OFF.

#### 4.6.1.2.2 [:TYPE] BPASs|NOTCh

CALCulate:FILTer:GATE:FREQuency:TYPE

Specifies what occurs to the data in the specific frequency region. The parameters have the following meanings:

- BPASs — Pass all information in specified time region and reject all else.
- NOTCh — Reject all information in specified frequency region and pass all else.

At \*RST, this value is device-dependent.

#### 4.6.1.2.3 :STARt <numeric\_value>

CALCulate:FILTer:GATE:FREQuency:STARt

Specifies the start frequency of the filter. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STARt is set to MIN.

#### 4.6.1.2.4 :STOP <numeric\_value>

CALCulate:FILTer:GATE:FREQuency:STOP

Specifies the stop frequency of the filter. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STOP is set to MIN.

**4.6.1.2.5 :SPAN <numeric\_value>**

CALCulate:FILTter:GATE:FREQuency:SPAN

Specifies the frequency span of the filter. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, SPAN is set to MIN.

**4.6.1.2.6 :CENTer <numeric\_value>**

CALCulate:FILTter:GATE:FREQuency:CENTER

Specifies the center frequency of the filter. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, CENTER is set to MIN.

**4.6.1.2.7 :POINts <numeric\_value>**

CALCulate:FILTter:GATE:FREQuency:POINts

Specifies the number of points output by the filter subsystem.

At \*RST, this value is device-dependent.

**4.6.1.2.7.1 :AUTO <Boolean>|ONCE**

CALCulate:FILTter:GATE:FREQuency:POINts:AUTO

Setting AUTO ON allows POINts to be set by device-dependent parameters (for example, the size of incoming SENSe data).

At \*RST, AUTO is ON.

**4.6.1.2.8 :WINDOW RECTangular|UNIFORM|FLATtop|HAMMING****|HANNing|KBESsel|FORCe|EXPonential**

CALCulate:FILTter:GATE:FREQuency:WINDOW

Specifies the type and parameter of data windowing (shaping) done prior to the filter.

Individual windowing algorithms are defined in “*On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform,*” F. J. Harris, Proc of the IEEE, Vol. 66-1, January 1978, pp 51-83.

At \*RST, this value is device-dependent.

**4.6.1.2.9 :KBESsel <numeric\_value>**

CALCulate:FILTter:GATE:FREQuency:KBESsel

The Kaiser BESsel command sets the parametric window parameter for the Kaiser Bessel window.

At \*RST, this value is device-dependent.

**4.6.1.2.10 :EXPonential <numeric\_value>**

CALCulate:FILTter:GATE:FREQuency:EXPonential

Enters the exponential decay time constant which characterizes the EXPonential window.

At \*RST, this value is device-dependent.

#### 4.6.1.2.11 :FORCe <numeric\_value>

CALCulate:FILTer:GATE:FREQuency:FORCe

Enters the time value parameter corresponding to the width of the gated portion of the input time record for FORCe windows.

At \*RST, this value is device-dependent.

#### 4.7

**:FORMat** **NONE|MLINear|MLOGarithmic|PHASe|REAL  
|IMAGinary|SWR|GDElay  
|COMPlex|NYQuist|NICHols|POLar  
|UPHase**

CALCulate:FORMat

Determines a simple post-processing of SENSe data. This subsystem is limited to simple point-by-point post processing. SENSe data description is  $x + jy$ , and if no  $y$  data, assume  $y=0$ . Regardless of the unprocessed data form, the data shall be interpreted as  $x + jy$  for the purposes of defining the formats in the table below.

NYQuist is an optional alias for COMPlex. If NYQuist is implemented, COMPlex shall also be implemented.

NAME	EQUATION	TYPE	
NONE	NONE	Unprocessed data	
MLINear	$\sqrt{x^2 + y^2}$	(scalar)	
MLOGarithmic	$10 \cdot \log(x^2 + y^2)$	(scalar, LOG is base 10)	
PHASe	$\arctan\left(\frac{y}{x}\right)$	(scalar in current angle units, $\pm 180$ degrees)	
IMAGinary	$y$	(linear scalar)	
REAL	$x$	(linear scalar)	
COMPlex NYQuist	pair (x,y)	(linear scalar)	1991
POLar	pair (r1, r2) $r1 = \sqrt{x^2 + y^2}$ $r2 = \arctan(y/x)$		1991 1991 1991
NICHols	pair (r1, r2) $r1 = 10 \cdot \log(x^2 + y^2)$ $r2 = \arctan(y/x)$		1991 1991 1991

## 1999 SCPI Command Reference

NAME	EQUATION	TYPE
SWR	$\frac{(1+r)}{(1-r)}$ WHERE $r = \text{MagLIN}(data)$	(scalar)
GDEDelay	$\frac{-d(\text{phase})}{360*d(\text{frequency})}$	(Scalar; units are seconds. Phase specified in current angle units. If current angle units are radians, replace the constant 360 with $2*\pi$ . The $d(\text{frequency})$ term is replaced with <code>CALCulate:GDA[P]erture:APERture</code> in practice.)
UPHase	UNWRAP(arctan(y/x))	(scalar in current angle units) 1991

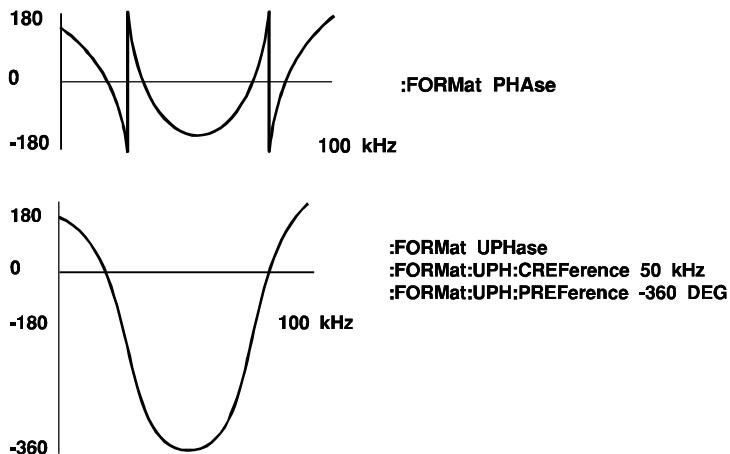
4

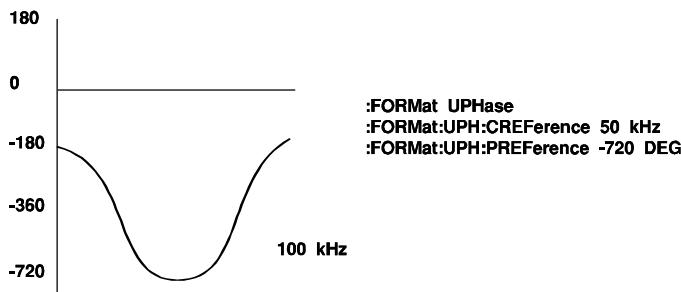
At \*RST, this value is device-dependent.

### 4.7.1 :UPHase

`CALCulate:FORMAT:UPHase`

The UPHase subsystem is used to specify how unwrapped phase is computed. A reference position is specified which is used as an estimate of what the unwrapped phase of calculated result should be. The calculated result is constrained to be within +/- 180 degrees of the reference position. Other points are calculated to produce a result which is compatible with the above constraint and does not “wrap” when cardinal points such as +/- 180 degrees are crossed. The reference position would typically be a well known point in a measurement such as the center of a band pass filter.





#### 4.7.1.1 :CREFerence <numeric\_value>

CALCulate:FORMAT:UPHase:CREFerence

The Control REFERENCE command designates the reference position on the independent dimension, called the control axis (typically the X axis) where the phase unwrap reference point is located. This value in combination with the :PREF value identifies the phase unwrap reference point.

At \*RST, this value is device dependent.

#### 4.7.1.2 :PREFerence <numeric value>

CALCulate:FORMAT:UPHase:PREFerence

The Phase REFERENCE command designates the phase of the phase unwrap reference point. This value in combination with the :CREFerence value identifies the reference point. The CALC result is constrained to be within +/- 180 degrees of this value at the point corresponding to the :CREFerence position.

At \*RST, this value is device dependent.

### 4.8 :GDAPerture

CALCulate:GDAPerture

Group Delay Aperture is defined as the negative of the derivative of phase with respect to frequency. The group delay format requires specifying an aperture over which the measurement is made. Since the frequency span of the measurement is known, the frequency step over the entire span can be thought of as  $d(freq)$ . This function permits the setting of that aperture. APERture is tightly coupled to SPAN by the equation:

$$APERture = GDAPerture:SPAN / (SENSe/SOURce):FREQuency:SPAN$$

#### 4.8.1 :SPAN <numeric\_value>

CALCulate:GDAPerture:SPAN

Specifies the aperture in Hertz. Minimum value is instrument-dependent.

At \*RST, this function is set to MIN.

#### 4.8.2 :APERture <numeric\_value>

CALCulate:GDAPerture:APERture

Specifies the aperture as a ratio of desired aperture span/measured frequency span.

At \*RST, this value is set to MIN.

4.9

**:IMMEDIATE**

CALCulate:IMMEDIATE

Causes the CALCulate subsystem to reprocess existing SENSe data without reacquiring the data. This permits the user to change a CALCulate subsystem, setting and output new results without initiating a data acquisition.

Note that CALC:IMMEDIATE? is semantically equivalent to CALC:IMM;DATA?. The query form outputs the results of the new calculation.

This command describes an event and therefore has no associated \*RST condition.

4

4.9.1

**:AUTO <Boolean>**

CALCulate:IMMEDIATE:AUTO

The AUTO command is used to specify if the CALCulate subsystem continually transforms the data (AUTO ON) whenever any changes are made which would affect the CALCulate subsystem output. With CALC:IMM:AUTO set to OFF, CALCulate only produces new data when data is received or when the CALC:IMMEDIATE command is received. Once CALC:IMM:AUTO is set to ON, the CALCulate Subsystem shall produce results when any CALC subsystem command is processed, even when new data is not being received. This allows the user to make configuration changes in the CALCulate subsystem and immediately have new CALC:DATA results on the same sensed data.

At \*RST, this value is set to OFF.

4.10

**:INTegral**

CALCulate:INTegral

This subsystem provides a point to point integration of a data set. The result is an estimate of the integral of the input. The units of the result of this process is the product of the currently selected amplitude units of the integrated data and the units of the X- axis.

4.10.1

**:STATE <Boolean>**

CALCulate:INTegral:STATE

Determines whether the integrate function is enabled.

At \*RST this function is OFF

4.10.2

**:TYPE SCALar | MOVing**

CALCulate:INTegral:TYPE

Specifies whether the result is a single value (SCALar) or a set of data points (MOVing). If SCALar is selected the result is an estimate of the definite integral over the entire data set. If MOVing is selected each point in the integrated data set is the integral up to the corresponding point in the original data set.

At \*RST this function is set to a device dependent value.

4.11

**:LIMit**

CALCulate:LIMit

The LIMit node collects together the commands associated with controlling and getting reports from a single LIMit test. The LIMit test may defined as an UPPer limit, a LOWER

limit, or both. A LIMit test may be defined as being scalar, that is UPPer or LOWER are provided a single value each.

Alternatively, UPPer or LOWER may be defined as a series of points (a vector) representing a waveform limit. For such a vector limit test, each point in the series (vector) must be qualified by where it occurs with respect to the control (typically the X) axis. The CONTrol node is used to define the points along the control axis. UPPer and LOWER represent the dependent (typically the Y) axis, and are given values corresponding to these control axis points. That is, the first value for UPPer and LOWER corresponds to the first value of CONTrol, the second value for UPPer and LOWER corresponds to the second value of CONTrol, and so on. UPPer, LOWER and CONTrol shall contain the same number of points, or an error shall be generated when data processing commences. For scalar limit tests, as described earlier, the CONTrol subsystem is not used.

If the data and the limit are scalar (single values), then LIMit shall produce results for each piece of scalar data tested. If the data and the limit are both vectors, then LIMit shall produce results for each vector data tested. If the limit is scalar and the data is a vector, then LIMit shall perform a the limit test by treating the scalar limit as an equivalent vector (all points having the same value).

#### 4.11.1 :STATe <Boolean>

CALCulate:LIMit:STATe

Is used to set or query if the LIMit test is active or not.

At \*RST, this is set to OFF.

#### 4.11.2 :CONTrol

CALCulate:LIMit:CONTrol

This node collects the commands and queries used to set up the control axis of the limit data, where the limit test is not based on a single point.

##### 4.11.2.1 [:DATA]<numeric\_value>{,<numeric\_value>}

CALCulate:LIMit:CONTrol:DATA

Sets or queries the CONTrol axis data.

##### 4.11.2.2 :POINts?

CALCulate:LIMit:CONTrol:POINts?

The POINts? query returns the number of points currently in CONTrol:DATA.

#### 4.11.3 :UPPer

CALCulate:LIMit:UPPer

UPPer is used to define the upper limit to be used in the LIMit test. When the data is greater than the value specified in UPPer, LIMit shall report a fail. That is, when the data is equal to the limit, a fail shall not be reported.

##### 4.11.3.1 [:DATA]<numeric\_value>{,<numeric\_value>}

CALCulate:LIMit:UPPer:DATA

Sets or queries the UPPer axis data.

**4.11.3.2 :POINts?**

CALCulate:LIMit:UPPer:POINts?

The POINts? query returns the number of points currently in UPPer:DATA.

**4.11.3.3 :STATe <Boolean>**

CALCulate:LIMit:UPPer:STATe

Sets or queries if the individual LIMit test is enabled.

At \*RST, this is set to ON.

**4.11.4 :LOWer**

CALCulate:LIMit:LOWer

LOWer is used to define the lower limit to be used in the LIMit test. When the data is less than the value specified in LOWer, LIMit shall report a fail. That is, when the data is equal to the limit, a fail shall not be reported.

**4.11.4.1 [:DATA] <numeric\_value>{,<numeric\_value>}**

CALCulate:LIMit:LOWer:DATA

Sets or queries the LOWer axis data.

**4.11.4.2 :POINts?**

CALCulate:LIMit:LOWer:POINts?

The POINts? query returns the number of points currently in LOWER:DATA.

**4.11.4.3 :STATe <Boolean>**

CALCulate:LIMit:LOWer:STATe

Sets or queries if the individual LIMit test is enabled.

At \*RST, this is set to ON.

**4.11.5 :FAIL?**

CALCulate:LIMit:FAIL?

This query returns a 0 or 1, to indicate if the LIMit test has failed or not; 0 represents pass and 1 represents fail.

**4.11.6 :FCOunt?**

CALCulate:LIMit:FCOunt?

This Fail COunt query returns the number of times that the LIMit test has failed. With a 0 indicating no failures. With vector limits and vector data, only one failure shall be recorded with each (measurement) vector that is tested, even if many points are outside the limits.

**4.11.7 :REPort**

CALCulate:LIMit:REPort

This node collects together commands to get a more detailed information on vector limit tests.

**4.11.7.1 [:DATA]?**

CALCulate:LIMit:REPort:DATA?

This query returns a number or list of numbers, each number corresponds to a value of the CONtrol variable where either the UPPer or LOWER limit test has failed. Note that: When

INTerpolate ON has been selected, the returned DATA shall correspond to the points along the CONTrol axis where a failure has occurred, which may or may not coincide with points defined for CONTrol. If there are no failures then NAN shall be returned.

#### 4.11.7.2 :POINts?

CALCulate:LIMit:REPort:POINts?

This query shall return the number of points in the DATA. If there are no failures then a 0 shall be returned.

#### 4.11.8 :CLEAR

CALCulate:LIMit:CLEar

The CLEAR node collects together the commands that are used to clear the information in DATA, FAIL and REPort.

#### 4.11.8.1 :AUTO <Boolean> | ONCE

CALCulate:LIMit:CLEar:AUTO

Sets or queries if the information is to be CLEared with each INITiate command, or more precisely when the TRIGger state machine exits the “IDLE state.” Such an INITiate shall cause the information in FAIL, FCOunt and REPort to be cleared if AUTO is set to ON. If AUTO is set to OFF, no such effect shall occur.

At \*RST, AUTO shall be set to ON.

#### 4.11.8.2 [:IMMEDIATE]

CALCulate:LIMit:CLEar:IMMEDIATE

IMMEDIATE shall cause the information in FAIL, FCOunt and REPort to be cleared IMMEDIATEly.

#### 4.11.9 :INTERPOLate <Boolean>

CALCulate:LIMit:INTERPOLate

Sets or queries whether or not limit tests will be performed at points between those specified by the CONTrol variable. The interpolation shall be straight line between points, however the algorithm determining the number of samples between points is device dependent.

At \*RST, it shall be OFF.

#### 4.12 :MATH

CALCulate:MATH

This subsystem permits processing of sense data in numerical expression format. The operators are +, -, \*, / and use of constants and data arrays are permitted.

#### 4.12.1 [:EXPReSSion] <numeric\_expression>

CALCulate:MATH:EXPReSSion

This subsystem controls which expression is used by the CALCulate block.

#### 4.12.1.1 :CATalog?

CALCulate:MATH:EXPReSSion:CATalog?

The CATalog query lists all the defined expressions. Response is a list of comma-separated strings. Each string contains the name of an expression. If no expressions are currently defined, the response is a null string ("").

### 4.12.1.2 [:DEFIne] <numeric\_expression>

CALCulate:MATH:EXPReSSion:DEFIne

Defines the expression used for the math operations. The <numeric\_expression> may contain any of the following as operands and they may occur in any combination:

- A <trace\_name>
- The reserved keyword IMPLied
- The <data\_handle> associated with another block

The reserved keyword IMPLied, refers to the IMPLied output of the previous CALCulate sub-block. The ordering of the sub-blocks is specified by the PATH command. If the MATH subsystem appears first in the PATH, then the IMPLied keyword refers to the <data\_handle> specified with the FEED command.

For example, to subtract a reference trace from the SENSe data, the following command would configure the MATH appropriately.

```
CALCulate:MATH:EXPReSSion ( IMPLied - MY-TRACE )
```

This example assumes that the FEED command has been configured to select the appropriate SENSe <data\_handle>. The IMPLied CALCulate data flow can be ignored and the SENSe data may be used directly in a <numeric\_expression>:

```
CALCulate:MATH:EXPReSSion ( "SENSe1:POWeR 3" - MY-TRACE )
```

At \*RST, this function is undefined.

### 4.12.1.3 [:DELete]

CALCulate:MATH:EXPReSSion:DELete

The DELetes subsystem deletes expressions.

#### 4.12.1.3.1 [:SELected] <expression\_name>

CALCulate:MATH:EXPReSSion:DELete:SELected

This command causes the selected expression to be deleted.

#### 4.12.1.3.2 [:ALL]

CALCulate:MATH:EXPReSSion:DELete:ALL

This command causes the all expressions to be deleted.

#### 4.12.1.4 [:NAME] <expression\_name>

CALCulate:MATH:EXPReSSion:NAME

The NAME command defines the name of the expression to be selected. If the expression name, <expression\_name>, already exists, then that existing expression shall be selected. If the expression name does not exist, then the new name shall be selected, but no expression shall be defined by this selection. <expression\_name> is character data.

At \*RST the value of NAME is device dependent.

**4.12.2 :STATe <Boolean>**

CALCulate:MATH:STATe

Determines whether math processing is done.

At \*RST, this function is OFF.

**4.13 :SMOothing**

CALCulate:SMOothing

This subsystem specifies point-to-point smoothing of a data set. A data point in a smoothed data set is the average of adjacent points from a single original data set. Smoothing differs from averaging, in which a given point is the average of corresponding points from multiple data sets or acquisitions.

**4.13.1 [:STATe] <Boolean>**

CALCulate:SMOothing:STATe

Determines whether the smoothing algorithm is enabled.

At \*RST, this function is OFF.

**4.13.2 :APERture <numeric\_value>**

CALCulate:SMOothing:APERture

Specifies the size of the smoothing APERture as a ratio of smoothing aperture points/trace points.

At \*RST, this function is device-dependent.

**4.13.3 :POINts <numeric\_value>**

CALCulate:SMOothing:POINts

Controls the number of points to be included in the running average. POINts is coupled to APERture by the equation:

$$POINts = APERture * TRACe:POINTS$$

**4.14 :STATe <Boolean>**

CALCulate:STATe

Controls whether postprocessing is enabled. If disabled, this subsystem is effectively transparent.

At \*RST, this is set ON.

**4.15 :TRANSform**

CALCulate:TRANSform

TRANSform is the subsystem that permits sense data to be transformed from one domain to another. Examples are from frequency domain to time domain or time domain to frequency domain. This is different from a SENSe:FUNC selection of domain because in SENSe, the data is actually collected in the selected domain, while in CALCulation:TRANSform, the data is computed, for example, by use of Fourier Transform.

**4.15.1 :HISTogram**

CALCulate:TRANSform:HISTogram

This subsystem transforms the data into a histogram. The histogram provides an amplitude distribution of the incoming signal. The points in a histogram specify the ratio between the number number of times a data point is within a particular amplitude belt and the total number of incoming data points that participate in the transformation. The ratio is expressed as a percentage.

4

The resolution of the histogram is specified by the number of points of the output signal. It determines the amplitude belt wherein the data points are counted, using the equation:

$$\text{amplitude belt} = \text{amplitude range} / \text{histogram points}$$

The units of the amplitude range are the currently selected amplitude units of the data to be transformed.

**4.15.1.1 :COUCount <numeric\_value>**

CALCulate:TRANSform:HISTogram:COUNT

This specifies the number of measurements to include in the histogram.

For scalar measurements, COUNT is the number of incoming data points that are used to create the histogram.

The COUNT for array based results is independent of the number of points in the result array. For example, an instrument which normally provides a measurement result which is a 401 point array would specify a histogram count of two to create a histogram based on 802 data points.

At \*RST this value is device dependent

**4.15.1.2 :ORDinate RATio | PERCent | PCT | COUNt**

CALCulate:TRANSform:HISTogram:ORDinate

When ORDinate is set to RATio, the output of the HISTogram function is an absolute ratio. When ORDinate is set to PERCent, the output of the HISTogram function is PERCent (RATio\*100). When ORDinate is set to COUNt, the output of the HISTogram function is the number of points in that amplitude belt.

PCT is provided as a synonym for PERCent since IEEE 488.2 allows the use of PCT as a numeric suffix.

At \*RST this value is device dependent.

**4.15.1.3 :POINts <numeric\_value>**

CALCulate:TRANSform:HISTogram:POINts

Specifies the number of POINts (amplitude belts) in the histogram.

At \*RST this value is device dependent.

**4.15.1.4 :RANGE**

CALCulate:TRANSform:HISTogram:RANGE

This node collects the commands used to control the size of the amplitude belts.

**4.15.1.4.1 :AUTO <Boolean>**

CALCulate:TRANSform:HISTogram:RANGE:AUTO

With AUTO ON, the number of output points is determined by the amplitude of the incoming data via a device dependent algorithm.

At \*RST this function is set to a device dependent value.

**4.15.1.5 :STATe <Boolean>**

CALCulate:TRANSform:HISTogram:STATe

Determines whether the histogram transformation is enabled.

At \*RST this function is OFF

**4.15.2 :TIME**

CALCulate:TRANSform:TIME

Specifies a transformation of data into its equivalent time representation. The parameters specify the time region over which the output data is transformed. The units are seconds.

**4.15.2.1 :STATe <Boolean>**

CALCulate:TRANSform:TIME:STATe

Determines whether the time transform is enabled.

At \*RST, this function is OFF.

**4.15.2.2 [:TYPE ] LPAsS|BPAsS**

CALCulate:TRANSform:TIME:TYPE

Selects a particular method to be used in band limiting information or the manner in which windows should be applied.

When LPAsS is selected, the windowing shall affect only the end of the data set. LPAsS requires data to start from zero in the appropriate units, such as Hertz and seconds. If the data set does not extend to zero, then the instrument shall either compute by extrapolation the extra points needed to complete the data set, or shall generate an execution (-200) error. When BPAsS is selected, the applied window shall affect both the start and end parts of the data set.

At \*RST, this value is device-dependent.

**4.15.2.3 :STIMulus STEP|IMPulse**

CALCulate:TRANSform:TIME:STIMulus

Specifies the type of stimulus to be simulated in the transform process.

At \*RST, this value is device-dependent.

**4.15.2.4 :STARt <numeric\_value>**

CALCulate:TRANSform:TIME:STARt

Specifies the start time of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STARt is set to MIN.

**4.15.2.5 :STOP <numeric\_value>**

CALCulate:TRANSform:TIME:STOP

Specifies the stop time of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STOP is set to MIN.

**4.15.2.6 :SPAN <numeric\_value>**

CALCulate:TRANSform:TIME:SPAN

Specifies the time span of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, SPAN is set to MIN.

**4.15.2.7 :CENTer <numeric\_value>**

CALCulate:TRANSform:TIME:CENTER

Specifies the center time of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, CENTER is set to MIN.

**4.15.2.8 :POINts <numeric\_value>**

CALCulate:TRANSform:TIME:POINts

Specifies the number of points output by the transform subsystem.

At \*RST, this value is device-dependent.

**4.15.2.8.1 :AUTO <Boolean>|ONCE**

CALCulate:TRANSform:TIME:POINts:AUTO

When AUTO is ON, the number of points is determined by the size of the incoming SENSe Data.

At \*RST, AUTO is ON.

**4.15.2.9 :WINDOW RECTangular|UNIForm|FLATtop|HAMMING  
|HANNing|KBESsel|FORCe|EXPonential**

CALCulate:TRANSform:TIME:WINDOW

Specifies the type and parameter of data windowing (shaping) done prior to the transformation.

At \*RST, this value is device-dependent.

Individual windowing algorithms are defined in “*On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform*,” F. J. Harris, Proc. of the IEEE, Vol. 66-1, January 1978, pp 51-83.

**4.15.2.10 :KBESsel <numeric\_value>**

CALCulate:TRANSform:TIME:KBESsel

The Kaiser BESsel command sets the parametric window parameter for the Kaiser Bessel window.

At \*RST, this value is device-dependent.

**4.15.2.11 :EXPonential <numeric\_value>**

CALCulate:TRANSform:TIME:EXPonential

Enters the exponential decay time constant which characterizes the EXPonential window.

At \*RST, this value is device-dependent.

4

**4.15.2.12 :FORCe <numeric\_value>**

CALCulate:TRANSform:TIME:FORCe

Enters the time value parameter corresponding to the width of the gated portion of the input time record for FORCe windows.

At \*RST, this value is device-dependent.

**4.15.3 :DISTance**

CALCulate:TRANSform:DISTance

This subsystem specifies a transformation of data into its equivalent distance representation. The parameters specify the distance region over which the output data is transformed. The units are meters.

**4.15.3.1 :STATe <Boolean>**

CALCulate:TRANSform:DISTance:STATE

Determines whether the distance transform is enabled.

At \*RST, this function is OFF.

**4.15.3.2 [:TYPE] LPASs|BPASs**

CALCulate:TRANSform:DISTance:TYPE

Selects a particular method to be used in limiting information or the manner in which windows shall be applied for the specified transform.

When LPASs is selected, the windowing shall affect only the end of the data set. LPASs requires data to start from zero in the appropriate units, such as Hertz and seconds. If the data set does not extend to zero, then the instrument shall either compute by extrapolation the extra points needed to complete the data set, or shall generate an execution (-200) error. When BPASs is selected, the applied window shall affect both the start and end parts of the data set.

At \*RST, this value is device-dependent.

**4.15.3.3 :STIMulus STEP|IMPulse**

CALCulate:TRANSform:DISTance:STIMulus

Specifies the type of stimulus to be simulated in the transform process.

At \*RST, this value is device-dependent.

**4.15.3.4 :STARt <numeric\_value>**

CALCulate:TRANSform:DISTance:STARt

Specifies the start distance of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STARt is set to MIN.

**4.15.3.5 :STOP <numeric\_value>**

CALCulate:TRANSform:DISTance:STOP

Specifies the stop distance of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STOP is set to MIN.

**4.15.3.6 :SPAN <numeric\_value>**

CALCulate:TRANSform:DISTance:SPAN

Specifies the distance span of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, SPAN is set to MIN.

**4.15.3.7 :CENTer <numeric\_value>**

CALCulate:TRANSform:DISTance:CENTER

Specifies the center distance of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, CENTER is set to MIN.

**4.15.3.8 :POINts <numeric\_value>**

CALCulate:TRANSform:DISTance:POINTs

Specifies the number of points output by the transform subsystem.

At \*RST, this is set to a device-dependent value.

**4.15.3.8.1 :AUTO <Boolean>|ONCE**

CALCulate:TRANSform:DISTance:POINTs:AUTO

When AUTO is ON, the number of points is determined by the size of the incoming SENSe data.

At \*RST, AUTO is ON.

**4.15.3.9 :WINDOW RECTangular|UNIFORM|FLATtop|HAMMING  
|HANNing|KBESsel|FORCe|EXPonential**

CALCulate:TRANSform:DISTance:WINDOW

Specifies the type and parameter of data windowing (shaping) done prior to the transformation.

At \*RST, this value is device-dependent.

Individual windowing algorithms are defined in “*On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform*,” F. J. Harris, Proc. of the IEEE, Vol. 66-1, January 1978, pp 51-83.

**4.15.3.10 :KBESsel <numeric\_value>**

CALCulate:TRANSform:DISTance:KBESsel

The Kaiser BESsel command sets the parametric window parameter for the Kaiser Bessel window.

At \*RST, this value is device-dependent.

**4.15.3.11 :EXPonential <numeric\_value>**

CALCulate:TRANSform:DISTance:EXPonential

Enters the exponential decay time constant which characterizes the EXPonential window.

At \*RST, this value is device-dependent.

**4.15.3.12 :FORCe <numeric\_value>**

CALCulate:TRANSform:DISTance:FORCe

Enters the time value parameter corresponding to the width of the gated portion of the input time record for FORCe windows.

At \*RST, this value is device-dependent.

**4.15.4 :FREQuency**

CALCulate:TRANSform:FREQuency

This subsystem specifies a transformation of data into its equivalent frequency representation. The parameters specify the frequency region over which the output data is transformed. The units are Hertz.

**4.15.4.1 :STATe <Boolean>**

CALCulate:TRANSform:FREQuency:STATe

Determines whether the frequency transform is enabled.

At \*RST, this function is OFF.

**4.15.4.2 [:TYPE] LPASs|BPASs**

CALCulate:TRANSform:FREQuency:TYPE

Selects a particular method to be used in limiting information or the manner in which windows shall be applied for the specified transform.

When LPASs is selected, the windowing shall affect only the end of the data set. LPASs requires data to start from zero in the appropriate units, such as Hertz and seconds. If the

data set does not extend to zero, then the instrument shall either compute by extrapolation the extra points needed to complete the data set, or shall generate an execution (-200) error. When BPASs is selected, the applied window shall affect both the start and end parts of the data set.

At \*RST, this value is device-dependent.

### 4.15.4.3 :**STIMulus** **STEP|IMPulse**

CALCulate:TRANSform:FREQuency:STIMulus

Specifies the type of stimulus to be simulated in the transform process.

4

At \*RST, this selection is device-dependent.

### 4.15.4.4 :**STARt** <numeric\_value>

CALCulate:TRANSform:FREQuency:STARt

Specifies the start frequency of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STARt is set to MIN.

### 4.15.4.5 :**STOP** <numeric\_value>

CALCulate:TRANSform:FREQuency:STOP

Specifies the stop frequency of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, STOP is set to MIN.

### 4.15.4.6 :**SPAN** <numeric\_value>

CALCulate:TRANSform:FREQuency:SPAN

Specifies the frequency span of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, SPAN is set to MIN.

### 4.15.4.7 :**CENTER** <numeric\_value>

CALCulate:TRANSform:FREQuency:CENTER

Specifies the center frequency of the output data record. Range is instrument- and setup-dependent.

The couplings between STARt, STOP, CENTER, and SPAN are the same as described in the SENSe subsystem.

At \*RST, CENTER is set to MIN.

**4.15.4.8 :POINts <numeric\_value>**

CALCulate:TRANSform:FREQuency:POINTs

Specifies the number of points output by the transform subsystem.

At \*RST, this value is device-dependent.

**4.15.4.8.1 :AUTO <Boolean>|ONCE**

CALCulate:TRANSform:FREQuency:POINTs:AUTO

When AUTO is ON, the number of points is determined by the size of the incoming SENSe Data.

At \*RST, AUTO is ON.

**4.15.4.9 :WINDow RECTangular|UNIForm|FLATtop|HAMMING****|HANNing|KBESsel|FORCe|EXPonential**

CALCulate:TRANSform:FREQuency:WINDOW

This specifies the type and parameter of data windowing (shaping) done prior to the transformation.

At \*RST, this value is device-dependent.

Individual windowing algorithms are defined in “*On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform*,” F. J. Harris, Proc. of the IEEE, Vol. 66-1, January 1978, pp 51-83.

**4.15.4.10 :KBESsel <numeric\_value>**

CALCulate:TRANSform:FREQuency:KBESsel

The Kaiser BESsel command sets the parametric window parameter for the Kaiser Bessel window.

At \*RST, this value is device-dependent.

**4.15.4.11 :EXPonential <numeric\_value>**

CALCulate:TRANSform:FREQuency:EXPonential

Enters the exponential decay time constant which characterizes the EXPonential window.

At \*RST, this value is device-dependent.

**4.15.4.12 :FORCe <numeric\_value>**

CALCulate:TRANSform:FREQuency:FORCe

Enters the time value parameter corresponding to the width of the gated portion of the input time record for FORCe windows.

At \*RST, this value is device-dependent.

**4.16 :PATH(MATH|TRANSform|FILTter|SMOothing|FORMAT|LIMit|AVERage)****{,(MATH|TRANSform|FILTTER|SMOothing|FORMAT|LIMIT|AVERage)}**

CALCulate:PATH

This subsystem defines the order in which CALCulate subsystems are to be performed. In many instruments this will be fixed and instrument-dependent. If it is possible to order the operations, then PATH is the method.

## 1999 SCPI Command Reference

For example, if TRANSform is first, then FILTER, and finally MATH, the following would be sent:

```
CALCulate:PATH TRANSform,FILTTer,MATH
```

Even if not settable, the query of PATH should return the actual order of operations.

The suffix of each subsystem used as a parameter in the PATH command corresponds to the instance of the specified CALCulate subsystem sub-block. For example, the command PATH FILTer1,FILTTer2, sets up the data flow through the filter defined by CALCulate:FILTTer1 and then through the filter defined by CALCulate:FILTTer2.

At \*RST, the PATH definition is set to a device-dependent setting.

## 5 CALibration Subsystem

This subsystem has the function of performing system calibration.

Instrument CALibration commands tend to be very class specific. Such commands properly belong in Volume 4 “Instrument Classes”. Therefore, expanded commands for controlling instrument calibration may not be found in Volume 2 “Command Reference”.

KEYWORD	PARAMETER FORM	NOTES
CALibration		
[:ALL]		[event; no query] 1994
[:ALL]?		[event; query]
:AUTO	<Boolean> ONCE	
:BINertia		1999
:AVERage?	<numeric_value>	[query only] 1999
:HSPEED	<numeric_value>	1999
:INITiate		[event; no query] 1999
:LSPeed	<numeric_value>	1999
:NRUNs	<numeric_value>	1999
:SDEviation?	<numeric_value>	[query only] 1999
:UPDate		[event; no query] 1999
:DATA	<arbitrary block program data>	
:PLOSS		1999
:APCoeffs	<numeric_value>,<numeric_value>,...	1999
:INITiate		[event; no query] 1999
:LATime	<numeric_value>	1999
:STIMe	<numeric_value>	1999
:UPDate		[event; no query] 1999
:SOURce	INTERNAL EXTERNAL	1999
:STATe	<Boolean>	
:VALue	<numeric_value>	
:WARMup		1999
:INITiate		[event; no query] 1999
:SPEED	<numeric_value>	1999
:TIMEout	<numeric_value>	1999
:ZERO		
:AUTO	<Boolean> ONCE	1999
:FSENsor		1999
:INITiate		[event; no query] 1999
:LATime	<numeric_value>	1999
:LEVEL?		[query only] 1999
:SPEED	<numeric_value>	1999
:STIM	<numeric_value>	1999
:UPDate		[event; no query] 1999

5.1

### [:ALL]

CALibration:ALL

The CALibrate:ALL command performs the same function as the CALibrate:ALL? except there is no response. An instrument shall report calibration errors through the status-reporting mechanism.

If this command is implemented as overlapped it shall be included in the No Operation Pending flag (see IEEE 488.2 chapter 12).

While this command is executing the CALibrating bit of the Operation Status Condition register shall be set (see 9.3 of Syntax & Style).

5

5.2

### [:ALL]?

CALibration:ALL?

The ALL? query performs a full calibration of the instrument and responds with a <numeric\_value> indicating the success of the calibration. A zero shall be returned if calibration is completed successfully; otherwise a nonzero value which represents the appropriate error number shall be returned. An instrument shall still report calibration errors through the status-reporting mechanism, even though an error is reported by the value of the query response.

5.3

### :AUTO <Boolean>|ONCE

CALibration:AUTO

AUTO sets whether or not the instrument should perform auto calibration at device-dependent intervals without user intervention.

At \*RST, AUTO OFF is selected.

5.4

### :BINertia

CALibration:BINertia

Base Inertia

For chassis dynamometers, this node describes the procedure to determine the base mechanical inertia of the dynamometer roll system (all rotating mechanical components outside of the control loop). This procedure consists of an acceleration and deceleration of the dynamometer rolls, from which a mechanical inertia calculation is made. The dynamometer is motored from an initial speed to a final speed at a given constant acceleration, and this interval is timed. Immediately, the dynamometer is motored back from the final speed to the initial speed at the same constant acceleration (inverted), and this interval is timed. For this pair of intervals, the force transducer output is continuously measured, and the average interval force calculated. The average acceleration is calculated using the interval initial and final speed values, and the interval time. This data is used to calculate the mechanical inertia (Inertia=Force/acceleration).

5.4.1

### :AVERage?

CALibration:BINertia:AVERage?

This query returns the running average of all base inertia values acquired since the last CALibration:BINertia:INITiate was invoked.

**5.4.2 :HSPEED <numeric\_value>**

CALibration:BINertia:HSPEED

High Speed

The higher magnitude speed (m/s) of the speed interval.

At \*RST, this value is set to 16.

**5.4.3 :INITiate**

CALibration:BINertia:INITiate

Begin motoring the dynamometer through the ACCELERation/deceleration pairs, using :SOURce:ACCELERation as the acceleration/deceleration rate, from LSPEED to HSPEED and back, NRUNS times. All query commands must be accepted and responded to while this command is executing. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Base Inertia Procedure bit indicating the Base Inertia is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Accelerate/decelerate the dynamometer through the speed intervals, recording time, average acceleration, average force for the specified number of runs. The data is placed in the BaseInertia table after each acceleration/deceleration is run. Update the :CALibration:BINertia:SDEVIation value based on this run. End the procedure when the specified runs are complete.
- Clear the DYNO operation condition register Base Inertia Procedure bit indicating the Base Inertia is not executing.
- Issue the process complete message.
- Execute the Idle Procedure.
- Update the on-line mechanical inertia value, if desired.

**5.4.4 :LSPEED <numeric\_value>**

CALibration:BINertia:LSPEED

Low Speed

The lower magnitude speed (m/s) of the speed interval.

At \*RST, this value is set to 6.

**5.4.5 :NRUNS <numeric\_value>**

CALibration:BINertia:NRUNS

Number of Runs - NRUNS

## 1999 SCPI Command Reference

The number of ACCeleration/deceleration pairs to perform. NRUNS will equal the number of base inertia values acquired.

At \*RST, this value is set to 10.

### 5.4.6 :SDEViation?

CALibration:BINertia:SDEViation?

Standard Deviation

This query returns the running standard deviation of all base inertia values acquired since the last CALibration:BINertia:INITiate was invoked.

### 5.4.7 :UPDate

CALibration:BINertia:UPDate

Place on-line the last determined average base mechanical inertia value.

This command describes an event and therefore has no associated \*RST condition.

### 5.5 :DATA <arbitrary block program data>

CALibration:DATA

Transfers the calibration data as arbitrary block program data. The query form returns the current calibration data. The block is transferred into the current calibration data. If an error occurs during the transfer causing the calibration to be lost, an error -313 shall be generated in addition to the execution error.

### 5.6 :PLOSS

CALibration:PLOSS

Parasitic Loss

For chassis dynamometers this node describes the commands associated with the parasitic loss procedure. This procedure measures and, if requested, updates the parasitic loss corrections for the dynamometer. The parasitic losses are measured by operating the dynamometer at various selected speeds and measuring the force required to maintain the constant speed. This procedure uses the :MEMory table 'PARasitic' that defines the speed points at which parasitic loss measurements are taken. The parasitic loss coefficients are calculated at the end of the procedure and reside in the memory table PCoefficient.

#### 5.6.1 :APCoeff

<numeric\_value>,<numeric\_value>,<numeric\_value>[,<numeric\_value>]

:CALibration:PLOSS:APCoeff

Active Parasitic Coefficients - APCoefficients

This is the presently active parasitic loss curve used by the dynamometer. The dynamometer will maintain coefficients for forward and reverse operations. The appropriate values must be placed in the APCoeff table, based on roll rotation.

#### 5.6.2 :INITiate

:CALibration:PLOSS:INITiate

This command starts the parasitic loss procedure. This is an overlapped command. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Parasitic Loss Procedure bit indicating the Parasitic Loss procedure is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Accelerate dynamometer to the first speed point defined in the PARasitic memory table. Stabilize at speed point for defined stabilization time.
- Average the force at the roll surface required to maintain the speed set point over the latency time. The average force value is placed in the PARasitic table for the specified speed point. Continue until all speed points have been executed in the PARasitic table. The first zero speed in the PARasitic table ends the procedure.
- Perform a curve fit on the PARasitic table data and place the coefficients in the memory table PCoeff.
- Clear the DYNO operation condition register Parasitic Loss Procedure bit indicating the parasitic loss procedure is not executing.
- Issue the process complete message.
- Execute the Idle Procedure.

#### **5.6.3 :LATime**

:CALibration:PLOSSs:LATime  
Loss Averaging Time - sec

The length of time in seconds to average the losses at a speed point.

At \*RST, this value is set to 5.

#### **5.6.4 :STIMe <numeric\_value>**

CALibration:PLOSSs:STIMe  
Stabilization Time - sec

Time for stabilization prior to LATime.

At \*RST, this value is set to 5.

#### **5.6.5 :UPDate**

:CALibration:PLOSSs:UPDate

Update the dynamometer parasitic loss coefficients contained in the table APCoeff with the coefficients contained in memory table PCoeff.

This command is an event and has no associated \*RST condition.

## 1999 SCPI Command Reference

### 5.7 :SOURce INTernal|EXTernal

CALibration:SOURce

Controls the source of the calibration signal. If set to INTernal, a reference internal to the instrument is selected. If SOURce is set to EXTernal, then an external reference shall be applied.

### 5.8 :STATe <Boolean>

CALibration:STATe

STATe is used to select if the calibration data is applied or not. If STATe is ON then the instrument uses the calibration data for correction. If STATe OFF is selected, then no correction using the calibration data shall be made.

5

At \*RST, STATe is set to ON.

### 5.9 :VALue <numeric\_value>

CALibration:VALue

Enters the value of the reference signal which is used in the calibration. For example, a voltmeter may calibrate its input to a reference signal which has a proven value of 4.99327513 Volts.

### 5.10 :WARMup

CALibration:WARMup

This node controls parameters relating to the warmup of the device.

For chassis dynamometers, this node describes the procedure to set up the conditions required to ready the dynamometer for testing. The warm-up consists of running the dynamometer at a fixed speed over a period of time. Warm-up completion is defined by either running the dynamometer for a fixed period of time, or by monitoring parasitic losses and comparing them to a target value tolerance.

#### 5.10.1 :INITiate

CALibration:WARMup:INITiate

Start the warmup procedure.

For chassis dynamometers, motor the dynamometer roll to SPEEd, and hold this speed for a TIMeout period of time. All query commands must be accepted and responded to while this command is executing. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Warm up Procedure bit indicating the Warm-up is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Accelerate dynamometer to the speed point defined by :CALibration:WARMup:SPEEd.

- Average the force at the roll surface required to maintain the speed set point over :CALIbrate:WARMup:LATime. The average force can be monitored by :SENSe:FORCe?
- The procedure will end when the elapsed time is greater than :CALibration:WARMup:TIMEout.
- Clear the DYNO operation condition register Warm up Procedure bit indicating the Warm-up is not executing.
- Issue the process complete message.
- Execute the Idle Procedure.

#### 5.10.2 **:SPEed <numeric\_value>**

CALibration:WARMup:SPEed

For chassis dynamometers this is the fixed roll speed (m/s) at which the warm-up will be run.

At \*RST, this value is set to 22.

#### 5.10.3 **:TIMEout <numeric\_value>**

CALibration:WARMup:TIMEout

For chassis dynamometers this is the length of time in seconds the warm-up will run before automatically returning to zero speed.

At \*RST, this value is device dependent.

#### 5.11 **:ZERO**

CALibration:ZERO

This subsystem controls the autozero calibration of the sensor.

For chassis dynamometers, this node describes the procedure to set up the conditions required and perform a mathematical zero correction of the roll force measurement system calibration. The correction is determined from a measurement of the roll force sensor at a given roll speed. The results are stored in the FSensorZero pre-defined table.

#### 5.11.1 **:AUTO <Boolean>|ONCE**

CALibration:ZERO:AUTO

Controls whether autozeroing calibration occurs.

At \*RST, this value is set to OFF.

#### 5.11.2 **:FSENsor**

CALibration:ZERO:FSENsor

Force Sensor

For chassis dynamometers this node sets up the conditions required and performs a mathematical zero correction of the roll force measurement system calibration. The correction is determined from a measurement of the roll force sensor at a given roll speed.

**5.11.2.1 :INITiate**

CALibration:ZERO:FSENsor:INITiate

This is an overlapped command. Perform the measurement of the roll force, acquiring for the length of time specified by :LATime. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Force Sensor Zero Procedure bit indicating the Zero is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Accelerate dynamometer to the speed specified in :CALibration:ZERO:FSENsor:SPEEd. Stabilize for :STIMe.
- Average the force at the roll surface required to maintain the speed set point over the :LATime. The average force value is placed in :LEVel.
- Clear the DYNO operation condition register Force Sensor Zero Procedure bit indicating the Zero is not executing.
- Issue the process complete message.
- Execute the Idle Procedure.

**5.11.2.2 :LATime <numeric\_value>**

CALibration:ZERO:FSENsor:LATime

Loss Averaging Time - sec

The length of time to average the measured zero reading.

At \*RST, this value is set to 0.5.

**5.11.2.3 :LEVel?**

CALibration:ZERO:FSENsor:LEVel?

Returns the measured force value used to determine the mathematical zero correction. This value is in units of force at the roll surface. This is query only.

**5.11.2.4 :SPEEd <numeric\_value>**

CALibration:ZERO:FSENsor:SPEEd

The roll speed set point for the force measurement period in m/s.

At \*RST, this value is set to 0.

**5.11.2.5 :STIMe <numeric\_value>**

CALibration:ZERO:FSENsor:STIMe

Stabilization Time - sec

Time for stabilization prior to LATime.

At \*RST, this value is set to 5.

### 5.11.2.6 :UPDate

CALibration:ZERO:FSENsor:UPDate

Once the mathematical zero correction has been determined, apply the results to the on-line calibration. This is not a queryable command and therefore has no \*RST associated with it.



## 6 CONTrol Subsystem

The CONTrol subsystem is used to turn on and off electromechanical devices or control some dynamometer states that are used in a procedure.

KEYWORD	PARAMETER FORM	NOTES
:CONTRol		1993
:APOWer		1993
[:STATe]	<Boolean>	1993
:BLOWer		1993
[:STATe]	<Boolean>	1993
:BRAKe		1999
[:STATe]	<Boolean>	1999
:COMPressor		1993
[:STATe]	<Boolean>	1993
:COVer		1999
[:ADJust]	OPEN CLOSe SOPen SCLOse	[event; no query] 1999
:POSIon?		[query only] 1999
:EBENch		1999
[:CLEan]		1999
:INITiate		[event; no query] 1999
:DURation	<numeric_value>	1999
:IDLE		1999
:INITiate		[event; no query] 1999
:LIFT		1999
[:ADJust]	UP DOWN	[event; no query] 1999
:POSIon?		[query only] 1999
:MCONTrol		1999
[:STATe]	<Boolean>	1999
:ROTation		1999
[:DIRection]	FORward REVerse	1999
:VCDevice		1999
[:STATe]	<Boolean>	1999
:TDIameter	<numeric_value>	1999

### 6.1 :APOWer

CONTrol:APoWer

Controls the Auxiliary POWER. Auxiliary POWER are additional power outlets that can be used to power devices.

#### 6.1.1 [:STATe] <Boolean>

CONTrol:APoWer[:STATe]

Turns the APoWer ON and OFF.

At \*RST this value is set to OFF.

- 6.2 :BLOWer**  
CONTrol:BLOWer  
Controls the BLOWer.  
For example, in order to keep a uniform temperature inside the environment chamber, the air inside needs to be thoroughly mixed. This is usually accomplished using a blower.
- 6.2.1 [:STATe] <Boolean>**  
CONTrol:BLOWer[:STATe]  
Turns the BLOWer ON and OFF.  
At \*RST this value is set to OFF.
- 6.3 :BRAKe**  
CONTrol:BRAKe  
This node contains the commands that control the brake.
- 6.3.1 [:STATe] <Boolean>**  
:CONTrol:BRAKe[:STATe]  
Sets or queries the state of the brake.  
At \*RST, this value is set to OFF.
- 6.4 :COMPressor**  
CONTrol:COMPressor  
Controls the COMPressor.  
For example in an environmental chamber, a compressor is often needed additional control of the heating and cooling.
- 6.4.1 [:STATe] <Boolean>**  
CONTrol:COMPressor[:STATe]  
Turns the COMPressor ON and OFF.  
At \*RST this value is set to OFF.
- 6.5 :COVer**  
CONTrol:COVer  
This node contains the commands which control the cover.
- 6.5.1 [:ADJust] OPEN|CLOSe|SOPEn|SCLOSe**  
CONTrol:COVer[:ADJust]  
This command controls the cover. This command is an event and has no associated \*RST condition
  - OPEN - open covers
  - CLOSe - close covers
  - SCLOSe - step covers incrementally toward CLOSe
  - SOPEn - step covers incrementally toward OPEN

**6.5.2 :POSITION?**

CONTrol:COVer:POSIon?

This command queries the position of the cover

- OPEN - covers are fully open
- CLOSe - covers are fully closed
- MID - covers are partially open

**6.6 :EBENch**

CONTrol:EBENch

This subsystem controls the state and activities of a gas analyzer emissions bench.

**6.6.1 :CLEan**

:CONTrol:EBENch:CLEan

This subsystem controls the bench's internal procedure to clean out its gas lines, commonly known as "purge" or "backflush".

**6.6.1.1 :INITiate**

:CONTrol:EBENch:CLEan:INITiate

Initiate the bench's procedure to clean out its gas lines.

This command is an event and therefore has no \*RST value.

**6.6.1.2 :DURation <numeric\_value>**

:CONTrol:EBENch:CLEan:DURation

This command sets or queries the duration in seconds of the "clean" procedure.

At \*RST, this value is device-dependent.

**6.7 :IDLE**

CONTrol:IDLE

This node contains commands that control the idle state of the device.

**6.7.1 :INITiate**

:CONTrol:IDLE:INITiate

Returns the device to idle.

For chassis dynamometers, the dynamometer must decelerate to 0 m/s at the :SOURce:ACCeeration rate and wait for a command. The DYNO operation condition register Idle Procedure bit should be set indicating that :CONTrol:IDLE is executing. Note: The DYNO operation condition register will be further defined in Volume 4.

This command is an event and has no associated \*RST condition.

**6.8 :LIFT**

CONTrol:LIFT

This node contains commands that control the lift.

## 1999 SCPI Command Reference

### 6.8.1 [:ADJust] UP|DOWN

CONTrol:LIFT[:ADJust]

This command controls the lift. This command is an event and has no associated \*RST condition.

- UP - raises lift
- DOWN - lowers lift

### 6.8.2 :POSIon?

CONTrol:LIFT:POSIon?

Queries the position of the lift.

- UP - lifts are up.
- DOWN - lifts are down.
- MID - lifts are positioned between UP and DOWN.

### 6.9 :MCOntrol

CONTrol:MCOntrol

Motor Control

Controls the motor.

### 6.9.1 [:STATe] <Boolean>

:CONTrol:MCOntrol[:STATe]

Turns the power of motor ON or Off.

At \*RST, this value is set to OFF.

### 6.10 :ROTration

CONTrol:ROTration

This node contains commands that control device rotation.

### 6.10.1 [:DIRection]

CONTrol:ROTration[:DIRection]

For chassis dynamometers, configures the dynamometer rotation direction. The dynamometer determines FORWARD or REVERSE based on configuration.

At \*RST, this value is set to FORWARD.

### 6.11 :VCDevice

CONTrol:VCDevice

Vehicle Centering Device

For chassis dynamometers, this node contains commands that control the Vehicle Centering Device.

### 6.11.1 [:STATe] <Boolean>

:CONTrol:VCDevice[:STATe]

When the state is ON, the dynamometer is performing the centering function. When the state is OFF, centering is not being performed.

At \*RST, this value is set to OFF.

### 6.11.2 :TDliameter <numeric\_value>

CONTrol:VCDevice:TDliameter

Tire Diameter

Centering device uses tire diameter (m) as limit to center the vehicle on the dynamometer.

At \*RST, this value is set to 0.7.

## **1999 SCPI Command Reference**

---

## 7 **DIAGnostic Subsystem**

The purpose of the DIAGnostic subsystem is to provide a tree node for all of the instrument service and diagnostic routines used in routine maintenance and repair. The actual commands in this subsystem are considered instrument-specific, and therefore are not included in this document. Instrument designers are free to add service and diagnostic commands to meet the needs of individual instruments.



## DISPlay Subsystem

The DISPlay subsystem controls the selection and presentation of textual, graphical, and TRACe information. This information includes measurement data, user-interaction displays, and data presented to the instrument by the controller. DISPlay is independent of, and does not modify, how data is returned to the controller.

Multiple DISPlay subsystems are used to represent independent display medium. A front panel mounted display and an attached terminal used only for information display are examples of independent displays. Conversely, an instrument with many dedicated indicators is considered to have one general display because of the dependencies in operation that exist between the indicators.

Within a DISPlay, information may be separated into individual WINDows (this is always the case for instruments with dedicated indicators). Each window is considered to consist of three overlapped planes, one each for text, graphics, and trace data. Thus, text, graphics, and trace information may be presented at the same time in a given window.

Most of the \*RST conditions in the DISPlay subsystem are device-dependent, to accommodate for typical display functions that exist for each of the various categories of instrumentation.

KEYWORD	PARAMETER FORM	NOTES
DISPlay		
:ANNotation		
[:ALL]	<Boolean>	
:AMPLitude	<Boolean>	
:FREQuency	<Boolean>	
:BRIGHTness	<numeric_value>	
:CMAP		
:DEFault	[event; no query]	
:COLor		1992
:HSL	<hue>,<sat>,<lum>	1992
:RGB	<red>,<green>,<blue>	1992
:CONTrast	<numeric_value>	
:ENABLE	<Boolean>	
:MENU		
[:NAME]	<menu_name>	
:STATE	<Boolean>	1992
:KEY	<string>	
[:WINDOW]		
:BACKground		
:COLOR	<color number>	
:GEOMetry		
:LLEFt	<numeric_value>,<numeric_value>	
:SIZE	<numeric_value>,<numeric_value>	
:URIGHT	<numeric_value>,<numeric_value>	

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:GRAPHics		
:CLEar		[event; no query]
:COLOR	<color number>	
:CSIZE	<numeric_value>	
[:DRAW]	<numeric_value>,<numeric_value>	
:PCL	<block>	
:HPGL	<block>	
:IDRaw	<numeric_value>,<numeric_value>	
:IMOVe	<numeric_value>,<numeric_value>	
:LABEL	<string>	
:LDIRection	<numeric_value>	
:LTYPe	<numeric_value>[,<numeric_value>]	
:MOVE	<numeric_value>,<numeric_value>	
:STATe	<Boolean>	
[:STATe]	<Boolean>	
:TEXT		
:ATTRibutes	<Boolean>	
:CLEar		[event; no query]
:COLOR	<numeric_value>	
:CSIZE	<numeric_value>	
[:DATA]	<string>   <block>	
:FEED	<data_handle>	1991
:LOCate	<row>[,<col>]	
:PAGE	<numeric_value>	1991
:STATe	<Boolean>	
:TRACe		
:COLOR	<numeric_value>	
:FEED	<data_handle>	1991
:GRATicule		
:AXIS		
[:STATe]	<Boolean>	
:COLOR	<numeric_value>	
:FRAME		
[:STATe]	<Boolean>	
:GRID		
:AUTO	<Boolean>	
[:STATe]	<Boolean>	
:PERSistence	<numeric_value>	
:AUTO	<Boolean> ONCE	
[:STATe]	<Boolean>	
:X		
:LABEL	<string>	
[:SCALe]		

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:AUTO	<Boolean>   ONCE	1991
:CENTer	<numeric_value>	1991
:LEFT	<numeric_value>	
:PDIVison	<numeric_value>	1991
:LINK	LEFT CENTer RIGHt	1991
:RIGHt	<numeric_value>	
:SPACing	LINear   LOGarithmic	
:Y		
:LABel	<string>	
:RLINe	<Boolean>	1991
[:SCALe]		
:AUTO	<Boolean>   ONCE	1991
:BOTTom		
:PDIVison	<numeric_value>	1991
:RLEVel	<numeric_value>	1991
:AUTO	<Boolean>	1991
:RPOSition	<numeric_value>	1991
:TOP	<numeric_value>	
:SPACing	LINear   LOGarithmic	
:R		
:LABel	<string>	
[:SCALe]		
:AUTO	<Boolean>   ONCE	1991
:CPOint	<numeric_value>	
:OEDGE	<numeric_value>	
:SPACing	LINear   LOGarithmic	

8.1

**:ANNAnnotation**

DISPlay:ANNAnnotation

This subsystem controls which annuciators are visible. ANNAnnotation does not affect the visibility of GRATicule. If SYSTem:SECurity is enabled, annotation cannot be switched from OFF to ON.

The ANNAnnotation parameters are set to ON at \*RST, unless SYSTem:SECurity is enabled.

8.1.1

**[:ALL] <Boolean>**

DISPlay:ANNAnnotation:ALL

Controls ALL of the annotation information.

8.1.2

**:AMPLitude <Boolean>**

DISPlay:ANNAnnotation:AMPLitude

Controls the amplitude annotation information.

8.1.3

**:FREQuency <Boolean>**

DISPlay:ANNAnnotation:FREQuency

Controls the frequency annotation information.

8

8.2

**:BRIGHTness <numeric\_value>**

DISPlay:BRIGHTness

Controls the intensity of the display. The range of the parameter is 0 to 1, where 1 is full intensity and 0 is fully blanked.

8.3

**:CMAP**

DISPlay:CMAP

The CMAP subsystem controls the physical color associated with logical color numbers in the color map of the display. On a monochrome DISPlay, the CMAP may be used to represent different intensity levels of the same shade. When a color is redefined, all information associated with that color number may change immediately.

Two methods are available for defining a particular color, HSL and RGB models. If both are implemented in a device, then they shall be coupled such that redefining a color with one model shall be reflected in the values of the other model.

At \*RST, COLOR[1] shall represent “black,” and COLOR2 shall represent “white” or the actual color of a monochrome display.

8.3.1

**:DEFault**

DISPlay:CMAP:DEFault

Sets the color map to the instrument’s default values for all colors, except that COLOR[1] shall be represent “black,” and COLOR2 shall represent “white” or the actual color of a monochrome display.

8.3.2

**:COLOR**

DISPlay:CMAP:COLOR

This subsystem controls the color map for the instrument.

**8.3.2.1 :HSL <hue>,<sat>,<lum>**

DISPlay:CMAP:COLor:HSL

The HSL command sets the instrument's color map based on the Hue/Saturation/Luminance levels color model. A query of HSL shall return three <NR2>s separated by commas.

Hue ranges from zero to one, circularly, with a value of zero resulting in the same hue as a value of one. The approximate color progression is (starting at zero): red, orange, yellow, green, cyan, blue, magenta, and back to red.

Saturation is the amount of pure color to be mixed with white. The saturation value ranges from zero to one, with zero specifying no color (only white or gray, depending on intensity) and one specifying no white.

Luminance specifies the brightness per unit area of the color. A luminance of zero results in black; a luminance of one results in the brightest color available.

At \*RST, these parameters are set to the DEFault values, equivalent to executing DEFault command. The HSL command is coupled to the RGB command. Changing values in either will affect the values of the other.

**8.3.2.2 :RGB <red>,<green>,<blue>**

DISPlay:CMAP:COLor:RGB

The RGB command sets the instrument's color map based on the Red/Green/Blue color model. A query of RGB shall return three <NR2>s separated by commas.

Red ranges from zero to one, with zero indicating “no red” and one indicating “full intensity red.”

Green ranges from zero to one, with zero indicating “no green” and one indicating “full intensity green.”

Blue ranges from zero to one, with zero indicating “no blue” and one indicating “full intensity blue.”

At \*RST, these parameters are set to the DEFault values, equivalent to executing DEFault command. The RGB command is coupled to the HSL command. Changing values in either will affect the values of the other.

**8.4 :CONTrast <numeric\_value>**

DISPlay:CONTrast

Determines the relative difference in brightness between “full” intensity and “no” intensity as displayed. The parameter ranges in value from 0 to 1; 0 indicates no difference between data and background, and 1 indicates maximum contrast.

**8.5 :ENABLE <Boolean>**

DISPlay:ENABLE

Controls whether the whole display is visible. If SYSTem:SECurity is enabled, DISPlay cannot be switched from OFF to ON.

ENABLE is set to ON at \*RST, unless SYSTem:SECurity is enabled.

**8.6 :MENU**  
 DISPlay:MENU  
 Allows selection of predefined device-dependent menus for display. A menu is basically a softkey help screen, and it may consist of only key labels or more extensive information.

**8.6.1 [:NAME] <menu\_name>**  
 DISPlay:MENU:NAME  
 The NAME command selects a menu by name from an instrument-predefined list of valid menu names.

**8.6.2 :STATe <Boolean>**  
 DISPlay:MENU:STATe  
 Turns the current menu page ON or OFF.

**8.6.3 :KEY <string>**  
 DISPlay:MENU:KEY  
 Assigns the soft key label <string> to key. This affects only the displayed key label; it does not change the key definition.

**8.7 [:WINDOW]**  
 DISPlay:WINDOW  
 The WINDOW subtree contains commands to enable a window, to set its size and location, and control its contents. Multiple instances of WINDOW may exist under a particular DISPlay.

**8.7.1 :BACKground**  
 DISPlay:WINDOW:BACKground  
 This subsystem controls the characteristics of the window background.

**8.7.1.1 :COLor <numeric\_value>**  
 DISPlay:WINDOW:BACKground:COLor  
 Selects from the CMAP the value given by <numeric\_value> to become the background color.

**8.7.2 :GEOmetry**  
 DISPlay:WINDOW:GEOmetry  
 This subtree is used to define the position and size of the WINDOW with respect to the display. A window may be specified by using any two of LLEFT, URIGHt or SIZE commands, since SIZE = URIGHt - LLEFT. If any one of these commands is implemented, then all three shall be implemented and the coupling between them shall be maintained.

For the GEOmetry subtree, the lower left corner of the whole display is defined to be (0,0); zero in the horizontal direction and zero in the vertical direction. The upper right corner is defined to be (1,1).

The effect of changing the size or aspect ratio of a WINDOW that already is displaying information is device-dependent. The device may choose to rescale the information to fit the new WINDOW or it may leave the information unchanged, clipping it where it exceeds the WINDOW boundaries.

**8.7.2.1 :LLEFt <numeric\_value>,<numeric\_value>**

DISPlay:WINDOW:GEOmetry:LLEFt

This specifies (or queries) the location of the lower left corner of the window in the range 0 to 1.

**8.7.2.2 :SIZE <numeric\_value>,<numeric\_value>**

DISPlay:WINDOW:GEOmetry:SIZE

This specifies (or queries) the size of the display window. The first parameter represents the length in the horizontal direction, and the second parameter represents the length in the vertical direction.

The parameters may be set in the range greater than zero to 1. Where the existence of a window can be represented by an icon, the device shall accept SIZE 0,0 to indicate the window is to be replaced by an icon.

If a window is represented by an icon, any valid SIZE command with non-zero parameters shall reset the size of the window, and shall cause the icon to be replaced by the window.

**8.7.2.3 :URIGht <numeric\_value>,<numeric\_value>**

DISPlay:WINDOW:GEOmetry:URIGht

This specifies (or queries) the location of the upper right corner of the display window in the range 0 to 1.

**8.7.3 :GRAPhics**

DISPlay:WINDOW:GRAPhics

The GRAPhics subtree allows for the display of graphical information in the WINDOW. For example, it allows direct control of the WINDOW by the controller, using it as a plotter output device.

For the GRAPhics subtree, the lower left corner of the WINDOW is defined to be (0,0); zero in the horizontal direction and zero in the vertical direction. The upper right corner is defined to be (1,1).

**8.7.3.1 :CLEar**

DISPlay:WINDOW:GRAPhics:CLEar

Erases the graphics from the WINDOW.

This command defines an event and therefore has no query form or \*RST value.

**8.7.3.2 :COLor <numeric\_value>**

DISPlay:WINDOW:GRAPhics:COLor

Selects from the CMAP the value given by <numeric\_value> to become the color for the next GRAPhics operation. It is similar in function to selecting a pen on a plotter.

**8.7.3.3 :CSIze <numeric\_value>[,<numeric\_value>]**

DISPlay:WINDOW:GRAPhics:CSIze

Sets the size and optional aspect ratio (width/height) of the character cell used by the LABel command. The actual physical size of the resulting character is device-dependent.

**8.7.3.4 [:DRAW] <numeric\_value>,<numeric\_value>**

DISPLAY:WINDOW:GRAPHICS:DRAW

Draws a line from the current “pen” position to the specified X and Y coordinate (the first parameter is X, the second is Y; both parameters are required). The current line type and pen number are used.

**8.7.3.5 :PCL <block>**

DISPLAY:WINDOW:GRAPHICS:PCL

Graphics and text data are sent to the instrument using Hewlett-Packard “<esc>\*” terminal-and printer-style escape sequences. These are described in the Hewlett-Packard “Printer Command Language” document. Some escape sequences may interact with global parameters such as color selection. The effect of these changes is global in scope and not restricted to the :GRAPHICS subsystem.

**8.7.3.6 :HPGL <block>**

DISPLAY:WINDOW:GRAPHICS:HPGL

Graphics and text data are sent to the instrument using HP-GL (Hewlett-Packard-Graphics Language) plotter language. Some HP-GL sequences will interact with global parameters such as color selection; the effect of these changes is global in scope and not restricted to the :GRAPHICS subsystem.

**8.7.3.7 :IDRaw <numeric\_value>,<numeric\_value>**

DISPLAY:WINDOW:GRAPHICS:IDRaw

Draws a line from the current pen position to a new position determined by adding an X offset (first parameter) and a Y offset (second parameter) to the current coordinates.

**8.7.3.8 :IMOVe <numeric\_value>,<numeric\_value>**

DISPLAY:WINDOW:GRAPHICS:IMOVe

Updates the current pen position by adding an X offset (first parameter) and a Y offset (second parameter) to the current coordinates. No line is drawn.

**8.7.3.9 :LABel <string>**

DISPLAY:WINDOW:GRAPHICS:LABel

Places text on the graphics display at the current pen position, using the current pen, with the label direction set by LDIR.

**8.7.3.10 :LDIRection <numeric\_value>**

DISPLAY:WINDOW:GRAPHICS:LDIRection

Defines the angle (in radians) at which labels are drawn. An angle of 0 is normal horizontal printing. The angle increases counter-clockwise; PI/2 is bottom-to-top.

**8.7.3.11 :LTYPe <numeric\_value>[,<numeric\_value>]**

DISPLAY:WINDOW:GRAPHICS:LTYPe

Selects a line type and optional repeat length for all subsequent lines drawn by IDRaw or DRAW. Although the specific line types are device-dependent, line type 1 is by tradition a “plain” line (solid), while other types include dots and dashes or combinations of dots and dashes.

**8.7.3.12 :MOVE <numeric\_value>,<numeric\_value>**

DISPlay:WINDOW:GRAphics:MOVE

Updates the pen position without drawing a new line.

**8.7.3.13 :STATe <Boolean>**

DISPlay:WINDOW:GRAphics:STATe

Controls whether the GRAphics is visible or not.

**8.7.4 [:STATe] <Boolean>**

DISPlay:WINDOW:STATe

Controls whether the WINDOW is visible or not. The command DISPlay &lt;Boolean&gt; refers to this node. If SYSTem:SECurity is enabled, WINDOW cannot be switched from OFF to ON.

At \*RST, DISPlay[:WINDOW][:STATe] is set to ON, unless SYSTem:SECurity is enabled.

**8.7.5 :TEXT**

DISPlay:WINDOW:TEXT

The TEXT subtree allows for the display of textual information in the WINDOW. For example, it allows direct control of the WINDOW by the controller, using it as a VDT (Visual Display Terminal) output device.

For the TEXT subtree, the upper left corner of the WINDOW is defined to be (1,1). The lower right corner is dependent on the WINDOW:SIZE and the TEXT:Character SIZE selected.

**8.7.5.1 :ATTRibutes <Boolean>**

DISPlay:WINDOW:TEXT:ATTRibutes

ATTRibutes allows the device to interpret the ANSI Standard Terminal escape sequences when displaying TEXT:DATA. Escape sequences that generate an inquiry are not permitted. The device may absorb escape sequences that it cannot interpret to prevent them from appearing on the display.

If ATTRibutes is ON, escape sequences embedded in the data are used to control the attributes of the string, such as blinking, or underline. ATTRibutes set to OFF disables the interpretation of display sequences, but does not affect the current display. A blinking display keeps on blinking until it is overwritten. The device may absorb the escape sequences to prevent them from appearing on the display.

**8.7.5.2 :CLEar**

DISPlay:WINDOW:TEXT:CLEar

Erases the text from the WINDOW.

This command defines an event, and therefore has no query form or \*RST value.

**8.7.5.3 :COLor <numeric\_value>**

DISPlay:WINDOW:TEXT:COLor

Selects from the CMAP the value given by <numeric\_value> to become the color for the next DATA stream.

**8.7.5.4 :CSIze <numeric\_value>[,<numeric\_value>]**

DISPlay:WINDow:TEXT:CSIze

Sets the size and optional aspect ratio (width/height) of the character cell used by DATA.  
Actual physical size of the resulting character is device-dependent.

**8.7.5.5 :FEED <data\_handle>**

DISPlay:WINDow:TEXT:FEED

Sets or queries what data flow is fed into the TEXT display window.

At \*RST, the <data\_handle> is set to a device-dependent value.

**8.7.5.6 [:DATA] <string> | <block>**

DISPlay:WINDow:TEXT:DATA

The data that gets written to the text display area. Note that writing to the display overwrites any data that may have already been written. The result of overwriting a string with a shorter string is not defined; the excess characters may or may not be cleared.

**8****8.7.5.7 :LOCate <numeric\_value>[,<numeric\_value>]**

DISPlay:WINDow:TEXT:LOCate

This command selects the display ROW and display COLumn where the next LINE of TEXT DATA is to appear. Row number 1 is the top row of the instrument display; column number 1 is the left-most column.

**8.7.5.8 :PAGE <numeric\_value>**

DISPlay:WINDow:TEXT:PAGE

The PAGE command sets the page to be displayed. This command may accept UP and DOWN as defined in <numeric\_value>.

**8.7.5.9 :STATe <Boolean>**

DISPlay:WINDow:TEXT:STATe

Controls whether the TEXT is visible or not.

**8.7.6 :TRACe**

DISPlay:WINDow:TRACe

The TRACe subtree allows for the display of trace data in the WINDow; for example, directly from measurement data or from MEMory. The format of the data (and associated information) is used by the TRACe subsystem to define how commands are to be interpreted and whether on not they are valid.

**8.7.6.1 :COLor <numeric\_value>**

DISPlay:WINDow:TRACe:COLor

Selects from the CMAP the value given by <numeric\_value> to become the color for the TRACe.

**8.7.6.2 :FEED <data\_handle>**

DISPlay:WINDow:TRACe:FEED

Sets or queries what data flow is fed into the TRACe display window.

At \*RST, the <data\_handle> is set to a device-dependent value.

**8.7.6.3 :GRATicule**

DISPlay:WINDOW:TRACe:GRATicule

The GRATicule subtree is used to set the type of graticule displayed in relation to the TRACe.

**8.7.6.3.1 :AXIS**

DISPlay:WINDOW:TRACe:GRATicule:AXIS

The axes are either X and Y AXIS with markings (divisions), or the R AXIS with markings (divisions) with respect to the TRACe.

**8.7.6.3.1.1 [:STATe] <Boolean>**

DISPlay:WINDOW:TRACe:GRATicule:AXIS:STATe

Determines if the AXIS is visible or not.

**8.7.6.3.2 :FRAME**

DISPlay:WINDOW:TRACe:GRATicule:FRAME

The FRAME is the perimeter boundary and markings with respect to the TRACe.

**8.7.6.3.2.1 [:STATe] <Boolean>**

DISPlay:WINDOW:TRACe:GRATicule:FRAME:STATe

Determines if the FRAME is visible or not.

**8.7.6.3.3 :GRID**

DISPlay:WINDOW:TRACe:GRATicule:GRID

The GRID provides constant lines of X, Y or R with respect to the TRACe.

**8.7.6.3.3.1 :AUTO <Boolean>**

DISPlay:WINDOW:TRACe:GRATicule:GRID:AUTO

AUTO couples the subtree of GRID to FRAME. Turning ON FRAME, with AUTO set to ON, shall cause GRID to turn on also.

**8.7.6.3.3.2 [:STATe] <Boolean>**

DISPlay:WINDOW:TRACe:GRATicule:GRID:STATe

Determines if the GRID is visible or not.

**8.7.6.4 :PERSistence <numeric\_value>**

DISPlay:WINDOW:TRACe:PERSistence

Sets how long trace data written to the screen will remain visible. The parameter is in seconds.

**8.7.6.4.1 :AUTO <Boolean>|ONCE**

DISPlay:WINDOW:TRACe:PERSistence:AUTO

When AUTO is set to ON, the persistence is determined by the device.

At \*RST, this value is set to ON.

**8.7.6.5 :STATe <Boolean>**

DISPlay:WINDOW:TRACe:STATe

Controls whether the TRACe and related information is visible or not.

**8.7.6.6 :X**

DISPlay:WINDOW:TRACe:X

This subsystem controls the setting of the X axis. The X axis is displayed horizontally, and it is usually the independent variable of the data, such as time or frequency. The :X node is used to define the parameters associated with the X axis. It is only available when the format of the data is Cartesian.

**8.7.6.6.1 :LABEL <string>**

DISPlay:WINDOW:TRACe:X:LABEL

Specifies custom axis labeling.

**8.7.6.6.2 [:SCALE]**

DISPlay:WINDOW:TRACe:X:SCALE

The SCALE subtree is used to define which portion of the (TRACe) data is to be displayed, by defining the end points of the axis as they appear in the WINDOW. The values are in the same UNITS as the UNIT of data for the x axis. The x scale can be defined in terms of CENTER, LEFT, RIGHT and/or PDIVision, where the following equations express the relationship of these parameters:

$$RIGHT - LEFT = (RIGHT - CENTER) * 2 = PDIVision * \# \text{ of graticule divisions.}$$

If an instrument implements x axis scaling, the commands RIGHT and LEFT must be implemented, while implementing PDIVision and CENTER are optional.

**8.7.6.6.2.1 :AUTO <Boolean> | ONCE**

DISPlay:WINDOW:TRACe:X:SCALE:AUTO

The AUTO ON command sets the display to always configure the scaling on the particular axis to best display the data. This rescaling may affect the values of any of the parameters under the SCALE node. When AUTO is ON the display may be rescaled after each measurement or sweep. Setting AUTO to ONCE, turns auto ON, adjusting the scaling of the axis for one measurement and then turns auto OFF. Turning AUTO ON can change any of the parameters which can be set under the SCALE node.

At \*RST AUTO is set to OFF.

**8.7.6.6.2.2 :CENTer <numeric\_value>**

DISPlay:WINDOW:TRACe:X:SCALE:CENTER

Set or queries the value represented by the center point of the x-axis. This value may be bounded by the range of the data. When a new CENTER value is entered, the PDIVision value is kept constant and the RIGHT and LEFT values are changed.

At \*RST, this parameter value is device dependent.

**8.7.6.6.2.3 :LEFT <numeric\_value>**

DISPlay:WINDOW:TRACe:X:SCALE:LEFT

Set or queries the value represented by the minimum (left) edge of the x-axis. This value may be bounded by the range of the data. When a new LEFT value is entered, the PDIVision value is kept constant and the CENTER and RIGHT values are changed.

At \*RST, this parameter value is device dependent.

**8.7.6.6.2.4 :PDIVision <numeric\_value>**

DISPlay:WINDOW:TRACe:X:SCALe:PDIvision

Set or query the value between two grid graticules (value “per division”). When a new PDIvision value is entered, the parameter set as the PDIvision:LINK is held constant and the other two are changed.

At \*RST, this parameter value is device dependent.

**8.7.6.6.2.4.1 :LINK LEFT | CENTer | RIGHT**

DISPlay:WINDOW:TRACe:X:SCALe:PDIvision:LINK

LINK selects the parameter, either CENTer, LEFT or RIGHT that shall not be changed when the PDIvision value is changed. For example, if LINK is set to LEFT, then changing PDIvision shall cause CENTer and RIGHT, not LEFT, to change.

At \*RST the value of LINK is device dependent.

**8.7.6.6.2.5 :RIGHT <numeric\_value>**

DISPlay:WINDOW:TRACe:X:SCALe:RIGHT

Set or queries the value represented by the maximum (right) edge of the x-axis. This value may be bounded by the range of the data. When a new RIGHT value is entered, the PDIvision value is kept constant and the CENTer and LEFT values are changed.

At \*RST, this parameter value is device dependent.

**8.7.6.7 :Y**

DISPlay:WINDOW:TRACe:Y

This subsystem controls the setting of the Y axis. The :Y node is displayed vertically, and it is usually the dependent variable of the data, such as power or voltage. The :Y node is used to define the parameters associated with the Y axis. It is only available when the format of the data is Cartesian.

**8.7.6.7.1 :LABel <string>**

DISPlay:WINDOW:TRACe:Y:LABel

Specifies custom axis labeling.

**8.7.6.7.2 :RLINe <Boolean>**

DISPlay:WINDOW:TRACe:Y:RLINe

Turn on/off a line which is positioned on the graticule at the current Reference POSition.

At \*RST RLINe is off.

**8.7.6.7.3 [:SCALe]**

DISPlay:WINDOW:TRACe:Y:SCALe

The SCALe subtree is used to define which portion of the (TRACe) data is to be displayed, by defining the end points of the axis as they appear in the WINDOW. The values are in the same UNITS as the UNIT of data for the y axis. The y scale can be defined in terms of TOP,

BOTTom, PDIVision or Reference LEVel and Reference POSition. These values are coupled together in the following manner:

$$TOP - BOTTom = PDIVision * \# \text{ of graticule divisions}$$

*TOP* = Reference LEVel when Reference POSITION is 100 %

*BOTTom* = Reference LEVel when Reference POSITION is 0 %

If an instrument implements y axis scaling, the commands TOP and BOTTom must be implemented, while implementing the RLEVel and RPOSITION and PDIVision commands are optional.

#### 8.7.6.7.3.1 :AUTO <Boolean> | ONCE

DISPlay:WINDow:TRACE:Y:SCALe:AUTO

The AUTO ON command sets the display to always configure the scaling on the particular axis to best display the data. This rescaling may affect the values of any of the parameters under the SCALe node. When AUTO is ON the display may be rescaled after each measurement or sweep. Setting AUTO to ONCE, turns auto ON, adjusting the scaling of the axis for one measurement and then turns auto OFF. Turning AUTO ON can change any of the parameters which can be set under the SCALe node.

At \*RST AUTO is set to OFF.

#### 8.7.6.7.3.2 :BOTTom <numeric\_value>

DISPlay:WINDow:TRACE:Y:SCALe:BOTTom

Sets or queries the value represented by the minimum (bottom) edge of the display. The value may be bounded by the range of the data.

#### 8.7.6.7.3.3 :PDIVision <numeric\_value>

DISPlay:WINDow:TRACE:Y:SCALe:PDIVision

Set or query the value between two grid graticules (value “per division”). When a new PDIVision value is entered, the current Reference LEVel is kept the same, while adjusting the top and bottom scaling for the new PDIVision value.

At \*RST, this parameter value is device dependent.

#### 8.7.6.7.3.4 :RLEVel <numeric\_value>

DISPlay:WINDow:TRACE:Y:SCALe:RLEVel

Set or query the value represented at the designated Reference POSition on the y-axis. Setting a new Reference LEVel does not affect the value of PDIVision.

At \*RST, this parameter value is device dependent.

#### 8.7.6.7.3.4.1 :AUTO <Boolean>

DISPlay:WINDow:TRACE:Y:SCALe:RLEVel:AUTO

Setting RLEVel:AUTO to ON causes the display to automatically choose a reference level to best display the particular data. When AUTO is ON, a new reference level may be selected each time new data is received.

At \*RST AUTO is off.

**8.7.6.7.3.5 :RPOSIon <numeric\_value>**

DISPlay:WINDow:TRACe:Y:SCALe:RPOSIon

Set or query the point on the y-axis to be used as the reference position as a percentage of the length of the y- axis. The top of the y axis is defined to have a Reference POSition of 100%, while the bottom of the y axis is defined to have a Reference POSition of 0%. The Reference POSition is the point on the y-axis which should equal the Reference LEVel.

At \*RST, this parameter value is device dependent.

**8.7.6.7.3.6 :TOP <numeric\_value>**

DISPlay:WINDow:TRACe:Y:SCALe:TOP

Sets or queries the value represented by the top edge of the display. The value may be bounded by the range of the data.

**8.7.6.7.4 :SPACing LOGarithmic | LINear**

DISPlay:WINDow:TRACe:Y:SPACing

The SPACing command is used to set the scale to either linear or log.

8

**8.7.6.8 :R**

DISPlay:WINDow:TRACe:R

This subsystem controls the R axis. The :R axis is displayed radially from the center point and usually represents magnitude of complex data. It is only available when the format of the data is in polar coordinates.

**8.7.6.8.1 :LABEL <string>**

DISPlay:WINDow:TRACe:R:LABEL

Specifies custom axis labeling.

**8.7.6.8.2 [:SCALE]**

DISPlay:WINDow:TRACe:R:SCALE

The SCALE subtree is used to define which portion of the (TRACe) data is to be displayed by defining the end points of the axis as they appear in the WINDow. The values are in the same UNITS as the UNIT of the data for that AXIS.

**8.7.6.8.2.1 :AUTO <Boolean> | ONCE**

DISPlay:WINDow:TRACe:R:SCALE:AUTO

The AUTO ON command sets the display to always configure the scaling on the particular axis to best display the data. This rescaling may affect the values of any of the parameters under the SCALE node. When AUTO is ON the display may be rescaled after each measurement or sweep. Setting AUTO to ONCE, turns auto ON, adjusting the scaling of the axis for one measurement and then turns auto OFF. Turning AUTO ON can change any of the parameters which can be set under the SCALE node.

At \*RST AUTO is set to OFF.

**8.7.6.8.2.2 :CPOint <numeric\_value>**

DISPlay:WINDow:TRACe:R:SCALE:CPOint

Sets or queries the value represented by the minimum (Center POint) of the circular display. The value may be bounded by the range of the data.

### 8.7.6.8.2.3 :OEDGe <numeric\_value>

DISPlay:WINDow:TRACe:R:SCALE:OEDGe

Sets or queries the value represented by the Outside EDGe of the circular display. The value may be bounded by the range of the data.

### 8.7.6.8.3 :SPACing LOGarithmic | LINear

DISPlay:WINDow:TRACe:R:SPACing

The SPACing command is used to set the scale to either linear or log.

## FORMat Subsystem

The FORMat subsystem sets a data format for transferring numeric and array information. This data format is used for both command and response data by those commands that are specifically designated to be affected by the FORMat subsystem. The designation is either given as part of a command description, or in the definition of block or array data used by a command. The data format for command data may override the definition of FORMat if the data received is self typing (indicates its type), for the duration of that data transfer.

KEYWORD	PARAMETER FORM	NOTES
FORMat	:BORDer [:DATA] :DINTerchange :SREGister	NORMAl SWAPPed <type> [,<length>] <Boolean> ASCii   BINary   HEXadecimal   OCTal

1992

9.1

### :BORDer NORMAl|SWAPPed

FORMat:READings:BORDer

Byte ORDER. Controls whether binary data is transferred in normal or swapped byte order.

At \*RST, this value is set to NORMAl.

9

9.2

### [:DATA] <type>[,<length>]

FORMat:READings:DATA

The DATA command selects the data format; the <type> and type <length>. Valid types are ASCii, INTeger, UNTeger, REAL, HEXadecimal, OCTal, BINary, and PACKed. The <length> parameter is optional for all types; its meaning is dependent on the type selected.

The <length> parameter is an <NRf>. In the query response <length> is returned as <NR1>.

If the type INTeger, UNTeger, REAL, or PACKed is selected the data is transferred in a block. The types ASCii, HEXadecimal, OCTal, and BINary are self typing. That is, the syntax designates the type. In these cases, the FORMat subsystem is only necessary to determine the output format.

At \*RST, ASCii is selected as the default type, the type length is device-dependent.

The various types are defined as:

<type>	Interpretation:
ASCii	Numeric data is transferred as ASCii bytes in <NR1>, <NR2> or <NR3> format, as appropriate. The numbers are separated by commas as specified in IEEE 488.2. Forgiving listening allows ASCii input to always be valid. The optional <length> parameter specifies the number of significant digits to be returned.

A <length> value of zero indicates that the device selects the number of significant digits to be returned. When a <length> of zero has been

## 1999 SCPI Command Reference

	specified, the FORMat[:DATA]? query shall return zero as its second parameter.
BINary	Data is encoded as a non-decimal numeric, base 2, preceded by "#B" as specified in IEEE 488.2. The <length> parameter is the number of binary digits in each number (not including the "#B").
HEXadecimal	Data is encoded as a non-decimal numeric, base 16, preceded by "#H" as specified in IEEE 488.2. The <length> parameter is the number of hexadecimal digits in each number (not including the "#H").
INTeger	Data is transferred in a definite length block as signed integers of the length specified (in bits; default is 16). Scaling and offset, if required for interpretation of measurement data, are determined by measurement parameters set in some other subsystem. Valid values of <length> for INTeger data are 8, 16 and 32.
OCTal	Data is encoded as a non-decimal numeric, base 8, preceded by "#Q" as specified in IEEE 488.2. The <length> parameter is the number of octal digits in each number (not including the "#Q").
PACKed	Data is transferred in a definite length block, in a manner specified in the device documentation. The <length> parameter may be used to convey additional information about the device-dependent format.
REAL	Data is transferred in a definite length block as IEEE 754 floating point numbers of the specified length (in bits; default is 64). Valid values of <length> for REAL data are 32 and 64.
UINTeger	Data is transferred in a definite length block as unsigned integers of the length specified (in bits; default is 16). Scaling and offset, if required for interpretation of measurement data, are determined by measurement parameters set in some other subsystem. Valid values of <length> for UINTeger data are 8, 16, 32 and 64.

9

### 9.3

#### :DINTerchange <Boolean>

FORMAT:READings:DINTerchange

Determines whether measurement data is formatted as a <dif\_expression>. See Data Interchange Format, Volume 3. When DINTerchange ON is selected, returned data is encapsulated in this structure.

The commands which are affected by this switch are a device dependent subset of those affected by the FORMAT:DATA command, but at a minimum shall include SENSe:DATA?, CALCulate:DATA?, TRACe:DATA?, MEASure?, READ?, and FETCh?

The DIF encoding format for its DATA(CURVe(VALues)) block follows that specified in FORMAT[:DATA] and FORMAT:BORDer.

At \*RST this function is set to OFF.

### 9.4

#### :SREGister ASCii | BINary | HEXadecimal | OCTal

FORMAT:SREGister

This command selects the data type of the response to queries for any CONDITION, EVENT and ENABLE register and for PTRansition and NTRansition filters that belong to the status subsystem. Note that this includes the IEEE 488.2 status register queries.

ASCii	The data is transferred as ASCII bytes in <NR1> format. This argument is required if the command is implemented.
BINary	The data is encoded as non-decimal numeric, base 2, preceded by '#B' as specified in IEEE 488.2
HEXadecimal	The data is encoded as non-decimal numeric, base 16, preceded by '#H' as specified in IEEE 488.2
OCTal	The data is encoded as non-decimal numeric, base 8, preceded by '#Q' as specified in IEEE 488.2

At \*RST, ASCII is selected as the default response data type.



---

## 10 HCOPy

The Hard COPy subsystem controls the setup of plotting and printing to an external device. The Hard COPy subsystem does not perform any data formatting, that is it does not convert the data from one representation to another, such as COMplex to POLar. Instead HCOPy adds the necessary page formatting (dependent upon the hard copy device language) to turn the data into an acceptable form for the hard copy device.

The HCOPy subsystem may FEED data from SENSe or CALCulate, and this data may be printed or plotted. A FEED may be established from a DISPLAY:WINDOW. In this case the HCOPy output shall reflect what is currently displayed in that WINDOW. In particular, any scaling and offset that applies shall be employed.

The selection of the interface and any address information is accomplished in the SYSTem:COMMunicate subsystem. DATA may be streamed to an external device without sending it through the HCOPy subsystem, where no page formatting commands are needed.

The HCOPy subsystem gives a user several ways to select and initiate a plot or print. A user may request all aspects of the data be plotted or printed, or the user may select a subset that is of interest.

The ITEM subsystem provides the means to select the desired aspects of the data in one of two ways. First, the ITEM of interest can be directly plotted or printed. For example, the TRACe data can be plotted or printed immediately by using the HCOPy:ITEM:...:IMMEDIATE command or the HCOPy:ITEM:...:DATA? query.

Alternatively, the user may set the STATE to ON for all the ITEMS of interest, and then use the HCOPy:IMMEDIATE command or the HCOPy:DATA? query to plot or print the selected items in one transaction with the hard copy device.

If the user wishes to plot or print all the data, pointed to by the FEED, and does not want to individually initiate each ITEM or change each ITEM's STATE to ON, the user can send the HCOPy:ITEM:ALL[:IMMEDIATE] command or the HCOPy:ITEM:ALL:DATA? query.

The :HCOPy:SDUMp:IMMEDIATE command and the :HCOPy:SDUMp:DATA? query allow the user to do a Screen DUMP of the whole DISPLAY, mimicking the traditional role of a camera recorder, and provide a simple hard copy scheme for instruments with a DISPLAY.

The HCOPy subsystem provides two schemes, IMMEDIATE and DATA?, for getting hard copy data out of an instrument. First, all of the IMMEDIATE events can be implemented and used where the instrument is capable of directly controlling the interface. For example, RS-232, Centronics, or GPIB where the instrument can talk directly to the hard copy device by becoming the active controller. Any instrument which requires control of the system interface to execute the IMMEDIATE command shall also implement the \*PCB common command, see IEEE 488.2 10.21.

IMMEDIATE can also be used on GPIB when the active controller addresses the instrument to talk and the hard copy device to listen while monitoring the data flow by shadow handshaking. The format of the data sent in response to an IMMEDIATE command is not necessarily an IEEE 488.2 format. This behavior is allowed by IEEE 488.2 section 8.1.

**WARNING:**

The IEEE 488.2 message exchange protocol requirements are violated when the IMMEDIATE command generates a response that is sent when the instrument is addressed to talk. The instrument shall revert to the normal message exchange protocol if it receives a device clear or any data on the system interface. Designers should use the control passing method rather than shadow handshaking method to avoid this escape of the IEEE 488.2 message exchange protocol.

Second, the DATA? query exists for use on buses, such as GPIB, where the data flow from the instrument to the hard copy device by way of a separate controller is acceptable or necessary. It is also useful when the controller wants to store formatted data for later use. The instrument returns the formatted data in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element, see IEEE 488.2 section 8.7.10. This response data element has a leading #0 and a trailing NL^END. The controller could strip the header and trailer and send the remainder to an appropriate printer or plotter. The indefinite form is used because the block's size or nature may prevent the instrument from computing the block's length until after it is sent.

A device may implement IMMEDIATE or DATA? or both of these forms to plot or print results.

The result from HCOPy can also be fed to the Mass MEMory subsystem. The user could then retrieve the results from the stored files at a more convenient time.

KEYWORD	PARAMETER FORM	NOTES	
HCOPy			1993
:ABORt		[event;no query]	1993
:DATA?			1993
:DESTination	<data_handle>	[event;no query]	1993
:DEVice			1993
:CMAP			1993
:COLor			1993
:HSL	<hue>,<sat>,<lum>		1993
:RGB	<red>,<green>,<blue>		1993
:DEFault		[event;no query]	1993
:COLor	<Boolean>		1993
:LANGage	PCL[<n>]   HPGL[<n>]   POSTscript[<n>]		1993
:MODE	TABLE   GRAPh		1993
:RESolution	<numeric_value>		1993
:UNIT	<SUFFIX PROGRAM DATA>		1993
:SPEed	<numeric_value>		1993
:UNIT	<SUFFIX PROGRAM DATA>		1993
:FEED	<data_handle>		1993
[:IMMEDIATE]			1993
:ITEM			1993

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:ALL		1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:ANNAnnotation		1993
:COLOR	<numeric_value>	1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
:CUT		1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
:FFEed		1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
:LABel		1993
:COLOR	<numeric_value>	1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
:TEXT	<string>	1993
:MENU		1993
:COLOR	<numeric_value>	1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
:TDSTamp		1993
:COLOR	<numeric_value>	1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
[:WINDow]		1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
:TEXT		1993
:COLOR	<numeric_value>	1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
:TRACe		1993
:COLOR	<numeric_value>	1993
:DATA?		[query only] 1993

## 1999 SCPI Command Reference

KEYWORD	PARAMETER FORM	NOTES
:GRATicule		1993
:COLOR	<numeric_value>	1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993
:STATe	<Boolean>	1993
[:IMMEDIATE]		[event;no query] 1993
:LTYPe	SOLid   DOTTed   DASHed   STYLe<n>	1993
:STATe	<Boolean>	1993
:PAGE		1993
:DIMensions		1993
:AUTO	<Boolean>	1993
:LLEft	<numeric_value>,<numeric_value>	1993
:QUADrant<n>		[event;no query] 1993
:URIGHT	<numeric_value>,<numeric_value>	1993
:LENGth	<numeric_value>	1993
:ORIentation	LANDscape   PORTrait	1993
:SCALe	<numeric_value>	1993
:SIZE	CUSTom  A   B   C   D   E   A0   A1   A2   A3   A4   B0   B1   B2   B3   B4   B5	1993 1993 1993
:UNIT	<SUFFIX PROGRAM DATA>	1993
:WIDTH	<numeric_value>	1993
:SDUMp		1993
:DATA?		[query only] 1993
[:IMMEDIATE]		[event;no query] 1993

10

### 10.1 :ABORt

Hard COPy:ABORt

Abort the current hard copy operation. A new program message automatically aborts any incomplete response from an :HCOPy:...:DATA? query as that is an INTERRUPTED condition.

### 10.2 :DATA?

Hard COPy:DATA?

This query initiates the plot or print according to the current Hard COPy setup parameters. All of the items under the ITEM node which are turned ON (STATe ON) are encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

### 10.3 :DESTination<data\_handle>

Hard COPy:DESTination

An event which sets all :FEED connections which had been set to “HCOPy” to “”. The :FEED connection pointed to by <data handle> is set to “HCOPy”. Any <data\_handle> which can be used as a parameter to :DESTination shall have a :FEED command under it which can be set to “HCOPy”.

**10.4 :DEvice**

Hard COPy:DEvice

This subsystem is used to set and query characteristics relating to the hard copy device and its output.

**10.4.1 :CMAP**

Hard COPy:DEvice:CMAP

The CMAP subsystem controls the physical color associated with logical color numbers in the color map of the display. On a monochrome DISPlay, the CMAP may be used to represent different intensity levels of the same shade. When a color is redefined, all information associated with that color number may change immediately.

Two methods are available for defining a particular color, HSL and RGB models. If both are implemented in a device, then they shall be coupled such that redefining a color with one model shall be reflected in the values of the other model.

At \*RST, COLor[1] shall represent “black,” and COLor2 shall represent “white” or the actual color of a monochrome display.

**10.4.1.1 :COLor**

Hard COPy:DEvice:CMAP:COLor

This subsystem controls the color map for the instrument.

10

**10.4.1.1.1 :HSL <hue>,<sat>,<lum>**

Hard COPy:DEvice:CMAP:COLor:HSL

The HSL command sets the instrument’s color map based on the Hue/Saturation/Luminance levels color model. A query of HSL shall return three <NR2>s separated by commas.

Hue ranges from zero to one, circularly, with a value of zero resulting in the same hue as a value of one. The approximate color progression is (starting at zero): red, orange, yellow, green, cyan, blue, magenta, and back to red.

Saturation is the amount of pure color to be mixed with white. The saturation value ranges from zero to one, with zero specifying no color (only white or gray, depending on intensity) and one specifying no white.

Luminance specifies the brightness per unit area of the color. A luminance of zero results in black; a luminance of one results in the brightest color available.

At \*RST, these parameters are set to the DEFault values, equivalent to executing DEFault command. The HSL command is coupled to the RGB command. Changing values in either will affect the values of the other.

**10.4.1.1.2 :RGB <red>,<green>,<blue>**

Hard COPy:DEvice:CMAP:COLor:RGB

The RGB command sets the instrument’s color map based on the Red/Green/Blue color model. A query of RGB shall return three <NR2>s separated by commas.

Red ranges from zero to one, with zero indicating “no red” and one indicating “full intensity red.”

## 1999 SCPI Command Reference

Green ranges from zero to one, with zero indicating “no green” and one indicating “full intensity green.”

Blue ranges from zero to one, with zero indicating “no blue” and one indicating “full intensity blue.”

At \*RST, these parameters are set to the DEFault values, equivalent to executing DEFault command. The RGB command is coupled to the HSL command. Changing values in either will affect the values of the other.

### 10.4.1.2 :DEFault

Hard COPy:DEVice:CMAP:DEFault

Sets the color map to the instrument’s default values for all colors, except that COLOR[1] shall be represent “black,” and COLOR2 shall represent “white” or the actual color of a monochrome display.

### 10.4.2 :COLOR <Boolean>

Hard COPy:DEVice:COLOR

Sets or queries whether color information should be sent as part of the plot or print information, in those languages that support both color and monochrome (black and white).

At \*RST, the value of this parameter is OFF.

### 10.4.3 :LANGuage PCL[<n>] | HPGL[<n>] | POSTscript[<n>]

Hard COPy:DEVice:LANGuage

Selects the control language or data format to be used when sending out plot or print information.

PCL            Hewlett-Packard’s Printer Control Language.

HPGL          Hewlett-Packard’s Graphics Language.

Information about PCL and HPGL is available by contacting:

HP Peripheral Developer Program  
16399 West Bernardo Drive

San Diego, CA 92127-1899

(619) 487-4100

POSTscript     Adobe Systems printer language.

Information about PostScript is available by contacting:

Adobe Systems Incorporated  
1585 Charleston Road  
Mountain View, CA 94039-7900  
(415) 961-4400

The optional number, [<n>], following the languages, refers to a specific version of that language.

At \*RST, the value of this parameter is device dependent.

**10.4.4 :MODE TABLE | GRAPh**

Hard COPy:DEvice:MODE

Sets or queries how the data is to be represented in the hard copy. TABLE implies the data is formatted into a table rather than a pictorial form. Only characters appear on the output.

GRAPh implies the format is a graph or picture. The graph may be generated using vectors, raster, or other available technology.

At \*RST, the value of this setting is device dependent.

**10.4.5 :RESolution <numeric\_value>**

Hard COPy:DEvice:RESolution

Sets or queries the resolution of the result on the hard copy device. This setting is generally used to affect the quality of the output from a raster printer.

At \*RST, the value of this parameter is device dependent.

**10.4.5.1 :UNIT <SUFFIX PROGRAM DATA>**

Hard COPy:DEvice:RESolution:UNIT

Sets or queries the units of the RESolution setting.

At \*RST, the value of this parameter is device dependent.

**10.4.6 :SPEed <numeric\_value>**

Hard COPy:DEvice:SPEed

Sets or queries the speed at which vectors are drawn. This setting is generally used to affect the quality of the output from a pen plotter.

At \*RST, the value of this parameter is device dependent.

**10.4.6.1 :UNIT <SUFFIX PROGRAM DATA>**

Hard COPy:DEvice:SPEed:UNIT

Sets or queries the units of the SPEed setting.

At \*RST, the value of this parameter is cm/sec.

**10.5 :FEED <data\_handle>**

Hard COPy:FEED

Sets or queries the data flow to be fed into the Hard COPy block.

At \*RST, the value of this parameter is device dependent.

**10.6 [:IMMEDIATE]**

Hard COPy:IMMEDIATE

This command immediately initiates the plot or print according to the current Hard COPy setup parameters. All of the items under the ITEM node which are turned ON (STATe ON) are plotted or printed.

**10.7 :ITEM**

Hard COPy:ITEM

The :ITEM subsystem collects together commands to configure what items of data will be plotted or printed, and how that data shall appear.

## 1999 SCPI Command Reference

Most data item nodes have :COLor nodes to allow the user to set the color of the specific data item. The parameter to the COLor node selects either the actual pen number of the hardcopy device, or the color map number, whichever value the hardcopy device expects. Some devices may elect not to support some or all of the COLor nodes. For example, a device which has a color DISPlay may elect to use just the existing colors in the DISPlay when plotting or printing a color hardcopy.

### 10.7.1 :ALL

Hard COPy:ITEM:ALL

This subsystem is used to plot or print all ITEMS.

#### 10.7.1.1 :DATA?

Hard COPy:ITEM:ALL:DATA?

All ITEMS, regardless of their individual states, are encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

#### 10.7.1.2 [:IMMEDIATE]

Hard COPy:ITEM:ALL:IMMEDIATE

Immediately plot or print all ITEMS, regardless of their individual states.

10

### 10.7.2 :ANNOTATION

Hard COPy:ITEM:ANNOTATION

Configures the plotting or printing of any display annotation.

#### 10.7.2.1 :COLor <numeric\_value>

Hard COPy:ITEM:ANNOTATION:COLor

Sets or queries the color to be used for plotting or printing the display annotation.

At \*RST, the value of this parameter is device dependent.

#### 10.7.2.2 :DATA?

Hard COPy:ITEM:ANNOTATION:DATA?

Returns the display annotation encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

#### 10.7.2.3 [:IMMEDIATE]

Hard COPy:ITEM:ANNOTATION:IMMEDIATE

Immediately plot or print the display annotation.

#### 10.7.2.4 :STATE <Boolean>

Hard COPy:ITEM:ANNOTATION:STATE

Sets or queries whether ANNOTATION should be plotted or printed when the HCOPy:IMMEDIATE command or HCOPy:DATA? query is sent.

At \*RST, the value of this parameter is device dependent.

### 10.7.3 :CUT

Hard COPy:ITEM:CUT

Configures cutting the page after the plot or print is complete.

**10.7.3.1 :DATA?**

Hard COPy:ITEM:CUT:DATA?

Returns what would be sent to the hard copy device to cut the page encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

**10.7.3.2 [:IMMEDIATE]**

Hard COPy:ITEM:CUT:IMMEDIATE

Immediately cut the page.

**10.7.3.3 :STATe <Boolean>**

Hard COPy:ITEM:CUT:STATe

Sets or queries whether a page cut should be performed as part of the HCOPy:IMMEDIATE command or HCOPy:DATA? query.

At \*RST, the value of this parameter is device dependent.

**10.7.4 :FFEd**

Hard COPy:ITEM:FFEEd

Configures form-feeding after the plot or print is complete.

**10.7.4.1 :DATA?**

Hard COPy:ITEM:FFEEd:DATA?

Returns what would be sent to the hard copy device to do a form feed encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

10

**10.7.4.2 [:IMMEDIATE]**

Hard COPy:ITEM:FFEEd:IMMEDIATE

Immediately form-feed the page.

**10.7.4.3 :STATe <Boolean>**

Hard COPy:ITEM:FFEEd:STATe

Sets or queries whether a form feed should be performed as part of the HCOPy:IMMEDIATE command or HCOPy:DATA? query.

At \*RST, the value of this parameter is device dependent.

**10.7.5 :LABEL**

Hard COPy:ITEM:LABEL

Configures the plotting or printing of the user entered label or title. The text for LABEL is not supplied by the source of the FEED connection. The TEXT is supplied by a setting in this subsystem since it could only be generated by the user. It might reflect specific operating conditions or refer to a specific part being tested. Where on the physical page the LABEL:TEXT is actually plotted/printed is device dependent.

**10.7.5.1 :COLor <numeric\_value>**

Hard COPy:ITEM:LABEL:COLor

Sets or queries the color to be used for plotting or printing the label.

At \*RST, the value of this parameter is device dependent.

### 10.7.5.2 :DATA?

Hard COPy:ITEM:LABEL:DATA?

Returns the label encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

### 10.7.5.3 [:IMMEDIATE]

Hard COPy:ITEM:LABEL:IMMEDIATE

Immediately plot or print the label.

### 10.7.5.4 :STATE <Boolean>

Hard COPy:ITEM:LABEL:STATE

Sets or queries whether the label should be plotted or printed when the HCOPy:IMMEDIATE command or HCOPy:DATA? query is sent.

At \*RST, the value of this parameter is device dependent.

### 10.7.5.5 :TEXT <string>

Hard COPy:ITEM:LABEL:TEXT

Sets or queries the contents of the user label. The contents of the <string> is placed in an appropriate place on the hard copy output. The string may contain formatting characters. A textual label containing the empty string ("") shall be plotted/printed as a single blank line.

At \*RST, all textual labels are set to the empty string.

### 10.7.6 :MENU

Hard COPy:ITEM:MENU

Configures the plotting or printing of the device menu.

#### 10.7.6.1 :COLor <numeric\_value>

Hard COPy:ITEM:MENU:COLor

Sets or queries the color to be used for plotting or printing the menu.

At \*RST, the value of this parameter is device dependent.

#### 10.7.6.2 :DATA?

Hard COPy:ITEM:MENU:DATA?

Returns the menu encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

#### 10.7.6.3 [:IMMEDIATE]

Hard COPy:ITEM:MENU:IMMEDIATE

Immediately plot or print the menu.

#### 10.7.6.4 :STATE <Boolean>

Hard COPy:ITEM:MENU:STATE

Sets or queries whether the menu should be plotted or printed when the HCOPy:IMMEDIATE command or HCOPy:DATA? query is sent.

At \*RST, the value of this parameter is device dependent.

**10.7.7 :TDSTamp**

Hard COPy:ITEM:TDSTamp

Configures the plotting or printing of a time and date stamp.

**10.7.7.1 :COLor <numeric\_value>**

Hard COPy:ITEM:TDSTamp:COLor

Sets or queries the color to be used for plotting or printing the time and date stamp.

At \*RST, the value of this parameter is device dependent.

**10.7.7.2 :DATA?**

Hard COPy:ITEM:TDSTamp:DATA?

Returns the time and date stamp encapsulated in an &lt;INDEFINITE LENGTH ARBITRARY RESPONSE DATA&gt; element.

**10.7.7.3 [:IMMEDIATE]**

Hard COPy:ITEM:TDSTamp:IMMEDIATE

Immediately plot or print the time and date stamp.

**10.7.7.4 :STATe <Boolean>**

Hard COPy:ITEM:TDSTamp:STATe

Sets or queries whether the time and date stamp should be plotted or printed when the HCOPy:IMMEDIATE command or HCOPy:DATA? query is sent.

At \*RST, the value of this parameter is device dependent.

**10.7.8 [:WINDOW]**

Hard COPy:ITEM:WINDOW

Configures the plotting or printing of windows.

**10.7.8.1 :DATA?**

Hard COPy:ITEM:WINDOW:DATA?

Returns the window encapsulated in an &lt;INDEFINITE LENGTH ARBITRARY RESPONSE DATA&gt; element.

**10.7.8.2 [:IMMEDIATE]**

Hard COPy:ITEM:WINDOW:IMMEDIATE

Immediately plot or print the window.

**10.7.8.3 :STATe <Boolean>**

Hard COPy:ITEM:WINDOW:STATe

Sets or queries whether the window should be plotted or printed when the HCOPy:IMMEDIATE command or HCOPy:DATA? query is sent.

At \*RST, the value of this parameter is device dependent.

**10.7.8.4 :TEXT**

Hard COPy:ITEM:WINDOW:TEXT

Configures the plotting or printing of text blocks or textual labels.

**10.7.8.4.1 :COLor <numeric\_value>**

Hard COPy:ITEM:WINDow:TEXT:COLor

Sets or queries the color to be used for plotting or printing TEXT.

At \*RST, the value of this parameter is device dependent.

**10.7.8.4.2 :DATA?**

Hard COPy:ITEM:WINDow:TEXT:DATA?

Returns the menu encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

**10.7.8.4.3 [:IMMEDIATE]**

Hard COPy:ITEM:WINDow:TEXT:IMMEDIATE

Immediately plot or print the text blocks or textual labels.

**10.7.8.4.4 :STATe <Boolean>**

Hard COPy:ITEM:WINDow:TEXT:STATe

Sets or queries whether the text blocks or textual labels should be plotted or printed when the HCOPy:IMMEDIATE command or HCOPy:DATA? query is sent.

At \*RST, the value of this parameter is device dependent.

10

**10.7.8.5 :TRACe**

Hard COPy:ITEM:WINDow:TRACe

Configures the plotting or printing of graphical trace data.

**10.7.8.5.1 :COLor <numeric\_value>**

Hard COPy:ITEM:WINDow:TRACe:COLor

Sets or queries the color to be used for plotting or printing the trace.

At \*RST, the value of this parameter is device dependent.

**10.7.8.5.2 :DATA?**

Hard COPy:ITEM:WINDow:TRACe:DATA?

Returns the trace encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

**10.7.8.5.3 :GRATicule**

Hard COPy:ITEM:WINDow:TRACe:GRATICULE

Configures the plotting or printing of the graticule.

**10.7.8.5.3.1 :COLOR <numeric\_value>**

Hard COPy:ITEM:WINDow:TRACe:GRATICULE:COLOr

Sets or queries the color to be used for plotting or printing the graticule.

At \*RST, the value of this parameter is device dependent.

**10.7.8.5.3.2 :DATA?**

Hard COPy:ITEM:WINDow:TRACe:GRATICULE:DATA?

Returns the graticule encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

**10.7.8.5.3.3 [:IMMEDIATE]**

Hard COPy:ITEM:WINDOW:TRACe:GRATicule:IMMEDIATE

Immediately plot or print the graticule.

**10.7.8.5.3.4 :STATE <Boolean>**

Hard COPy:ITEM:WINDOW:TRACe:GRATicule:STATE

Sets or queries whether the graticule should be plotted or printed when the HCOPy:IMMEDIATE command or HCOPy:DATA? query is sent.

At \*RST, the value of this parameter is device dependent.

**10.7.8.5.4 [:IMMEDIATE]**

Hard COPy:ITEM:WINDOW:TRACe:IMMEDIATE

Immediately plot or print the trace.

**10.7.8.5.5 :LTYPe SOLId | DOTTeD | DASHed | STYLe<n>**

Hard COPy:ITEM:WINDOW:TRACe:LTYPe

Sets or queries the line type to be used for plotting or printing the trace.

For each value of &lt;n&gt; in the STYLe parameter, the appearance of the line type shall be different. The STYLe line types should be chosen so they appear different on the specific hard copy device being used. The actual appearance is device dependent.

At \*RST, the value of this parameter is device dependent.

**10.7.8.5.6 :STATE <Boolean>**

Hard COPy:ITEM:WINDOW:TRACe:STATE

Sets or queries whether the trace should be plotted or printed when the HCOPy:IMMEDIATE command or HCOPy:DATA? query is sent.

At \*RST, the value of this parameter is device dependent.

**10.8 :PAGE**

Hard COPy:PAGE

This subsystem contains commands to set up parameters relating to the medium of the plot or print output.

**10.8.1 :DIMensions**

Hard COPy:PAGE:DIMensions

Sets the plot or print dimensions. Different plotters or printers allow the user to set the page dimensions in different ways, such as length, width dimensions of the page, selecting a standard page size, or setting “corners” of the page in terms of plotter units (that is p1 and p2). A device will likely support the setting of plot or print dimensions in a subset of these ways, depending on which plotter or printer(s) it supports. The interaction of the various dimension commands is device dependent.

**10.8.1.1 :AUTO <Boolean>**

Hard COPy:PAGE:DIMensions:AUTO

When AUTO is on the device selects the default page dimensions. When AUTO is OFF the page dimensions are set by the other setting in this subsystem.

At \*RST, the value of this parameter is ON.

#### 10.8.1.2 :LLEFt <numeric\_value>,<numeric\_value>

Hard COPy:PAGE:DIMensions:LLEFt

Sets or queries the x,y position of lower left corner of the page. Units are percent of the width and length of the page.

At \*RST, the value of this parameter is device dependent.

#### 10.8.1.3 :QUADrant[<n>]

Hard COPy:PAGE:DIMensions:QUADrant

Sets dimensions so that the plot/print occupies one quadrant of the page.

<n>	quadrant		
1	upper right	2	1
2	upper left		
3	lower left		
4	lower right	3	4

This command is an event only.

#### 10.8.1.4 :URIGht <numeric\_value>,<numeric\_value>

Hard COPy:PAGE:DIMensions:URIGht

Specifies the x,y position of upper right corner of the page. Units are percent of the width and length of the page.

At \*RST, the value of this parameter is device dependent.

#### 10.8.2 :LENGth <numeric\_value>

Hard COPy:PAGE:LENGTH

Sets or queries the length of the page to be plotted or printed.

At \*RST, the value of this parameter is device dependent.

#### 10.8.3 :ORIentation LANDscape | PORTrait

Hard COPy:PAGE:ORIentation

Sets or queries the orientation of the plot or print. Switching between LANDscape and PORTrait rotates the hardcopy result by 90 degrees. No other settings are changed.

At \*RST, the value of this parameter is device dependent.

#### 10.8.4 :SCALE <numeric\_value>

Hard COPy:PAGE:SCALE

Sets or queries a scaling factor which is applied to the hard copy output. This factor applies to both dimensions. The output from HCOPy shall not attempt to place anything on the page outside the limits set under :DIMensions even when :SCALE is greater than one.

At \*RST, the value of this setting is one.

**10.8.5 :SIZE CUSTom|A|B|C|D|E|A0|A1|A2|A3|A4|B0|B1|B2|B3|B4|B5**

Hard COPy:PAGE:SIZE

Sets or queries the size of the paper in terms of a standard paper size. If LENGTH or WIDTH are set, the value of this setting becomes CUSTom.

At \*RST, the value of this parameter is device dependent.

**10.8.6 :UNIT <SUFFIX PROGRAM DATA>**

Hard COPy:PAGE:UNIT

Sets or queries the units for LENGTH and WIDTH under DIMensions.

At \*RST, the value of this parameter is device dependent.

**10.8.7 :WIDTh <numeric\_value>**

Hard COPy:PAGE:WIDTh

Sets or queries the width of the page to be plotted or printed.

At \*RST, the value of this parameter is device dependent.

**10.9 :SDUMp**

Hard COPy:SDUMp

The Screen DUMp subsystem obtains its FEED from the whole DISPlay subsystem, and shall only be used where a DISPlay subsystem is present in an instrument.

**10.9.1 :DATA?**

Hard COPy:SDUMp:DATA?

Returns the whole DISPlay encapsulated in an <INDEFINITE LENGTH ARBITRARY RESPONSE DATA> element.

**10.9.2 [:IMMEDIATE]**

Hard COPy:SDUMp:IMMEDIATE

This event shall cause the whole DISPlay to be plotted or printed.

## 1999 SCPI Command Reference

## 11 INPut Subsystem

The INPut subsystem controls the characteristics of a sensor's input ports.

KEYWORD	PARAMETER FORM	NOTES
INPut		
:ATTenuation	<numeric_value>	
:AUTO	<Boolean> ONCE	
:STATe	<Boolean>	
:BIAS		
:CURRent		
:AC	<numeric_value>	1991
[:DC]	<numeric_value>	1991
[:STATe]	<Boolean>	1991
:TYPE	CURRent   VOLTagE	1991
:VOLTagE		1991
:AC	<numeric_value>	1991
[:DC]	<numeric_value>	1991
:COUPling	AC DC  GROund	1991
:FILTer		
:AWEighting		1991
[:STATe]	<Boolean>	1991
:HPASs		
:FREQuency	<numeric_value>	
[:STATe]	<Boolean>	
[:LPASs]		
:FREQuency	<numeric_value>	
[:STATe]	<Boolean>	
:GAIN	<numeric_value>	
:AUTo	<Boolean> ONCE	
:STATe	<Boolean>	
:GUARD	LOW FLOat	1994
:IMPedance	<numeric_value>	
:LOW	FLOAT GROund	
:OFFSet	<numeric_value>	1991
[:STATe]	<Boolean>	1991
:POLarity	NORMal INVerted	1994
:POLarization	<numeric_value>	1994
:HORizontal		[no query] 1994
:VERTical		[no query] 1994
:POStion		1994
[:X]		1994
:ANGLE		1994
:DIRection	UP DOWN	1994

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
	[:IMMEDIATE] <numeric_value>	1994
	:LIMit	1994
	:HIGH   <numeric_value>	1994
	:LOW   <numeric_value>	1994
	[:STATe] <Boolean>	1994
	:OFFSet <numeric_value>	1994
	:VELOCITY <numeric_value>	1994
	[:DISTANCE]	1994
	:DIRection UP DOWN	1994
	[:IMMEDIATE] <numeric_value>	1994
	:LIMit	1994
	:HIGH   <numeric_value>	1994
	:LOW   <numeric_value>	1994
	[:STATe] <Boolean>	1994
	:OFFSet <numeric_value>	1994
	:VELOCITY <numeric_value>	1994
:Y		1994
	:ANGLE	1994
	:DIRection UP DOWN	1994
	[:IMMEDIATE] <numeric_value>	1994
	:LIMit	1994
	:HIGH   <numeric_value>	1994
	:LOW   <numeric_value>	1994
	[:STATe] <Boolean>	1994
	:OFFSet <numeric_value>	1994
	:VELOCITY <numeric_value>	1994
	[:DISTANCE]	1994
	:DIRection UP DOWN	1994
	[:IMMEDIATE] <numeric_value>	1994
	:LIMit	1994
	:HIGH   <numeric_value>	1994
	:LOW   <numeric_value>	1994
	[:STATe] <Boolean>	1994
	:OFFSet <numeric_value>	1994
	:VELOCITY <numeric_value>	1994
:Z		1994
	:ANGLE	1994
	:DIRection UP DOWN	1994
	[:IMMEDIATE] <numeric_value>	1994
	:LIMit	1994
	:HIGH   <numeric_value>	1994
	:LOW   <numeric_value>	1994
	[:STATe] <Boolean>	1994
	:OFFSet <numeric_value>	1994

KEYWORD	PARAMETER FORM	NOTES
:VELOCITY	<numeric_value>	1994
[:DISTANCE]		1994
:DIRECTION	UP DOWN	1994
[:IMMEDIATE]	<numeric_value>	1994
:LIMIT		1994
:HIGH	<numeric_value>	1994
:LOW	<numeric_value>	1994
[:STATE]	<Boolean>	1994
:OFFSET	<numeric_value>	1994
:VELOCITY	<numeric_value>	1994
[:STATE]	<Boolean>	
:TYPE	<character data>	1992

**11.1 :ATTenuation <numeric\_value>**

INPut:ATTenuation

Sets the input ATTenuation. The default units are in the current relative amplitude units.

This should only be used where the setting is not reflected within the SENSe system. For example, setting the input attenuation to 10 dB will cause the measured signal to decrease in amplitude by 10 dB.

This setting is not coupled to INPut:GAIN. If both ATTenuation and GAIN were set to 10 dB, the net result would be a signal of the original amplitude.

At \*RST, this value is device-dependent.

**11.1.1 :AUTO <Boolean>|ONCE**

INPut:ATTenuation:AUTO

AUTO ON changes the ATTenuation to achieve maximum sensitivity without overloading the input channel. Programming a specified ATTenuation sets AUTO OFF.

At \*RST, this value is set to ON.

**11.1.2 :STATe <Boolean>**

INPut:ATTenuation:STATe

Turns the input attenuator ON and OFF. OFF means, that any attenuation circuits are taken out of the signal path, thus, no distortion to the signal will happen.

At \*RST, this value is device dependent.

**11.2 :BIAS**

INPut:BIAS

This subsystem is used to control the bias applied to an external tranducer/detector.

**11.2.1 :CURRent**

INPut:BIAS:CURRent

Specifies the specific bias current being generated.

### 11.2.1.1 :AC <numeric\_value>

INPut:BIAS:CURRent:AC

Specifies the AC component of the bias current. The parameter is specified in Amperes.

At \*RST, this value is device-dependent.

### 11.2.1.2 [:DC] <numeric\_value>

INPut:BIAS:CURRent:DC

Specifies the DC component of the bias current. The parameter is specified in Amperes.

At \*RST, this value is device-dependent.

### 11.2.2 [:STATe] <Boolean>

INPut:BIAS:STATe

Sets or queries whether or not input biasing is enabled.

At \*RST, this value is device-dependent.

### 11.2.3 :TYPE CURRent | VOLTage

INPut:BIAS:TYPE CURRent

Specifies the type of bias to be generated. CURRent specifies a current bias and VOLTage specifies a voltage bias.

At \*RST, this value is device-dependent.

### 11.2.4 :VOLTage

INPut:BIAS:VOLTage

Specifies the specific bias voltage being generated.

#### 11.2.4.1 :AC <numeric\_value>

INPut:BIAS:VOLTage:AC

Specifies the AC component of the bias voltage. The parameter is specified in Volts.

At \*RST, this value is device-dependent.

#### 11.2.4.2 [:DC] <numeric\_value>

INPut:BIAS:VOLTage:DC

Specifies the DC component of the bias voltage. The parameter is specified in Volts.

At \*RST, this value is device-dependent.

### 11.3 :COUPLing AC|DC|GROund

INPut:COUPLing

Selects AC or DC coupling for the specified signal. DC coupling allows the measurement signal's AC and DC component to pass. AC coupling passes only the measurement signal's AC component. GROund coupling allows no signal to pass at all and grounds the input amplifier without grounding the input signal.

At \*RST, this value is device-dependent.

**11.4 :FILTer**

INPut:FILTter

The FILTer function allows a filter to be inserted in the path of the measurement signal. The FREQuency function allows the programming of a specific break frequency. The STATe command enables the filter.

**11.4.1 :AWEighting**

INPut:FILTter:AWEighting

Controls the “A” weighting input filter. This is a human hearing compensation filter described in the *IEC Recommendation 179* and *ANSI S1.4*.

**11.4.1.1 [:STATe] <Boolean>**

INPut:FILTter:AWEighting:STATe

Turns the “A” weighting input filter on and off.

At \*RST this value is device-dependent.

**11.4.2 :HPASs**

INPut:FILTter:HPASs

Controls the high pass filter.

**11.4.2.1 :FREQuency <numeric\_value>**

INPut:FILTter:HPASs:FREQuency

Determines the cutoff frequency of the high pass filter. Default units are Hertz.

At \*RST, this value is device-dependent.

**11.4.2.2 [:STATe] <Boolean>**

INPut:FILTter:HPASs:STATe

Turns the high pass filter ON and OFF.

At \*RST, this value is device-dependent.

**11.4.3 [:LPASs]**

INPut:FILTter:LPASs

Controls the low pass filter.

**11.4.3.1 :FREQuency <numeric\_value>**

INPut:FILTter:LPASs:FREQuency

Determines the cutoff frequency of the low pass filter. Default units are Hertz.

At \*RST, this value is device-dependent.

**11.4.3.2 [:STATe] <Boolean>**

INPut:FILTter:LPASs:STATe

Turns the low pass filter ON and OFF.

At \*RST, this value is device-dependent.

### 11.5 :GAIN <numeric\_value>

INPut:GAIN

Sets the input GAIN. The default units are in the current relative amplitude units.

This should only be used where the setting is not reflected within the SENSe system. For example, setting the input gain to 10 dB will cause the measured signal to increase in amplitude by 10 dB.

This setting is not coupled to INPut:ATTenuation. If both ATTenuation and GAIN were set to 10 dB, the net result would be a signal of the original amplitude.

At \*RST, this value is device-dependent.

#### 11.5.1 :AUTO <Boolean>|ONCE

INPut:GAIN:AUTO

AUTO ON changes the GAIN to achieve maximum sensitivity without overloading the input channel.

AT \*RST, this value is set to ON.

#### 11.5.2 :STATe <Boolean>

INPut:GAIN:STATe

Turns the input gain (preamplifier) ON and OFF. OFF means, that any preamplifier circuits are taken out of the signal path, thus, no distortion to the signal will happen. OFF is not the same as INPut:GAIN 0 dB, because an isolation preamplifier will have that gain.

At \*RST, this value is device dependent.

### 11.6 :GUARd LOW|FLOat

INPut:GUARD

Connects guard to signal low terminal or allows guard to float.

At \*RST, this value is device-dependent.

### 11.7 :IMPedance <numeric\_value>

INPut:IMPedance

Termination input impedance for the input signal specified in Ohms.

At \*RST, this value is device-dependent.

### 11.8 :LOW FLOat|GROund

INPut:LOW

Connects the low signal terminal to ground or allows it to float.

At \*RST, this value is device-dependent.

**11.9 :OFFSet <numeric\_value>**

INPut:OFFSet

Sets or queries the input OFFSet. OFFSet may take on positive and negative values. The OFFSet shall be subtracted from the input, thus making OFFSet more positive shall make the signal more negative. The default units are in the current amplitude units

Introducing an OFFSet in the input has the effect of offsetting the input signal, prior to detection in the SENSe block. Any offset introduced in the input block shall not be corrected for in the sensor and subsequently the sensor results. For example, if the current amplitude units are in volts, a value of 5 shall decrease the input signal by 5 volts.

At \*RST, the value of OFFSet is device-dependent.

**11.9.1 :STATe <Boolean>**

INPut:OFFSet:STATe

Turns the effect of OFFSet ON or OFF.

At \*RST, STATe shall be OFF.

**11.10 :POLarity NORMAL|INVerted**

INPut:POLarity

Sets or queries the input polarity. NORMAL causes the input signal to pass through the INPut block without inversion. INVerted causes the polarity of the input signal to be reversed. Any inversion introduced in the INPut block shall not be corrected for in the sensor and subsequently the sensor results.

At \*RST, this value is NORMAL.

**11.11 :POLarization <numeric\_value>**

INPut:POLarization

Sets the polarization of the input. The default units are radian.

At \*RST, this value is device-dependent.

**11.11.1 :HORizontal**

INPut:POLarization:HORizontal

Sets the polarization of the input to zero. This command is an event. Therefore it has no associated query or meaning at \*RST.

**11.11.2 :VERTical**

INPut:POLarization:VERTical

Sets the polarization of the input to PI/2 radians. This command is an event. Therefore it has no associated query or meaning at \*RST.

**11.12 :POStion**

INPut:POStion

The characteristics of an incoming signal can be affected by the input's spatial position and orientation relative to the origin. The definition of the origin is application specific. This subsystem uses a righthand coordinate system and has only one origin.

In this subsystem only the IMMEDIATE commands initiate movement.

In this subsystem velocity is always positive.

### 11.12.1 [:X]

INPut:POSition:X

Controls the settings concerning the X-axis.

#### 11.12.1.1 :ANGLE

INPut:POSition:X:ANGLE

Specifies the angle of rotation around the X-axis.

##### 11.12.1.1.1 :DIRECTION UP|DOWN

INPut:POSition:X:ANGLE:DIRECTION

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

##### 11.12.1.1.2 [:IMMEDIATE] <numeric\_value>

INPut:POSition:X:ANGLE:IMMEDIATE

This command indicates that the input is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

11

##### 11.12.1.1.3 :LIMIT

INPut:POSition:X:ANGLE:LIMIT

Sets the maximum bounds on the physical POSITION.

##### 11.12.1.1.3.1 :HIGH <numeric\_value>

INPut:POSition:X:ANGLE:LIMIT:HIGH

Sets the maximum for the POSITION.

At \*RST, this value is set to the clockwise physical limit.

##### 11.12.1.1.3.2 :LOW <numeric\_value>

INPut:POSition:X:ANGLE:LIMIT:LOW

Sets the minimum for the POSITION.

At \*RST, this value is set to the counterclockwise physical limit.

##### 11.12.1.1.3.3 :STATE <Boolean>

INPut:POSition:X:ANGLE:LIMIT:STATE

Controls whether the LIMIT is enabled.

At \*RST, this value is set to ON.

##### 11.12.1.1.4 :OFFSET <numeric\_value>

INPut:POSition:X:ANGLE:OFFSET

This defines a single value that is subtracted from the physical POSITION.

At \*RST, this value is device-dependent.

**11.12.1.1.5 :VELocity <numeric\_value>**

INPut:POSIon:X:ANGLE:VELocity

Controls the velocity of the angular rotation around the x-axis.

At \*RST, this value is device dependent.

**11.12.1.2 [:DISTance]**

INPut:POSIon:X:DISTance

Controls the settings in linear direction of the X-axis.

**11.12.1.2.1 :DIRection UP|DOWN**

INPut:POSIon:X:DISTance:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

**11.12.1.2.2 [:IMMEDIATE] <numeric\_value>**

INPut:POSIon:X:DISTance:IMMEDIATE

This command indicates that the input is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

**11.12.1.2.3 :LIMit**

INPut:POSIon:X:DISTance:LIMit

Sets the maximum bounds on the physical POSITION.

**11.12.1.2.3.1 :HIGH <numeric\_value>**

INPut:POSIon:X:DISTance:LIMit:HIGH

Sets the maximum for the POSITION.

At \*RST, this value is set to the upper physical limit.

**11.12.1.2.3.2 :LOW <numeric\_value>**

INPut:POSIon:X:DISTance:LIMit:LOW

Sets the minimum for the POSITION.

At \*RST, this value is set to the lower physical limit.

**11.12.1.2.3.3 :STATE <Boolean>**

INPut:POSIon:X:DISTance:LIMit:STATE

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

**11.12.1.2.4 :OFFSet <numeric\_value>**

INPut:POSIon:X:DISTance:OFFSet

This defines a single value that is subtracted from the physical POSITION.

At \*RST, this value is device-dependent.

**11.12.1.2.5 :VELocity <numeric\_value>**

INPut:POSition:X:DISTance:VELocity

Controls the velocity of the displacement on the x-axis.

At \*RST, this value is device dependent.

**11.12.2 :Y**

INPut:POSition:Y

Controls the settings concerning the Y-axis.

**11.12.2.1 :ANGLE**

INPut:POSition:Y:ANGLE

Specifies the angle of rotation around the Y-axis.

**11.12.2.1.1 :DIRection UP|DOWN**

INPut:POSition:Y:ANGLE:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

**11.12.2.1.2 [:IMMEDIATE] <numeric\_value>**

INPut:POSition:Y:ANGLE:IMMEDIATE

This command indicates that the input is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

**11.12.2.1.3 :LIMIT**

INPut:POSition:Y:ANGLE:LIMit

Sets the maximum bounds on the physical POSition.

**11.12.2.1.3.1 :HIGH <numeric\_value>**

INPut:POSition:Y:ANGLE:LIMit:HIGH

Sets the maximum for the POSition.

At \*RST, this value is set to the clockwise physical limit.

**11.12.2.1.3.2 :LOW <numeric\_value>**

INPut:POSition:Y:ANGLE:LIMit:LOW

Sets the minimum for the POSition.

At \*RST, this value is set to the counterclockwise physical limit.

**11.12.2.1.3.3 :STATE <Boolean>**

INPut:POSition:Y:ANGLE:LIMit:STATe

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

**11.12.2.1.4 :OFFSET <numeric\_value>**

INPut:POSition:Y:ANGLE:OFFSet

This defines a single value that is subtracted from the physical POSition.

At \*RST, this value is device-dependent.

#### 11.12.2.1.5 :VELOCITY <numeric\_value>

INPut:POSition:Y:ANGLE:VELOCITY

Controls the velocity of the angular rotation around the y-axis.

At \*RST, this value is device dependent.

#### 11.12.2.2 [:DISTANCE]

INPut:POSition:Y:DISTance

Controls the settings in linear direction of the Y-axis.

#### 11.12.2.2.1 :DIRECTION UP|DOWN

INPut:POSition:Y:DISTance:DIRECTION

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

#### 11.12.2.2.2 [:IMMEDIATE] <numeric\_value>

INPut:POSition:Y:DISTance:IMMEDIATE

This command indicates that the input is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

11

#### 11.12.2.2.3 :LIMIT

INPut:POSition:Y:DISTance:LIMIT

Sets the maximum bounds on the physical POSITION.

#### 11.12.2.2.3.1 :HIGH <numeric\_value>

INPut:POSition:Y:DISTance:LIMIT:HIGH

Sets the maximum for the POSITION.

At \*RST, this value is set to the upper physical limit.

#### 11.12.2.2.3.2 :LOW <numeric\_value>

INPut:POSition:Y:DISTance:LIMIT:LOW

Sets the minimum for the POSITION.

At \*RST, this value is set to the lower physical limit.

#### 11.12.2.2.3.3 :STATE <Boolean>

INPut:POSition:Y:DISTance:LIMIT:STATE

Controls whether the LIMIT is enabled.

At \*RST, this value is set to ON.

#### 11.12.2.2.4 :OFFSET <numeric\_value>

INPut:POSition:Y:DISTance:OFFSET

This defines a single value that is subtracted from the physical POSITION.

At \*RST, this value is device-dependent.

**11.12.2.2.5 :VELocity <numeric\_value>**

INPut:POSition:Y:DISTance:VELocity

Controls the velocity of the displacement on the y-axis.

At \*RST, this value is device dependent.

**11.12.3 :Z**

INPut:POSition:Z

Controls the settings concerning the Z-axis

**11.12.3.1 :ANGLE**

INPut:POSition:Z:ANGLE

Specifies the angle of rotation around the Z-axis.

**11.12.3.1.1 :DIRection UP|DOWN**

INPut:POSition:Z:ANGLE:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

**11.12.3.1.2 [:IMMEDIATE] <numeric\_value>**

INPut:POSition:Z:ANGLE:IMMEDIATE

This command indicates that the input is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

**11.12.3.1.3 :LIMIT**

INPut:POSition:Z:ANGLE:LIMIT

Sets the maximum bounds on the physical POSITION.

**11.12.3.1.3.1 :HIGH <numeric\_value>**

INPut:POSition:Z:ANGLE:LIMIT:HIGH

Sets the maximum for the POSITION.

At \*RST, this value is set to the clockwise physical limit.

**11.12.3.1.3.2 :LOW <numeric\_value>**

INPut:POSition:Z:ANGLE:LIMIT:LOW

Sets the minimum for the POSITION.

At \*RST, this value is set to the counterclockwise physical limit.

**11.12.3.1.3.3 :STATE <Boolean>**

INPut:POSition:Z:ANGLE:LIMIT:STATE

Controls whether the LIMIT is enabled.

At \*RST, this value is set to ON.

**11.12.3.1.4 :OFFSET <numeric\_value>**

INPut:POSition:Z:ANGLE:OFFSET

This defines a single value that is subtracted from the physical POSITION.

At \*RST, this value is device-dependent.

#### 11.12.3.1.5 :VELOCITY <numeric\_value>

INPut:POSition:Z:ANGLE:VELOCITY

Controls the velocity of the angular rotation around the z-axis.

At \*RST, this value is device dependent.

#### 11.12.3.2 [:DISTANCE]

INPut:POSition:Z:DISTance

Controls the settings in linear direction of the Z-axis.

##### 11.12.3.2.1 :DIRECTION UP|DOWN

INPut:POSition:Z:DISTance:DIRECTION

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

##### 11.12.3.2.2 [:IMMEDIATE] <numeric\_value>

INPut:POSition:Z:DISTance:IMMEDIATE

This command indicates that the input is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

11

##### 11.12.3.2.3 :LIMIT

INPut:POSition:Z:DISTance:LIMIT

Sets the maximum bounds on the physical POSITION.

##### 11.12.3.2.3.1 :HIGH <numeric\_value>

INPut:POSition:Z:DISTance:LIMIT:HIGH

Sets the maximum for the POSITION.

At \*RST, this value is set to the upper physical limit.

##### 11.12.3.2.3.2 :LOW <numeric\_value>

INPut:POSition:Z:DISTance:LIMIT:LOW

Sets the minimum for the POSITION.

At \*RST, this value is set to the lower physical limit.

##### 11.12.3.2.3.3 :STATE <Boolean>

INPut:POSition:Z:DISTance:LIMIT:STATE

Controls whether the LIMIT is enabled.

At \*RST, this value is set to ON.

##### 11.12.3.2.4 :OFFSet <numeric\_value>

INPut:POSition:Z:DISTance:OFFSet

This defines a single value that is subtracted from the physical POSITION.

At \*RST, this value is device-dependent.

### 11.12.3.2.5 :VELOCITY <numeric\_value>

INPut:POSition:Z:DISTance:VELOCITY

Controls the velocity of the displacement on the z-axis.

At \*RST, this value is device dependent.

### 11.13 [:STATe] <Boolean>

INPut:STATe

The STATe command connects the input terminal to the measurement signal path when ON. It can be turned OFF to allow maximum isolation of the measurement signal. Note that this does not mean ground the input signal.

At \*RST, STATe ON shall be selected.

### 11.14 :TYPE <character data>

INPut:TYPE

Selects the input characteristic to be used if more than one is available.

The following characteristics are currently defined for instruments:

- BALanced – balanced or differential input.
- UNBalanced – unbalanced with respect to ground.

At \*RST the TYPE is device-dependent.

## 12 INSTRument Subsystem

An instrument may support multiple logical instruments. A dual channel power supply for example is considered as two logical instruments. The logical instruments are not required to have identical functionality, nor does the functionality have to be available simultaneously. The INSTRument subsystem provides a mechanism to identify and select logical instruments by either name or number. In this way a particular logical instrument could be selected and it would respond to commands, such as MEASure, in the same manner as a dedicated instrument having the same functionality as the logical instrument. The INSTRument identifiers have no fixed correspondence to the numeric suffixes allowed with command headers.

KEYWORD	PARAMETER FORM	NOTES
INSTRument		
:CATalog?		[query only] 1994
:FULL?		[query only] 1994
:COUPle		
	[<subsystem>] ALL NONE <list>	
:DEFine		1999
:GROup	<identifier>,<identifier_list>	1999
[:NAME]	<identifier>,<numeric_value>	1999
:DElete		1999
[:NAME]	<identifier>	[event; no query] 1994
:ALL		[event; no query] 1994
:NSElect	<numeric_value>	
[:SElect]	<identifier>	1999
:STATe	<Boolean>	

### 12.1 :CATalog?

INSTRument:CATalog?

The CATalog query returns a comma-separated list of strings which contains the names of all logical instruments and groups. If no logical instruments are defined, a single null string is returned.

Some instruments may have one or more intrinsic (permanent) <identifier>'s. These are listed along with any user-named <identifier>'s.

#### 12.1.1 :FULL?

INSTRument:CATalog?:FULL?

CATalog:FULL? query returns a list of string - number pairs. The string contains the name of the logical instrument. The immediately following NR1-formatted number is its associated logical instrument number. All response data elements are comma separated. If no logical instrument is defined, a null string followed by a zero is returned.

Intrinsic <identifier>'s associated with logical instrument numbers are listed along with any user-named <identifier>'s and numbers.

### 12.2 :COUPle[:<subsystem>] ALL|NONE|<list>

INSTrument:COUPLE

Defines a coupling between various logical instruments or between various channels within an instrument. The COUPle command consists of an optional subsystem node followed by a single parameter.

The subsystem node denotes a SCPI subsection which can either affect each instrument individually, or affect all instruments/channels collectively. This node is used only when the user can turn this coupling ON or OFF. Predefined couplings need not use this node. If no node follows the COUPle command, then the entire instrument is assumed to be coupled. Coupling a node lower in the tree overrides any couplings which are assigned at a higher level.

The parameter indicates to which logical instruments the specified coupling is to apply. ALL indicates that specified coupling is to apply to all logical instruments. NONE indicates that specified coupling is to removed. A list of instruments specifies a particular set of logical instruments to be coupled.

Within SCPI certain subsystems, such as SYSTem, STATus, MEMory, MMEMory, TEST and DIAGnostic, are considered common to all logical instruments. That is, they appear coupled across all logical instruments, and they cannot be uncoupled. The IEEE common commands are assumed to always operate on all logical instruments.

At \*RST, all subsystems are uncoupled as far as the instrument hardware allows.

### 12.3 :DEFIne

INSTrument:DEFIne

#### 12.3.1 :GROup <identifier>,<identifier\_list>

INSTrument:DEFIne:GROup

Defines or queries a group of instruments. The <identifier> is character data. The <identifier\_list> is a comma-separated list of strings which contains the names of logical instruments. The defined instruments can be queried using :INSTrument:DEFIne:GROup? <identifier>. No instrument selection is performed by this command. Any instrument implementing the :GROup command shall define which commands are affected by groups.

When groups are defined and used, and data is set or queried, the following rules shall apply unless otherwise noted:

- When setting a list of values, each value will apply to the corresponding instrument in the selected group. If the number of values does not correspond to the number of instruments selected, no action is taken and error -115 is reported.
- When a set of values is required by a command, one set of values may be sent for each instrument in the group. For instance,

FOO a,b;

becomes

FOO a1,b1,a2,b2,...

- If a single parameter value is supplied for a command that expects one value per group member, the value is applied to every instrument in the currently selected group.
- Note that commands that take an arbitrary number of optional parameters need to have some way to determine how many parameters apply to each member of the group.
- When queried, the command shall return the value(s) for each instrument in the currently selected group.

When groups are defined and used, events will be sent to all instruments in the currently selected group.

### 12.3.2 **[:NAME] <identifier>,<numeric\_value>**

INSTRument:DEFine[:NAME]

The command form defines a logical instrument name, <identifier>, and associates it with a logical instrument number, <numeric\_value>. The <identifier> is in character data format.

The query form takes only <identifier> as a parameter and returns the logical instrument number.

### 12.4 **:DEDelete**

INSTRument:DEDelete

The DEDelete subsystem disassociates <identifier>'s from logical instrument numbers or groups of instrument <identifier>'s. After the :DEDelete command executes, the <identifier> is no longer accessible until it is redefined. The instrument behaves as if the <identifier> never existed.

#### 12.4.1 **:ALL**

INSTRument:DEDelete:ALL

This command is an event that disassociates all identifiers from their logical instrument numbers or identifier list, except for the currently selected identifier and any intrinsic identifier definitions. There is no query form.

The situation of “nothing to delete” does not cause an execution error.

This command is an event, and therefore has no \*RST value associated with it.

\*RST has no effect.

#### 12.4.2 **[:NAME] <identifier>**

INSTRument:DEDelete:NAME

This command is an event, which disassociates the named <identifier> from its logical instrument number or identifier list. There is no query form.

Attempting to delete the currently selected logical instrument or any intrinsic (permanent) <identifier> - number associations results in an -221, (settings conflict) execution error.

This command is an event, and therefore has no \*RST value associated with it.

**12.5 :NSElect <numeric\_value>**

INSTrument:NSElect

This command is used in conjunction with the SElect command. It serves the same purpose, except that it uses a numeric value instead of the identifier used in the SElect command.

When queried it shall return the logical instrument number. Note that the numbering used for logical instruments shall directly correspond to the numbers used in status reporting for multiple instruments; specifically the STAhus:QUESTIONable:INSTrument and STATus:OPERation:INSTrument commands. By selecting an instrument with either SElect or NSElect and querying the selected instrument with the other command, the equivalence between the instrument number and identifier can be determined. For example, by selecting a logical instrument using the SElect command with its name identifier and then querying which logical instrument is selected with the NSElect command, the corresponding numeric value shall be returned.

Assigning numeric values to logical instrument identifiers is not appropriate for instrument groups. Therefore, NSElect returns an error if a group has been selected.

**12.6 [:SELect] <identifier>**

INSTrument:SELect

This command selects a path or entire instrument or group as the default. When a logical instrument or a group is selected, all other logical instruments or groups are unavailable for programming until selected. If group logical identifiers are not used, and either :SELect or :NSElect commands are implemented, then both of the commands must be implemented.

Character data names may be predefined by the instrument.

At \*RST, a device-dependent instrument is selected.

**12.7 :STATe <Boolean>**

INSTrument:STATe

Turns the selected logical instrument ON or OFF. A logical instrument does not have to be turned OFF before another logical instrument is selected. That is, several logical instruments may be active, while only one may be selected. When an instrument is active, measurements occur and signals are generated. When inactive, measurements do not occur and signals are not generated. When a logical instrument is selected, yet is inactive, all commands shall be processed so that the logical instrument shall reflect the state changes requested when the logical instrument is turned on. The exception to this are commands which would normally cause trigger events in an active logical instrument. Any of these commands shall cause an execution error (-200) to be generated.

At \*RST, this value is device-dependent.

## 13 MEMory Subsystem

The purpose of the memory subsystem is to manage instrument memory. This specifically excludes memory used for mass storage, which is defined in the MMEMory Subsystem. Instrument memory may be used to store several data types (e.g. ASCii, BINary, instrument STAtE, TRACe or MACRo.) The data types available for a given instrument are device-dependent.

An instrument may support either fixed or dynamic allocation of instrument memory, between the different data types. With fixed allocation a certain amount of memory is dedicated to each of the supported data types. The dedicated memory is unavailable for use by any other data type. Alternatively, with dynamic allocation, instrument memory is shared between the various data types supported on a demand basis.

In the MEMory:TABLE subsystems, many fundamental measurement are defined as subnodes below :TABLE, such as MEMory:TABLE:FREQuency, :VOLTage, :POWER, etc. These nodes define standardized entries in the table (note that these entries can be conceptualized as either a column or a row). While it makes sense to define widely used entries in the SCPI Volume 2 standard, there is sometimes a desire to standardize very instrument class specific names. Where instrument class specific names require standardization, and little or no overlap is expected with other instrument classes, these instrument class specific MEMory:TABLE nodes are defined in the Volume 4, Instrument classes chapter to which they apply.

As an example, the SCPI commands for Chassis Dynamometers :FORCe :SPEed and TIME are defined in Volume 2 under :MEMory:TABLE. These are fundamental measurements, and might be used by many instruments. However Deceleration RATE is deemed to be specific to dynamometers and is defined in Volume 4.

13

KEYWORD	PARAMETER FORM	NOTES
MEMory		
:CATalog		
[:ALL]?		[query only]
:ASCii?		[query only]
:BINary?		[query only]
:MACRo?		[query only]
:STAtE?		[query only]
:TABLE?		[query only] 1992
:CLEar		1993
[:NAME]	<name>	[event; no query] 1993
:TABLE	<name>	[event; no query] 1993
:COPY		1993
[:NAME]	<name>,<name>	[event; no query] 1993
:TABLE	<name>	[event; no query] 1993
:DATA	<name>,<data>	1994
:DELETE		
:ALL		[event; no query]

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
[:NAME]	<name>	[event; no query] 1993
:EXChange		
[:NAME]	<name>,<name>	[event; no query] 1993
:TABLE	<name>	[event; no query] 1993
:FREE		
[:ALL]?		[query only]
:ASCIi?		[query only]
:BINary?		[query only]
:MACRo?		[query only]
:STATe?		[query only]
:TABLE?		[query only] 1992
:NSTates?		[query only]
:STATe		
:CATAlog?		
:DEFIne	<name>,<register_number>	[query only] 1994
:DEFIne?	<name>	1994
:TYPE?	<name>	[query only] 1994
:TABLE		
:BNUMber<n>	<numeric_value>{,<numeric_value>}	1992
POINts		1999*
:CCURve	<numeric_value>{,<numeric_value>}	[query only] 1999*
POINts		1999*
:CONCetration	<numeric_value>{,<numeric_value>}	[query only] 1999*
POINts		1999*
:CONDition		
[:MAGNitude]	<Boolean>{,<Boolean>}	1993
:POINts?		[query only] 1993
:CPOint	<numeric_value>{,<numeric_value>}	1999*
POINts		[query only] 1999*
:CURREnt		
[:MAGNitude]	<numeric_value>{,<numeric_value>}	1992
:POINts?		[query only] 1992
:PHASE	<numeric_value>{,<numeric_value>}	1992
:POINts?		[query only] 1992
:DEFIne	<identifier>[,<identifier>]	1999
:DFACtory	<numeric_value>{,<numeric_value>}	1999*
POINts		[query only] 1999*
:DLAST	<numeric_value>{,<numeric_value>}	1999*
POINts		[query only] 1999*
:DLINEarize	<numeric_value>{,<numeric_value>}	1999*
POINts		[query only] 1999*
:EXPected	<numeric_value>{,<numeric_value>}	1999*
POINts		[query only] 1999*
:FORCe		1999

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:POINts?	<numeric_value> {,<numeric_value>}	1999
:FREQuency	<numeric_value>{,<numeric_value>}	1999
:POINts?		1992
:LABel	<string>{,<string>}	1999*
POINts		1999*
:LLIMit	<numeric_value>{,<numeric_value>}	1999
:POINts?		1999
:LOG	<string>{,<string>}	1999*
POINts		1999*
:LOSS		1992
:POINts?	<numeric_value>{,<numeric_value>}	1992
:PHASe	<numeric_value>{,<numeric_value>}	1992
:POINts?		1992
:NCURve	<numeric_value>{,<numeric_value>}	1999*
POINts		1999*
:POWER		1992
:POINts?	<numeric_value>{,<numeric_value>}	1992
:RAW	<numeric_value>{,<numeric_value>}	1999*
POINts		1999*
:RESistance		1992
:POINts?	<numeric_value>{,<numeric_value>}	1992
:SElect	<identifier>	1999
:SPEEd		1999
:POINts?	<numeric_value> {,<numeric_value>}	1999
:TIME		1999
:POINts?	<numeric_value> {,<numeric_value>}	1999
:TOLERance	<numeric_value>{,<numeric_value>}	1999*
POINts		1999*
:ULIMit	<numeric_value>{,<numeric_value>}	1999
:POINts?		1999
:VOLTage		1992
:POINts?	<numeric_value>{,<numeric_value>}	1992
:PHASe	<numeric_value>{,<numeric_value>}	1992
:POINts?		1992
:WFACTOR	<numeric_value>{,<numeric_value>}	1999*
POINts		1999*

## 1999 SCPI Command Reference

\* NOTE: The commands marked with an asterisk query or set values inside of predefined emissions bench memory tables. The tables involved are as follows:

Table	Column Name(s)	Description
CPIxxRyy	CPOint,TOLerance,WFACTorWT	Linearization input cut point tables
GasBottle	BNUMber,CONCenTration,LABel	Gas bottle table
LSIxxRyy	CPOint,EXPected,RAW,CCURve,NCURve	Linearization results tables
VERify	BNUMber1,BNUMber2,...BNUMber	Verify gas table
QLIxRyy	LOG	Linearization log tables
QNIxxRyy	LOG	NOx efficiency log tables
SDRift	DFACTory,DLAST,DLINearize	Span Drift results table
SpanGas	BNUMber1,BNUMber2,...BNUMber	Span Gas table
TopGas	BNUMber1,BNUMber2,...BNUMber	Top Gas table
ZDRift	DFACTory,DLAST,DLINearize	Zero Drift results table
ZeroGas	BNUMber1,BNUMber2,...BNUMber	Zero Gas table

The <name> parameter is used to define a user label, which shall be <character program data>. This label can then be used by the application in place of the specified data defined here or in other subsystems. The <name> parameter is sent as <character data>, rather than as a string, because that is the simplest IEEE 488.2 type that meets the requirements. When a name is returned, for example, in response to a CATalog? query, it must be sent as <string response data>, so that a null string ("") can indicate that nothing is defined.

The MEMory capabilities of an instrument are not part of the instrument state and are not affected by \*RST or \*RCL. They may survive a power cycle, but this capability is optional. If the instrument implements SYSTem:SECure mode, MEMory:DElete:ALL must be executed when SYSTem:SEC OFF is executed.

### 13.1

#### :CATalog

MEMory:CATalog

The CATalog queries return information on the current contents and state of the instrument's memory. The returned data shall be in the following form:

<numeric\_value>,<numeric\_value> {,<string>}

The instrument shall return two numeric parameters and as many strings as there are items in the directory list. The first parameter shall indicate the total amount of storage currently used, in bytes. The second parameter shall indicate the total amount of storage available, also in bytes. Each string parameter returned shall indicate the name, type and size of one in the directory list, thus:

<name>,<type>,<size>

The <name>, <type> and <size> are all character data. The <name> is the exact name of a file as it appears in the directory list, including an extension if present. The <size> provides the size of the file in bytes. The <type> is indicated by the name of the major subsystem or one of the following:

- ASCII — An ASCII text file

- BINary — A binary file
- MACRo — An instrument macro
- STATe — An instrument state
- TABLE — A table of data

The qualified CATalog query for TRACe memory is specified in the TRACe subsystem.

### 13.1.1 **:ALL?**

MEMory:CATalog:ALL?

If the CATalog query is qualified by the optional ALL node then the query shall return a directory list and memory sizes related to all allocatable items in instrument memory. Note that this includes all allocatable objects, such as trace, etc.

### 13.1.2 **:ASCII?**

MEMory:CATalog:ASCII?

If the CATalog query is qualified by the ASCII node then the query shall return the directory list related only to ASCII items in instrument memory. The memory usage and availability parameters shall reflect values that correspond to values for ASCII data. In an instrument that has dedicated memory for ASCII data types, values reported shall be different from the total given by a CATalog of ALL types.

### 13.1.3 **:BINary?**

MEMory:CATalog:BINary?

If the CATalog query is qualified by the BINary node then the query shall return the directory list related only to BINary items in instrument memory. The memory usage and availability parameters shall reflect values that correspond to values for BINary data. In an instrument that has dedicated memory for BINary data types, values reported shall be different from the total given by a CATalog of ALL types.

13

### 13.1.4 **:MACRo?**

MEMory:CATalog:MACRo?

If the CATalog query is qualified by the MACRo node then the query shall return the directory list related only to MACRo items in instrument memory. The memory usage and availability parameters shall reflect values that correspond to values for MACRo data. In an instrument that has dedicated memory for MACRo data types, values reported shall be different from the total given by a CATalog of ALL types.

### 13.1.5 **:STATe?**

MEMory:CATalog:STATe?

If the CATalog query is qualified by the STATe node then the query shall return the directory list related only to STATe items in instrument memory. The memory usage and availability parameters shall reflect values that correspond to values for STATe data. In an instrument that has dedicated memory for STATe data types, values reported shall be different from the total given by a CATalog of ALL types.

### 13.1.6 :TABLE?

MEMory:CATalog:TABLE?

If the CATalog query is qualified by the TABLE node then the query shall return the directory list related only to TABLE items in instrument memory. The memory usage and availability of parameters shall reflect values that correspond to values for TABLE data. In an instrument that has dedicated memory for TABLE data types, the values reported shall be different from the total given by a CATalog of ALL types.

### 13.2 :CLEar

MEMory:CLEar

The commands from this subsystem remove the data that is stored in instrument memory. This subsystem differs from the DELETED subsystem in that CLEar removes the data contents but does not affect the memory definitions such as memory type and structure, available space, and the name of the associated instrument memory.

#### 13.2.1 [:NAME] <name>

MEMory:CLEar:NAME

This command is a default node and removes all data from the instrument memory associated with <name>. The NAME command does not affect the related memory definitions.

This command is an event with no associated query form and \*RST condition.

#### 13.2.2 :TABLE

MEMory:CLEar:TABLE

The CLEar:TABLE command removes the data values from all element lists in the SElected TABLE. This command does not affect the space and structure of the table as specified with the table's DEFine command. Executing a POINTs? query for any table element after the CLEar:TABLE command, would return 0.

This command is an event with no associated query form and \*RST condition.

### 13.3 :COPY

MEMory:COPY

The COPY subsystem copies the data from an existing instrument memory item into another, named memory item.

#### 13.3.1 [:NAME] <name>,<name>

MEMory:COPY:NAME

This command is a default node that copies the data contents from the memory item associated with the <name> specified in the first parameter to a memory item associated with the <name> specified by the second parameter.

When the memory item to which data is copied is of a different type, or has an incompatible structure, an execution error - 294 (Incompatible type) is generated. When the specified destination name does not exist, the device may create a new memory item of the same type and structure as the source. In that case, the destination name is to be associated with the new created memory item.

This command is an event with no associated query form and \*RST condition.

### 13.3.2 :TABLE <name>

MEMory:COPY:TABLE

The COPY:TABLE command copies the data values of all element lists from the SElected TABLE to the memory table associated with the specified <name>.

When the memory item to which data is copied is of a different type, or has an incompatible structure, an execution error -294 (Incompatible type) is generated. When the TABLE name does not exist, the device may define a new memory table with the same structure as the selected table. In that case, the specified TABLE name is associated with the new created memory table.

This command is an event with no associated query form and \*RST condition.

### 13.4 :DATA<name>,<data>

MEMory:DATA

The command form is MEMory:DATA <name>,<data>. The command loads <data> into the memory location <name>. <data> is in 488.2 block format. <name> is character data.

The query form is MEMory:DATA? <name> with the response being the associated <data> in block format.

MEMory:DATA? response is not affected by the FORMat subsystem.

### 13.5 :DELETED

MEMory:DELETED

The DELETED command purges the specified definition from instrument memory and makes it available for reuse.

13

### 13.5.1 :ALL

MEMory:DELETED:ALL

The ALL command removes all memory names, key definitions, and data, and it returns memory to "available."

### 13.5.2 [:NAME] <name>

MEMory:DELETED:NAME

The NAME command is an optional node and deletes the definition associated with <name>.

### 13.6 :EXCHange

MEMory:EXCHange

The EXCHange subsystem allows for swapping the data contents between two memory items.

### 13.6.1 [:NAME] <name>,<name>

MEMory:EXCHange:NAME

This command is a default node which swaps the data contents between the memory items associated with the <name> parameters.

When the memory items between which the data is swapped are of different type, or have an incompatible structure, an execution error -294 (Incompatible type) is generated.

## 1999 SCPI Command Reference

This command is an event with no associated query form and \*RST condition.

### 13.6.2 :TABLE <name>

MEMory:EXChange:TABLE

This command swaps the data contents between the SElected TABLE and the memory table that is associated with the <name> parameter.

When the memory item associated with the specified <name> parameter is not a TABLE, or the tables between which the data is swapped have an incompatible structure, an execution error -294 (Incompatible type) is generated.

This command is an event with no associated query form and \*RST condition.

### 13.7 :FREE

MEMory:FREE

Returns information on the user memory space as follows:

<bytes available>,<bytes in use>

where the returned data for both shall be in <NR1> format.

The qualified FREE query for TRACe only memory usage is specified in the TRACe subsystem.

### 13.7.1 [:ALL]?

MEMory:FREE:ALL?

If the FREE query is qualified by the optional ALL node then the query shall return memory sizes related to all items in instrument memory.

### 13.7.2 :ASCII?

MEMory:FREE:ASCII?

If the FREE query is qualified by the ASCII node then the memory usage and availability shall reflect the values that correspond to values for ASCII data. In an instrument that has dedicated memory for ASCII data types, these values reported shall be different from the total given by a FREE query of ALL types.

### 13.7.3 :BINary?

MEMory:FREE:BINary?

If the FREE query is qualified by the BINary node then the memory usage and availability shall reflect the values that correspond to values for BINary data. In an instrument that has dedicated memory for BINary data types, these values reported shall be different from the total given by a FREE query of ALL types.

### 13.7.4 :MACRo?

MEMory:FREE:MACRo?

If the FREE query is qualified by the MACRo node then the memory usage and availability shall reflect the values that correspond to values for MACRo data. In an instrument that has dedicated memory for MACRo data types, these values reported shall be different from the total given by a FREE query of ALL types.

**13.7.5 :STATE?**

MEMory:FREE:STATE?

If the FREE query is qualified by the STATE node then the memory usage and availability shall reflect the values that correspond to values for STATE data. In an instrument that has dedicated memory for STATE data types, these values reported shall be different from the total given by a FREE query of ALL types.

**13.7.6 :TABLE?**

MEMory:FREE:TABLE?

If the FREE query is qualified by the TABLE node then the memory usage and availability shall reflect the values that correspond to values for TABLE data. In an instrument that has dedicated memory for TABLE types, the values reported shall be different from the total given by a FREE query of ALL types.

**13.8 :NSTates?**

MEMory:NSTates?

The Number of STates query requests that the instrument return the number of \*SAV/\*RCL states available in the instrument. The instrument shall return a <NR1> value which is one greater than the maximum that can be sent as a parameter to the \*SAV and \*RCL commands.

**13.9 :STATE**

MEMory:STATE

Collects commands relating to memory STATE data.

**13.9.1 :CATalog?**

MEMory:STATE:CATalog?

13

This query requests a list of defined names in the MEMory:STATE subsystem. The instrument returns a list of defined <name>'s in a comma separated list. Each <name> is returned in a quoted string. If no names are defined, a single null string is returned. There is no command form.

**13.9.2 :DEFine <name> , <register\_number>**

MEMory:STATE:DEFine

:DEFine associates a <name> with a \*SAV/\*RCL register number where <name> is in character data format. <register\_number> is in <NRf> format.

The query form of this command, MEMory:STATE:DEFine? <name> returns the associated <register\_number> in <NR1> format.

\*RST has no effect on this command.

**13.10 :TYPE? <name>**

MEMory:TYPE?

The TYPE query requests information about a defined name. The instrument must return a string indicating the subsystem where the name was defined. If the name is not defined, a null string is returned. For example, if VAT3 was previously defined by:

*ROUTE:MODULE:DEFine VAT3,(@20000(0:9))*

then the query:

*MEMory:TYPE? VAT3*

requests a return of

*"ROUT:MOD:"*

### 13.11 :TABLE

MEMory:TABLE

The MEMory:TABLE command allows the user to DEFine a TABLE and to write data to and read data from it.

#### 13.11.1 :BNUMber <numeric\_value> {,<numeric\_value>}

:MEMory:TABLE:BNUMber

BNUMber is the gas bottle number. This command sets the gas bottle number, which is a positive integer.

NOTE: For Emissions benches, BNUMber<n> are bottle ID number columns in the GasBottle, ZeroGas, SpanGas, Verify, and TopGas tables. In the ZeroGas, SpanGas, Verify, and TopGas tables, there are columns BNUMber<n>, where n corresponds to the analyzer range number. In GasBottle, there is a single BNUMber column.

#### 13.11.1.1 :POINts?

:MEMory:TABLE:BNUMber:POINts?

This command returns the number of points in column BNUMber, for the currently selected table.

#### 13.11.2 :CCURve <numeric\_value> {,<numeric\_value>}

:MEMory:TABLE:CCURve

CCURve sets the value of concentration in ppm using the current curve in linearization tables. This command sets the values of CCURve.

#### 13.11.2.1 :POINts?

:MEMory:TABLE:CCURve:POINts?

This command returns the number of points in column CCURve, for the currently selected table.

#### 13.11.3 :CONCntration <numeric\_value> {,<numeric\_value>}

:MEMory:TABLE:CONCntration

CONCntration is the gas concentration in ppm. This command sets the values of CONCntration.

#### 13.11.3.1 :POINts?

:MEMory:TABLE:CONCntration:POINts?

This command returns the number of points in column CONCntration, for the currently selected table.

#### 13.11.4 :CONDition

MEMory:TABLE:CONDition

**13.11.4.1 [:MAGNitude] <Boolean>{,<Boolean>}**

MEMORY:TABLE:CONDition:MAGNitude

Specifies the CONDITION[:MAGNitude] points of the TABLE. If there are more <Boolean>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as CONDITION[:MAGNitude]

At \*RST, the contents of this list is device dependent.

**13.11.4.1.1 :POINts?**

MEMORY:TABLE:CONDition:MAGNitude:POINts?

Returns the number of points in the CONDITION[:MAGNitude] list of the SElected TABLE. If no CONDITION[:MAGNitude] value have been sent, POINts? returns a 0. If no TABLE has been selected, POINts? returns a NAN.

**13.11.5 :CPOint <numeric\_value>{,<numeric\_value>}**

:MEMORY:TABLE:CPOint

CPOint is the linearization gas dilution percentage or “cut point”. This command sets the values of CPOint.

**13.11.5.1 :POINts?**

:MEMORY:TABLE:CPOint:POINts?

This command returns the number of points in column CPOint, for the currently selected table.

**13.11.6 :CURRent**

MEMORY:TABLE:CURRent

13

**13.11.6.1 [:MAGNitude] <numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:CURRent:MAGNitude

Specifies the CURRent[:MAGNitude] points of the TABLE. <numeric\_value>'s may include current units as listed in the UNIT subsystem. The default unit is specified by UNIT:CURRent. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as CURRent[:MAGNitude].

At \*RST, the contents of this list item is device dependent.

**13.11.6.1.1 :POINts?**

MEMORY:TABLE:CURRent:MAGNitude:POINts?

Returns the number of points in the CURRent[:MAGNitude] list of the SElected TABLE. If no CURRent[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.6.2 :PHASe <numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:CURRent:PHASe

Specifies the CURRent:PHASe points of the TABLE. <numeric\_value>'s may include angle units as listed in the UNIT subsystem. The default unit is specified by UNIT:ANGLE. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error

## 1999 SCPI Command Reference

-223 (Too much data) will be generated. This item list is included in the table DEFine command as CURREnt:PHASe.

At \*RST, the contents of this list item is device dependent.

### 13.11.6.2.1 :POINts?

:MEMory:TABLE:CURREnt:PHASE:POINts?

Returns the number of points in the CURREnt:PHASE list of the SElected TABLE. If no CURREnt:PHASE values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

### 13.11.7 :DFACtory <numeric\_value> {<numeric\_value>}

:MEMory:TABLE:DFACtory

DFACtory is the maximum allowable drift from factory settings.

For emissions benches, this is the zero or span factory setting in the tables ZDRift and SDRift respectively. This command sets the values of DFACtory. Drift from factory setting is calculated as follows:

Zero Drift       $ZD_f = \frac{(Z^* - Z_f)}{f_s} * 100$

Span Drift       $SD_f = \frac{(S^* - S_f)}{f_s} * 100$

where       $Z^*$  and  $S^*$  are the currently read zero and span values in ppm

$Z_f$  and  $S_f$  are the factory settings stored in DFACtory in ppm

$f_s$  is the full scale CONCenration in ppm

$ZD_f$  is the zero drift in PCT

$SD_f$  is the span drift in PCT

Note: a ratio of 1.0 = 100 PCT

### 13.11.7.1 :POINts?

:MEMory:TABLE:DFACtory:POINts?

This command returns the number of points in column DFACtory, for the currently selected table.

### 13.11.8 :DLASt <numeric\_value> {<numeric\_value>}

:MEMory:TABLE:DLASt

DLASt is the maximum allowable drift from the most recently saved zero and set point.

For emissions benches, this is the most recently saved zero or span setting in the tables ZDRift and SDRift respectively. This command sets the values of DLASt. Drift from most recently saved zero or span (SPOint) setting is calculated as follows:

Zero Drift       $ZD_0 = \frac{(Z^* - Z_0)}{f_s} * 100$

$$\text{Span Drift} \quad SD_0 = \frac{(S^* - S_0)}{f_s} * 100$$

where  $Z^*$  and  $S^*$  are the currently read zero and span values in ppm  
 $Z_0$  and  $S_0$  are the most recently saved settings stored in DLAST in ppm  
 $f_s$  is the full scale CONCcentration in ppm  
 $ZD_0$  is the zero drift in PCT  
 $SD_0$  is the span drift in PCT

Note: a ratio of 1.0 = 100 PCT

### 13.11.8.1 :POINts?

:MEMory:TABLE:DLASt:POINts?

This command returns the number of points, for the currently selected table.

### 13.11.9 :DLINearize <numeric\_value> {,<numeric\_value>}

:MEMory:TABLE:DLINearize

DLINearize is the maximum allowable drift from the most recent linearization.

For emissions benches, this is the zero or span setting saved at the time of the most recent linearization in the tables ZDRift and SDRift respectively. This command sets the values of DLINearization. Drift from most recent linearization is calculated as follows:

$$\text{Zero Drift} \quad ZD_l = \frac{(Z^* - Z_l)}{f_s} * 100$$

$$\text{Span Drift} \quad SD_l = \frac{(S^* - S_l)}{f_s} * 100$$

where  $Z^*$  and  $S^*$  are the currently read zero and span values in ppm  
 $Z_l$  and  $S_l$  are the settings from the most recent linearization stored in DLINearize in ppm  
 $f_s$  is the full scale CONCcentration in ppm  
 $ZD_l$  is the zero drift in PCT  
 $SD_l$  is the span drift in PCT

Note: a ratio of 1.0 = 100 PCT

### 13.11.9.1 :POINts?

:MEMory:TABLE:DLINearize:POINts?

This command returns the number of points in column DLINearize, for the currently selected table.

### 13.11.10 :EXPected <numeric\_value> {,<numeric\_value>}

:MEMory:TABLE:EXPected

EXPected is the expected value.

For emissions benches, EXPected is the expected gas concentration in ppm. This command sets the values of EXPected.

### 13.11.10.1 :POINts?

:MEMORY:TABLE:EXPected:POINts?

This command returns the number of points in column EXPected , for the currently selected table.

### 13.11.11 :DEFIne <structure\_string>[,<numeric\_value>]

MEMory:TABLE:DEFIne

The DEFIne subsystem is used to allocate space and define the structure of TABLE in the instrument memory. The <name> of the TABLE is specified by SElect.

The string parameter <structure\_string> specifies the structure of the TABLE being defined, it contains a comma separated list of the elements. Any item lists under MEMORY:TABLE, such as FREQuency or VOLTage:PHASE, may be specified as elements in the table structure. The item lists maybe specified in long or short form, optional nodes if sent shall be accepted without error. The item lists available and the legal combinations of these item lists are device dependent. All lists are empty (0 length) when DEFIned.

The second parameter, is the size of the TABLE. This is the maximum number of POINts in each of the lists. Computations based on device dependent internal data formats will be made to determine the number of bytes. For example, if the TABLE will have two lists and the internal storage requires 8 bytes for a number, then an entry of 10 will allocate 160 bytes. If there is insufficient room, execution error -225 (Out of memory) will be generated. The size of the table parameter is optional. The algorithm to allocate memory is device dependent when this parameter is omitted.

The DEFIne query returns both the <structure\_string> and a numeric value for the size (in points) of memory allocated for the table. The elements of the structure string are returned in the short form, with any optional nodes omitted.

### 13.11.12 :FORCe

MEMory:TABLE:FORCe

Specifies the FORCe points of the TABLE.

### 13.11.12.1 [:MAGNitude] <numeric\_value>{,<numeric\_value>}

MEMory:TABLE:FORCe:MAGNitude

Specifies the FORCe[:MAGNitude] points of the TABLE. The default units of this command are newtons.

At \*RST, the contents of this list item is device dependent.

### 13.11.12.1.1 :POINts?

MEMory:TABLE:FORCe:MAGNitude:POINts?

Returns the number of points in the FORCe[:MAGNitude] list of the selected TABLE. If no FORCe[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.13 :FREQuency <numeric\_value>{,<numeric\_value>}**

:MEMORY:TABLE:FREQuency

Specifies the FREQuency points of the TABLE. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as FREQuency.

At \*RST, the contents of this list item is device dependent.

**13.11.13.1 :POINts?**

:MEMORY:TABLE:FREQuency:POINts?

Returns the number of points in the FREQuency list of the SElected TABLE. If no FREQuency values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.14 :LABel <string> {,<string>}**

:MEMORY:TABLE:LABel

LABel is the label name in a table in character string format. This command sets the values of LABel.

For emissions benches, LABel is the label name in the GasBottle table.

**13.11.14.1 :POINts?**

:MEMORY:TABLE:LABel:POINts?

This command returns the number of points in column LABel, for the currently selected table.

**13.11.15 :LLIMit <numeric\_value> {,<numeric\_value>}**

:MEMORY:TABLE:LLIMit

LLIMit is the lower limit value. This command sets the values of LLIMit. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as LLIMit.

At \*RST, the contents of this list item is device dependent.

**13.11.15.1 :POINts?**

:MEMORY:TABLE:LLIMit:POINts?

This command returns the number of points in column LLIMit, for the currently selected table. If no LLIMit values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.16 :LOG <string> {,<string>}**

:MEMORY:TABLE:LOG

LOG is a series of string data elements.

For emissions benches, LOG is a collection of audit-related strings. This command sets the values of LOG.

**13.11.16.1 :POINts?**

:MEMORY:TABLE:LOG:POINts?

This command returns the number of points in column LOG, for the currently selected table.

**13.11.17 :LOSS**

MEMORY:TABLE:LOSS

**13.11.17.1 [:MAGNitude] <numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:LOSS:MAGNitude

Specifies the LOSS[:MAGNitude] points of the TABLE. The default unit is DB. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as LOSS[:MAGNitude].

At \*RST, the contents of this list item is device dependent.

**13.11.17.1.1 :POINts?**

MEMORY:TABLE:LOSS:MAGNitude:POINts?

Returns the number of points in the LOSS[:MAGNitude] list of the SElected TABLE. If no LOSS[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.17.2 :PHASe <numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:LOSS:PHASe

Specifies the LOSS:PHASe points of the TABLE. <numeric\_value>'s may include angle units as listed in the UNIT subsystem. The default unit is specified by UNIT:ANGLE. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as LOSS:PHASe.

At \*RST, the contents of this list item is device dependent.

**13.11.17.2.1 :POINts?**

MEMORY:TABLE:LOSS:PHASe:POINts?

Returns the number of points in the LOSS:PHASe list of the SElected TABLE. If no LOSS:PHASE values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.18 :NCURve <numeric\_value>{,<numeric\_value>}**

:MEMORY:TABLE:NCURve

NCURve sets the value of concentration in ppm using the newly calculated curve (New CURve) in linearization tables. This command sets the values of NCURve.

**13.11.18.1 :POINts?**

:MEMORY:TABLE:NCURve:POINts?

This command returns the number of points in column NCURve, for the currently selected table.

**13.11.19 :POWer**

MEMory:TABLE:POWer

**13.11.19.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}**

MEMory:TABLE:POWer:MAGNitude

Specifies the POWER[:MAGNitude] points of the TABLE. <numeric\_value>'s may include POWER units as listed in the UNIT subsystem. The default unit is specified by UNIT:POWER. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as POWER[:MAGNitude].

At \*RST, the contents of this list item is device dependent.

**13.11.19.1.1 :POINts?**

MEMory:TABLE:POWer:MAGNitude:POINts?

Returns the number of points in the POWER[:MAGNitude] list of the SElected TABLE. If no POWER[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.20 :RAW <numeric\_value> {,<numeric\_value>}**

:MEMory:TABLE:RAW

RAW is a collection of points in instrument internal units.

For emissions benches, it is the raw gas concentration in bench internal units from a linearization. This command sets the values of RAW.

**13.11.20.1 :POINts?**

:MEMory:TABLE:RAW:POINts?

This command returns the number of points in column RAW, for the currently selected table.

**13.11.21 :RESistance**

MEMory:TABLE:RESistance

**13.11.21.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}**

MEMory:TABLE:RESistance:MAGNitude

Specifies the RESistance[:MAGNitude] points of the TABLE. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as RESistance[:MAGNitude].

At \*RST, the contents of this list item is device dependent.

**13.11.21.1.1 :POINts?**

MEMory:TABLE:RESistance:MAGNitude:POINts?

Returns the number of points in the RESistance[:MAGNitude] list of the SElected TABLE. If no RESistance[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.22 :SELect <name>**

MEMory:TABLE:SELect

Sets or queries the name of the TABLE selected. If the TABLE <name> already exists, then the existing TABLE shall be selected. If the TABLE <name> does not exist, then the new name shall be selected, but no TABLE shall be defined by this selection. The TABLE <name> is character data.

At \*RST the <name> select shall be device dependent.

**13.11.23 :SPEed**

MEMory:TABLE:SPEed

Specifies the SPEed points of the TABLE.

**13.11.23.1 [:MAGNitude] <numeric\_value>{,<numeric\_value>}**

MEMory:TABLE:SPEed:MAGNitude

Specifies the SPEed[:MAGNitude] points of the TABLE. The default units of this command are meters per second.

At \*RST, the contents of this list item is device dependent.

**13.11.23.1.1 :POINts?**

MEMory:TABLE:SPEed:MAGNitude:POINts?

Returns the number of points in the SPEed[:MAGNitude] list of the selected TABLE. If no SPEed[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

13

**13.11.24 :TIME**

MEMory:TABLE:TIME

Specifies the TIME points of the TABLE.

**13.11.24.1 [:MAGNitude] <numeric\_value>{,<numeric\_value>}**

MEMory:TABLE:TIME:MAGNitude

Specifies the TIME[:MAGNitude] points of the TABLE. The default units of this command are seconds.

At \*RST, the contents of this list item is device dependent.

**13.11.24.1.1 :POINts?**

MEMory:TABLE:TIME:MAGNitude:POINts?

Returns the number of points in the TIME[:MAGNitude] list of the selected TABLE. If no TIME[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**13.11.25 :TOLerance <numeric\_value> {,<numeric\_value>}**

:MEMory:TABLE:TOLerance

TOLerance is a series of tolerance values.

For emissions benches, it is the allowable deviation in gas concentration in percentage of full scale (%FS) between expected and current (or new curve) values stored from a linearization.

The bench will set an error bit in the status register if the difference between expected and actual values exceed the tolerance. This command sets the values of TOLerance.

#### 13.11.25.1 :POINts?

:MEMory:TABLE:TOLerance:POINts?

This command returns the number of points in column TOLerance, for the currently selected table.

#### 13.11.26 :ULIMit <numeric\_value>{,<numeric\_value>}

:MEMory:TABLE:ULIMit

ULIMit is the upper limit value. This command sets the values of ULIMit. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as ULIMit.

At \*RST, the contents of this list item is device dependent.

#### 13.11.26.1 :POINts?

:MEMory:TABLE:ULIMit:POINts?

This command returns the number of points in column ULIMit, for the currently selected table. If no ULIMit values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

#### 13.11.27 :VOLTage

MEMory:TABLE:VOLTage

#### 13.11.27.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMory:TABLE:VOLTage:MAGNitude

Specifies the VOLTage[:MAGNitude] points of the TABLE. <numeric\_value>'s may include voltage units as listed in the UNIT subsystem. The default unit is specified by UNIT:VOLTage. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as VOLTage[:MAGNitude].

At \*RST, the contents of this list item is device dependent.

#### 13.11.27.1.1 :POINts?

MEMory:TABLE:VOLTage:MAGNitude:POINts?

Returns the number of points in the VOLTage[:MAGNitude] list of the SElected TABLE. If no VOLTage[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

#### 13.11.27.2 :PHASe <numeric\_value>{,<numeric\_value>}

MEMory:TABLE:VOLTage:PHASe

Specifies the VOLTage:PHASe points of the TABLE. <numeric\_value>'s may include angle units as listed in the UNIT subsystem. The default unit is specified by UNIT:ANGLE. If there are more <numeric\_value>'s than in the associated DEFine statement, execution error -223 (Too much data) will be generated. This item list is included in the table DEFine command as VOLTage:PHASe.

## 1999 SCPI Command Reference

At \*RST, the contents of this list item is device dependent.

### 13.11.27.2.1 :POINts?

:MEMory:TABLE:VOLTage:PHASE:POINts?

Returns the number of points in the VOLTage:PHASE list of the SElected TABLE. If no VOLTage:PHASE values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

### 13.11.28 :WFACtor <numeric\_value> {<numeric\_value>}

:MEMory:TABLE:WFACtor

Values in column WFACtor are weighting factors.

For emissions benches, a value in the WFACtor column is assigned to each cut point in a linearization. If weighting is not desired, all values are set to the same non-zero number. WFACtor is a column in linearization tables. This command sets the values of WFACtor.

### 13.11.28.1 :POINts?

:MEMory:TABLE:WFACtor:POINts?

This command returns the number of points in column WFACtor, for the currently selected table.

## 14 MMEMORY Subsystem

The Mass MEMory subsystem provides mass storage capabilities for instruments. All mass memory device commands are contained in the MMEMORY subsystem. The mass storage may be either internal or external to the instrument. If the mass memory device is external, then the instrument must have controller capability on the mass memory device bus.

The CLOSE, FEED, NAME, and OPEN commands permit the streaming of data from anywhere in the data flow into a file; this is particularly useful for saving HCOPy output.

Mass storage media may be formatted in any one of a number of standard formats.

### Selecting Mass Memory Devices

When an instrument has multiple attached mass storage devices, the desired mass storage unit is selected with the mass storage unit specifier <msus>. The syntax of the <msus> string is instrument specific. Each instrument's documentation shall describe the syntax applicable to that instrument. For example:

```
"C:  
":700,2"  
"fileserv"
```

Note that some file systems do not make use of the <msus> concept.

### File Names

The <file\_name> parameter described in the commands in this subsystem is a string. The contents of the string are dependent on the needs of the format of the mass storage media. In particular, the file name may contain characters for specifying subdirectories (that is, \ for DOS, / for HFS) and the separator for extensions in DOS (that is, period).

14

The following commands are provided for mass memory device operations.

KEYWORD	PARAMETER FORM	NOTES
MMemory		
:CATalog?	[<msus>]	[query only]
:CDIRectory	[<directory_name>]	
:CLOSE		[event; no query] 1993
:COPY	<file_name>,<file_name> <file_name>,<msus>,<file_name>,<msus>	
:DATA	<file_name>,<data>	1995
:DELETE	<file_name>[,<msus>]	[no query]
:FEED	<data_handle>	1993
:INITialize	[<msus>[,,(LIF DOS HFS) [,<numeric_value>]]]	
:LOAD		
:DINTerchange	<label>,<file_name>[,<msus>]	[no query]
:TRACE	<label>,<file_name>[,<msus>]	[no query]
:MACRo	<label>,<file_name>[,<msus>]	[no query]
:STATE	<numeric_value>,<file_name>[,<msus>]	[no query]

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:TABLE	<label>,<file_name>[,<msus>]	[no query] 1994
:TRACe	<label>,<file_name>[,<msus>]	[no query]
:MSIS	[<msus>]	
:MOVE	(<src_file>,<dest_file>)  (<src_file>,<src_msus>,<dest_file>,<dest_msus>)	[no query] 1991 1991
:NAME	<file_name>[,<msus>]	1993
:OPEN		[event; no query] 1993
:PACK	<msus>	[no query] 1991
:STORE		
:DINTerchange	<label>,<file_name>[,<msus>]	[no query]
:TRACe	<label>,<file_name>[,<msus>]	[no query] 1992
:MACRo	<label>,<file_name>[,<msus>]	[no query]
:STATE	<numeric_value>,<file_name>[,<msus>]	[no query]
:TABLE	<label>,<file_name>[,<msus>]	[no query] 1994
:TRACe	<label>,<file_name>[,<msus>]	[no query]

14.1

### **:CATalog? [<msus>]**

MMEMemory:CATalog?

The CATalog command is query-only and returns information on the current contents and state of the mass storage media. Upon a CATalog? query, the instrument shall read the specified mass memory device and returns its directory information in the following format:

*<numeric\_value>,<numeric\_value> {,<file\_entry>}*

The instrument shall return two numeric parameters and as many strings as there are files in the directory list. The first parameter shall indicate the total amount of storage currently used in bytes. The second parameter shall indicate the total amount of storage available, also in bytes. The <file\_entry> is a string. Each <file\_entry> shall indicate the name, type, and size of one file in the directory list:

*<file\_name>,<file\_type>,<file\_size>*

Note that this syntax places some restrictions on the <file\_name> (for example, commas are not allowed.) The <file\_name> is the exact name of a file as it appears in the directory list, including an extension if present. The <file\_size> provides the size of the file in bytes. The <file\_type> is indicated by one of the following:

- ASCii — An ASCii text file
- BINary — A binary file
- STATE — Instrument (setting) state
- TRACe — Trace (display) data
- MACRo — An instrument macro

For media which cannot determine the file type from the directory information (i.e. DOS, HFS) this field shall be left blank.

**14.2 :CDIRectory [<directory\_name>]**

MMEMory:CDIRectory

Changes the default directory for a mass memory file system. The <directory\_name> parameter is a string. The contents of the <directory\_name> parameter are dependent on the file system being accessed. If no parameter is specified, the directory is set to the \*RST value.

At \*RST, this value is set to the root for DOS file systems and to a device-dependent value for other file systems such as HFS.

**14.3 :CLOSE**

MMEMory:CLOSE

The CLOSE event causes the selected file specified in NAME to be closed. An attempt to CLOSE a file that was not open shall cause an error -256, (File name not found) to be generated.

**14.4 :COPY <file\_source>,<file\_destination>**

MMEMory:COPY

The COPY command copies an existing file to a new file.

Two forms of parameters are allowed. The first form has two parameters. In this form, the first parameter specifies the source file, and the second parameter specifies the destination file.

The second form has four parameters. In this form, the first and third parameter specify the file\_names. The second and fourth parameters specify the <msus>. The first pair of parameters specifies the source file. The second pair specifies the destination file. An error is generated if the source file doesn't exist. An error may be generated if the destination file already exists.

This command defines an event and thus has no \*RST state or query form.

**14.5 :DATA <file\_name>,<data>**

MMEMory:DATA

The command form is MMEMory:DATA <file\_name>,<data>. The command loads <data> into the file <file\_name>. <data> is in 488.2 block format. <file\_name> is string data.

The query form is MMEMory:DATA? <file\_name> with the response being the associated <data> in block format.

MMEMory:DATA? response is not affected by the FORMat subsystem.

**14.6 :DELETED <file\_name>[,<msus>]**

MMEMory:DELETED

The DELETED command removes a file from the specified mass storage device. The <file\_name> parameter specifies the file\_name to be removed.

This command defines an event, and it has no \*RST condition or query form.

## 14.7 :FEED &lt;data\_handle&gt;

MMEMory:FEED

Sets or queries the <data\_handle> to be used to feed data into the file specified by NAME. New data arriving from <data\_handle> always overwrites the contents of the file specified by NAME. If the <data\_handle> generates new data and the file is not OPEN an error -256, (File name not found) shall be generated.

At \*RST, the value of FEED is device dependent.

## 14.8 :INITialize [&lt;msus&gt;[,(LIF|DOS|HFS)[,&lt;numeric\_value&gt;]]]

MMEMory:INITialize

The INITialize command is used to initialize the specified mass storage media. If the <msus> is not specified, the default is used. The default is the <msus> selected at \*RST, not the currently selected <msus>. The second parameter specifies the type of media to be formatted. The <numeric\_value> parameter is used to specify the format, interleave, sector sizes, and other media-dependent information. Execution of the command attempts to initialize the media. Previous data on the media, if any, is destroyed.

This command is an event and does not have a query form or a \*RST condition.

## 14.9 :LOAD and :STORE

MMEMory:LOAD and :STORE

The LOAD and STORE subsystems transfer information between the mass memory device and the instrument's memory. The LOAD commands transfer from mass memory device to internal memory, and STORE transfers from internal memory to the mass memory device.

When information is loaded, the instrument uses the file type information to determine whether the data is in internal (BINary) or transportable (ASCii) form.

The <file\_name> parameter is a quoted string and the <label> refers to an internal identifier for the instrument trace or macro.

The LOAD and STORE commands do not have query forms.

## 14.9.1 :DINTerchange &lt;label&gt;,&lt;file\_name&gt;[,&lt;msus&gt;]

MMEMory:LOAD:DINTerchange

Specifies that the device transfer DINTerchange formatted data.

## 14.9.1.1 :TRACe &lt;label&gt;,&lt;file\_name&gt;[,&lt;msus&gt;]

MMEMory:LOAD:DINTerchange:TRACe

The specified trace or array is loaded from or stored to the mass memory device in DINTerchange format.

## 14.9.2 :MACRo &lt;label&gt;,&lt;file\_name&gt;[,&lt;msus&gt;]

MMEMory:LOAD:MACRo

The specified macro is loaded from or stored to the mass memory device.

## 14.9.3 :STATe &lt;numeric\_value&gt;,&lt;file\_name&gt;[,&lt;msus&gt;]

MMEMory:LOAD:STATE

The specified instrument state is stored or loaded. The <numeric\_value> parameter refers to the state number, which corresponds to the numbers found in \*SAV and \*RCL.

**14.9.4 :TABLE <label>,<file\_name>[,<msus>]**

MMEMory:LOAD:TABLE

The MEMory:TABLE specified by <label> is loaded from or stored to the mass memory device.

**14.9.5 :TRACe <label>,<file\_name>[,<msus>]**

MMEMory:LOAD:TRACe

The specified trace or array is loaded from or stored to the mass memory device.

**14.10 :MSIS [<msus>]**

MMEMory:MSIS

The “Mass Storage IS” command selects a default mass storage device which is used by all MMEMory commands except INITialize. If the parameter is omitted, the device-dependent setting for default mass storage device is selected.

At \*RST, the value of this parameter is device-dependent.

**14.11 :MOVE (<src\_file>,<dest\_file>)  
|(<src\_file>,<src\_msus>,<dest\_file>,<dest\_msus>)**

MMEMory:MOVE

The MOVE command moves an existing file to another file name.

Two forms of the parameter are allowed. The first form has two parameters. In this form the first parameter, <src\_file>, specifies the file to be renamed, and the second parameter, <dest\_file>, specifies the new name of the file.

The move operation is performed on the default mass storage device.

The second form has four parameters. The first two parameters, <src\_file> and <src\_msus>, specify the file name and storage device of the source file and the second two parameters, <dest\_file> and <dest\_msus>, specify the new file name and new storage device for the source file.

Some instruments may not support the moving of a file across mass storage devices and thus shall generate error -250, “Mass Storage Error”, when the msus parameters are not the same.

If the <src\_file> does not exist, the instrument shall generate error -256, “File name not found”. If the <dest\_file> already exists, the instrument shall generate error -257, “File name error”.

This command is an event-only command and thus has no \*RST value, and no query form.

**14.12 :NAME <file\_name>[,<msus>]**

MMEMory:NAME

Sets or queries the name of the file specification used by the CLOSe and OPEN commands.

At \*RST, the value of this parameter is device dependent.

## 1999 SCPI Command Reference

### 14.13 :OPEN

MMEMory:OPEN

The OPEN event causes the selected file specified in NAME to be opened. An attempt to OPEN a file that is already opened shall cause an error -257, (File name error) to be generated. If the selected file is non-existent, it is created.

### 14.14 :PACK [<msus>]

MMEMory:PACK

This command causes the mass storage device to be packed, such that unused memory between files is recovered and packed together. If an instrument, which supports file packing (ie: implements the MMEM:PACK command), receives the MMEM:PACK command but the current disk format does not support file packing, the instrument shall generate an error -250 (Mass Storage Error). If the <msus> is not specified the default is used.

This command defines an event and thus has no \*RST state or query form.

## 15 OUTPut Subsystem

The OUTPut subsystem controls the characteristics of the source's output port.

A source which is sourcing impedance (programmable load) is allowed to alias this subsystem as INPut, providing that the OUTPut keyword is also recognized and the INPut subsystem is not used for its intended purpose within the instrument.

KEYWORD	PARAMETER FORM	NOTES
OUTPut INPut		See text
:ATTenuation	<numeric_value>	
:COUPling	AC DC	
:FILTer		
:AUTO	<Boolean>   ONCE	1992
:EXTernal		1992
[:STATe]	<Boolean>	1992
:HPASs		
:FREQuency	<numeric_value>	1991
[:STATe]	<Boolean>	
:TYPE	BESSel   CHEByshev	1992
[:LPASs]		
:FREQuency	<numeric_value>	1991
[:STATe]	<Boolean>	
:TYPE	BESSel   CHEByshev	1992
:IMPedance	<numeric_value>	
:LOW	FLOAT GROund	1991
:POLarity	NORMal   INVerted	1992
:POLarization	<numeric_value>	1994
:ORIZONTAL		1994
:VERTical		1994
:POStion		1994
[:X]		1994
:ANGLE		1994
:DIRection	UP DOWN	1994
[:IMMediate]	<numeric_value>	1994
:LIMit		1994
:HIGH	<numeric_value>	1994
:LOW	<numeric_value>	1994
[:STATe]	<Boolean>	1994
:OFFSet	<numeric_value>	1994
:VELOCITY	<numeric_value>	1994
[:DISTance]		1994
:DIRection	UP DOWN	1994
[:IMMediate]	<numeric_value>	1994
:LIMit		1994
:HIGH	<numeric_value>	1994

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:LOW	<numeric_value>	1994
[:STATe]	<Boolean>	1994
:OFFSet	<numeric_value>	1994
:VELOCITY	<numeric_value>	1994
:Y		1994
:ANGLE		1994
:DIRection	UP DOWN	1994
[:IMMEDIATE]	<numeric_value>	1994
:LIMit		1994
:HIGH	<numeric_value>	1994
:LOW	<numeric_value>	1994
[:STATe]	<Boolean>	1994
:OFFSet	<numeric_value>	1994
:VELOCITY	<numeric_value>	1994
[:DISTANCE]		1994
:DIRection	UP DOWN	1994
[:IMMEDIATE]	<numeric_value>	1994
:LIMit		1994
:HIGH	<numeric_value>	1994
:LOW	<numeric_value>	1994
[:STATe]	<Boolean>	1994
:OFFSet	<numeric_value>	1994
:VELOCITY	<numeric_value>	1994
:Z		1994
:ANGLE		1994
:DIRection	UP DOWN	1994
[:IMMEDIATE]	<numeric_value>	1994
:LIMit		1994
:HIGH	<numeric_value>	1994
:LOW	<numeric_value>	1994
[:STATe]	<Boolean>	1994
:OFFSet	<numeric_value>	1994
:VELOCITY	<numeric_value>	1994
[:DISTANCE]		1994
:DIRection	UP DOWN	1994
[:IMMEDIATE]	<numeric_value>	1994
:LIMit		1994
:HIGH	<numeric_value>	1994
:LOW	<numeric_value>	1994
[:STATe]	<Boolean>	1994
:OFFSet	<numeric_value>	1994
:VELOCITY	<numeric_value>	1994
:PROTECTION		1994
:DELAY	<numeric_value>	1994

KEYWORD	PARAMETER FORM	NOTES
[:STATE]	<Boolean>	
:TRIPped?		[query only]
:CLEar		[no query]
:ROSCillator		1991
[:STATE]	<Boolean>	1991
:TTLTrg<n>		
:ECLTrg<n>		1991
:IMMEDIATE		1991
:LEVel	<Boolean>	1991
:POLarity	NORMal INVerted	1994
:PROTocol	SYNChronous SSYNChronous ASYNChronous	1991
:WIDTh	<numeric_value>	1991
[:STATE]	<Boolean>	
:SOURce	<character data>	
[:STATE]	<Boolean>	
:TYPE	<character data>	1992

**15.1 :ATTenuation <numeric\_value>**

OUTPut:ATTenuation

Sets the output ATTenuation level. The default units are in the current relative amplitude units.

This should only be used where the setting is not reflected within the SOURce subsystem. For example, setting the output attenuation to 10 dB will cause the generated signal to decrease in amplitude by 10 dB. However, the power level setting will be unaffected by changing the output attenuation.

At \*RST, this value is device-dependent.

**15.2 :COUPling AC|DC**

OUTPut:COUPling

Controls whether the signal is AC or DC coupled to the output port.

At \*RST, this value is device-dependent.

**15.3 :FILTer**

OUTPut:FILTter

The FILTer function allows filters to be inserted in the path of the signal.

**15.3.1 :AUTO <Boolean> | ONCE**

OUTPut:FILTer:AUTO

Allows the system to determine the best filter characteristic and cutoff frequency. Explicitly selecting any output filter or output filter characteristic shall set AUTO OFF.

At \*RST, this value is device dependent.

**15.3.2 :EXTernal**

OUTPut:FILTer:EXTernal

Controls an external, user provided, filter.

**15.3.2.1 [:STATe] <Boolean>**

OUTPut:FILTter:EXTernal:STATe

Selects a user provided filter. When STATe is ON the filter is placed in the signal path.

At \*RST, this value is OFF.

**15.3.3 :HPASs**

OUTPut:FILTter:HPASs

Controls the high pass filter.

**15.3.3.1 :FREQuency <numeric\_value>**

OUTPut:FILTter:HPASs:FREQuency

Determines the cutoff frequency of the high pass filter.

At \*RST, this value is device-dependent.

**15.3.3.2 [:STATe] <Boolean>**

OUTPut:FILTter:HPASs:STATe

Turns the high pass filter ON and OFF.

At \*RST, this value is device-dependent.

**15.3.3.3 :TYPE BESSel | CHEByshев**

OUTPut:FILTter:HPASs:TYPE

Determines the characteristic of the high pass filter.

At \*RST, this value is device dependent.

**15.3.4 [:LPASs]**

OUTPut:FILTter:LPASs

Controls the low pass filter.

**15.3.4.1 :FREQuency <numeric\_value>**

OUTPut:FILTter:LPASs:FREQuency

Determines the cutoff frequency of the low pass filter.

At \*RST, this value is device-dependent.

**15.3.4.2 [:STATe] <Boolean>**

OUTPut:FILTter:LPASs:STATe

Turns the low pass filter ON and OFF.

At \*RST, this value is device-dependent.

**15.3.4.3 :TYPE BESSel | CHEByshев**

OUTPut:FILTter:LPASs:TYPE

Determines the characteristic of the low pass filter.

At \*RST, this value is device dependent.

- 15.4 :IMPedance <numeric\_value>**  
 OUTPut:IMPedance  
 OUTPut source impedance for the signal specified in Ohms.  
 At \*RST, this value is device-dependent.
- 15.5 :LOW FLOat|GROund**  
 OUTPut:LOW  
 Connects the low signal terminal to ground or allows it to float.  
 This parameter is set to a device-dependent value at \*RST.
- 15.6 :POLarity NORMAL | INVerted**  
 OUTPut:POLarity  
 Sets or queries the output polarity. NORMAL causes the source block's signal to appear at the output with the same polarity as it was generated. INVerted causes the polarity of the source block's signal to be reversed when it appears at the output terminals.  
 At \*RST, this value is NORMAL.
- 15.7 :POLarization <numeric\_value>**  
 OUTPut:POLarization  
 Sets the polarization of the output. The default units are radian.  
 At \*RST, this value is device-dependent.
- 15.7.1 :HORizontal**  
 OUTPut:POLarization:HORizontal  
 Sets the polarization of the output to zero. This command is an event. Therefore it has no associated query or meaning at \*RST.
- 15.7.2 :VERTical**  
 OUTPut:POLarization:VERTical  
 Sets the polarization of the output to PI/2 radians. This command is an event. Therefore it has no associated query or meaning at \*RST.
- 15.8 :POSIon**  
 OUTPut:POSIon  
 The characteristics of an incoming signal can be affected by the output's spatial position and orientation relative to the origin. The definition of the origin is application specific. This subsystem uses a righthand coordinate system and has only one origin.  
 In this subsystem only the IMMEDIATE commands initiate movement.  
 In this subsystem velocity is always positive.
- 15.8.1 [:X]**  
 OUTPut:POSIon:X  
 Controls the settings concerning the X-axis.

**15.8.1.1 :ANGLE**

OUTPut:POSition:X:ANGLE

Specifies the angle of rotation around the X-axis.

**15.8.1.1.1 :DIRection UP|DOWN**

OUTPut:POSition:X:ANGLE:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

**15.8.1.1.2 [:IMMEDIATE] <numeric\_value>**

OUTPut:POSition:X:ANGLE:IMMEDIATE

This command indicates that the output is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

**15.8.1.1.3 :LIMit**

OUTPut:POSition:X:ANGLE:LIMit

Sets the maximum bounds on the physical POSition.

**15.8.1.1.3.1 :HIGH <numeric\_value>**

OUTPut:POSition:X:ANGLE:LIMit:HIGH

Sets the maximum for the POSition.

At \*RST, this value is set to the clockwise physical limit.

**15.8.1.1.3.2 :LOW <numeric\_value>**

OUTPut:POSition:X:ANGLE:LIMit:LOW

Sets the minimum for the POSition.

At \*RST, this value is set to the counterclockwise physical limit.

**15.8.1.1.3.3 :STATE <Boolean>**

OUTPut:POSition:X:ANGLE:LIMit:STATE

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

**15.8.1.1.4 :OFFSet <numeric\_value>**

OUTPut:POSition:X:ANGLE:OFFSet

This defines a single value that is subtracted from the physical POSition.

At \*RST, this value is device-dependent.

**15.8.1.1.5 :VELOCITY <numeric\_value>**

OUTPut:POSition:X:ANGLE:VELOCITY

Controls the velocity of the angular rotation around the x-axis.

At \*RST, this value is device dependent.

**15.8.1.2 [:DISTance]**

OUTPut:POSition:X:DISTance

Controls the settings in linear direction of the X-axis.

**15.8.1.2.1 :DIRection UP|DOWN**

OUTPut:POSition:X:DISTance:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

**15.8.1.2.2 [:IMMEDIATE] <numeric\_value>**

OUTPut:POSition:X:DISTance:IMMEDIATE

This command indicates that the output is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

**15.8.1.2.3 :LIMit**

OUTPut:POSition:X:DISTance:LIMit

Sets the maximum bounds on the physical POSition.

**15.8.1.2.3.1 :HIGH <numeric\_value>**

OUTPut:POSition:X:DISTance:LIMit:HIGH

Sets the maximum for the POSition.

At \*RST, this value is set to the upper physical limit.

**15.8.1.2.3.2 :LOW <numeric\_value>**

OUTPut:POSition:X:DISTance:LIMit:LOW

Sets the minimum for the POSition.

At \*RST, this value is set to the lower physical limit.

**15.8.1.2.3.3 :STATE <Boolean>**

OUTPut:POSition:X:DISTance:LIMit:STATE

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

**15.8.1.2.4 :OFFSet <numeric\_value>**

OUTPut:POSition:X:DISTance:OFFSet

This defines a single value that is subtracted from the physical POSition.

At \*RST, this value is device-dependent.

**15.8.1.2.5 :VELOCITY <numeric\_value>**

OUTPut:POSition:X:DISTance:VELOCITY

Controls the velocity of the displacement on the x-axis.

At \*RST, this value is device dependent.

**15.8.2 :Y**

OUTPut:POSition:Y

Controls the settings concerning the Y-axis.

**15.8.2.1 :ANGLE**

OUTPut:POSition:Y:ANGLE

Specifies the angle of rotation around the Y-axis.

**15.8.2.1.1 :DIRection UP|DOWN**

OUTPut:POSition:Y:ANGLE:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

**15.8.2.1.2 [:IMMEDIATE] <numeric\_value>**

OUTPut:POSition:Y:ANGLE:IMMEDIATE

This command indicates that the output is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

**15.8.2.1.3 :LIMit**

OUTPut:POSition:Y:ANGLE:LIMit

Sets the maximum bounds on the physical POSition.

**15.8.2.1.3.1 :HIGH <numeric\_value>**

OUTPut:POSition:Y:ANGLE:LIMit:HIGH

Sets the maximum for the POSition.

At \*RST, this value is set to the clockwise physical limit.

**15.8.2.1.3.2 :LOW <numeric\_value>**

OUTPut:POSition:Y:ANGLE:LIMit:LOW

Sets the minimum for the POSition.

At \*RST, this value is set to the counterclockwise physical limit.

**15.8.2.1.3.3 :STATE <Boolean>**

OUTPut:POSition:Y:ANGLE:LIMit:STATE

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

**15.8.2.1.4 :OFFSet <numeric\_value>**

OUTPut:POSition:Y:ANGLE:OFFSet

This defines a single value that is subtracted from the physical POSition.

At \*RST, this value is device-dependent.

**15.8.2.1.5 :VELOCITY <numeric\_value>**

OUTPut:POSition:Y:ANGLE:VELOCITY

Controls the velocity of the angular rotation around the y-axis.

At \*RST, this value is device dependent.

### 15.8.2.2 [:DISTance]

OUTPut:POSIon:Y:DISTance

Controls the settings in linear direction of the Y-axis.

#### 15.8.2.2.1 :DIRection UP|DOWN

OUTPut:POSIon:Y:DISTance:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

#### 15.8.2.2.2 [:IMMEDIATE] <numeric\_value>

OUTPut:POSIon:Y:DISTance:IMMEDIATE

This command indicates that the output is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

#### 15.8.2.2.3 :LIMit

OUTPut:POSIon:Y:DISTance:LIMit

Sets the maximum bounds on the physical POSition.

##### 15.8.2.2.3.1 :HIGH <numeric\_value>

OUTPut:POSIon:Y:DISTance:LIMit:HIGH

Sets the maximum for the POSition.

At \*RST, this value is set to the upper physical limit.

##### 15.8.2.2.3.2 :LOW <numeric\_value>

OUTPut:POSIon:Y:DISTance:LIMit:LOW

Sets the minimum for the POSition.

At \*RST, this value is set to the lower physical limit.

#### 15.8.2.2.3.3 :STATe <Boolean>

OUTPut:POSIon:Y:DISTance:LIMit:STATE

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

#### 15.8.2.2.4 :OFFSet <numeric\_value>

OUTPut:POSIon:Y:DISTance:OFFSet

This defines a single value that is subtracted from the physical POSition.

At \*RST, this value is device-dependent.

#### 15.8.2.2.5 :VELOCITY <numeric\_value>

OUTPut:POSIon:Y:DISTance:VELOCITY

Controls the velocity of the displacement on the y-axis.

At \*RST, this value is device dependent.

**15.8.3 :Z**

OUTPut:POSIon:Z

Controls the settings concerning the Z-axis

**15.8.3.1 :ANGLE**

OUTPut:POSIon:Z:ANGLE

Specifies the angle of rotation around the Z-axis.

**15.8.3.1.1 :DIRection UP|DOWN**

OUTPut:POSIon:Z:ANGLE:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

**15.8.3.1.2 [:IMMEDIATE] <numeric\_value>**

OUTPut:POSIon:Z:ANGLE:IMMEDIATE

This command indicates that the output is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

**15.8.3.1.3 :LIMIT**

OUTPut:POSIon:Z:ANGLE:LIMit

Sets the maximum bounds on the physical POSition.

**15.8.3.1.3.1 :HIGH <numeric\_value>**

OUTPut:POSIon:Z:ANGLE:LIMit:HIGH

Sets the maximum for the POSition.

At \*RST, this value is set to the clockwise physical limit.

**15.8.3.1.3.2 :LOW <numeric\_value>**

OUTPut:POSIon:Z:ANGLE:LIMit:LOW

Sets the minimum for the POSition.

At \*RST, this value is set to the counterclockwise physical limit.

**15.8.3.1.3.3 :STATE <Boolean>**

OUTPut:POSIon:Z:ANGLE:LIMit:STATe

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

**15.8.3.1.4 :OFFSet <numeric\_value>**

OUTPut:POSIon:Z:ANGLE:OFFSet

This defines a single value that is subtracted from the physical POSition.

At \*RST, this value is device-dependent.

**15.8.3.1.5 :VELOCITY <numeric\_value>**

OUTPut:POSIon:Z:ANGLE:VELOCITY

Controls the velocity of the angular rotation around the z-axis.

At \*RST, this value is device dependent.

### 15.8.3.2 [:DISTance]

OUTPut:POSIon:Z:DISTance

Controls the settings in linear direction of the Z-axis.

#### 15.8.3.2.1 :DIRection UP|DOWN

OUTPut:POSIon:Z:DISTance:DIRection

Controls the direction of the movement when started with INITiate.

At \*RST, this is set to UP

#### 15.8.3.2.2 [:IMMEDIATE] <numeric\_value>

OUTPut:POSIon:Z:DISTance:IMMEDIATE

This command indicates that the output is moved to the specified position without waiting for further commands.

\*RST has no effect on the IMMEDIATE value except that movement shall stop.

#### 15.8.3.2.3 :LIMit

OUTPut:POSIon:Z:DISTance:LIMit

Sets the maximum bounds on the physical POSition.

##### 15.8.3.2.3.1 :HIGH <numeric\_value>

OUTPut:POSIon:Z:DISTance:LIMit:HIGH

Sets the maximum for the POSition.

At \*RST, this value is set to the upper physical limit.

##### 15.8.3.2.3.2 :LOW <numeric\_value>

OUTPut:POSIon:Z:DISTance:LIMit:LOW

Sets the minimum for the POSition.

At \*RST, this value is set to the lower physical limit.

#### 15.8.3.2.3.3 :STATe <Boolean>

OUTPut:POSIon:Z:DISTance:LIMit:STATE

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

#### 15.8.3.2.4 :OFFSet <numeric\_value>

OUTPut:POSIon:Z:DISTance:OFFSet

This defines a single value that is subtracted from the physical POSition.

At \*RST, this value is device-dependent.

#### 15.8.3.2.5 :VELocity <numeric\_value>

OUTPut:POSIon:Z:DISTance:VELocity

Controls the velocity of the displacement on the z-axis.

At \*RST, this value is device dependent.

- 15.9 :PROtection**  
OUTPut:PROtection  
Controls the characteristics of the output protection or foldback circuits.
- 15.9.1 :DElay <numeric\_value>**  
OUTPut:PROtection:DElay  
Controls the amount of time the protection trip condition must be true before the subsequent protection is taken (for example, tripping a foldback circuit or reporting an error). The units for the parameter are seconds.  
At \*RST, this value is device-dependent.
- 15.9.2 [:STATe] <Boolean>**  
OUTPut:PROtection:STATe  
Controls whether the output protection circuit is enabled.  
At \*RST, this value is device-dependent.
- 15.9.3 :TRIPped?**  
OUTPut:PROtection:TRIPped?  
This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped. TRIPped only has a query form and thus has no associated \*RST condition.
- 15.9.4 :CLEar**  
OUTPut:PROtection:CLEar  
Causes the protection circuit to be cleared.  
This command is an event and has no associated \*RST condition.
- 15.10 :ROSCillator**  
OUTPut:ROSCillator  
The ROSCillator subtree controls the output on the ROSCillator OUTPut port.
- 15.10.1 [:STATe] <Boolean>**  
OUTPut:ROSCillator:STATe  
The STATe command selects whether or not the device outputs a signal on its ROSCillator port. When ON is selected, the device outputs a signal; when OFF, it does not.  
At \*RST, STATe OFF shall be selected.
- 15.11 :TTLTrg<n>[:ECLTrg<n>]**  
OUTPut:TTLTrg[:ECLTrg]  
The TTLTrg and ECLTrg subtrees control the driving of the VXI backplane TTL and ECL trigger buses. A numeric suffix is required, indicating the particular line in the bus. Valid lines are TTLTrg0 through TTLrg7, and ECLTrg0 through ECLTrg5.
- 15.11.1 :IMMediate**  
OUTPut:TTLTrg[:ECLTrg]:IMMediate  
The IMMediate command causes a pulse to appear on the specified trigger line, regardless of the level.

**15.11.2 :LEVel <Boolean>**

OUTPut:TTLTrg|ECLTrg:LEVel

The LEVel command sets the selected TTLTrg or ECLTrg line to a logical level 0 or 1. For TTL, level 0 is 5 volts and level 1 is 0 volts. For ECL level 0 is -2 volts and level 1 is 0 volts.

At \*RST, LEVel is set to OFF.

**15.11.3 :POLarity NORMAL|INVerted**

OUTPut:TTLTrg|ECLTrg:POLarity

This command sets and queries the polarity of the specified VXI TTL or ECL trigger line.

At \*RST, the value of this parameter is NORMAl.

**15.11.4 :PROTocol SYNChronous|SSYNchronous|ASYNchronous**

OUTPut:TTLTrg|ECLTrg:PROTocol

This command selects the trigger protocol for the VXI TTLTrg or ECLTrg trigger lines. The protocols are specified in the VXIbus System Specification Revision 1.4 under sections B.6.2.3 and B.6.2.4.

- SYNChronous      selects the synchronous trigger protocol.
- SSYNchronous      selects the semi-synchronous trigger protocol.
- ASYNchronous      selects the asynchronous trigger protocol.

At \*RST, the SYNChronous protocol is selected for all trigger lines.

**15.11.5 :WIDTh <numeric\_value>**

OUTPut:TTLTrg|ECLTrg:WIDTh

Selects width of the pulse being generated by the OUTPut:TTLTrg<n>:IMMEDIATE or OUTPut:ECLTrg<n>:IMMEDIATE command. The default units are seconds.

15

The \*RST value is device dependent.

**15.11.6 [:STATe] <Boolean>**

OUTPut:TTLTrg|ECLTrg:STATE

The STATE command selects whether or not the VXI module drives the VXI backplane trigger line. When ON is selected, the module shall drive the trigger line; when OFF, it shall not.

At \*RST, STATe OFF shall be selected.

**15.11.7 :SOURce <character data>**

OUTPut:TTLTrg|ECLTrg:SOURce

Controls which signal from the VXI module shall drive the VXI backplane trigger line. Source names are character data, such as INTERNAL or EXTERNAL.

At \*RST, the value of this parameter is device dependent.

### 15.12 [:STATe] <Boolean>

OUTPut:STATe

The STATe function controls whether the output terminals are open or closed. When the function is OFF, the terminals are at maximum isolation from the signal.

At \*RST, this function shall be set to OFF.

### 15.13 :TYPE <character data>

OUTPut:TYPE

Selects the output characteristic to be used if more than one is available.

The following characteristics are currently defined for instruments:

- BALanced – balanced or differential output.
- UNBalanced – unbalanced with respect to ground.

At \*RST the TYPE is device-dependent.

## 16 PROGram Subsystem

The purpose of the PROGram subsystem is to provide the administrative features needed to generate and control one or more user-programmed tasks resident in an instrument. Two distinct methods of accessing a particular program are provided in the PROGram subsystem. One method employs EXPLicit reference for each command, allowing access to all programs, but requires the program name to be always specified. The other method allows a single program to be SElected at a time, and the related commands access only the currently selected program. If the EXPLicit method is implemented, then the SElected method shall also be implemented.

A \*RST received from a remote controller, via the device interface, shall cause all programs to be stopped.

A \*RST generated by a program running in the device shall cause the same effect as a \*RST received from a remote controller, with the exception that the program that generated the \*RST shall not be forced to stop. It is necessary for a \*RST to behave in this manner so that a program, when run on either a remote controller or the device, shall have the same effect.

KEYWORD	PARAMETER FORM	NOTES
PROGram		
:CATalog?		[query only]
[:SElected]		
:DEFine	<program>	
:DELeTe		[no query]
[:SELECTed]		
:ALL		
:EXECute	<program_command>	[no query]
:MALLOCate	<nbytes> DEFault	
:NAME	<progname>	
:NUMBER	<varname>{,<nvalues>}	
:STATe	(RUN PAUSE STOP CONTinue)	
:STRing	<varname>{,<svalues>}	
:WAIT		1991
:EXPLicit		
:DEFine	<progname>,<program>	
:DEFine?	<progname>	[query only]
:DELeTe	<progname>	
:EXECute	<progname>,<program_command>	[no query]
:MALLOCate	<progname>,(<nbytes> DEFault)	
:NUMBER	<progname>,<varname>{,<nvalues>}	
:STATe	<progname>,(RUN PAUSE STOP CONTinue)	
:STRing	<progname>,<varname>{,<svalues>}	
:WAIT	<progname>	1991

- 16.1 :CATalog?**  
PROGram:CATalog?
- The CATalog query commands lists all the defined programs. Response is a list of comma-separated strings. Each string contains the name of a program. If no programs are currently defined, then response is a null string ( “ ” ).
- 16.2 [:SElected]**  
PROGram:SESelected
- All the commands under this node access only the program that has been selected by the NAME command.
- 16.2.1 :DEFine <program>**  
PROGram:SESelected:DEFine
- The DEFine command is used to create and download programs. The DEFine query is used to upload programs. The program name used for the definition is the currently selected program name.
- In the DEFine command, the selected program name shall represent a unique name. That is, to download and overwrite an existing program with the same program name, the existing program must be DELETED first. If such an attempt is made without deleting first, an “Illegal program name” error (-282) shall be generated.
- The <program> shall be block data. The contents are device-dependent. When overflow occurs a program error (-28X) shall be generated.
- In the DEFine query, the selected program shall be returned. The <program> shall be uploaded as definite length arbitrary block response data.
- 16.2.2 :DElete**  
PROGram:SESelected:DEDelete
- The DEDelete command deletes downloaded programs.
- 16.2.2.1 [:SElected]**  
PROGram:SESelected:DEDelete:SESelected
- This command causes the selected program to be deleted.
- 16.2.2.2 :ALL**  
PROGram:SESelected:DEDelete:ALL
- ALL is used to specify that all programs in the device are to be deleted. If any of the programs are in the RUN state, a “Program currently running” error (-284) shall be generated and no programs shall be deleted.
- 16.2.3 :EXECute <program\_command>**  
PROGram:SESelected:EXECute
- The EXECute command executes the <program\_command> in the selected program environment. <program\_command> is string data representing any legal program command. If the string data is not legal, then a “Program syntax error” (-285) shall be generated. The selected program shall be in either the PAUSED or STOPped state before the EXECute

command shall be allowed. If the program is in the RUN state, a “Program currently running” error (-284) shall be generated.

#### 16.2.4 :MALLocate <nbytes>|DEFault

PROGram:SELected:MALLocate

The Memory ALLocate command is used to reserve memory space in the device for use by the selected program. If DEFault is specified, then the device shall calculate the amount of memory needed. <nbytes> is numerical data which specifies the required memory in bytes. If the required memory is greater than the space available, then a parameter error “Data out of range” (-222) shall be generated.

\*RST has no effect on the value of MALLocate.

#### 16.2.5 :NAME <progname>

PROGram:SELected:NAME

The NAME command defines the name of the program to be selected. If the program name <progname> already exists, then that existing program shall be selected. If the program name does not exist, then the new name shall be selected, but no program shall be defined by this selection. <progname> is character data.

\*RST causes the selected name to become PROG.

#### 16.2.6 :NUMB<sub>E</sub>r <varname>{,<nvalues>}

PROGram:SELected:NUMB<sub>E</sub>r

The NUMB<sub>E</sub>r command is used to set and query the contents of numeric program variables and arrays in the currently selected program. The currently selected program must be a DEFined program, otherwise an “Illegal program name” error (-282) shall be generated. The variable specified in <varname> shall be the name of an existing variable in the selected program, otherwise an “Illegal variable name” error (-283) shall be generated. <varname> can be either character data or string data. <nvalues> is a list of comma separated <numeric\_values> which are used to set <varname>. If the specified variable cannot hold all of the specified <numeric\_values> then a “Parameter not allowed” error ( -108 ) shall be generated.

The query form NUMB<sub>E</sub>r? <varname> returns the contents of the variable as a comma separated list

#### 16.2.7 :STATe RUN|PAUSe|STOP|CONTinue

PROGram:SELected:STATe

The STATe command is used to either set the state or query the state of a selected program. The following matrix defines the effect of setting the STATe to the desired value from each of the possible current states. In certain cases a parameter error “Settings conflict” (-221) shall be generated.

The states are described as follows:

**RUNNING:** The program is currently executing

**PAUSED:** The program has reached a break in execution but can be continued.

**STOPPED:** Execution has been terminated.

state requested	current state		
	RUNNING	PAUSED	STOPPED
RUN	error (-221)	RUNNING	RUNNING
CONT	error (-221)	RUNNING	error (-221)
PAUSE	PAUSED	PAUSED	STOPPED

**16.2.8 :STRing <varname>{,<svalues>}**

PROGram:SELected:STRing

The STRing command is used to set and query the contents of string program variables and arrays in the currently selected program. The currently selected program must be a DEFined program, otherwise an “Illegal program name” error (-282) shall be generated. The variable specified in <varname> shall be the name of an existing variable in the selected program, otherwise an “Illegal variable name” error (-283) shall be generated. <varname> can be either character data or string data. <svalues> is a list of comma-separated strings which are used to set <varname>. If the specified variable cannot hold all of the specified string values then a “Parameter not allowed” error (-108 ) shall be generated. If a string value is too long then it will be truncated when stored in the programs variable.

The query form STRing? <varname> returns the contents of the variable as a comma separated list

**16.2.9 :WAIT**

PROGram:SELected:WAIT

The WAIT command and its query form shall cause no further commands or queries to be executed, until the selected program exits from the RUN state, that is either STOPped or PAUSed. For the query, a 1 is returned in NR1 format at this time, when the program is either STOPped or PAUSed.

**16.3 :EXPLicit**

PROGram:EXPLicit

All the commands under the EXPLicit node directly reference the desired program by name, thus allowing access to a program without having to change the selected program. The <progname> parameter is required for these commands.

**16.3.1 :DEFIne <progname>,<program>**

PROGram:EXPLicit:DEFIne

The DEFIne command is used to create and download programs. The DEFIne query is used to upload programs.

In the DEFIne command, the specified program name shall represent a unique name. That is, to download and overwrite an existing program with the same program name, the existing program must be DELETED first. If such an attempt is made, without deleting first, an “Illegal program name” error (-282) shall be generated.

The <program> shall be block data. The contents are device-dependent. When overflow occurs a program error (-28X) shall be generated.

In the DEFIne query, the specified program name shall be the name of an existing program; otherwise an “Illegal program name” error (-282) shall be generated. The <program> shall be uploaded as definite length arbitrary block response data.

#### 16.3.2 :DELetE <progname>

PROGram:EXPLicit:DELetE

The delete command deletes the specified downloaded program.

#### 16.3.3 :EXECute <progname>,<program\_command>

PROGram:EXPLicit:EXECute

The EXECute command executes the <program\_command> in the specified program environment. <program\_command> is string data representing any legal program command. If the string data is not legal, then a “Program syntax error” (-285) shall be generated. The specified program shall be in either the PAUSed or STOPped state before the EXECute command shall be allowed. If the program is in the RUN state, a “Program currently running” error (-284) shall be generated.

#### 16.3.4 :MALLocate <progname>,(<nbytes>|DEFault)

PROGram:EXPLicit:MALLocate

The Memory ALLocate command is used to reserve memory space in the device for use by the specified program. The specified program shall be the name of an existing program, otherwise an “Illegal program name” error (-282) shall be generated. If DEFault is specified, then the device shall calculate the amount of memory needed. <nbytes> is numerical data which specifies the required memory in bytes. If the required memory is greater than the space available, then a parameter error “Data out of range” (-222) shall be generated.

\*RST has no effect on the value of MALLocate.

#### 16.3.5 :NUMBer <progname>,<varname>{,<nvalues>}

PROGram:EXPLicit:NUMBer

The NUMBER command is used to set and query the contents of numeric program variables and arrays in the specified program. The specified program must be a DEFIned program, otherwise an “Illegal program name” error (-282) shall be generated. The variable specified in <varname> shall be the name of an existing variable in the specified program, otherwise an “Illegal variable name” error (-283) shall be generated. <varname> can be either character data or string data. <nvalues> is a list of comma-separated <numeric\_values> which are used to set <varname>. If the specified variable cannot hold all of the specified <numeric\_values>, then a “Parameter not allowed” error (-108) shall be generated.

The query form NUMBER? <varname> returns the contents of the variable as a comma separated list

#### 16.3.6 :STATe <progname>,(<RUN|PAUSE|STOP|CONTinue>)

PROGram:EXPLicit:STATe

The STATe command is used to either set the state or query the state of the specified program. The matrix below defines the effect of setting the STATe to the desired value from

each of the possible current states. In certain cases a parameter error “Settings conflict” (-221) shall be generated.

desired state	current state		
	RUNNING	PAUSED	STOPPED
RUN	error (-221)	RUNNING	RUNNING
CONT	error (-221)	RUNNING	error (-221)
PAUSE	PAUSED	PAUSED	STOPPED
STOP	STOPPED	STOPPED	STOPPED

The states are described as follows:

**RUNNING:** The program is currently executing

**PAUSED:** The program has reached a break in execution but can be continued.

**STOPPED:** Execution has been terminated.

### 16.3.7 :STRing <progname>,<varname>{,<svalues>}

PROGram:EXPLicit:STRing

The STRing command is used to set and query the contents of string program variables and arrays in the specified program. The specified program must be a DEFined program, otherwise an “Illegal program name” error (-282) shall be generated. The variable specified in <varname> shall be the name of an existing variable in the specified program, otherwise an “Illegal variable name” error (-283) shall be generated. <varname> can be either character data or string data. <svalues> is a list of comma-separated strings which are used to set <varname>. If the specified variable cannot hold all of the specified string values, then a “Parameter not allowed” error (-108) shall be generated. If a string value is too long then it will be truncated when stored in the programs variable.

The query form STRing? <varname> returns the contents of the variable as a comma separated list

### 16.3.8 :WAIT <progname>

PROGram:EXPLicit:WAIT

The WAIT command and its query form shall cause no further commands or queries to be executed, until the defined program exits from the RUN state, that is either STOPped or PAUSed. For the query, a 1 is returned in NR1 format at this time, when the program is either STOPped or PAUSed.

## 17 ROUTe Subsystem

In the SCPI instrument model, the block where user accessibility to the actual signals occurs is called “signal routing.” In some instruments this function may be trivial and thus commands for this block would not be required. In such cases the ports associated with the INPut or OUTput blocks are available directly to the user, typically on the instrument’s front panel. The ROUTe commands apply equally to instruments whose primary purpose is to provide signal routing and to instruments that provide some routing capability “in front of” the INPut and/or OUTput blocks.

The ROUTe node may be optional in a particular instrument. This capability is intended for instruments whose primary function is signal routing. The ROUTe node cannot be optional if either SENSe or SOURce is optional, since only one node at a given level may be optional.

KEYWORD	PARAMETER FORM	NOTES
:ROUTe		
:CLOSe	<channel_list>	1992
:.STATe?		[query only]
:MODule		
:CATalog?		[query only]
:DEFine	<module_name>,<module_address>	
:DEFine?	<module_name>	
:DELetE		
:ALL		[event; no query]
[:NAME]	<module_name>	[event; no query]
:OPEN	<channel_list>	1992
:ALL		[event; no query] 1993
:PATH		
:CATalog?		[query only]
:DEFine	<path_name>,<channel_list>	
:DEFine?	<path_name>	
:DELetE		
:ALL		[event; no query]
[:NAME]	<path_name>	[event; no query]
:SAMPLE		1999
:CATalog?		1999
[:OPEN]	BAG DILute PRE POST MID  CEFFiciency NONE ZERO SPAN  VERify MANifold	[query only] 1999
:SCAN	<channel_list>	1992,4
:TERMinals	FRONt REAR BOTH NONE	1992

### 17.1 :CLOSe <channel\_list>

ROUTe:CLOSE

The CLOSe command allows specific individual channels to be closed or queried. If all the specified channels cannot be closed, an execution error is reported.

For example, suppose a module contains a coaxial multiplexor, and only one channel on a bank is able to be closed at a time. If a command tries to close two channels on the same bank, the instrument should report an execution error.

CLOSE may be an overlapped command. The \*OPC mechanism may be used to determine when switches are actually closed or opened. The sequence in which channels are closed is not guaranteed.

The CLOSE? query allows the condition of individual switches to be queried. The instrument returns a 1 or 0 for each channel in the list, in the same order that the list is specified. A response of 1 means the channel is closed and a 0 means the channel is open.

At \*RST, this value is device-dependent.

### 17.1.1 :STATE?

ROUTe:CLOSE:STATE?

The ROUTe:CLOSE:STATE? query, which has no parameters, returns an IEEE 488.2 definite length block which contains a <channel\_list> of all the closed switches in the entire instrument.

### 17.2 :MODULE

ROUTe:MODULE

The MODULE subsystem controls the assignment of <module\_name>.

#### 17.2.1 :CATalog?

ROUTe:MODULE:CATalog?

The ROUTe:MODULE:CATalog? query returns a list of all currently defined <module\_names>.

Response is a list of comma-separated strings. Each string contains a <module\_name>. If no <module\_name> elements are currently defined, then response is a null string ("").

#### 17.2.2 [:DEFine] <module\_name>,<module\_address>

ROUTe:MODULE:DEFine

Name a module <module\_address> as <module\_name>. This command assigns a user-defined name to a number representing a card address in hardware-dependent form (for example, a VME address).

The query form of this command, “ROUTe:MODULE:DEFine? <module\_name>” returns an <NR1> response showing what <module\_address> is bound to <module\_name>.

#### 17.2.3 :DElete

ROUTe:MODULE:DElete

##### 17.2.3.1 :ALL

ROUTe:MODULE:DElete:ALL

The ROUTe:MODULE:DElete:ALL command deletes all module name bindings, and frees all names created by the MODULE command.

**17.2.3.2 [:NAME] <module\_name>**

ROUTe:MODule:DElete:NAME

The ROUTe:MODule:DElete[:NAME] command removes a module name created by a MODule command.

**17.3 :OPEN <channel\_list>**

ROUTe:OPEN

The OPEN command allows specific channels to be opened or queried.

OPEN may be an overlapped command. The \*OPC mechanism may be used to determine when switches are actually closed or opened. The sequence in which channels are opened is not guaranteed.

The query OPEN? allows the condition of switches to be queried. The instrument returns a 1 or 0 for each channel in the list, in the same order that the list is specified. A response of 0 means the channel is closed and a 1 means the channel is open.

At \*RST, this value is device-dependent.

**17.3.1 :ALL**

ROUTe:OPEN:ALL

The OPEN:ALL command opens all the channels.

OPEN:ALL may be an overlapped command. The \*OPC mechanism may be used to determine when switches are actually closed or opened. The sequence in which channels are opened is not guaranteed.

**17.4 :PATH**

ROUTe:PATH

The PATH subsystem controls the binding of <channel\_list> elements.

**17.4.1 :CATalog?**

ROUTe:PATH:CATalog?

The ROUTe:PATH:CATalog? query returns a list of all currently defined <path\_names>.

Response is a list of comma-separated strings. Each string contains a <path\_name>. If no <path\_name> elements are currently defined, then response is a null string ("").

**17.4.2 [:DEFine] <path\_name>,<channel\_list>**

ROUTe:PATH:DEFine

The ROUTe:PATH:DEFine command assigns <path\_name> as a user-specified way of referring to <channel\_list>. This command allows the user to define a list of relays that can be opened and closed with the use of a <path\_name>.

The query form of this command, “ROUTe:PATH:DEFine? <path\_name>” returns a block containing the <channel\_list> bound to <path\_name>.

**17.4.3 :DELete**

ROUTe:PATH:DELete

**17.4.3.1 :ALL**

ROUTe:PATH:DELete:ALL

The ROUTe:PATH:DELete:ALL command deletes all path name bindings, and frees all names created by the PATH command.

ROUTe:SAMPLE

**17.4.3.2 [:NAME] <path\_name>**

ROUTe:PATH:DELete:NAME

The ROUTe:PATH:DELete[:NAME] command removes a path name created by a PATH command.

**17.5 :SAMPLE**

ROUTe:SAMPLE

This subsystem sets up the route for the sample gas to flow.

**17.5.1 :CATalog?**

:ROUTe:SAMPLE:CATalog?

Returns a comma-separated list of strings containing available sample points used for :ROUTe:SAMPLE:OPEN.

**17.5.2 [:OPEN]BAG|DILute|PRE|POST|MID|CEFFiciency|NONE|ZERO|SPAN  
|VERify|MANifold**

:ROUTe:SAMPLE[:OPEN]

This command causes flow from the specified sample point to the selected instrument(s). An execution error -200 shall occur if the <sample\_point> references a configuration that cannot be achieved by the device. This command can also query the currently selected <sample\_point>. The query response is a string. Each string contains a <sample\_point>. If a <sample\_point> is not currently selected, then the response is NONE.

At \*RST the selected path goes to the idle state (NONE).

For emissions benches, the :ROUTe:SAMPLE command is used to route gases from the source to the measurement instrument. This command causes gases to flow to the instruments from the specified sample point. No measurement sequence is executed, however subsequent SENSe:DATA? queries will reflect the current values of the gases flowing through the analyzers. Gas routing may be implied during predefined procedures. On \*RST the selected path goes to the idle state (NONE).

When the following <sample\_point>s are implemented they shall conform to the following naming conventions:

- **BAG** — Causes gases from the specified bag sampling system to flow through the instruments. (Note: an appropriate CVS command is needed to complete this function.)
- **DILute** — Gases from the Sampler after dilution with ambient air, for example, the tunnel of a diesel sampling system

- **PRE** — Gasses from between the engine and the catalytic converter.
- **POST** — Gasses from the vehicle exhaust after the catalyst but before dilution in the sampling system.
- **MID** — Gases from between the oxidizing and reducing catalysts.
- **CEFFiciency** — Directs gases from each of the necessary sample points to each of the necessary sample manifolds in order to perform a catalyst efficiency test.
- **NONE** — Places the flow of gases in an idle state.
- **ZERO** — Directs zero gases to the selected instruments.
- **SPAN** — Directs span gases to the selected instruments for the selected ranges.
- **VERify** — Flows verification gases to all selected instruments
- **MANifold** — Enables Manifold as locally configured at the bench.

Examples:

```
:INSTRument:DEFIne:GROup MyGroup, "CONC1", "CONC2", "CONC3"
:INSTRument:SElect MyGroup
:ROUTE:SAMPle ZERO
```

Routes zero gas to all analyzers defined by the group “MyGroup”.

```
:INSTRument:DEFIne:GROup MyGroup, "CONC1", "CONC3", "CONC4"
:INSTRument:SElect MyGroup
:ROUTE:SAMPle NONE
```

Discontinues any flows in progress and places the flow of gases into an idle state, for the group of instruments defined by “MyGroup”.

```
:INSTRument:DEFIne:GROup MyGroup, "CONC2", "CONC3"
:INSTRument:SElect MYGROUP
:ROUTE:SAMPle MANifold1
```

Flows engine exhaust from manifold setting 1 for the group of instruments defined by “MyGroup”.

```
:INSTRument:DEFIne:GROup MyGroup, "CONC1", "CONC2", "CONC3"
:INSTRument:SElect MyGroup
:ROUTE:SAMPle?
```

Returns: MANifold1

### 17.6 :SCAN <channel\_list>

ROUTe:SCAN

The SCAN command specifies or queries a list of channels for the instrument to sequence through. The instrument will automatically step through the list of channels provided when the trigger subsystem device action occurs. The device action is to open the channel previously closed, then to close the next channel in the list. Whether this action is break-before-make is device-dependent.

The INITiate[:IMMEDIATE] command must be sent in order to initiate scanning.

Execution of the SCAN command, and the resulting scan operation, may or may not open any channels which are already closed.

If the <channel\_list> parameter contains a <channel\_range>, the instrument will expand the range into its individual channels, and scan each channel individually. For example, (@1,3:5,9) will be scanned identically to (@1,3,4,5,9).

The query, which has no parameter, returns the scan list. The format of the response is an IEEE 488.2 definite length block which contains a <channel\_list>. The instrument may return the scan list exactly as it was received, or it may return an equivalent list with paths or ranges expanded.

Note: The query form was added in 1994.

### 17.7 TERMinals FRONT|REAR|BOTH|NONE

ROUTe:TERMinals

FRONt opens the rear terminals and closes the front terminals (break before make). REAR opens the front terminals and closes the rear terminals (also break before make). BOTH closes both the front and rear terminals in a device-dependent order. NONE opens both the front and rear terminals in a device-dependent order.

At \*RST, TERM is set to FRONt.

## 18 SENSe Subsystem

The SENSe setup commands are divided into several sections. Each section or subsystem deals with controls that directly affect device-specific settings of the device and not those related to the signal-oriented characteristics. These commands are referenced through the SENSe node.

The SENSe node may be optional in a particular device. The reason that the SENSe is optional is to allow devices which are primarily sensors to accept shorter commands. The SENSe node cannot be optional if either SOURce or ROUTe is optional, since only one node at a given level may be optional. For example, a typical counter may elect to make the SENSe node optional, since the most frequently used commands would be under the SENSe subsystem. If the counter also contained either source or routing functionality, these would be controlled through their respective nodes; however, the SOURCE or ROUTE nodes would be required. An optional node, such as SENSe, implies that the device shall accept and process commands with or without the optional node and have the same result. That is, the device is required to accept the optional node, if sent, without error.

In some instances, a sensor may contain subservient source or sensor functions. In such cases, the additional subservient functionality shall be placed as SENSe or SOURCE subnodes under the SENSe subsystem. Further, no optional nodes (SENSe, SOURCE or ROUTE) are permitted if such subservient functionality exists in a device. Otherwise conflicts in interpreting commands would occur, for example, between SENSe and [SOURCE:]SENSe if the SOURCE were allowed to be optional.

The FREQuency subsystem contains several sets of commands which have complex couplings. This includes the swept commands START, STOP, CENTER, and SPAN as one set. Sending any one of a set singly will affect two others of the set. However, if two are sent in a single message unit, then these two should be set to the values specified and the other two changed. This is in accordance with the style guidelines and IEEE 488.2. If any command in the set is implemented then all the commands in the set shall be implemented.

If the requested setting is unachievable given the other instrument settings, then an error must be generated (-221 “Settings conflict”). The device may resolve the conflict in a device-dependent manner (for example, change START, STOP, CENTER, and/or SPAN) to resolve the error. Note that when more than one of the four sweep settings are received in the same program message unit, the sweep will be determined by the last two received.

The coupled commands may define commands as subnodes which alter these couplings. The command description will define how the couplings are altered. However, any command which alters couplings must default to the couplings described in this section as the \*RST couplings.

When CENTER or SPAN is changed the following couplings apply:

SPAN	=	SPAN
CENTER	=	CENTER
STOP	=	CENTER + (SPAN/2)
START	=	CENTER - (SPAN/2)

## 1999 SCPI Command Reference

When START or STOP changed the following couplings apply:

$$\begin{array}{lll} \text{STARt} & = & \text{STARt} \\ & & \text{CENTer} = (\text{STARt} + \text{STOP})/2 \\ \text{STOP} & = & \text{STOP} \\ & & \text{SPAN} = \text{STOP} - \text{STARt} \end{array}$$

The CURRent, VOLTage and POWER subsystems contain several commands which have complex couplings. These commands exist under RANGE: they are UPPer, LOWER, OFFset, and PTPeak. UPPer and LOWER are considered as one set, with OFFSet and PTPeak as another set. Sending any one of a set singly shall affect the two values of those commands in the other set. However, if two are sent in a single message unit, then these two should be set to the values specified and the unspecified two values shall be changed. This is in accordance with the style guidelines and IEEE 488.2. If either OFFSet or PTPeak is implemented, then all four commands, UPPer, LOWER, OFFSet, and PTPeak shall also be implemented.

When UPPer or LOWER is changed the following couplings apply:

$$\begin{array}{lll} \text{UPPer} & = & \text{UPPer} \\ & & \text{OFFSet} = (\text{UPPer} + \text{LOWER})/2 \\ \text{LOWEr} & = & \text{LOWEr} \\ & & \text{PTPeak} = \text{UPPer} - \text{LOWER} \end{array}$$

When OFFSet or PTPeak is changed the following couplings apply:

$$\begin{array}{lll} \text{OFFSet} & = & \text{OFFSet} \\ & & \text{UPPer} = \text{OFFSet} + (\text{PTPeak}/2) \\ \text{PTPeak} & = & \text{PTPeak} \\ & & \text{LOWEr} = \text{OFFSet} - (\text{PTPeak}/2) \end{array}$$

The SENSe subsystem for the dynamometer contains the commands needed to sense real-time measurements taken from the dynamometer system. This includes force, speed, frictional losses, acceleration rate, distance, and simulation error. These commands are used in conjunction with the TRIGer Subsystem for collecting data in blocks. See SCPI 1995 Volume 2 Command Reference chapter 18 section 11 for further detail on the DATA and FUNCtion subsystems.

## 1999 SCPI Command Reference

### First Level SENSe Keywords

[SENSe]

:AM	1991
:AVERage	1992
:BANDwidth :BWIDth	
:CONCcentration	1999
:CONDITION	
:CORRection	
:CURRent	
:DATA?	1999
:DETector	
:DISTance	1999
:FILTer	1991
:FM	1991
:FREQuency	
:FUNCTION	
:LIST	
:MIXer	
:PM	1991
:POWER	
:RESistance :FRESistance	
:ROSCillator	
:SMOOthing	
:SSB	
:STABilize	1999
:SWEep	
:VOLTage	
:WINDow	

**18.1 AM Subsystem**

SENSe:AM

The Amplitude Modulation subsystem is used to set up the modulation controls and parameters associated with sensing and demodulating amplitude modulated signals.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:AM		1991
[:DEPTh]		1991
:RANGE		1991
:AUTO	<Boolean> ONCE	1991
[:UPPer]	<numeric_value>	1991
:LOWER	<numeric_value>	1991
:TYPE	LINear LOGarithmic	1994

**18.1.1 [:DEPTh]**

SENSe:AM:DEPTh

This node is used to collect together the depth parameters.

**18.1.1.1 :RANGE**

SENSe:AM:DEPTh:RANGE

Sets the range for the sensor function.

**18.1.1.1.1 :AUTO <Boolean> | ONCE**

SENSe:AM:DEPTh:RANGE:AUTO

Sets the RANGE to the value determined to give the most dynamic range without overloading. The actual algorithm is device-dependent. Selecting AUTO ONCE will have the effect of setting AUTO to ON and then OFF.

At \*RST, this value is set to ON.

**18.1.1.1.2 [:UPPer] <numeric\_value>**

SENSe:AM:DEPTh:RANGE:UPPer

Specifies the maximum signal level expected for the sensor input.

At \*RST, this value is device-dependent.

**18.1.1.1.3 :LOWER <numeric\_value>**

SENSe:AM:DEPTh:RANGE:LOWER

Specifies the smallest signal level expected for the sensor input. This allows setting for the best dynamic range. This function is coupled to UPPer.

At \*RST, this value is device-dependent.

**18.1.2 :TYPE LINear|LOGarithmic**

SENSe:AM:TYPE

SENSe:AM:TYPE sets or queries the type of amplitude demodulation technique the demodulator uses. LINear means the demodulated signal is derived directly from the amplitude of the signal in a linear way. LOGarithmic means the demodulated signal is derived from the amplitude of the signal in a logarithmic way. The amplitude of the demodulated signal is proportional to the log-value of the amplitude of the signal.

## **1999 SCPI Command Reference**

At \*RST, this value is set to LINear.

## 18.2 AVERage Subsystem

SENSe:AVERage

The purpose of this subsystem is to control averaging used to improve the measurement. For example to improve the accuracy or to acquire maximum/minimum measurements.

Analytical post-acquisition statistical functions are controlled within the CALCulate block. The AVERage subsystem combines successive measurements to produce a new composite result. The new result has the same number of points and control axis as the original measurement.

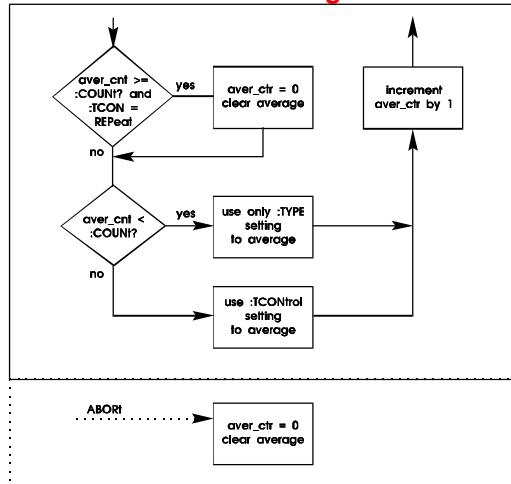
A :COUNt command is provided to specify the number of measurements to combine. When :COUNT averages are reached, the TerminalCONtrol switch specifies the operation of the average subsystem. The TerminalCONtrol switch can be configured to continue replacing old values with new values (MOVing), to exponentially weight new measurements (EXPonential), to automatically repeat (REPeat), or to continue adding new measurements in the NORMal fashon, as specified by :TYPE.

The average operation differs from CALC:SMOOthing in the fact that AVERage combines the jth point of each measurement with the jth point of preceeding measurements, whereas in SMOOthing the jth point is processed with adjacent points (...., j-2, j-1, j, j+1, j+2, ...) of the current measurement set.

Note: The notation  $X_n^*$  indicates the complex conjugate of a number. So:

$$XX^* = \text{MAG}^2(X) = |X|^2$$

### 18.2.1 A typical device action for SENSe:AVERAGE



KEYWORD	PARAMETER FORM	NOTES
:AVERage		1992
:COUNT	<numeric_value>	1992
:AUTO	<Boolean>   ONCE	1992
[:STATe]	<Boolean>	1992
:TCONtrol	EXPonential   MOVing   NORMal   REPeat	1992
:TYPE	COMplex   ENVelope   MAXimum   MINimum   RMS   SCALar	1992

### 18.2.2 :COUNT <numeric\_value>

SENSe:AVERage:COUNT

Specifies the number of measurements to combine using the :TYPE setting. After :COUNT measurements have been averaged, the operation of the AVERage subsystem is controlled by the setting of the :TCONtrol node.

The count for array based results is independent of the number of points in the result array. For example, an instrument which normally provides a measurement result which is a 401 point array would specify an average count of two to average two results, not 802.

When averaging is ON, some devices may automatically set :COUNT values in the TRIGger subsystem based on the AVER:COUNT value, such that the TRIGger subsystem provides enough triggers for the average.

At \*RST, this value is device dependent.

### 18.2.2.1 :AUTO <Boolean> | ONCE

SENSe:AVERage:COUNT:AUTO

AUTO ON causes the device to select a value for :COUNT which is appropriate for the current measurement. Selecting AUTO ONCE will have the effect of setting AUTO ON and then OFF.

At \*RST, this value is set to OFF.

### 18.2.3 [:STATE] <Boolean>

SENSe:AVERage:STATE

The STATe command is used to turn averaging ON and OFF.

At \*RST, this value is set to OFF.

### 18.2.4 :TCONtrol EXPonential | MOVing | NORMal | REPeat

SENSe:AVERage:TCONtrol

TerminalCONtrol specifies the action of the AVERage subsystem when more than AVERage:COUNT measurement results are generated.

The parameter has the following meanings:

EXPonential Continue the average with an exponential weighting applied to old values.  
The additional averages are weighted using:

## 1999 SCPI Command Reference

For :TYPE SCALar when X<sub>n</sub> is real and :TYPE COMPlex when X<sub>n</sub> is either real or complex

$$AVG_n = \frac{1}{k} X_n + \frac{k-1}{k} AVG_{n-1}$$

For :TYPE SCALar when X<sub>n</sub> is complex

$$AVG_n = \frac{1}{k} MAG(X_n) + \frac{k-1}{k} AVG_{n-1}$$

For :TYPE RMS when X<sub>n</sub> is real

$$AVG_n = \sqrt{\frac{1}{k} X_n^2 + \frac{k-1}{k} AVG_{n-1}^2}$$

For :TYPE RMS when X<sub>n</sub> is complex

$$AVG_n = \sqrt{\frac{1}{k} X_n X_n^* + \frac{k-1}{k} AVG_{n-1}^2}$$

The exponential factor k is equal to the AVERage:COUNt value. Some instruments may use an approximation to true exponential.

Operation with EXPonential :TCON is undefined with a :TYPE setting of MINimum, MAXimum, or ENVelope.

MOVing As new data is added the oldest data is discarded.

REPeat Clear average data and counter and restart the average process.

NORMal Additional averages continue to be accumulated according to the :TYPE selection.

At \*RST, this value is device dependent.

### 18.2.5 :TYPE COMPlex | ENVelope | MAXimum | MINimum | RMS | SCALar SENSe:AVERage:TYPE

This command selects the type of averaging, as follows:

COMPlex The points in the summation are treated as real/imaginary pairs.

$$AVG(n) = \frac{1}{n} \sum_{i=1}^n X_i$$

ENVelope Both the MIN and MAX values are retained.

If X<sub>n</sub> is real

MAXimum  $AVG(n) = MAX(X_1 \dots X_n)$

MINimum  $AVG(n) = MIN(X_1 \dots X_n)$

## 1999 SCPI Command Reference

RMS

$$AVG(n) = \sqrt{\frac{1}{n} \sum_{i=1}^n X_i^2}$$

SCALar      The scalar magnitude of each point is averaged.

$$AVG(n) = \frac{1}{n} \sum_{i=1}^n X_i$$

If  $X_n$  is complex

MAXimum      AVG(n) = MAX( MAG(X<sub>1</sub>)... MAG(X<sub>n</sub>) )

MINimum      AVG(n) = MIN( MAG(X<sub>1</sub>)...MAG(X<sub>n</sub>) )

RMS

$$AVG(n) = \sqrt{\frac{1}{n} \sum_{i=1}^n X_i X_i^*}$$

SCALar      The scalar magnitude of each point is averaged.

$$AVG(n) = \frac{1}{n} \sum_{i=1}^n |X_i|$$

At \*RST, this value is device dependent.

### 18.3 BANDwidth|BWIDth Subsystem

SENSe:BANDwidth

This subsystem controls the characteristics of the filter bandwidth of the instrument. Instruments must accept both forms, if they accept either.

KEYWORD	PARAMETER FORM	NOTES
:BANDwidth :BWIDth		
[:RESolution]	<numeric_value>	
:AUTO	<Boolean> ONCE	
:RATio	<numeric_value>	
:TRACK	<Boolean>	
:VIDeo	<numeric_value>	
:AUTO	<Boolean> ONCE	
:RATio	<numeric_value>	

#### 18.3.1 [:RESolution] <numeric\_value>

SENSe:BANDwidth:RESolution

Controls the resolution bandwidth of the instrument. Resolution Bandwidth is defined as the bandwidth in Hz of the predetection bandpass filtering. This command has units of Hz.

At \*RST, the value of this function is device-dependent.

##### 18.3.1.1 :AUTO <Boolean>|ONCE

SENSe:BANDwidth:RESolution:AUTO

When enabled, the value of resolution bandwidth is coupled to other parameters of the measurement. The exact coupling equation is machine-dependent.

At \*RST, this function is set to ON.

##### 18.3.1.2 :RATio <numeric\_value>

SENSe:BANDwidth:RESolution:RATio

Controls the ratio of resolution bandwidth to span when the two are coupled. Coupled is enabled when AUTO is ON. The <numeric\_value> is RESolution(Hz)/SPAN(Hz).

At \*RST, the value of this function is instrument-dependent.

##### 18.3.1.3 :TRACK <Boolean>

SENSe:BANDwidth:RESolution:TRACK

This function allows the resolution bandwidth to dynamically change during a logarithmic frequency sweep. Logarithmic sweep is enabled via the SENSe:SWEep:SPACing command.

At \*RST, the value of this function is ON.

#### 18.3.2 :VIDeo <numeric\_value>

SENSe:BANDwidth:VIDeo

Controls the video filtering, a particular type of post-detection filtering. This has units of Hertz.

At \*RST, the value of this function is instrument-dependent.

### 18.3.2.1 :**AUTO <Boolean>|ONCE**

SENSe:BANDwidth:VIDeo:AUTO

Couples the value of video bandwidth to instrument-determined values.

At \*RST, the value of this setting is ON.

### 18.3.2.2 :**RATio <numeric\_value>**

SENSe:BANDwidth:VIDeo:RATio

Controls the ratio of video bandwidth to resolution bandwidth when the two are coupled.

Coupled is enabled when AUTO is ON. The <numeric\_value> is  
VIDeo(Hz)/RESolution(Hz).

At \*RST, the value of this function is instrument-dependent.

## 18.4 CONCenTration Subsystem

SENSe:CONCenTration

This subsystem configures the CONCenTration measurement functions and parameters. An instance of this subsystem exists for each range in an instrument. The data defined and queried and the events will only be performed on the currently selected range of the currently selected instrument(s).

KEYWORD	PARAMETER FORM	COMMENTS
:CONCenTration		1999
:CSET	<numeric_value>,<numeric_value>	1999
:LOWer	<numeric_value>	1999
:LSET	POLYnomial<n>   SRATional<n>,<numeric_value>	1999
:RANGE		1999
:AUTO		1999
:LOWer	<numeric_value>	1999
[:STATe]	<Boolean>	1999
:UPPer	<numeric_value>	1999
[:FIXed]	<numeric_value>	1999
:TALign	<numeric_value>	1999
:UPPer	<numeric_value>	1999

### 18.4.1 :CSET <numeric\_value>,<numeric\_value>

:SENSe:CONCenTration:CSET

This command specifies or queries the correction set(s) used for the current range(s). The parameters are the slope and the Y intercept used in the latest correction calculation.

At \*RST these values are not changed.

### 18.4.2 :LOWer <numeric\_value>

:SENSe:CONCenTration:LOWer

This command specifies or queries the lowest measurable concentration for the currently-selected range for the selected instruments.

At \*RST these values are not changed.

### 18.4.3 :LSET POLYnomial<n> | SRATional<n>,<numeric\_value>{,<numeric\_value>}

:SENSe:CONCenTration:LSET

This command sets or returns the curve fit type and coefficients that are currently in use. Coefficients are listed starting with the 0th order term. The first parameter is the curve type, which consists of either POLYnomial for a polynomial curve fit or SRATional for a simple rational polynomial curve fit, followed by a positive integer <n>, which is the curve order. The additional parameters are the curve coefficients. These are listed starting with the 0th order term. Note that the number of coefficients depends on the order of the curve - there are exactly n+1 coefficients for a curve of order n. The parameter list for LSET includes a curve type and associated coefficients for the currently selected range of each currently selected analyzer. See :CALibration:LINEarize:CURVe:FIT for examples.

Note that the :CALibration:LINEarize:ACCept command sets the parameter values.

At \*RST these values are device dependent.

#### 18.4.4 :RANGE

:SENSe:CONCenTration:RANGE

This node allows the setting of a specific range or turns off/on auto-ranging mode for the pre-selected instrument(s).

##### 18.4.4.1 :AUTO

:SENSe:CONCenTration:RANGE:AUTO

The AUTO command controls the automatic range switching feature.

###### 18.4.4.1.1 :LOWER <numeric\_value>

:SENSe:CONCenTration:RANGE:AUTO:LOWER

This command specifies or queries the lower CONCenTration(s) in ppm for automatic ranging for the selected instrument(s). When a CONCenTration is decreasing, this value is the switch point for this range.

At \*RST, these values are not changed.

###### 18.4.4.1.2 [:STATe] <Boolean>

:SENSe:CONCenTration:RANGE:AUTO[:STATe]

The STATe command turns the automatic range switching feature on or off. In automatic range mode, the instrument will switch ranges to keep the measured values within the limits defined in SENSe:CONCenTration:RANGE:AUTO:LOWER and :UPPer. Turning AUTO off will keep the device at the current range.

At \*RST, the STATe is device-dependent.

###### 18.4.4.1.3 :UPPer <numeric\_value>

:SENSe:CONCenTration:RANGE:AUTO:UPPer

This command specifies or queries the upper CONCenTration(s) in ppm for automatic ranging for the selected instrument(s). When a CONCenTration is increasing, this value is the switch point for this range.

At \*RST, these values are not changed.

##### 18.4.4.2 [:FIXed] <numeric\_value>

:SENSe:CONCenTration:RANGE:FIXed

This command specifies or queries the range number of the already-selected instrument(s). The instrument(s) will continue to measure in this range until a subsequent range command changes the range(s). If auto-ranging is enabled, it will be disabled upon specifying range(s) with this command, but will remain unchanged if the range(s) are queried with this command.

At \*RST, the settings of :FIXed are device-dependent.

#### 18.4.5 :TALign <numeric\_value>

:SENSe:CONCenTration:TALign <numeric\_value>

Sets (or queries) the time delay in seconds for the Time Alignment of the continuously measured data.

At \*RST these values are not changed.

### 18.4.6 :UPPer <numeric\_value>

:SENSe:CONCcentration:UPPer

This command specifies or queries the full scale concentration for the currently-selected range for the selected instrument(s).

At \*RST these values are not changed.

18.5 **CONDITION Subsystem**

SENSe:CONDition

This subsystem controls the electrical characteristics of a sensor that determines the ON or OFF condition of a digital signal.

KEYWORD	PARAMETER FORM	COMMENTS
:CONDition		1993
LEVel	<numeric_value>   TTL   ECL	1993

18.5.1 **:LEVel <numeric\_value> | TTL ECL**

SENSe:CONDition:LEVel

This command specifies the characteristics that causes the transition between an ON and OFF condition of the sensed signal. When the value of the input signal exceeds the specified level, the condition is qualified as ON or 1 (True) for positive logic type, and qualified as OFF or 0 (False) when it is associated with negative types of logic. For positive logic, the condition is sensed as OFF or 0 (False) when the input value is below the level; for negative logic this value is qualified as an ON or 1 (True) condition.

The default units are the amplitude units associated with the currently sensed signal.

At \*RST this value is device dependent.

## 18.6 CORRection Subsystem

SENSe:CORRection

The correction subsystem provides for known external losses or gains. In contrast to CALibration, these losses usually are dependent on individual measurement setups.

Correction data may be entered and stored in several ways. A mechanism is provided to automatically measure a reference standard(s) and store the resulting “correction set” in a trace. The trace can be selected at any time as the active “correction set”. This trace could also be generated by an external device and transferred to the instrument over a computer interface. Commands are also provided to enter individual losses, gains, offsets, delays, etc. that the instrument should use in correcting the data. An instrument is not required to implement all of the methods.

The commands in this subsystem are designed to correct for losses, gains, and offsets in external probes or transducers. For instance, they can be used to inform the sensor that a 10X probe has been attached to the instrument. This way, the data and measurement results will be corrected so that they are not 1/10th their actual value. In instruments that automatically detect the attachment of a probe, these commands may not be necessary.

KEYWORD	PARAMETER FORM	COMMENTS
:CORRection		
:AUTO		[event] 1999
:CALCulate		[event] 1999
:COLLect		
[:ACQuire ]	STANDARD	
:METHOD	TPORt	
:SAVE	<trace_name>	
:CSET		
[:SELect]	<name>	1992
:STATE	<Boolean>	
:EDELay		
:DISTance	<numeric_value>	
:STATE	<Boolean>	
[:TIME]	<numeric_value>	
:IMPedance		1993
[:INPut]:OUTPut		1993
[:MAGNitude]	<numeric_value>	1993
:STATE	<Boolean>	1993
:LOSS :GAIN :SLOPe		
[:INPut]:OUTPut		1992
:AUTO	ON OFF	
[:MAGNitude]	<numeric_value>	
:PHASe	<numeric_value>	
:STATE	<Boolean>	
:OFFSet		
[:MAGNitude]	<numeric_value>	

KEYWORD	PARAMETER FORM	COMMENTS
:PHASe	<numeric_value>	
:STATe	<Boolean>	
:RVELOCITY		
:COAX	<numeric_value>	
:MEDIUM	COAX WAVeguide	
:STATe	<Boolean>	
:WAVeguide	<numeric_value>	
:FCUToff	<numeric_value>	
:SPOint		1999
:ACQuire		[event] 1999
:DTOLerance	<numeric_value>	1999
[:STATe]	<Boolean>	
:ZERO		1999
:ACQuire		[event] 1999
:DTOLerance	<numeric_value>	1999

**18.6.1 :AUTO**

:SENSe:CORRection:AUTO

AUTO performs an automatic correction of the selected instruments. The procedure shall be defined according to the manufacturer's recommended procedure. This command is an event and therefore has no \*RST value associated with it.

**18.6.2 :CALCulate**

:SENSe:CORRection:CALCulate

If a calculation is needed for the CORRection subsystem, this command initiates the calculation.

This command is an event, and has no \*RST value.

**18.6.3 :COLlect**

SENSe:CORRection:COLlect

Deals with the collection of the correction data. This data is collected directly by the device by measurements of standards.

**18.6.3.1 [:ACQuire] STANdard**

SENSe:CORRection:COLlect:ACQuire

A measurement will be performed and saved as the data for standard of the chosen correction method. The number of standards required to be measured for the correction is determined by the method chosen. If a standard is specified that does not exist in the chosen method, then an error is generated and no measurement is taken. This measurement will not affect any of the users settings for regular measurements.

For complex corrections, several standards will be measured and then followed by a SAVE command. The parameter on the save command will be the name of the correction set. The standards themselves are device-specific.

This command is an event, does not have a \*RST value, and is not queryable.

**18.6.3.2 :METHOD TPORt**

SENSe:CORRection:COLLect:METHOD

Selects the correction method to be used for the correction that is about to be performed. If only one correction method exists in the instrument then this command is not required.

The various settings of the parameters have the following meanings:

**TPORT** Two Port

At \*RST, this value is device-dependent.

**18.6.3.3 :SAVE [<trace\_name>]**

SENSe:CORRection:COLLect:SAVE

Calculates the correction data using the current method selection and then saves the correction data. If only one correction array exists then it is saved there. Otherwise it is saved in the set specified by <trace\_name>. Note that the parameter <trace\_name> must be defined and exist in the instrument. This command will not create a trace. Trace definition is done in the trace subsystem. Note that the correction array may be a single value.

Depending on the complexity of the correction method chosen, a calculation may or may not be performed on the measured arrays to determine the correction data. For example, if the chosen method has only one standard, then no calculation will be performed.

This command is an event, does not have a \*RST value, and is not queryable.

**18.6.4 :CSET**

SENSe:CORRection:CSET

The Correction SET subsystem is used to select the active correction set.

**18.6.4.1 [:SELect] <name>**

SENSe:CORRection:CSET:SELect

Specifies the active CORRection set. This is the set that is used when CORR:CSET:STATE ON is specified. <name> may be either a <trace\_name>, or a <table\_name> which must be defined and exist in the instrument. This command will not create either <trace\_name>, or a <table\_name>. Trace definition is done in the TRACe subsystem. Table definition is done in the MEMory subsystem.

If <table\_name> specifies a MEMORY:TABLE the various lists must have an equal number of points, or execution error -226 (Lists not same length) will be generated.

Those lists in a TABLE which are defined in the CORRection subsystem will be applied when CORRection:STATE is ON. Other items in a TABLE will be ignored without generating an error. This allows the same TABLE to be shared with other subsystems.

At \*RST this value is device dependent.

**18.6.4.2 :STATE <Boolean>**

SENSe:CORRection:CSET:STATE

Determines whether the correction data defined in the selected set is applied to the measurement.

At \*RST, this function is OFF. Off is chosen because all of the correction sets may be empty. The instrument must collect data before correction can be applied.

#### 18.6.5 :EDElay

SENSe:CORRection:EDElay

The Electrical DELay is used to compensate for delays in the signal path.

This command is used to correct external factors that the instrument cannot measure or detect.

TIME and DISTance are coupled by the equation:

$$TIME = RVELOCITY * DISTANCE$$

RVELOCITY is not coupled to either TIME or DISTANCE.

##### 18.6.5.1 :DISTance <numeric\_value>

SENSe:CORRection:EDElay:DISTance

This command will set the electrical delay with the distance parameter given. The effective delay will be computed using this parameter and the relative velocity set in the instrument. The units for this command are meters. A value accepted here will modify the time setting because they are the same parameter in different forms. Implementation of this command requires implementation of CORRection:EDElay[:TIME].

At \*RST, the value of this setting is zero.

##### 18.6.5.2 :STATE <Boolean>

SENSe:CORRection:EDElay:STATE

This function is used to enable or disable the electrical delay correction.

At \*RST, this function is set to OFF.

##### 18.6.5.3 [:TIME] <numeric\_value>

SENSe:CORRection:EDElay:TIME

This command will set the electrical delay with the time parameter given. The units for this command are seconds. A value accepted here will modify the distance setting because they are the same parameter in different forms.

At \*RST, the value of this setting is zero.

#### 18.6.6 :IMPedance

SENSe:CORRection:IMPedance

This is the termination impedance provided external to the device.

##### 18.6.6.1 [:INPut]:OUTPut

SENSe:CORRection:IMPedance:INPut

The INPut and OUTPut nodes are used to reference measurements made at the instrument's input and output, respectively. Most sensor instruments are only capable of making measurements at its input ports; however, some instruments are capable of making measurements on output ports that are sourcing a signal.

**18.6.6.1.1 [:MAGNitude] <numeric\_value>**

SENSe:CORRection:IMPedance:INPut:MAGNitude

Sets the magnitude of the impedance. The default units are Ohms

At \*RST, this value is device-dependant.

**18.6.6.2 :STATE <Boolean>**

SENSe:CORRection:IMPedance:STATE

The STATE command is used to enable or disable external IMPedance correction factors.

At \*RST, this function is set to OFF.

**18.6.7 :LOSS|:GAIN|:SLOPe**

SENSe:CORRection:LOSS

These commands are used either to provide a mechanism to apply simple correction or to correct external factors that have not been accounted for in the correction set. These correction factors are typically applied to measurements made at the device input; however, in cases where measurements can be at the device output a separate set of correction factors may be applied.

Loss, gain and slope effect the measurement values as a multiplier for linear units such as VOLT and as additive (or subtractive) value for logarithmic units such as DB.

:LOSS This command is used to set the anticipated loss in the system, excluding those factors covered by the active correction set. The device then corrects every measurement by this factor to compensate for the loss. LOSS is coupled to GAIN by the equation  $\text{LOSS} = 1/\text{GAIN}$  when the default unit is linear and  $\text{LOSS}=-\text{GAIN}$  when the default unit is logarithmic.

:GAIN This command is used to set the anticipated gain in the system, excluding those factors covered by the active correction set. The device then corrects every measurement by this factor to compensate for the gain. GAIN Is coupled to LOSS by the equation  $\text{GAIN}=1/\text{LOSS}$  when the default unit is linear and  $\text{GAIN}=-\text{LOSS}$  when the default unit is logarithmic.

:SLOPe This node defines an increasing or decreasing value that is added to the signal. This correction value is zero at a zero value of the x axis and increase or decreases from there.

**18.6.7.1 [:INPut]|[:OUTPut]**

SENSe:CORRection:LOSS:INPut

The INPut and OUTPut nodes are used to reference measurements made at the instrument's input and output, respectively. Most sensor instruments are only capable of making measurements at its input ports; however, some instruments are capable of making measurements on output ports that are sourcing a signal.

**18.6.7.1.1 :AUTO ON|OFF**

SENSe:CORRection:LOSS:INPut:AUTO

When AUTO is ON the OUTPut correction values must track those of the INPut.

At \*RST, AUTO is set to ON.

#### 18.6.7.1.2 [:MAGNitude] <numeric\_value>

SENSe:CORRection:LOSS:INPut:MAGNitude

This is the magnitude value of the correction data for LOSS, GAIN or SLOPe. The units associated with LOSS and GAIN are the current relative amplitude unit. The unit associated with SLOPe is:

*<current realtive amplitude unit>/Hz*

After a \*RST, this value should no longer affect the data being measured. Thus its value is set to zero at \*RST.

#### 18.6.7.1.3 [:PHASe] <numeric\_value>

SENSe:CORRection:LOSS:INPut:PHASe

This is the phase value of the correction data for LOSS, GAIN or SLOPe. The unit associated with SLOPe is current angle units/Hertz. The units associated with LOSS and GAIN are current angle units.

After a \*RST, this value should no longer affect the data being measured. Thus its value is set to zero at \*RST.

#### 18.6.7.2 [:STATe] <Boolean>

SENSe:CORRection:LOSS:STATE

The STATE command is used to enable or disable the LOSS, GAIN or SLOPe correction factors.

At \*RST, this function is set to OFF.

#### 18.6.8 [:OFFSet]

SENSe:CORRection:OFFSet

This node defines a single value that is added to every point measured by the instrument. If the instrument is a trace-based machine, then this value is added to every point in the trace. If the instrument measures a single value then that value is modified by this setting. This command shall be used with the currently selected units. It is used to correct external factors that the instrument cannot measure or detect.

#### 18.6.8.1 [:MAGNitude] <numeric\_value>

SENSe:CORRection:OFFSet:MAGNitude

This is the magnitude value of the correction offset data.

After a \*RST, this value should no longer affect the data being measured. Its value is set to zero at \*RST.

#### 18.6.8.2 [:PHASe] <numeric\_value>

SENSe:CORRection:OFFSet:PHASe

This is the phase value of the correction offset data. The units associated with this are the currently selected angle units.

**18.6.8.3 :STATe <Boolean>**

SENSe:CORRection:OFFSet:STATe

This function is used to enable or disable the offset correction.

At \*RST, this function is set to OFF.

After a \*RST, this value should no longer affect the data being measured. Its value is set to zero at \*RST .

**18.6.9 :RVELocity**

SENSe:CORRection:RVELocity

This command corrects for the relative velocity of the medium of the Device Under Test. This factor will be used in any delay calculation performed in the instrument.

This command is used to correct external factors that the instrument cannot measure or detect.

TIME and DISTance are coupled by the equation:

$$TIME = RVELocity \cdot DISTance$$

RVELocity is not coupled to either TIME or DISTance.

**18.6.9.1 :COAX <numeric\_value>**

SENSe:CORRection:RVELocity:COAX

This command sets the relative velocity factor for coaxial lines. The parameter is unitless.

At \*RST, the value of this setting is one.

**18.6.9.2 :MEDIUM COAX|WAVeguide**

SENSe:CORRection:RVELocity:MEDIUM

This command selects the correction algorithm for the media through which the detected signal is transmitted. It will determine the method of calculation for any operation that uses relative velocity. For example, for COAX a linear phase shift would be computed if a delay was added, but for WAVeguide (a dispersive medium) a different phase shift is calculated.

At \*RST, this value is device-dependent.

**18.6.9.3 :STATe <Boolean>**

SENSe:CORRection:RVELocity:STATe

This function is used to enable or disable the relative velocity correction.

At \*RST, this function is set to OFF.

**18.6.9.4 :WAVeguide <numeric\_value>**

SENSe:CORRection:RVELocity:WAVeguide

This command sets the relative velocity factor for waveguide. The parameter is unitless.

At \*RST, the value of this setting is one.

**18.6.9.4.1 :FCUToff <numeric\_value>**

SENSe:CORRection:RVELocity:WAVeguide:FCUToff

This command specifies the frequency cutoff of the waveguide medium. The units are Hertz.

At \*RST, this value is device-dependent.

#### 18.6.10 :SPOint

:SENSe:CORRection:SPOint

This subsystem controls the set point correction of the sensor.

For Emissions benches, this subsystem references the span gas of the instrument.

This command is an event and therefore has no \*RST value associated with it.

##### 18.6.10.1 :ACQuire

:SENSe:CORRection:SPOint:ACQuire

This command supplies a known signal to the selected instruments and waits for the stabilization criteria to be met..

This command is an event, and has no \*RST value.

##### 18.6.10.2 :DTOLerance <numeric\_value>

:SENSe:CORRection:SPOint:DTOLerance

This command specifies or queries the drift tolerance for a set point correction procedure for the current range of the selected instrument(s).

For emission benches, this tolerance is the difference between the specified concentration referenced in the SpanGas and GasBottle tables and the measured concentration. Drift tolerance is in units of percent of full scale value.

The data defined and queried will only apply to the currently selected range of the currently selected instrument(s).

At \*RST, this value is not changed.

#### 18.6.11 [:STATe] <Boolean>

SENSe:CORRection:STATe

Determines whether the correction data defined in this section is applied to the measurement.

At \*RST, this function is OFF.

#### 18.6.12 :ZERO

:SENSe:CORRection:ZERO

This subsystem controls the zero correction of the sensor.

##### 18.6.12.1 :ACQuire

:SENSe:CORRection:ZERO:ACQuire

This command sends a zero signal to the selected instruments and waits for the stabilization criteria to be met.

This command is an event, and has no \*RST value

##### 18.6.12.2 :DTOLerance <numeric\_value>

:SENSe:CORRection:ZERO:DTOLerance

This command specifies the drift tolerance for a zero procedure for the selected instrument(s).

## **1999 SCPI Command Reference**

At \*RST, this value is not changed.

## 18.7

**CURRent Subsystem**

SENSe:CURRent

This subsection controls the current amplitude characteristics of the sensor.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:CURRent		
:AC[:DC]		
:APERture	<numeric_value>	
:NPLCycles	<numeric_value>	1991
:ATTenuation	<numeric_value>	
:AUTO	<Boolean>	
:PROTection		
[:LEVel]	<numeric_value>	
:STATe		
:TRIPped?		[query only]
:CLEar		[no query]
:RANGe		
[:UPPer]	<numeric_value>	
:LOWer	<numeric_value>	
:AUTO	<Boolean> ONCE	
:DIRection	UP DOWN EITHER	
:LLIMit	<numeric_value>	1996
:ULIMit	<numeric_value>	1996
:OFFSet		1991
:PTPeak		1991
:REFerence	<numeric_value>	
:STATe	<Boolean>	
:RESolution	<numeric_value>	
:AUTO	<Boolean> ONCE	
:DETector	INTernal   EXTernal	1991

## 18.7.1 :AC[:DC]

SENSe:CURRent:AC

This subsystem selects the alternating current or the direct current sensor. The default units are amperes.

## 18.7.1.1 :APERture &lt;numeric\_value&gt;

SENSe:CURRent:AC:APERture

Specifies the acquisition/sampling/gate time for a single measurement point. The parameter has a unit of seconds.

At \*RST, this value is device-dependent.

**18.7.1.2 :NPLCycles <numeric\_value>**

SENSe:CURRent:AC:NPLCycles

Specifies the acquisition/sampling/gate time for a single measurement point in terms of the number of power line cycles. The parameter is unitless. If this command is implemented, the APERture command must also be implemented.

The value is coupled to APERture by the equation:

$$\text{APERture} = \text{NPLCycles}/\text{selected line frequency}$$

At \*RST, this value is device-dependent.

**18.7.1.3 :ATTenuation <numeric\_value>**

SENSe:CURRent:AC:ATTenuation

Sets the ATTenuation level. Note that when setting the level to 10 dB the magnitude of the incoming signal will be decreased by 10 dB. Changing the ATTenuation changes RANGe by the same amount.

Default units are as determined in the UNITS system.

**18.7.1.3.1 :AUTO <Boolean>**

SENSe:CURRent:AC:ATTenuation:AUTO

Couples the attenuator to RANGe such that maximum dynamic range of the incoming signal is assured without overloading.

Programming a specified attenuation sets AUTO OFF.

At \*RST, AUTO is set to ON.

**18.7.1.4 :PROTection**

SENSe:CURRent:AC:PROTection

Controls the protection circuits.

**18.7.1.4.1 [:LEVel] <numeric\_value>**

SENSe:CURRent:AC:PROTection:LEVel

This command sets the input level at which the input protection circuit will trip.

18

**18.7.1.4.2 :STATe <Boolean>**

SENSe:CURRent:AC:PROTection:STATE

Controls whether the input protection circuit is enabled.

At \*RST, this value is set to ON.

**18.7.1.4.3 :TRIPped?**

SENSe:CURRent:AC:PROTection:TRIPped?

This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped. TRIPped only has a query form and thus has no associated \*RST condition.

**18.7.1.4.4 :CLEar**

SENSe:CURRent:AC:PROTection:CLEar

Causes the protection circuit to be cleared.

This command is an event and has no associated \*RST condition.

**18.7.1.5 :RANGE**

SENSe:CURRent:AC:RANGE

Sets the range for the amplitude sensor function. The RANGE endpoints may be specified in terms of the UPPer endpoint and the LOWER endpoint, alternatively the midpoint of the RANGE (OFFSet) and the span of the RANGE (PTPeak) may be used. A non-symmetrical range around zero may cause some instruments to introduce an actual offset to the signal entering the SENSe block.

**18.7.1.5.1 [:UPPer] <numeric\_value>**

SENSe:CURRent:AC:RANGE:UPPer

Specifies the most positive signal level expected for the sensor input.

At \*RST, this value is device-dependent.

**18.7.1.5.2 :LOWER <numeric\_value>**

SENSe:CURRent:AC:RANGE:LOWER

Specifies the most negative signal level expected for the sensor input. This allows setting for the best dynamic range. This function is coupled to UPPer in a device-dependent manner.

At \*RST, this value is device-dependent.

**18.7.1.5.3 :AUTO <Boolean>|ONCE**

SENSe:CURRent:AC:RANGE:AUTO

Sets the RANGE to the value determined to give the most dynamic range without overloading. The actual algorithm is device-dependent.

At \*RST, AUTO is set to ON.

**18.7.1.5.3.1 :DIRection UP|DOWN|EITHER**

SENSe:CURRent:AC:RANGE:AUTO:DIRection

The DIRection command defines the manner in which AUTO works. When UP is selected, the instrument shall only change the range automatically to ranges larger than the current range, as the input signal dictates. When DOWN is selected, the instrument shall only change the range automatically to ranges smaller than the current range, as the input signal dictates. When EITHER is selected the instrument may range in either direction.

At \*RST, DIRection shall be set to EITHER.

**18.7.1.5.3.2 :LLIMit <numeric\_value>**

SENSe:CURRent:AC:RANGE:AUTO:LLIMit

The Lower LIMit command sets the smallest range to which the instrument will go while autoranging. When AUTO is ON and the present RANGE is outside the new limits, the instrument may change the RANGE to a value inside the limits or wait until the next time the autoranging algorithm is performed.

At \*RST, LLIMit shall be set to MINimum.

**18.7.1.5.3.3 :ULIMit <numeric\_value>**

SENSe:CURRent:AC:RANGE:AUTO:ULIMit

The Upper LIMit command sets the largest range to which the instrument will go while

autoranging. When AUTO is ON and the present RANGe is outside the new limits, the instrument may change the RANGe to a value inside the limits or wait until the next time the autoranging algorithm is performed.

At \*RST, ULIMit shall be set to MAXimum.

### 18.7.1.5.4 :OFFSet <numeric\_value>

SENSe:CURRent:AC:RANGE:OFFSet

OFFSet determines the midpoint of the range.

At \*RST, this value is device-dependent.

### 18.7.1.5.5 :PTPeak <numeric\_value>

SENSe:CURRent:AC:RANGE:PTPeak

Peak To Peak specifies the dynamic range required for the sensor.

At \*RST, this value is device-dependent.

### 18.7.1.6 :REFerence <numeric\_value>

SENSe:CURRent:AC:REFerence

Sets a reference amplitude for sensor instruments. This command differs in intent from the reference correction in that this is intended to be a measurement offset where a user wishes to reference a measurement to some known value.

At \*RST, this value is device-dependent, but STATe is set to OFF.

### 18.7.1.6.1 :STATe <Boolean>

SENSe:CURRent:AC:REFERENCE:STATe

Determines whether amplitude is measured in absolute or relative mode. If STATe is ON, then amplitude is referenced to the value set in REFerence.

At \*RST, this value is set to OFF.

### 18.7.1.7 :RESolution <numeric\_value>

SENSe:CURRent:AC:RESolution

Specifies the absolute resolution of the measurement. For example, if a measurement of 10 Volts is specified with a resolution of .01 Volts, the measurement is returned with a resolution of .01 Volts (that is, 10.00).

RANGe is not coupled to RESolution. Setting RANGe shall not cause RESolution to change. Setting RESolution shall not cause RANGe to change.

At \*RST, this value is device-dependent.

### 18.7.1.7.1 :AUTO <Boolean>|ONCE

SENSe:CURRent:AC:RESolution:AUTO

Allows the system to determine the best resolution for the other measurement conditions.

Explicitly setting a value for :RESolution will turn AUTO OFF.

At \*RST, this value is set to ON.

### 18.7.2 :DETector INTernal | EXTernal

SENSe:CURRent:DETector

The DETector can be specified as internal or external.

At \*RST, this value is set to INT.

## 18.8 DETector Subsystem

SENSe:DETector

This subsystem controls the acquisition of data points, including how to choose points and the linear characteristics of the sensor.

KEYWORD	PARAMETER FORM	NOTES
:DETector		
:BANDwidth	<numeric_value>	1991
:BWIDth		1991
[:FUNCtion]	GROund NEGative POSitive SAMPlE SIGNAl  PAVerage   AAVerage RMS	1991
:AUTO	<Boolean> ONCE	
:SHAPe	LINear LOGarithmic	

### 18.8.1 :BANDwidth | BWIDth

&lt;numeric\_value&gt;

SENSe:DETector:BANDwidth

The BANDwidth function controls the bandwidth after an intermediate signal has been processed within the SENSe block. Instruments must accept both forms, if they accept either.

### 18.8.2 [:FUNCtion] <detector function>

SENSe:DETector:FUNCtion

Specifies the detector sampling characteristics. The various settings have the following meanings:

- GROund: The input detector is grounded. Thus no signal is acquired.
- NEGative: The minimum signal detected during the acquisition of the particular point is chosen.
- POSitive: The maximum signal detected during the acquisition of the particular point is chosen.
- SAMPlE: A single arbitrary point detected is chosen.
- SIGNAl: The detector chooses points so that envelope noise is maximized and positive signals above some threshold are also maximized.
- PAVerage: PeakAVerage is the average of the minimum and maximum excursions of the signal. PAV = ( absolute minimum peak + absolute maximum peak ) / 2
- AAVerage: AbsoluteAVerage averages the absolute values of a sine wave input and processes the value to read RMS values.
- RMS: True Root Mean Square signal detector.

At \*RST, this value is device-dependent.

### 18.8.2.1 :**AUTO <Boolean>|ONCE**

SENSe:DETector:FUNCtion:AUTO

Couples the detector function to several settings within the instrument. When AUTO ON is selected, the detector type is determined by the system in order to give the “best” quality of data.

Explicitly selecting a detector will set AUTO OFF.

At \*RST, this parameter is set to ON.

### 18.8.3 :**SHAPe LINear|LOGarithmic**

SENSe:DETector:SHAPe

Controls the linearity of the detector. This setting is coupled to DETector:FUNCtion in a device-dependent manner.

At \*RST, this value is device-dependent.

## 1999 SCPI Command Reference

### 18.9 DISTance Subsystem

SENSe:DISTance

This node is used to set the parameters associated with sensing distance.

KEYWORD	PARAMETER FORM	NOTES
:DISTance		1999
:RESet		[event; no query]1999

#### 18.9.1 :RESet

:SENSe:DISTance:RESET

This command will set DISTance to 0. This command is not queryable.

At \*RST the DISTance is set to zero.

**18.10 FILTer Subsystem**

SENSe:FILTer

The FILTer function allows a filter to be inserted after an intermediate signal has been processed within the SENSe block. This FILTer subsystem is not interchangeable with the INPut:FILTer subsystem.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:FILTer		1991
[:LPASs]		1991
[:STATe]	<Boolean>	1991
[:FREQuency	<numeric_value>	1991
[:HPASs]		1991
[:STATe]	<Boolean>	1991
[:FREQuency	<numeric_value>	1991
:DEMPhasis		1991
[:STATe]	<Boolean>	1991
[:TCONstant	<numeric_value>	1991
:CCITt		1991
[:STATe]	<Boolean>	1991
:CMESsage		1991
[:STATe]	<Boolean>	1991
:CCIR		1991
[:STATe]	<Boolean>	1991
:CARM		1991
[:STATe]	<Boolean>	1991
:AWEighting		1991
[:STATe]	<Boolean>	1991

**18.10.1 [:LPASs]**

SENSe:FILTer:LPASs

Controls the sensor low pass filter.

**18.10.1.1 [:STATe] <Boolean>**

SENSe:FILTer:LPASs:STATe

Turns the low pass filter on and off.

At \*RST, this value is device-dependent.

**18.10.1.2 :FREQuency <numeric\_value>**

SENSe:FILTer:LPASs:FREQuency

Determines the cutoff frequency of the low pass filter.

At \*RST, this value is device-dependent.

**18.10.2 :HPASs**

SENSe:FILTer:HPASs

Controls the sensor high pass filter.

### 18.10.2.1 [:STATe] <Boolean>

SENSe:FILTter:HPASs:STATe

Turns the high pass filter on and off.

At \*RST, this value is device-dependent.

### 18.10.2.2 :FREQuency <numeric\_value>

SENSe:FILTter:HPASs:FREQuency

Determines the cutoff frequency of the high pass filter.

At \*RST, this value is device-dependent.

### 18.10.3 :DEMPhasis

SENSe:FILTter:DEMPhasis

Controls the sensor FM De-EMPhasis filter. De-emphasis filters compensate for pre-emphasis on FM signals.

### 18.10.3.1 [:STATe] <Boolean>

SENSe:FILTter:DEMPhasis:STATe

Turns the FM de-emphasis filter on and off.

At \*RST, this value is device-dependent.

### 18.10.3.2 :TCONstant <numeric\_value>

SENSe:FILTter:DEMPhasis:TCONstant

Determines the Time CONstant of the FM de-emphasis filter. The default units are in seconds.

At \*RST, this value is device-dependent.

### 18.10.4 :CCITt

SENSe:FILTter:CCITt

Controls the CCITT sensor filter. This standard filter is described in "CCITT Recommendation P53". It is a voice bandwidth filter used in radio or telephone applications.

### 18.10.4.1 [:STATe] <Boolean>

SENSe:FILTter:CCITt:STATe

Turns the CCITT filter on and off.

At \*RST, this value is device-dependent.

### 18.10.5 :CMESsage

SENSe:FILTter:CMESsage

Controls the C-Message sensor filter. This Bell Telephone standard filter is described in "Per BTSM 41004". It is a voice bandwidth filter used in radio or telephone applications.

### 18.10.5.1 [:STATe] <Boolean>

SENSe:FILTter:CMESsage:STATe

Turns the C-Message filter on and off.

At \*RST, this value is device-dependent.

### 18.10.6 :CCIR

SENSe:FILTer:CCIR

Controls the CCIR weighting sensor filter. This standard filter is described in “CCIR Recommendation 468-2”. It is a music bandwidth filter used in applications such as broadcasting and FM receivers.

#### 18.10.6.1 [:STATe] <Boolean>

SENSe:FILTer:CCIR:STATe

Turns the CCIR weighting filter on and off.

At \*RST, this value is device-dependent.

### 18.10.7 :CARM

SENSe:FILTer:CARM

Controls the CCIR/ARM weighting sensor filter. This is a standard music bandwidth filter described in “CCIR Recommendation 468-2” and “Dolby Labs Bulletin No. 19/4”.

#### 18.10.7.1 [:STATe] <Boolean>

SENSe:FILTer:CARM:STATe

Turns the CCIR/ARM weighting filter on and off.

At \*RST, this value is device-dependent.

### 18.10.8 :AWEighting

SENSe:FILTer:AWEighting

Controls the “A” weighting sensor filter. This is a human hearing compensation filter described in “IEC Recommendation 179” and “ANSI S1.4”.

#### 18.10.8.1 [:STATe] <Boolean>

SENSe:FILTer:AWEighting:STATe

Turns the “A” weighting filter on and off.

At \*RST, this value is device-dependent.

**18.11 FM Subsystem**

SENSe:FM

The Frequency Modulation subsystem is used to set up the modulation controls and parameters associated with sensing and demodulating frequency modulated signals.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:FM		1991
[:DEViation]		1991
:RANGE		1991
:AUTO	<Boolean> ONCE	1991
[:UPPer]	<numeric_value>	1991
:LOWER	<numeric_value>	1991

**18.11.1 [:DEViation]**

SENSe:FM:DEViation

This node is used to collect together the deviation parameters.

**18.11.1.1 :RANGE**

SENSe:FM:DEViation:RANGE

Sets the range for the sensor function.

**18.11.1.1.1 :AUTO <Boolean> | ONCE**

SENSe:FM:DEViation:RANGE:AUTO

Sets the RANGE to the value determined to give the most dynamic range without overloading. The actual algorithm is device-dependent. Selecting AUTO ONCE will have the effect of setting AUTO to ON and then OFF.

At \*RST, this value is set to ON.

**18.11.1.1.2 [:UPPer] <numeric\_value>**

SENSe:FM:DEViation:RANGE:UPPer

Specifies the maximum signal level expected for the sensor input.

At \*RST, this value is device-dependent.

**18.11.1.1.3 :LOWER <numeric\_value>**

SENSe:FM:DEViation:RANGE:LOWER

Specifies the smallest signal level expected for the sensor input. This allows setting for the best dynamic range. This function is coupled to UPPer.

At \*RST, this value is device-dependent.

**18.12 FREQuency Subsystem**

SENSe:FREQuency

The FREQuency subsystem controls the frequency characteristics of the instrument. Two types of controls are included in this subsystem. The first is frequency sensor controls, such as RANGE. The second is frequency sweep controls, such as STARt, or tuning controls such as CW.

Unless otherwise noted, the default unit for all <numeric\_values> in this subsystem are in Hz.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:FREQuency	<numeric_value>	
:APERture	<numeric_value>	
:CENTer	<numeric_value>	
[:CW :FIXed]	<numeric_value>	
:AFC	<Boolean> ONCE	1994
:AUTO	<Boolean> ONCE	
:MANual	<numeric_value>	
:MODE	CW FIXed SWEep LIST SOURce	
:MULTiplier	<numeric_value>	
:OFFSet	<numeric_value>	
:RANGE		
[:UPPer]	<numeric_value>	
:LOWER	<numeric_value>	
:AUTO	<Boolean> ONCE	
:RESolution	<numeric_value>	
:AUTO	<Boolean> ONCE	
:SPAN	<numeric_value>	
:HOLD	<Boolean>	
:LINK	CENTer STARt STOP	
:FULL		[no query]
:STARt	<numeric_value>	
:STOP	<numeric_value>	

**18.12.1 :APERture <numeric\_value>**

SENSe:FREQuency:APERture

Specifies the acquisition/sampling/gate time for a single measurement point. The parameter has a unit of seconds.

At \*RST, this value is device-dependent.

**18.12.2 :CENTer <numeric\_value>**

SENSe:FREQuency:CENTer

Set the center frequency of the sweep or measurement.

At \*RST, this function will be set to (MINimum+MAX)/2.

**18.12.3 [:CW|FIXed] <numeric\_value>**

SENSe:FREQuency:CW

The Continuous Wave or FIXed node is used to select a frequency of a non-swept signal. This node is optional. An optional node, such as [:CW|FIXed], implies that the device shall accept and process commands with or without the optional node and have the same result. That is, the device is required to accept the optional node, if sent, without error. A device must accept both :CW and :FIXed if it implements this node.

At \*RST, this value is device-dependent.

**18.12.3.1 :AFC <Boolean>|ONCE**

SENSe:FREQuency:CW:AFC

If AFC is ON the sensor is coupled to the frequency of the signal. The default frequency is determined by SENSe:FREQuency[:CW].

The actual value of FREQuency[:CW] is adjusted to match the frequency of the signal. The query FREQuency[:CW]? returns the adjusted value, not the value sent with the last FREQuency[:CW] command.

At \*RST, this value is set to OFF.

**18.12.3.2 :AUTO <Boolean>|ONCE**

SENSe:FREQuency:CW:AUTO

Couples the CW frequency to center frequency. Explicitly setting FREQuency:CW will set AUTO OFF.

At \*RST, this value is set to ON.

**18.12.4 :MANual <numeric\_value>**

SENSe:FREQuency:MANual

Same as CW except limited by START and STOP. This command only affects the frequency if SENSe:SWEep:MODE Manual and SENSe:FREQuency:MODE SWEep are selected. If the sweep limits are changed such that manual frequency would be outside the sweep limits, then the manual frequency will be set to the nearest end point of the sweep.

At \*RST, this value is device-dependent.

**18.12.5 :MODE CW|FIXed|SWEep|LIST|SOURce**

SENSe:FREQuency:MODE

Determines which set of commands control the frequency. If CW or FIXed is selected, the frequency is determined by FREQuency:CW. CW and FIXed are aliases. Both must be implemented if either is implemented. If SWEep is selected, the sensor is in the swept mode and frequency is determined by FREQuency:START, STOP, CENT, SPAN, and MAN. If LIST is selected, the frequencies are determined by LIST:FREQuency.

If SOURce is selected, the sensor is coupled to the source. All frequency controls are determined by the SOURCe:FREQuency block, except for OFFSet and MULTiplier. These two commands remain active under SENSe to allow for offset and harmonic tracking.

At \*RST, this value is set to CW.

**18.12.6 :MULTiplier <numeric\_value>**

SENSe:FREQuency:MULTiplier

Sets a reference multiplier for all other frequency settings in the instrument. All entered and displayed frequencies are affected by this command. The coupling equation is:

$$\text{Entered/Displayed frequency} = (\text{Hardware frequency} * \text{multiplier}) + \text{offset}.$$

This command affects the values of all functions which affect sensing of the signal. This includes such things as CW frequency and the sweep frequency controls. It does not affect things like input filter bandwidths. It does affect the settings of relative frequency parameters such as delta markers or FM deviation. Also note that this command does not affect the hardware settings of the instrument, but only the entered and displayed frequencies.

At \*RST, this value is set to 1.

**18.12.7 :OFFSet <numeric\_value>**

SENSe:FREQuency:OFFSet

Sets a reference frequency for all other absolute frequency settings in the instrument. This command affects the values of all functions which affect sensing of the signal. This includes such things as CW frequency and the sweep frequency controls. It does not affect things like input filter bandwidths. It also does not affect the settings of relative frequency parameters such as delta markers or SPAN. Also note that this command does not affect the hardware settings of the instrument, but only the entered and displayed frequencies. The coupling equation is:

$$\text{Entered/Displayed frequency} = (\text{Hardware frequency} * \text{multiplier}) + \text{offset}.$$

At \*RST, this value is set to 0.

**18.12.8 :RANGe**

SENSe:FREQuency:RANGe

Sets the frequency range for the sensor function.

**18.12.8.1 [:UPPer] <numeric\_value>**

SENSe:FREQuency:RANGe:UPPer

Specifies the maximum value expected for the sensor input.

At \*RST, this value is device-dependent.

**18.12.8.2 :LOWer <numeric\_value>**

SENSe:FREQuency:RANGe:LOWer

Specifies the lowest value expected for sensor input. This function is coupled to UPPer in a device-dependent manner.

At \*RST, this value is device-dependent.

**18.12.8.3 :AUTO <Boolean>|ONCE**

SENSe:FREQuency:RANGe:AUTO

Sets the RANGe to the value determined by the instrument.

At \*RST, AUTO is set to ON.

### 18.12.9 :RESolution <numeric\_value>

SENSe:FREQuency:RESolution

Specifies the absolute resolution of the frequency measurement. For example, if a measurement of 10 MHZ is made with a resolution of .01, the measurement is returned with a resolution of .01 Hz (that is, 10000000.00Hz)

At \*RST, this value is device-dependent.

### 18.12.9.1 :AUTO <Boolean>|ONCE

SENSe:FREQuency:RESolution:AUTO

Allows the system to determine the best resolution for the other measurement conditions.

Explicitly selecting a value for FREQuency:RESolution will set AUTO OFF.

At \*RST, this value is set to ON.

### 18.12.10 :SPAN <numeric\_value>

SENSe:FREQuency:SPAN

Sets the frequency span.

At \*RST, this function will be set to MAXimum.

### 18.12.10.1 :HOLD <Boolean>

SENSe:FREQuency:SPAN:HOLD

Provides a mechanism to prevent the SPAN from being changed implicitly by the defined coupling between START, STOP, CENTER and SPAN. When HOLD is set to ON, SPAN shall only be changed by explicitly sending the SPAN command.

At \*RST, this value is set to OFF.

### 18.12.10.2 :LINK CENTer|STARt|STOP

SENSe:FREQuency:SPAN:LINK

Allows the default couplings for SPAN to be overridden. LINK selects the parameter, either CENTER, STARt or STOP, that shall not be changed when SPAN's value is changed. For example, if LINK is set to STARt, then changing SPAN shall cause CENTER and STOP, not STARt, to change.

At \*RST, this value is set to CENTER, to be compatible with the default definition for the couplings.

### 18.12.10.3 :FULL

SENSe:FREQuency:SPAN:FULL

When this command is received, start frequency is set to its minimum value, and stop frequency is set to its maximum value. Center frequency and span are set to their coupled values.

This command is an event. Therefore it has no associated query, or meaning at \*RST.

### 18.12.11 :STARt <numeric\_value>

SENSe:FREQuency:STARt

Sets the starting frequency for a sweep or measurement. It also sets one limit of a manual frequency adjustment. If the instrument does not support negative frequency span, having the start frequency greater than the stop frequency will cause an error to be generated.

At \*RST, the parameter will be set to MINimum.

### 18.12.12 :STOP <numeric\_value>

SENSe:FREQuency:STOP

Sets the stop frequency of a sweep or measurement. It also sets the other limit of a manual sweep adjustment. If the instrument does not support negative frequency span, having the stop frequency less than the start frequency will cause an error to be generated.

At \*RST, this parameter will be set to MAXimum.

## 18.13 FUNCtion & DATA Subsystem

SENSe:FUNCtion & DATA

The FUNCtion subsystem selects the <sensor\_function>(s) to be sensed by the instrument. These functions differ from the signal-oriented MEASure instructions in that a <sensor\_function> is something that the sensor can be configured to directly detect.

The DATA subsystem is used to query data from the SENSe block.

KEYWORD	PARAMETER FORM	NOTES
:DATA?	[<data_handle>]	[query only] 1991
:DATA		1995
:PREamble?	[<data_handle>]	[query only] 1994
:FUNCtion		
:CONCurrent	<Boolean>	1991
:OFF	<sensor_function>{,<sensor_function>}	1991
:ALL		[no query] 1991
:COUNT?		[query only] 1991
[:ON]	<sensor_function>{,<sensor_function>}	1991
:ALL		[no query] 1991
:COUNT?		[query only] 1991
:STATe?	<sensor_function>	[query only] 1991

### 18.13.1 DATA? [<data\_handle>]

SENSe:DATA?

Provides access to the result(s) of the SENSe block. This query retrieves the current SENSe result(s).

The <data\_handle> parameter to this query is optional. Instruments that implement this query are required to accept the optional parameter <data\_handle> if sent. If a sensor function is specified, and that function is currently being sensed the result is returned. If the function is not being sensed an error -221 (Settings Conflict) shall be generated.

If a <data\_handle> is not specified, then the result(s) of all functions being sensed are returned. If CONCurrent is OFF there will be only one FUNCtion result. If CONCurrent is ON there will be one or more results, depending on how many functions are currently on. The order of the function results returned by the DATA? query is device-dependent, although the order of these results shall be the same as the order of the functions returned by the FUNCtion[:ON]? query. If no functions are currently being sensed an error -221 (Settings Conflict) shall be generated. The data returned by the DATA? query is affected by the FORMAT subsystem.

This command is query-only and therefore has no \*RST condition.

#### 18.13.1.1 SENSe <data\_handle>s

If the SENSe block only emits one data flow, then the <data\_handle> is just the string "SENSe<n>", where <n> is a numeric suffix. However, for the case where the SENSe block outputs more than one result, each data result needs to be assigned a unique <data\_handle>, symbolic of the data flow. Since the SENSe:FUNCtion command is already defined to take a

well-defined string as a parameter, these <sensor\_function> strings are used as part of the <data\_handle>. If more than one SENSe block in the instrument outputs the same <sensor\_function> the SENSe block must also be included in the data handle. The syntax for the SENSe block <data\_handle> is:

<data\_handle> := “[SENSe:]<sensor\_function>” / “SENSe[:<sensor\_function>]”

#### 18.13.1.2 :PREamble? [<data\_handle>]

SENSe:DATA:PREamble?

PREamble? returns the preamble information supporting the DATA(CURVe(VALUes)) but omits the following data:

- DIF blocks of waveform, measurement, and delta.
- DIF curve block keywords VALUes and CSUM and their parameters.

The DIF difid block must contain the keyword SCOPe with the parameter of PREamble.

While the PREamble? query is designed to efficiently transmit data relating directly to the curve, the various notes and identification parameters may be sent if absolutely required.

The optional <data\_handle> parameter functions the same as it does in SENSe:DATA?.

#### 18.13.2 :FUNCtion

SENSe:FUNCTION

The FUNCtion subsystem selects the <sensor\_function>(s) to be sensed by the instrument. For compatibility, instruments that support CONCurrent ON are required to support CONCurrent OFF mode. This is so some level of compatibility can be maintained with non-concurrent instruments which expect data results for only one FUNCtion to be returned by the SENSe:DATA? query.

#### 18.13.2.1 :CONCurrent <Boolean>

SENSe:FUNCTION:CONCurrent

The CONCurrent command indicates whether the SENSoR block should be configured to SENSe one function at a time or SENSe more than one function at a time (concurrently).

If CONCurrent is OFF, the FUNCtion[:ON] command acts as a one-of-n switch selecting the indicated function to be the only sensed function. If CONCurrent is ON, the function(s) specified as parameter(s) to the FUNCtion[:ON] command are turned on, while the state of other functions are not affected.

When CONCurrent is turned OFF, the instrument shall select one function to be turned on, and turn off any other functions that were on. The function which is selected to be on shall always be the same one, even if that function is not currently ON.

When CONCurrent is turned ON, the states of individual functions shall not be changed.

At \*RST the value of CONCurrent is device-dependent. However if an instrument implements CONCurrent ON, then it must implement CONCurrent OFF also for purposes of instrument compatibility.

**18.13.2.2 :OFF <sensor\_function>{,<sensor\_function>}**

SENSe:FUNCtion:OFF

The FUNCTION:OFF command selects the <sensor\_function>s to be turned off. This command only needs to be implemented by instruments which allow users to turn OFF individual functions while leaving others on. The state of any other functions are not affected by turning the specified functions OFF.

The query response of FUNCtion:OFF? shall return a comma separated list of functions which are currently OFF, each of which are <STRING RESPONSE DATA>. If no functions are OFF, a null string is returned. The query shall return the short form mnemonics and shall omit any default nodes in the <sensor\_function>.

At \*RST, this value is device-dependent.

**18.13.2.2.1 :ALL**

SENSe:FUNCtion:OFF:ALL

This command turns OFF all of the <sensor\_function>s which the instrument can concurrently sense. There is no query for this command.

**18.13.2.2.2 :COUNt?**

SENSe:FUNCtion:OFF:COUNT?

This query returns the number of <sensor\_function>s which are OFF. This command must be implemented if CONCurrent ON and FUNCtion:OFF are implemented.

This command is query-only and therefore has no \*RST condition.

**18.13.2.3 [:ON]<sensor\_function>{,<sensor\_function>}**

SENSe:FUNCtion:ON

The FUNCtion[:ON] command selects the <sensor\_function>(s) to be SENSeD by the instrument. This command only needs to be implemented by instruments which allow users to select what functions will be SENSeD. The <sensor\_function> is specified as a quoted string. For example:

FUNCtion "VOLTage:AC"

If CONCurrent is OFF, a single <sensor\_function> is sent as the parameter. This selects this function to be sensed. If more than one function is sent an error -108 (Parameter not allowed) shall be generated.

If CONCurrent is ON, a comma separated list of <sensor\_function> may be sent as parameters. These functions are turned on, while the state of other functions are not affected.

The query response of FUNCtion[:ON]? shall return a comma separated list of functions which are on, each of which are <STRING RESPONSE DATA>. If no functions are ON, a null string is returned. The query shall return the short form mnemonics and shall omit any default nodes in the <sensor\_function>.

For example FUNCtion[:ON]? returns the quoted string "VOLT:AC".

At \*RST, this value is device-dependent.

**18.13.2.3.1 :ALL**

SENSe:FUNCTION:ON:ALL

This command turns ON all of the <sensor\_function>s which the instrument can concurrently sense. There is no query for this command.

**18.13.2.3.2 :COUNT?**

SENSe:FUNCTION:ON:COUNT?

This query returns the number of <sensor\_function>s which are ON. This command must be implemented if CONCurrenT ON and FUNCTION[:ON] are implemented.

This command is query-only and therefore has no \*RST condition.

**18.13.2.4 :STATE? <sensor\_function>**

SENSe:FUNCTION:STATE?

Returns a Boolean indicating whether the specified <sensor\_function> is currently ON or OFF. This command is query only.

At \*RST, the state of each <sensor\_function> is device-dependent.

**18.13.2.5 <sensor\_function>**

SENSe:FUNCTION

The FUNCtion subsystem uses the <sensor\_function> to select the function on which the FUNCtion commands operate on, <sensor\_function> has the following syntax:

$$\begin{aligned} \langle \text{sensor\_function} \rangle ::= & " \langle \text{presentation\_layer} \rangle \langle \text{function\_name} \rangle \\ & [ \langle \text{input\_block} \rangle \{, \langle \text{input\_block} \rangle\} ] [ \text{ON } \langle \text{subnode} \rangle \{, \langle \text{subnode} \rangle\} ] \end{aligned}$$

The <sensor\_function> is STRING PROGRAM DATA.

The names inside angle brackets (<>) and (>) are called productions and the commas (,) and 'ON' are literals. White space is defined in section 7.4.1.2 of IEEE 488.2 as one or more white space characters.

For a <sensor\_function> sent to an instrument, white space is not allowed inside any production or literal. White space is not allowed between the <presentation\_layer> and <function\_name> productions. At least one white space character is required before and after the literal ON. At least one white space character is required before the first <input\_block>. White space is permitted, but not required, everywhere else in the <sensor\_function>. The FUNCtion commands shall accept the long and short forms of <sensor\_function> mnemonics. Instruments shall accept both upper and lowercase characters without distinguishing between cases.

When a <sensor\_function> is returned in response to a query, it shall contain no white space except for a single space before the first <input\_block> and single spaces around the literal ON if either is present. The mnemonics in the query response shall use short form with default nodes omitted. All alpha characters in the response shall be in uppercase.

<input\_block> elements are omitted in the query response only if all have been set to their default values. If any <input\_block> element has been set to anything other than its default value, then all <input\_block> elements are returned in the query response.

## 1999 SCPI Command Reference

The <function\_name> is a well-defined hierarchical sensor function indicating the type of function the sensor will be configured to directly detect.

The optional <input\_block> parameters are numeric values indicating which INPut blocks shall be “fed” into the sensor function. The numeric value corresponds to the same-numbered INPut block. Specifying an input block in a sensor function makes an implicit internal routing connection such that the output of the designated INPut block will flow into the sensor. If no <input\_block> parameters are specified the instrument defined default data flow is used.

For example, SENSe:FUNCtion “VOLTage:AC 2” would configure the sensor to sense the AC Voltage of the data flowing out of INPut2. The SENSe:FUNCtion? query would then return the quoted string “VOLT:AC 2”.

The optional <subnode> parameter is used when the SENSe block contains user configurable selections of subnodes to be used in computation of the SENSe FUNCtion. For example if a single SENSe block contained two SWEEP subblocks and the sensor could use either SWEEP block to measure the AC Voltage of INPut1. Currently SWEEP and AVERage are the only subnodes which are allowed after the ON keyword.

## 18.13.2.6 &lt;presentation\_layer&gt;

1991

**PRESENTATION LAYER**

[XNONE:]	1991
XTIME:	1991
XFREQUENCY:	1991
XPOWER:	1991
XVOLTAGЕ:	1991
XCURRENT:	1991

18.13.2.6.1 **[XNONE:]**

SENSe:FUNCTION "XNONE:&lt;function&gt;"

This specifies that the X-axis has no specific units or number of points. No knowledge can be associated with the X-axis. XNONE is only scalar.

18.13.2.6.2 **XTIME:**

SENSe:FUNCTION "XTIME:&lt;function&gt;"

This specifies that the sensor function results are acquired with respect to time. The X-axis is in units of time.

18.13.2.6.3 **XFREQUENCY:**

SENSe:FUNCTION "XFREQUENCY:&lt;function&gt;"

This specifies that the sensor function results are acquired with respect to frequency. The X-axis is in units of frequency.

18.13.2.6.4 **XPOWER:**

SENSe:FUNCTION "XPOWER:&lt;function&gt;"

This specifies that the sensor function results are acquired with respect to power. The X-axis is in units of power.

18.13.2.6.5 **XVOLTAGЕ:**

SENSe:FUNCTION "XVOLTAGЕ:&lt;function&gt;"

This specifies that the sensor function results are acquired with respect to voltage. The X-axis is in units of voltage.

18

18.13.2.6.6 **XCURRENT:**

SENSe:FUNCTION "XCURRENT:&lt;function&gt;"

This specifies that the sensor function results are acquired with respect to current. The X-axis is in units of current.

### 18.13.2.7 <function\_name>

SENSe:FUNCTION

A SENSe:FUNCTION <function\_name> is a well-defined hierarchical sensor function indicating the type of function the sensor will be configured to directly detect. The <function\_name> comprises the <function> and is optionally followed by a <function\_suffix>.

*<function\_name>* := <function>[<function\_suffix>]

Where: <function\_suffix> is of type <CHARACTER PROGRAM DATA> with the following values defined:

*<function\_suffix>* := RATio / SUM

When the RATio suffix is included, then a ratio between two input blocks shall be selected.

When the SUM suffix is included, then the sum of two or more input blocks shall be selected.

Excluding the <function\_suffix> shall cause the sensor function to be selected.

For example:

:SENSe:FUNCTION "FREQuency"

Selects frequency sensing.

:SENSe:FUNCTION "FREQuency:RATio"

Selects a ratio of two frequencies as a function. If the user specifies input blocks after the sensor function, then the first input block specified is the numerator and the second input block is the denominator.

:SENSe:FUNCTION "VOLTage:DC:SUM 1,2"

Selects the sum of DC voltages on inputs 1 and 2 as a function.

## 18.13.2.8 &lt;function&gt;

The following is the sensor <function> tree and descriptions:

<b>FUNCTION</b>	<b>DESCRIPTION</b>	
ACCeleration	Acceleration	1999
AM		1991
[:DEPTh]	AM Depth	1991
:DISTortion	AM Distortion	1991
:FREQuency	AM Frequency	1991
:SNDRatio	AM Signal, Noise & Distortion Ratio (SINAD)	1991
:SNR	AM Signal to Noise Ratio	1991
:THD	AM Total Harmonic Distortion	1991
CONCcentration	Non-time-aligned concentration	1999
:RAW	Raw concentration value in device dependent units	1999
:SDEViation	Standard Deviation corresponding to data point CONCcentration	1999
:TALign	Time-aligned analyzer for analyzer number (1,2,...n) in device-dependent units	1999
CONDITION	True/False condition of a signal	1993
CURRent		
[:DC]	DC Current	1999
:AC	AC Current	1999
DISTance	Distance	1999
FERRor	Force Error	1991
FM		
[:DEViation]	FM Deviation	1991
:DISTortion	FM Distortion	1991
:FREQuency	FM Frequency	1991
:SNDRatio	FM Signal, Noise & Distortion Ratio (SINAD)	1991
:SNR	FM Signal to Noise Ratio	1991
:THD	FM Total Harmonic Distortion	1991
FORCe	Force	1999
FREQuency	Direct sensing of frequency	
FRESistance	Four-wire resistance	1991
PERiod	Inverse of frequency	
PHASe	Direct sensing of phase	1992
PM		1991
[:DEViation]	PM Deviation	1991
:DISTortion	PM Distortion	1991
:FREQuency	PM Frequency	1991
:SNDRatio	PM Signal, Noise & Distortion Ratio (SINAD)	1991
:SNR	PM Signal to Noise Ratio	1991
:THD	PM Total Harmonic Distortion	1991
POWER		
:AC	AC Power	

## 1999 SCPI Command Reference

<b>FUNCTION</b>	<b>DESCRIPTION</b>	
:AChannel	Adjacent Channel Power	1991
:LOWER	Lower Adjacent Channel Power	1991
[:UPPer]	Upper Adjacent Channel Power	1991
:COHerence		1992
:CROSs	Cross Product	1992
[:DC]	DC Power	
:DISTortion	Distortion	1991
:PSDensity	Power Spectral Density	1992
:S11  S21  S22  S12	Scattering Parameter Function	1992
:SNDRatio	Signal, Noise & Distortion Ratio (SINAD)	1991
:SNR	Signal to Noise Ratio	1991
:THD	Total Harmonic Distortion	1991
PULM	Pulse Modulation	1994
RESistance	Two wire resistance	
SPEed	Speed	1999
:FRONt	Front Roll Speed	1999
[:REAR]	Rear Roll Speed	1999
SSB	Single Sideband	1994
TEMPerature	Temperature	1999
TIMer	Value of the timer (see :SYSTem:TIME:TIMer)	1999
TINTerval	Time Interval	1991
TOTalize	Totalize Events	1991
TPLoss	Target Parasitic Loss	1999
VOLTage		
:AC	AC Voltage	
:CDFunction	Cumulative Density Function	1992
[:DC]	DC Voltage	
:HISTogram	Histogram	1992
:PDFunction	Probability Density Function	1992

18

### 18.13.2.8.1 ACCeleration

SENSe:FUNCtion “ACCeleration”

Acceleration.

### 18.13.2.8.2 AM

SENSe:FUNCtion “AM”

Amplitude modulation.

### 18.13.2.8.2.1 [:DEPTh]

SENSe:FUNCtion “AM:DEPTh”

AM depth.

### 18.13.2.8.2.2 :DISTortion

SENSe:FUNCtion “AM:DISTortion”

DISTortion is the ratio of noise plus distortion to the signal plus noise plus distortion.

**18.13.2.8.2.3 :FREQuency**

SENSe:FUNCTION "AM:FREQuency"

AM frequency.

**18.13.2.8.2.4 :SNDRatio**

SENSe:FUNCTION "AM:SNDRatio"

Signal, Noise and Distortion Ratio (SINAD or SNDRatio) is the ratio of the unfiltered signal to the signal with its fundamental removed (ratio of signal plus noise plus distortion to the noise plus distortion).

$$\text{Signal to Noise Ratio (SNR)} = (\text{signal+noise}) / \text{noise}.$$

**18.13.2.8.2.5 :SNR**

SENSe:FUNCTION "AM:SNR"

AM signal to noise ratio.

**18.13.2.8.2.6 :THD**

SENSe:FUNCTION "AM:THD"

Total Harmonic Distortion (THD) is the ratio of the signal with its fundamental removed to the unfiltered signal.

**18.13.2.8.3 CONCetration**

SENSe:FUNCTION "CONCetration"

Non-time-aligned concentration.

**18.13.2.8.3.1 :RAW**

SENSe:FUNCTION "CONCetration:RAW"

Raw concentration value in device dependent units.

**18.13.2.8.3.2 :SDEViation**

SENSe:FUNCTION "CONCetration:SDEViation"

Standard Deviation corresponding to data point CONCetration.

**18.13.2.8.3.3 :TALign**

SENSe:FUNCTION "CONCetration:TALign"

Time-aligned analyzer for analyzer number (1,2,...n) in device-dependent units.

**18.13.2.8.4 CONDition**

SENSe:FUNCTION "CONDition"

True/False condition of a signal.

**18.13.2.8.5 CURRent**

SENSe:FUNCTION "CURRent"

**18.13.2.8.5.1 [:DC]**

SENSe:FUNCTION "CURRent:DC"

DC current.

**18.13.2.8.5.2 :AC**

SENSe:FUNCTION "CURRent:AC"

AC current.

### 18.13.2.8.6 DISTance

SENSe:FUNCtion "DISTance"

Distance. For chassis dynamometers this is the distance accumulated by the dynamometer.

### 18.13.2.8.7 FM

SENSe:FUNCtion "FM"

Frequency modulation.

### 18.13.2.8.7.1 :[DE]Viation

SENSe:FUNCtion "FM:DEViation"

FM Deviation.

### 18.13.2.8.7.2 :DISTortion

SENSe:FUNCtion "FM:DISTortion"

DISTortion is the ratio of noise plus distortion to the signal plus noise plus distortion.

### 18.13.2.8.7.3 :FREQuency

SENSe:FUNCtion "FM:FREQuency"

FM frequency.

### 18.13.2.8.7.4 :SNDRatio

SENSe:FUNCtion "FM:SNDRatio"

Signal, Noise and Distortion Ratio (SINAD or SNDRatio) is the ratio of the unfiltered signal to the signal with its fundamental removed (ratio of signal plus noise plus distortion to the noise plus distortion).

$$\text{Signal to Noise Ratio (SNR)} = (\text{signal} + \text{noise}) / \text{noise}.$$

### 18.13.2.8.7.5 :SNR

SENSe:FUNCtion "FM:SNR"

FM signal to noise ratio.

### 18.13.2.8.7.6 :THD

SENSe:FUNCtion "FM:THD"

Total Harmonic Distortion (THD) is the ratio of the signal with its fundamental removed to the unfiltered signal.

### 18.13.2.8.8 FERRor

SENSe:FUNCtion "FERRor"

Force Error. For chassis dynamometers this specifies the error inherent in the measurement of the real-time force.

### 18.13.2.8.9 FORCE

SENSe:FUNCtion "FORCe"

Force. For chassis dynamometers this is the force at the roll surface.

### 18.13.2.8.10 FREQuency

SENSe:FUNCtion "FREQuency"

Direct sensing of frequency.

**18.13.2.8.11 FRESistance**

SENSe:FUNCTION "FRESistance"

Four-wire resistance.

**18.13.2.8.12 PERiod**

SENSe:FUNCTION "PERiod"

Inverse of frequency.

**18.13.2.8.13 PHASE**

SENSe:FUNCTION "PHASE"

Direct sensing of phase.

**18.13.2.8.14 PM**

SENSe:FUNCTION "PM"

Pulse modulation.

**18.13.2.8.14.1 [:DEViation]**

SENSe:FUNCTION "PM:DEViation"

PM deviation.

**18.13.2.8.14.2 :DISTortion**

SENSe:FUNCTION "PM:DISTortion"

DISTortion is the ratio of noise plus distortion to the signal plus noise plus distortion.

**18.13.2.8.14.3 :FREQuency**

SENSe:FUNCTION "PM:FREQuency"

PM Frequency.

**18.13.2.8.14.4 :SNDRatio**

SENSe:FUNCTION "PM:SNDRatio"

Signal, Noise and Distortion Ratio (SINAD or SNDRatio) is the ratio of the unfiltered signal to the signal with its fundamental removed (ratio of signal plus noise plus distortion to the noise plus distortion).

$$\text{Signal to Noise Ratio (SNR)} = (\text{signal} + \text{noise}) / \text{noise}.$$

**18.13.2.8.14.5 :SNR**

SENSe:FUNCTION "PM:SNR"

PM signal to noise ratio.

**18.13.2.8.14.6 :THD**

SENSe:FUNCTION "PM:THD"

PM Total Harmonic Distortion (THD) is the ratio of the signal with its fundamental removed to the unfiltered signal.

**18.13.2.8.15 POWer**

SENSe:FUNCTION "POWER"

**18.13.2.8.15.1 :AC**

SENSe:FUNCtion "POWER:AC"

AC Power.

**18.13.2.8.15.2 :ACHannel**

SENSe:FUNCtion "POWER:ACHannel"

Adjacent Channel Power measurements involve measuring the power at a specified carrier for a given bandwidth and then ratio it to the power detected at the upper (or lower) adjacent channel.

**18.13.2.8.15.2.1 :LOWER**

SENSe:FUNCtion "POWER:ACHannel:LOWER"

Lower Adjacent Channel Power.

**18.13.2.8.15.2.2 [:UPPer]**

SENSe:FUNCtion "POWER:ACHannel:UPPer"

Upper Adjacent Channel Power.

**18.13.2.8.15.3 :COHerence**

SENSe:FUNCtion "POWER:COHerence"

COHerence is the portion of the first <input\_block> spectrum related to the second <input\_block> spectrum. It is a measure of the statistical validity of a frequency response measurement.

**18.13.2.8.15.4 :CROSs**

SENSe:FUNCtion "POWER:CROSS"

This is the Cross Product of two input\_blocks. It is computed by multiplying the complex conjugate of the first <input\_block> by the second <input\_block>.

**18.13.2.8.15.5 [:DC]**

SENSe:FUNCtion "POWER:DC"

DC Power.

**18.13.2.8.15.6 :DISTortion**

SENSe:FUNCtion "POWER:DISTortion"

DISTortion is the ratio of noise plus distortion to the signal plus noise plus distortion.

**18.13.2.8.15.7 :PSDensity**

SENSe:FUNCtion "POWER:PSDensity"

This is the Power Spectral Density. It is the power spectrum normalized to a 1 Hz bandwidth.

**18.13.2.8.15.8 :S11|:S12|:S22|:S21**

SENSe:FUNCtion "POWER:S11"

These are the Scattering Parameter Functions.

Scattering parameters or S parameters are a tool used in frequency characterization of devices to determine how the device modifies signal flow. An S parameter is a ratio of two energy values and is defined as:

$$S_{xy} = \frac{\text{energy emerging from port } x}{\text{energy entering port } y}$$

with all ports other than port Y terminated in a Z0 load.

By convention, a devices ports are numbered (1,2,...n). An n port device has  $n^2$  S parameters labelled  $S_{xy}$ , where  $\{x,y\} = \{1,2,\dots,n\}$ .

For more information please refer to “Scattering and Transmission Coefficients” chapter in “Fields and Waves in Communication Electronics”, Ramo, Whinnery and Van Duzer.

#### 18.13.2.8.15.9 :SNDRatio

SENSe:FUNCTION “POWER:SNDRatio”

Signal, Noise and Distortion Ratio (SINAD or SNDRatio) is the ratio of the unfiltered signal to the signal with its fundamental removed (ratio of signal plus noise plus distortion to the noise plus distortion).

$$\text{Signal to Noise Ratio (SNR)} = (\text{signal+noise}) / \text{noise}.$$

#### 18.13.2.8.15.10 :SNR

SENSe:FUNCTION “POWER:SNR”

Signal to Noise Ratio.

#### 18.13.2.8.15.11 :THD

SENSe:FUNCTION “POWER:THD”

Total Harmonic Distortion (THD) is the ratio of the signal with its fundamental removed to the unfiltered signal.

#### 18.13.2.8.16 PULM

SENSe:FUNCTION “PULM”

Pulse modulation.

#### 18.13.2.8.17 RESistance

SENSe:FUNCTION “RESistance”

Two wire resistance.

#### 18.13.2.8.18 SPEed

SENSe:FUNCTION “SPEed”

Speed.

#### 18.13.2.8.18.1 :FRONt

SENSe:FUNCTION “SPEed:FRONT”

Front Roll Speed. For chassis dynamometers this is the roll speed. On twin roll systems, this is the front roll speed.

#### 18.13.2.8.18.2 [:REAR]

SENSe:FUNCTION “SPEed[:REAR]”

Rear Roll Speed. For chassis dynamometers this is the roll speed. On twin roll systems, this is the rear roll speed.

### 18.13.2.8.19 **SSB**

SENSe:FUNCtion "SSB"

Single sideband modulation.

### 18.13.2.8.20 **TEMPerature**

SENSe:FUNCtion "TEMPerature"

Temperature.

### 18.13.2.8.21 **TIMer**

SENSe:FUNCtion "TIMer"

Value of the timer (see :SYSTem:TIME:TIMer).

### 18.13.2.8.21.1 **COUNT**

SENSe:FUNCtion "TIMer:COUNT"

Returns the current timer count.

### 18.13.2.8.22 **TINTERval**

SENSe:FUNCtion "TINTERval"

Time Interval.

### 18.13.2.8.23 **TOTALize**

SENSe:FUNCtion "TOTALize"

Totalize events.

### 18.13.2.8.24 **TPLoss**

SENSe:FUNCtion "TPLoss"

Target Parasitic Loss. For chassis dynamometers this is the value of frictional loss at the present speed based on the active parasitic loss curve.

### 18.13.2.8.25 **VOLTage**

SENSe:FUNCtion "VOLTage"

#### 18.13.2.8.25.1 **:AC**

SENSe:FUNCtion "VOLTage:AC"

AC Voltage.

18

#### 18.13.2.8.25.2 **:CDFunction**

SENSe:FUNCtion "VOLTage:CDFunction"

The Cumulative Density Function is the integral of a normalized histogram.

#### 18.13.2.8.25.3 **[:DC]**

SENSe:FUNCtion "VOLTage:DC"

DC Voltage.

#### 18.13.2.8.25.4 **:HISTogram**

SENSe:FUNCtion "VOLTage:HISTogram"

The histogram measures how the amplitude of the input signal is distributed between its maximum and minimum values.

### 18.13.2.8.25.5 :PDFunction

SENSe:FUNCtion "VOLTage:PDFunction"

The Probability Density Function is computed by normalizing a histogram and is a statistical measure of the probability that a specific level occurred.

**18.14 LIST Subsystem**

SENSe:LIST

The list subsystem controls automatic sequencing through associated lists of frequency, and dwell time values. The various values are specified in the LIST:FREQuency, LIST:DWEli commands. Points in each list correspond to the associated point in all other lists.

The LIST subsystem controls a characteristic of the instrument when that subsystem's mode is set to LIST (for example, FREQuency:MODE LIST).

When employing multiple lists, all lists must be of equal length. If they are not of the same length, an error is generated at sequencing time. An exception is made for the case where the shorter list is of length 1. In this case, the particular list is treated as though it were a list of equal length, with all values being the same.

Lists are not affected by \*RST.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:LIST		
:COUNt <numeric_value>		
:DIRection UP DOWN		
:DWEli <numeric_value>{,<numeric_value>}		
:POINts?		[query only]
:FREQuency <numeric_value>{,<numeric_value>}		
:POINts?		[query only]
:SEQUence <numeric_value>{,<numeric_value>}		
:AUTO <Boolean> ONCE		
:POINts?		[query only]

**18.14.1 :COUNt <numeric\_value>**

SENSe:LIST:COUNt

Controls the number of times the sequence list is scanned when a trigger is received.

**18.14.2 :DIRection UP|DOWN**

SENSe:LIST:DIRection

Specifies the direction that the sequence list is scanned. If UP is selected, the list is scanned from the first to the last item in the sequence list. If DOWN is selected, the list is scanned from the last item to the first item in the sequence list.

**18.14.3 :DWEli <numeric\_value>{,<numeric\_value>}**

SENSe:LIST:DWEli

Specifies the dwell time points of the lists. The units are in seconds.

This specifies the amount of time spent at each point.

**18.14.3.1 :POINts?**

SENSe:LIST:DWEli:POINts?

Returns the number of points currently in the DWEli list.

### 18.14.4 :FREQuency <numeric\_value>{,<numeric\_value>}

SENSe:LIST:FREQuency

Specifies the frequency points of the list set. The units are Hz.

#### 18.14.4.1 :POINts?

SENSe:LIST:FREQuency:POINts?

This query returns the number of points currently in the FREQuency list.

#### 18.14.5 :SEQuence <numeric\_value>{,<numeric\_value>}

SENSe:LIST:SEQuence

Defines a sequence for stepping through the list. Individual points may be specified as many times as desired in a single sequence. The points specified are indexes into the lists. For example, if 3 was selected, the third point in the frequency and dwell lists would be sequenced.

#### 18.14.5.1 :AUTO <Boolean>|ONCE

SENSe:LIST:SEQuence:AUTO

When on, the sequence list is set to 1 through N, where N is the longest list.

#### 18.14.5.2 :POINts?

SENSe:LIST:SEQuence:POINts?

This query returns the number of points currently in the SEQuence list.

**18.15 MIXer Subsystem**

SENSe:MIXer

This subsystem controls external mixers for tuned receivers. It also provides controls to compensate for known fixed losses in external mixers.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:MIXer		
:BIAS	<numeric_value>	
:AUTO	<Boolean> ONCE	
:LIMit	<numeric_value>	
:HARMonic	<numeric_value>	
:AUTO	<Boolean> ONCE	
:LOSS	<numeric_value>	
:AUTO	<Boolean>	1991

**18.15.1 :BIAS <numeric\_value>**

SENSe:MIXer:BIAS

Controls the mixer bias.

At \*RST, this value is device-dependent.

**18.15.1.1 :AUTO <Boolean>|ONCE**

SENSe:MIXer:BIAS:AUTO

Allows the mixer bias to be automatically set by other parameters in the system. Explicitly setting a value for BIAS shall cause AUTO to be set to OFF.

At \*RST, this value shall be set to ON.

**18.15.1.2 :LIMit <numeric\_value>**

SENSe:MIXer:BIAS:LIMit

Controls the maximum mixer bias level which can be set by the BIAS command or by other parameters when AUTO is ON.

The limits and \*RST conditions are instrument-dependent.

**18.15.2 :HARMonic <numeric\_value>**

SENSe:MIXer:HARMonic

Selects which harmonic of the local oscillator will be used for mixing.

At \*RST, this value is device-dependent.

**18.15.2.1 :AUTO <Boolean>|ONCE**

SENSe:MIXer:HARMonic:AUTO

When AUTO is ON, the harmonic is chosen by the system or instrument, and may change during the sweep. When OFF, the harmonic is locked to the selected value. Setting a value for MIXer:HARMonic sets AUTO to OFF.

At \*RST, this value is set to ON.

### 18.15.3 :LOSS <numeric\_value>

SENSe:MIxer:LOSS

Compensates for losses external to the instrument. This parameter is added to all readings.  
The unit for this command is the current relative amplitude unit.

At \*RST, this value is set to 0.

#### 18.15.3.1 :AUTO <Boolean>

SENSe:MIxer:LOSS:AUTO

Sets the LOSS to the value appropriate to the connected external mixer.

At \*RST, this value is device-dependent.

**18.16 PM Subsystem**

SENSe:PM

The Phase Modulation subsystem is used to set the modulation controls and parameters associated with sensing phase modulated signals.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:PM		1991
[:DEViation]		1991
:RANGE		1991
:AUTO	<Boolean>   ONCE	1991
[:UPPer]	<numeric_value>	1991
:LOWER	<numeric_value>	1991

**18.16.1 [:DEViation]**

SENSe:PM:DEViation

This node is used to collect together the deviation parameters.

**18.16.1.1 :RANGE**

SENSe:PM:DEViation:RANGE

Sets the range for the sensor function.

**18.16.1.1.1 :AUTO <Boolean> | ONCE**

SENSe:PM:DEViation:RANGE:AUTO

Sets the RANGE to the value determined to give the most dynamic range without overloading. The actual algorithm is device-dependent. Selecting AUTO ONCE will have the effect of setting AUTO to ON and then OFF.

At \*RST, this value is set to ON.

**18.16.1.1.2 [:UPPer] <numeric\_value>**

SENSe:PM:DEViation:RANGE:UPPer

Specifies the maximum signal level expected for the sensor input.

At \*RST, this value is device-dependent.

**18.16.1.1.3 :LOWER <numeric\_value>**

SENSe:PM:DEViation:RANGE:LOWER

Specifies the smallest signal level expected for the sensor input. This allows setting for the best dynamic range. This function is coupled to UPPer.

At \*RST, this value is device-dependent.

**18.17 POWER Subsystem**

SENSe:POWer

This subsection controls the power amplitude characteristics of the sensor.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:POWER		
:ACHannel		1991
:SPACing		1991
:LOWER	<numeric_value>	1991
:AUTO	<Boolean>	1991
[:UPPer]	<numeric_value>	1991
:AC[:DC]		
:APERture	<numeric_value>	
:NPLCycles	<numeric_value>	1991
:ATTenuation	<numeric_value>	
:AUTO	<Boolean>	
:PROTection		
[:LEVel]	<numeric_value>	
:STATe		
:TRIPped?		[query only]
:CLEar		[no query]
:RANGE		
[:UPPer]	<numeric_value>	
:LOWER	<numeric_value>	
:AUTO	<Boolean> ONCE	
:DIRection	UP DOWN EITHER	
:LLIMit	<numeric_value>	1996
:ULIMit	<numeric_value>	1996
:OFFSet		1991
:PTPeak		1991
:REFerence	<numeric_value>	
:STATe	<Boolean>	
:RESolution	<numeric_value>	
:AUTO	<Boolean> ONCE	
:DETector	INTernal   EXTernal	1991

**18.17.1 :ACHannel**

SENSe:POWer:ACHannel

Adjacent Channel Power Measurements involve measuring the power at a specified carrier for a given bandwidth and then ratio it to the power detected at the upper (or lower) adjacent channel.

**18.17.1.1 :SPACing**

SENSe:POWer:ACHannel:SPACing

Controls the channel SPACing which is the distance from the modulated carrier signal to the upper or lower Adjacent Channel.

### 18.17.1.1.1 :LOWER <numeric\_value>

SENSe:POWer:ACHannel:SPACing:LOWER

Controls the channel SPACing to the lower Adjacent Channel.

At \*RST, this value is device-dependent.

### 18.17.1.1.1.1 :AUTO <Boolean>

SENSe:POWer:ACHannel:SPACing:LOWER:AUTO

When AUTO is ON, the value of LOWER is coupled to the value of UPPER.

### 18.17.1.1.2 [:UPPer] <numeric\_value>

SENSe:POWer:ACHannel:SPACing:UPPer

Controls the channel SPACing which is the distance from the modulated carrier signal to the upper Adjacent Channel.

At \*RST, this value is device-dependent.

## 18.17.2 :AC[:DC]

SENSe:POWer:AC

This subsystem selects the alternating current or the direct current sensor. The default units are watts.

### 18.17.2.1 :APERture <numeric\_value>

SENSe:POWer:AC:APERture

Specifies the acquisition/sampling/gate time for a single measurement point. The parameter has a unit of seconds.

At \*RST, this value is device-dependent.

### 18.17.2.2 :NPLCycles <numeric\_value>

SENSe:POWer:AC:NPLCycles

Specifies the acquisition/sampling/gate time for a single measurement point in terms of the number of power line cycles. The parameter is unitless. If this command is implemented, the APERture command must also be implemented.

The value is coupled to APERture by the equation:

$$\text{APERture} = \text{NPLCycles}/\text{selected line frequency}$$

At \*RST, this value is device-dependent.

### 18.17.2.3 :ATTenuation <numeric\_value>

SENSe:POWer:AC:ATTenuation

Sets the ATTenuation level. Note that when setting the level to 10 dB the magnitude of the incoming signal will be decreased by 10 dB. Changing the ATTenuation changes RANGe by the same amount.

Default units are as determined in the UNITS system.

**18.17.2.3.1 :AUTO <Boolean>**

SENSe:POWer:AC:ATTenuation:AUTO

Couples the attenuator to RANGe such that maximum dynamic range of the incoming signal is assured without overloading.

Programming a specified attenuation sets AUTO OFF.

At \*RST, AUTO is set to ON.

**18.17.2.4 :PROTection**

SENSe:POWer:AC:PROTection

Controls the protection circuits.

**18.17.2.4.1 [:LEVel] <numeric\_value>**

SENSe:POWer:AC:PROTection:LEVel

This command sets the input level at which the input protection circuit will trip.

**18.17.2.4.2 :STATe <Boolean>**

SENSe:POWer:AC:PROTection:STATe

Controls whether the input protection circuit is enabled.

At \*RST, this value is set to ON.

**18.17.2.4.3 :TRIPped?**

SENSe:POWer:AC:PROTection:TRIPped?

This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped. TRIPped only has a query form and thus has no associated \*RST condition.

**18.17.2.4.4 :CLEar**

SENSe:POWer:AC:PROTection:CLEar

Causes the protection circuit to be cleared.

This command is an event and has no associated \*RST condition.

**18.17.2.5 :RANGe**

SENSe:POWer:AC:RANGE

Sets the range for the amplitude sensor function. The RANGe endpoints may be specified in terms of the UPPer endpoint and the LOWER endpoint, alternatively the midpoint of the RANGe (OFFSet) and the span of the RANGe (PTPeak) may be used. A non-symmetrical range around zero may cause some instruments to introduce an actual offset to the signal entering the SENSe block.

**18.17.2.5.1 [:UPPer] <numeric\_value>**

SENSe:POWer:AC:RANGE:UPPer

Specifies the most positive signal level expected for the sensor input.

At \*RST, this value is device-dependent.

**18.17.2.5.2 :LOWER <numeric\_value>**

SENSe:POWER:AC:RANGE:LOWER

Specifies the most negative signal level expected for the sensor input. This allows setting for the best dynamic range. This function is coupled to UPPer in a device-dependent manner.

At \*RST, this value is device-dependent.

**18.17.2.5.3 :AUTO <Boolean>|ONCE**

SENSe:POWER:AC:RANGE:AUTO

Sets the RANGE to the value determined to give the most dynamic range without overloading. The actual algorithm is device-dependent.

At \*RST, AUTO is set to ON.

**18.17.2.5.3.1 :DIRection UP|DOWN|EITHER**

SENSe:POWER:AC:RANGE:AUTO:DIRection

The DIRection command defines the manner in which AUTO works. When UP is selected, the instrument shall only change the range automatically to ranges larger than the current range, as the input signal dictates. When DOWN is selected, the instrument shall only change the range automatically to ranges smaller than the current range, as the input signal dictates. When EITHER is selected the instrument may range in either direction.

At \*RST, DIRection shall be set to EITHER.

**18.17.2.5.3.2 :LLIMit <numeric\_value>**

SENSe:POWER:AC:RANGE:AUTO:LLIMit

The Lower LIMit command sets the smallest range to which the instrument will go while autoranging. When AUTO is ON and the present RANGE is outside the new limits, the instrument may change the RANGE to a value inside the limits or wait until the next time the autoranging algorithm is performed.

At \*RST, LLIMit shall be set to MINimum.

**18.17.2.5.3.3 :ULIMit <numeric\_value>**

SENSe:POWER:AC:RANGE:AUTO:ULIMit

The Upper LIMit command sets the largest range to which the instrument will go while autoranging. When AUTO is ON and the present RANGE is outside the new limits, the instrument may change the RANGE to a value inside the limits or wait until the next time the autoranging algorithm is performed.

At \*RST, ULIMit shall be set to MAXimum.

**18.17.2.5.4 :OFFSet <numeric\_value>**

SENSe:POWER:AC:RANGE:OFFSet

OFFSet determines the midpoint of the range.

At \*RST, this value is device-dependent.

**18.17.2.5.5 :PTPeak <numeric\_value>**

SENSe:POWER:AC:RANGE:PTPeak

Peak To Peak specifies the dynamic range required for the sensor.

At \*RST, this value is device-dependent.

#### 18.17.2.6 :REFerence <numeric\_value>

SENSe:POWer:AC:REFerence

Sets a reference amplitude for sensor instruments. This command differs in intent from the reference correction in that this is intended to be a measurement offset where a user wishes to reference a measurement to some known value.

At \*RST, this value is device-dependent, but STATe is set to OFF.

#### 18.17.2.6.1 :STATe <Boolean>

SENSe:POWer:AC:REFerence:STATe

Determines whether amplitude is measured in absolute or relative mode. If STATe is ON, then amplitude is referenced to the value set in REFerence.

At \*RST, this value is set to OFF.

#### 18.17.2.7 :RESolution <numeric\_value>

SENSe:POWer:AC:RESolution

Specifies the absolute resolution of the measurement. For example, if a measurement of 10 Volts is specified with a resolution of .01 Volts, the measurement is returned with a resolution of .01 Volts (that is, 10.00).

RANGe is not coupled to RESolution. Setting RANGe shall not cause RESolution to change. Setting RESolution shall not cause RANGe to change.

At \*RST, this value is device-dependent.

#### 18.17.2.7.1 :AUTO <Boolean>|ONCE

SENSe:POWer:AC:RESolution:AUTO

Allows the system to determine the best resolution for the other measurement conditions.

Explicitly setting a value for :RESolution will turn AUTO OFF.

At \*RST, this value is set to ON.

#### 18.17.3 :DETector INTERNAL | EXTERNAL

SENSe:POWer:DETector

The DETector can be specified as internal or external.

At \*RST, this value is set to INT.

**18.18 RESistance|FRESistance Subsystem**

SENSe:RESistance

This subsection controls the signal characteristics of a resistance sensor. RESistance should be used for a two-wire measurement. FRESistance (four-wire resistance) should be used for a four-wire measurement. The default units are ohms.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:RESistance		
:FRESistance		1991
:APERture	<numeric_value>	
:NPLCycles	<numeric_value>	1991
:OCOMPensated	<Boolean>	1991
:RANGE		
[:UPPer]	<numeric_value>	
:LOWER	<numeric_value>	
:AUTO	<Boolean> ONCE	
:DIRection	UP DOWN EITHER	
:LLIMit	<numeric_value>	1996
:ULIMit	<numeric_value>	1996
:REFERENCE	<numeric_value>	
:STATe	<Boolean>	
:RESolution	<numeric_value>	
:AUTO	<Boolean> ONCE	

**18.18.1 :APERture <numeric\_value>**

SENSe:RESistance:APERture

Specifies the acquisition/sampling/gate time for a single measurement point. The parameter has a unit of seconds.

At \*RST, this value is device-dependent.

**18.18.2 :NPLCycles <numeric\_value>**

SENSe:RESistance:NPLCycles

Specifies the acquisition/sampling/gate time for a single measurement point in terms of the number of power line cycles. The parameter is unitless. If this command is implemented, the APERture command must also be implemented.

The value is coupled to APERture by the equation:

$$\text{APERture} = \text{NPLCycles}/\text{selected line frequency}$$

At \*RST, this value is device-dependent.

**18.18.3 :OCOMPensated <Boolean>**

SENSe:RESistance:OCOMPensated

Enables or disables the offset compensation when measuring resistance. (For example, offset compensation first measures the voltage on the input terminals before turning on the current used to measure resistance.)

At \*RST, OCOMPensated shall be set to OFF.

**18.18.4 :RANGE**

SENSe:RESistance:RANGE

Sets the range for the resistance sensor.

**18.18.4.1 [:UPPer] <numeric\_value>**

SENSe:RESistance:RANGE:UPPer

Specifies the maximum resistance expected for the sensor input.

At \*RST, this value is device-dependent.

**18.18.4.2 :LOWER <numeric\_value>**

SENSe:RESistance:RANGE:LOWER

Specifies the smallest resistance expected for the sensor input. This allows setting for the best dynamic range. This function is coupled to the UPPer command in a device-dependent manner.

At \*RST, this value is device-dependent.

**18.18.4.3 :AUTO <Boolean>|ONCE**

SENSe:RESistance:RANGE:AUTO

Sets the RANGE to the value determined to give the most dynamic range without over loading. The actual algorithm is device-dependent.

At \*RST, AUTO is set to ON.

**18.18.4.3.1 :DIRection UP|DOWN|EITHer**

SENSe:RESistance:RANGE:AUTO:DIRection

The DIRection command defines the manner in which AUTO works. When UP is selected, the instrument shall only change the range automatically to ranges larger than the current range, as the input signal dictates. When DOWN is selected, the instrument shall only change the range automatically to ranges smaller than the current range, as the input resistance dictates. When EITHer is selected, the instrument may range in either direction.

AT \*RST, DIRection shall be set to EITHer.

**18.18.4.3.2 :LLIMit <numeric\_value>**

SENSe:RESistance:RANGE:AUTO:LLIMit

The Lower LIMit command sets the smallest range to which the instrument will go while autoranging. When AUTO is ON and the present RANGE is outside the new limits, the instrument may change the RANGE to a value inside the limits or wait until the next time the autoranging algorithm is performed.

At \*RST, LLIMit shall be set to MINimum.

**18.18.4.3.3 :ULIMit <numeric\_value>**

SENSe:RESistance:RANGE:AUTO:ULIMit

The Upper LIMit command sets the largest range to which the instrument will go while autoranging. When AUTO is ON and the present RANGE is outside the new limits, the instrument may change the RANGE to a value inside the limits or wait until the next time the autoranging algorithm is performed.

## 1999 SCPI Command Reference

At \*RST, ULIMit shall be set to MAXimum.

### 18.18.5 :REFerence <numeric\_value>

SENSe:RESistance:REFerence

Sets a reference resistance for sensor instruments. This command differs in intent from the reference correction in that this is intended to be a measurement offset where a user wishes to reference a measurement to some known value.

At \*RST, this value is device-dependent, but STATe is set to OFF.

#### 18.18.5.1 :STATe <Boolean>

SENSe:RESistance:REFerence:STATe

Determines whether resistance is measured in absolute or relative mode. If STATe is ON, then resistance is referenced to the value set in REFerence.

At \*RST, this value is set to OFF.

### 18.18.6 :RESolution <numeric\_value>

SENSe:RESistance:RESolution

Specifies the absolute resolution of the measurement. For example, if a measurement of 10 Ohms is made with a resolution of .01, the measurement is returned with a resolution of .01 Ohms (that is, 10.00).

At \*RST, this value is device-dependent.

#### 18.18.6.1 :AUTO <Boolean>|ONCE

SENSe:RESistance:RESolution:AUTO

Allows the system to determine the best resolution for the other measurement conditions.

Explicitly setting a value for RESolution will turn AUTO OFF.

At \*RST, this value is set to ON.

**18.19 ROSCillator Subsystem**

SENSe:ROSCillator

This subsystem controls the frequency reference oscillator.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:ROSCillator		
[:INTERNAL]		1992
:FREQuency	<numeric_value>	
:EXTERNAL		1992
:FREQuency	<numeric_value>	1992
:SOURce	INTERNAL EXTERNAL NONE  CLK10 CLK100	
:AUTO	<Boolean> ONCE	1991

**18.19.1 [:INTERNAL]**

SENSe:ROSCillator:INTERNAL

This subsystem configures the internal reference oscillator(s).

**18.19.1.1 :FREQuency <numeric\_value>**

SENSe:ROSCillator:INTERNAL:FREQuency

Specifies the frequency of the internal reference oscillator. The default units are Hz.

At \*RST, this value is device-dependent.

**18.19.2 :EXTERNAL**

SENSe:ROSCillator:EXTERNAL

This subsystem configures the external reference oscillator(s).

**18.19.2.1 :FREQuency <numeric\_value>**

SENSe:ROSCillator:EXTERNAL:FREQuency

Specifies the frequency of the external reference oscillator. The default units are Hz.

At \*RST, this value is device-dependent.

**18.19.3 :SOURce INTERNAL|EXTERNAL|NONE|CLK10|CLK100**

SENSe:ROSCillator:SOURce

Controls the selection of the reference oscillator source. This typically refers to a precision, stabilized time base. The parameter values have the following meanings:

- INTERNAL — The reference frequency is derived from an internal precision oscillator.
- EXTERNAL — The reference frequency is derived from an external signal supplied to the device.
- NONE — The reference frequency is not derived from any precision oscillator.
- CLK10 — The reference frequency is derived from VXIbus signal CLK10.
- CLK100 — The reference frequency is derived from VXIbus signal CLK100.

At \*RST, this value is device-dependent.

### 18.19.3.1 :AUTO <Boolean>|ONCE

SENSe:ROSCillator:SOURce:AUTO

When AUTO is ON, the system automatically selects which oscillator will be used by the instrument.

Explicitly selecting a reference oscillator turns AUTO OFF.

At \*RST, AUTO is set to ON.

**18.20 SMOothing Subsystem**

SENSe:SMOothing

This subsystem controls the point-to-point smoothing of the signal. Smoothing differs from averaging in that with smoothing, adjacent points of one set of data are averaged, while in averaging the average is taken over several acquisitions (that is, smoothing is a moving average). This subsystem controls smoothing done to compensate for measurement artifacts. Analytical post-acquisition smoothing is controlled with CALCulate:SMOothing.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
SMOothing		
[:STATE]	<Boolean>	
:APERture	<numeric_value>	
:POINts	<numeric_value>	

**18.20.1 [:STATE] <Boolean>**

SENSe:SMOothing:STATe

Determines whether the smoothing algorithm is enabled.

At \*RST, this function is set to OFF.

**18.20.2 :APERture <numeric\_value>**

SENSe:SMOothing:APERture

Specifies the size of the smoothing APERture as a ratio of smoothing window points/trace points. POINts is coupled to APERture by the equation:

$$POINts = APERture * SWEEP:POINts$$

For example, if an APERture of .01 were chosen for a 1000-point sweep, then the smoothing window would be 10 points wide.

At \*RST, this value is device-dependent.

**18.20.3 :POINts <numeric\_value>**

SENSe:SMOothing:POINts

Controls the number of points to be included in the running average. POINts is coupled to APERture by the equation:

$$POINts = APERture * SWEEP:POINts$$

At \*RST, this value is device-dependent.

### 18.21 SSB Subsystem

SENSe:SSB

The single sideband modulation subsystem is used to set up the modulation controls and parameters associated with sensing and demodulating single sideband modulated signals.

#### 18.21.1 :TYPE USB|LSB|A1

SENSe:SSB:TYPE

SENSe:SSB:TYPE sets or queries the type of SSB demodulation technique the demodulator uses. USB means the demodulator is configured to upper sideband signals. LSB means the demodulator is configured to lower sideband signals. A1 means the demodulator is configured to A1 signals.

At \*RST, this value is device-dependent.

## 18.22 STABilize Subsystem

SENSe:STABilize

This subsystem contains the commands that control the stability parameters for the designated instrument and range.

KEYWORD	PARAMETER FORM	NOTES
:STABilize		1999
:NTOLerance	<numeric_value>	1999
[:STATE]	<Boolean>	1999
:TIME<n>	<numeric_value>	1999

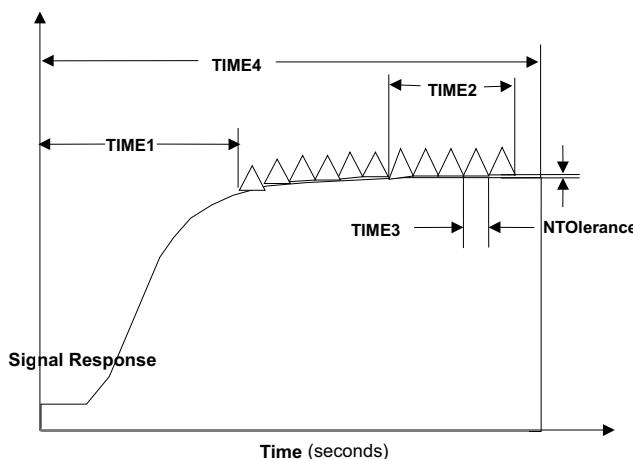


Figure 18.1 Stabilization Use

### 18.22.1 :NTOLerance <numeric\_value>

:SENSe:STABilize:NTOLerance

This command specifies or returns the allowable tolerance between averaged readings for a stabilized read, in percentage of full scale (%FS).

At \*RST these values are not changed.

### 18.22.2 [:STATE] < Boolean >

:SENSe:STABilize[:STATe]

This command sets or queries the stabilization state for all sensors. If stabilization is enabled, a procedure involving use of the stabilization times TIME1, TIME2, TIME3, and TIME4 and the stabilization tolerance is invoked in response to commands such as :SENSe:DATA?, and :SENSe:CORRection:ZERO:ACQuire. An error in stabilization shall result in error #-365.

At \*RST, stabilization is OFF.

### 18.22.3 :TIME<n> <numeric\_value>

:SENSe:STABilize:TIME<n>

This command sets the nth time parameter for the stabilization procedure, for the selected instrument(s).

The numeric suffix of :TIME, which is NOT optional, is defined as follows:

TIME1: Stabilized read initial rise or delay time.

TIME2: Stabilized read period, consisting of a number of TIME3's, over which stabilization is considered.

TIME3: Stabilized read duration of a single averaging period.

TIME4: Stabilized read maximum time of stabilization procedure.

At \*RST these values are not changed.

18.23

**SWEEp Subsystem**

SENSe:SWEEp

This subsystem controls the timebase and/or sweep generator in instruments that acquire data either directly or indirectly as a function of time.

Examples of instruments that acquire data directly as a function of time (non-swept instruments) are digitizers and logic analyzers. The X axis has units of time. The <sensor\_function> presentation layer is :XTIME.

Examples of instruments that acquire data indirectly as a function of time (swept instruments) are spectrum analyzers. In this case, the X axis has units of frequency, but the frequency is swept with relationship to time. The <sensor\_function> presentation layer is :XFREQuency. The parameter to be swept is specified by setting the :MODE node in the associated subsystem to SWEEP (FREQuency:MODE SWEEp).

Some commands in this subsystem are applicable to both swept and non-swept instruments. Others may apply to only one of these classes.

Commands in this subsystem may be coupled to those in implementor-defined trigger sequences such as TRIGger:SAMPle. The TRIGger sequences can provide additional flexibility by allowing the user more precise control over the acquisition.

The commands in the :SWEEp subsystem specify the timing and spacing of the values returned in :DATA?, FETCh?, and other measurement queries. The commands in the TRIGger subsystem specify when actual measurements are acquired. The precise relationship between the parameters in the two subsystems is device dependent. For example, the :SWEEp:TINTerval command specifies the timing period between two successive samples of the SENSe:DATA? response. The TRIGger:SAMPLE:TIMER command specifies the timing period between two acquisition samples that are taken by the trigger subsystem.

The :SWEEp subsystem contains several sets of commands which have complex couplings. These include the commands :TIME, :POINTS, :DWELl, and TINTerval. Sending any one of a set singly will affect others of the set. However, if two are sent in a single program message, then these two shall set the values specified and the other changed. This is in accordance with the style guidelines and IEEE 488.2. See Syntax and Style section 7.4 for additional rules.

Default units in this subsystem are seconds unless otherwise specified.

KEYWORD	PARAMETER FORM	NOTES
:SWEEp		
:COUNt	<numeric_value>	
:DIRection	UP DOWN	
:DWELl	<numeric_value>	
:AUTO	<Boolean> ONCE	
:GENeration	STEPped ANALog	
:MODE	AUTO MANual	
:OFFSet		

KEYWORD	PARAMETER FORM	NOTES
:POINts	<numeric_value>	1994
:TIME	<numeric_value>	1994
:OREference		1994
:LOCation	<numeric_value>	1994
:POINts	<numeric_value>	1994
:POINts	<numeric_value>	1994
:REALtime		1993
[:STATe]	<Boolean>	1993
:SPACing	LINear LOGarithmic	
:STEP	<numeric_value>	
:TIME	<numeric_value>	
:AUTO	<Boolean> ONCE	
:LLIMit	<numeric_value>	
:TINTerval	<numeric_value>	1994

**18.23.1 :COUNt <numeric\_value>**

SENSe:SWEep:COUNT

This command determines the number of sweeps which are initiated or vectors (arrays of samples, waveforms, etc) which are acquired by a single trigger event.

At \*RST, this value is set to 1.

**18.23.2 :DIRection UP|DOWN**

SENSe:SWEep:DIRection

Controls the direction of the sweep. If UP is selected, the sweep is carried out from start to stop. If DOWN is selected, the sweep is carried out from stop to start. This command only applies to swept instruments.

At \*RST, this value is set to UP.

**18.23.3 :DWELI <numeric\_value>**

SENSe:SWEep:DWELI

Controls the amount of time spent at each point during a sweep. This command only applies to swept instruments. Explicitly setting a value for SWEep:DWELI sets SWEep:DWELL:AUTO OFF.

The following coupling equation applies.

$$SWEep:TINTerval = SWEep:DWELI + \langle stepping\_time \rangle$$

Changing DWELI will cause either TINTerval or <stepping\_time> to change in a device dependent manner. See also introductory comments in the SWEEP subsystem.

<stepping\_time> is the device dependent time required for the sweep generator to move from one point (swept entity) to the next.

Note that DWELI cannot exceed TINTerval. Setting DWELI to a greater value will cause error -222, "Data out of range" to be generated.

At \*RST, this value is device-dependent.

**18.23.3.1 :AUTO <Boolean>|ONCE**

SENSe:SWEep:DWELl:AUTO

When ON is selected, the dwell time is calculated internally when SWEep:TINTerval is changed. Explicitly setting a value for SWEep:DWELL sets SWEep:DWELL:AUTO OFF.

At \*RST, AUTO is set to ON.

**18.23.4 :GENeration STEPped|ANALog**

SENSe:SWEep:GENeration

Determines if the sweep or acquisition is stepped or analog.

- STEPped: A stepped sweep or acquisition is selected. The number of points is set by the SWEep:POINts command.
- ANALog: The sweep or acquisition is controlled by an analog signal. SWEep:POINts is not applicable.

At \*RST, this value is device-dependent.

**18.23.5 :MODE AUTO|MANual**

SENSe:SWEep:MODE

Determines whether the sweep is performed automatically, or if the actual swept entity is controlled by the MANual command in the appropriate subsystem (for example, FREQuency:MANual). This command only applies to swept instruments.

At \*RST, this value is set to AUTO.

**18.23.6 :OFFSet**

SENSe:SWEep:OFFSet

This subsystem sets the offset between the offset reference point (SWEep:OREference) and the trigger point.

**18.23.6.1 :POINts <numeric\_value>**

SENSe:SWEep:OFFSet:POINts

Sets the offset in terms of points. This value may be positive or negative. A positive value specifies that the offset reference point occurs after the trigger point.

The following coupling equation applies.

$$\text{SWEep:OFFSet:POINts} = \text{SWEep:OFFSet:TIME} / \text{SWEep:TINTerval}$$

Changing SWEep:OFFSet:POINts will cause SWEep:OFFSet:TIME to change, but not SWEep:TINTerval. See also introductory comments in the SWEep subsystem.

At \*RST, this value is set to 0.

**18.23.6.2 :TIME <numeric\_value>**

SENSe:SWEep:OFFSet:TIME

Sets the offset in units of time. This value may be positive or negative. A positive value specifies that the offset reference point occurs after the trigger point.

The following coupling equation applies.

$$\text{SWEep:OFFSet:POINts} = \text{SWEep:OFFSet:TIME} / \text{SWEep:TINTerval}$$

Changing SWEep:OFFSet:TIME will cause SWEep:OFFSet:POINts to change, but not SWEep:TINTerval. See also introductory comments in the SWEep subsystem.

At \*RST, this value is set to 0.

### 18.23.7 :OREFerence

SENSe:SWEep:OREFerence

This subsystem sets the location of the offset reference point in the sweep or acquisition. The offset reference is used by the SWEep:OFFSet commands.

#### 18.23.7.1 :LOCation <numeric\_value>

SENSe:SWEep:OREFerence:LOCAtion

Sets the offset reference by specifying a relative location. SWEep:OREFerence:LOCAtion accepts a range of values from 0 to 1. A value of 0 selects the first point of acquisition as the reference. A value of 1 selects the last point.

The following coupling equation applies.

$$\begin{aligned} \text{SWEep:OREFerence:LOCAtion} = \\ (\text{SWEep:OREFerence:POINts} - 1) / (\text{SWEep:POINts} - 1) \end{aligned}$$

Changing SWEep:OREFerence:LOCAtion will cause SWEep:OREFerence:POINts to change, but not SWEep:POINts. See also introductory comments in the SWEep subsystem.

At \*RST, this value is device-dependent.

#### 18.23.7.2 :POINts <numeric\_value>

SENSe:SWEep:OREFerence:POINts

Sets the offset reference by specifying a number of points. SWEep:OREFerence:POINts accepts values from 1 to N, where N is the maximum value of SWEep:POINts. A value of 1 selects the first point of acquisition as the reference.

The following coupling equation applies.

$$\begin{aligned} \text{SWEep:OREFerence:LOCAtion} = \\ (\text{SWEep:OREFerence:POINts} - 1) / (\text{SWEep:POINts} - 1) \end{aligned}$$

Changing SWEep:OREFerence:POINts will cause SWEep:OREFerence:LOCAtion to change, but not SWEep:POINts. See also introductory comments in the SWEep subsystem.

At \*RST, this value is device-dependent.

#### 18.23.8 :POINts <numeric\_value>

SENSe:SWEep:POINts

Sets the number of points in a stepped sweep or acquisition. POINts is ignored when GENeration is ANALog.

The following coupling equations apply.

$$\text{SWEep:TINTerval} = \text{SWEep:TIME} / (\text{SWEep:POINts} - 1)$$

Changing SWEep:POINts will cause either SWEep:TINTerval or SWEep:TIME to change in a device dependent manner. See also introductory comments in the SWEep subsystem.

*SWEep:OREFerence:LOCation =  
 (SWEep:OREFerence:POINts - 1) / (SWEep:POINts - 1)*

Changing SWEep:POINts will cause either SWEep:OREFerence:LOCation or SWEep:OREFerence:POINts to change in a device dependent manner. See also introductory comments in the SWEep subsystem.

When SPACing is LINear:

*SWEep:STEP = <swept\_subsystem>:SPAN / (SWEep:POINts - 1)*

Changing SWEep:POINts will cause SWEep:STEP to change, but not <swept\_subsystem>:SPAN.

When SPACing is LOGarithmic:

*<points\_per\_decade> = (SWEep:POINts-1) / <decades\_in\_span>*

Changing SWEep:POINts will cause <points\_per\_decade> to change, but not <decades\_in\_span>. <decades\_in\_span> is calculated internally from <swept\_subsystem>:STARt, :STOP, :CENTer, and :SPAN. Normally, <decades\_in\_span> equals  $\log_{10}(:STOP/:STARt)$ , when :STOP > :STARt. See also introductory comments in the SWEep subsystem.

At \*RST, this value is device-dependent.

### 18.23.9 :REALtime

SENSe:SWEep:REALtime

Instruments that acquire data directly as function of time, may use several data acquisition methods to sense the data points of the sweep.

There are techniques where the points in the requested sweep are collected from a number of successive acquisitions. In this case the result is a reconstruction of the original signal out of several acquisitions; this is not real time.

An acquisition is known as real time if no reconstruction is done; that is, the points in the resulting data set are in the order in which they were acquired.

#### 18.23.9.1 [:STATe] <Boolean>

SENSe:SWEep:REALtime:STATe

When STATe is ON, the instrument is required to collect data in real time. When STATe is OFF, the instrument is allowed to use reconstruction techniques.

At \*RST this function is OFF

### 18.23.10 :SPACing LINear|LOGarithmic

SENSe:SWEep:SPACing

Determines the time vs. swept entity characteristics of the sweep. The various settings have the following meanings:

- LINear: The swept entity is incremented (decremented) by the stepsize until the sweep limit is reached if the sweep generation is stepped. If the sweep is analog,

selecting this parameter specifies a ramp.

- LOGarithmic: Step size is determined by a logarithmic curve fitted between the start and stop value of the swept entity. Stepping is determined by the number of steps function.

At \*RST, this value is set to LINear.

### 18.23.11 :STEP <numeric\_value>

SENSe:SWEep:STEP

Controls the swept entity step size for a stepped linear sweep. STEP is ignored when GENeration is ANALog or when SPACing is LOGarithmic.

The following coupling equation applies.

$$SWEep:STEP = \langle swept\_subsystem \rangle:SPAN / (SWEep:POINts - 1)$$

Changing SWEep:STEP will cause SWEep:POINts to change, but not  $\langle swept\_subsystem \rangle:SPAN$ . See also introductory comments in the SWEep subsystem.

The default units are those of the associated  $\langle swept\_subsystem \rangle:SPAN$  command.

At \*RST, this value is device-dependent.

### 18.23.12 :TIME <numeric\_value>

SENSe:SWEep:TIME

Sets the duration of the sweep or acquisition. Note that this does not turn sweeping or acquisition on. Explicitly setting a value for SWEep:TINTerval or SWEep:TIME sets SWEep:TIME:AUTO OFF.

The following coupling equation applies.

$$SWEep:TINTerval = SWEep:TIME / (SWEep:POINts - 1)$$

Changing SWEep:TIME will cause SWEep:TINTerval to change, but not SWEep:POINts. See also introductory comments in the SWEep subsystem.

At \*RST, this value is device-dependent.

### 18.23.12.1 :AUTO <Boolean>|ONCE

SENSe:SWEep:TIME:AUTO

When ON is selected, SWEep:TIME and SWEep:TINTerval are calculated internally when related settings outside the time domain coupling equations are changed. Explicitly setting a value for SWEep:TIME or SWEep:TINTerval sets SWEep:TIME:AUTO OFF.

This command only applies to swept instruments, where explicitly setting SWEep:TIME and SWEep:TINTerval is the exception. For instance, in a spectrum analyzer, a user normally sets the frequency parameters (FREQuency:SPAN and :CENTER) and expects the instrument to calculate the appropriate sweep time and interval to produce a reasonable resolution bandwidth. The :AUTO node should not be implemented in non-swept instruments that normally expect the user to explicitly set SWEep:TIME or SWEep:TINTerval.

At \*RST, AUTO is set to ON.

**18.23.12.2 :LLIMit <numeric\_value>**

SENSe:SWEep:TIME:LLIMit

Defines a lower limit for sweep time. This lower limit restricts the sweep time value set either explicitly or automatically.

At \*RST, this value is device-dependent.

**18.23.13 :TINTerval <numeric\_value>**

SENSe:SWEep:TINTerval

Sets the time interval between points of the sweep or acquisition. Explicitly setting a value for SWEep:TINTerval or SWEep:TIME sets SWEep:TIME:AUTO OFF.

The following coupling equations apply.

$$\text{SWEep:TINTerval} = \text{SWEep:TIME} / (\text{SWEep:POINts} - 1)$$

Changing SWEep:TINTerval will cause SWEep:TIME to change, but not SWEep:POINts.  
See also introductory comments in the SWEep subsystem.

$$\text{SWEep:OFFSet:POINts} = \text{SWEep:OFFSet:TIME} / \text{SWEep:TINTerval}$$

Changing SWEep:TINTerval will cause either SWEep:OFFSet:POINts or SWEep:OFFSet:TIME to change in a device dependent manner. See also introductory comments in the SWEep subsystem.

At \*RST, this value is device-dependent.

18.24 **VOLTage Subsystem**

SENSe:VOLTage

This subsection controls the voltage amplitude characteristics of the sensor.

KEYWORD	PARAMETER FORM	NOTES
:VOLTage		
:AC[:DC]		
:APERture	<numeric_value>	
:NPLCycles	<numeric_value>	1991
:ATTenuation	<numeric_value>	
:AUTO	<Boolean>	
:PROTection		
[:LEVel]	<numeric_value>	
:STATe		
:TRIPped?		[query only]
:CLEar		[no query]
:RANGE		
[:UPPer]	<numeric_value>	
:LOWer	<numeric_value>	
:AUTO	<Boolean> ONCE	
:DIRection	UP DOWN EITHER	
:LLIMit	<numeric_value>	1996
:ULIMit	<numeric_value>	1996
:OFFSet	<numeric_value>	1991,2
:PTPeak	<numeric_value>	1991,2
:REFerence	<numeric_value>	
:STATe	<Boolean>	
:RESolution	<numeric_value>	
:AUTO	<Boolean> ONCE	
:DETector	INTERNAL   EXTERNAL	1991

18

18.24.1 **:AC[:DC]**

SENSe:VOLTage:AC

This subsystem selects the alternating current or the direct current sensor. The default units are volts.

18.24.1.1 **:APERture <numeric\_value>**

SENSe:VOLTage:AC:APERture

Specifies the acquisition/sampling/gate time for a single measurement point. The parameter has a unit of seconds.

At \*RST, this value is device-dependent.

**18.24.1.2 :NPLCycles <numeric\_value>**

SENSe:VOLTage:AC:NPLCycles

Specifies the acquisition/sampling/gate time for a single measurement point in terms of the number of power line cycles. The parameter is unitless. If this command is implemented, the APERture command must also be implemented.

The value is coupled to APERture by the equation:

$$\text{APERture} = \text{NPLCycles}/\text{selected line frequency}$$

At \*RST, this value is device-dependent.

**18.24.1.3 :ATTenuation <numeric\_value>**

SENSe:VOLTage:AC:ATTenuation

Sets the ATTenuation level. Note that when setting the level to 10 dB the magnitude of the incoming signal will be decreased by 10 dB. Changing the ATTenuation changes RANGe by the same amount.

Default units are as determined in the UNITS system.

**18.24.1.3.1 :AUTO <Boolean>**

SENSe:VOLTage:AC:ATTenuation:AUTO

Couples the attenuator to RANGe such that maximum dynamic range of the incoming signal is assured without overloading.

Programming a specified attenuation sets AUTO OFF.

At \*RST, AUTO is set to ON.

**18.24.1.4 :PROTection**

SENSe:VOLTage:AC:PROTection

Controls the protection circuits.

**18.24.1.4.1 [:LEVel] <numeric\_value>**

SENSe:VOLTage:AC:PROTection:LEVel

This command sets the input level at which the input protection circuit will trip.

**18.24.1.4.2 :STATe <Boolean>**

SENSe:VOLTage:AC:PROTection:STATE

Controls whether the input protection circuit is enabled.

At \*RST, this value is set to ON.

**18.24.1.4.3 :TRIPped?**

SENSe:VOLTage:AC:PROTection:TRIPped?

This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped. TRIPped only has a query form and thus has no associated \*RST condition.

**18.24.1.4.4 :CLEar**

SENSe:VOLTage:AC:PROTection:CLEar

Causes the protection circuit to be cleared.

This command is an event and has no associated \*RST condition.

**18.24.1.5 :RANGE**

SENSe:VOLTage:AC:RANGE

Sets the range for the amplitude sensor function. The RANGE endpoints may be specified in terms of the UPPer endpoint and the LOWER endpoint, alternatively the midpoint of the RANGE (OFFSet) and the span of the RANGE (PTPeak) may be used. A non-symmetrical range around zero may cause some instruments to introduce an actual offset to the signal entering the SENSe block.

**18.24.1.5.1 [:UPPer] <numeric\_value>**

SENSe:VOLTage:AC:RANGE:UPPer

Specifies the most positive signal level expected for the sensor input.

At \*RST, this value is device-dependent.

**18.24.1.5.2 :LOWER <numeric\_value>**

SENSe:VOLTage:AC:RANGE:LOWER

Specifies the most negative signal level expected for the sensor input. This allows setting for the best dynamic range. This function is coupled to UPPer in a device-dependent manner.

At \*RST, this value is device-dependent.

**18.24.1.5.3 :AUTO <Boolean>|ONCE**

SENSe:VOLTage:AC:RANGE:AUTO

Sets the RANGE to the value determined to give the most dynamic range without overloading. The actual algorithm is device-dependent.

At \*RST, AUTO is set to ON.

**18.24.1.5.3.1 :DIRection UP|DOWN|EITHER**

SENSe:VOLTage:AC:RANGE:AUTO:DIRection

The DIRection command defines the manner in which AUTO works. When UP is selected, the instrument shall only change the range automatically to ranges larger than the current range, as the input signal dictates. When DOWN is selected, the instrument shall only change the range automatically to ranges smaller than the current range, as the input signal dictates. When EITHER is selected the instrument may range in either direction.

At \*RST, DIRection shall be set to EITHER.

**18.24.1.5.3.2 :LLIMit <numeric\_value>**

SENSe:VOLTage:AC:RANGE:AUTO:LLIMit

The Lower LIMit command sets the smallest range to which the instrument will go while autoranging. When AUTO is ON and the present RANGE is outside the new limits, the instrument may change the RANGE to a value inside the limits or wait until the next time the autoranging algorithm is performed.

At \*RST, LLIMit shall be set to MINimum.

**18.24.1.5.3.3 :ULIMit <numeric\_value>**

SENSe:VOLTage:AC:RANGE:AUTO:ULIMit

The Upper LIMit command sets the largest range to which the instrument will go while

autoranging. When AUTO is ON and the present RANGe is outside the new limits, the instrument may change the RANGe to a value inside the limits or wait until the next time the autoranging algorithm is performed.

At \*RST, ULIMit shall be set to MAXimum.

#### **18.24.1.5.4 :OFFSet <numeric\_value>**

SENSe:VOLTage:AC:RANGE:OFFSet

OFFSet determines the midpoint of the range.

At \*RST, this value is device-dependent.

#### **18.24.1.5.5 :PTPeak <numeric\_value>**

SENSe:VOLTage:AC:RANGE:PTPeak

Peak To Peak specifies the dynamic range required for the sensor.

At \*RST, this value is device-dependent.

#### **18.24.1.6 :REFerence <numeric\_value>**

SENSe:VOLTage:AC:REFERENCE

Sets a reference amplitude for sensor instruments. This command differs in intent from the reference correction in that this is intended to be a measurement offset where a user wishes to reference a measurement to some known value.

At \*RST, this value is device-dependent, but STATe is set to OFF.

#### **18.24.1.6.1 :STATe <Boolean>**

SENSe:VOLTage:AC:REFERENCE:STATE

Determines whether amplitude is measured in absolute or relative mode. If STATe is ON, then amplitude is referenced to the value set in REFerence.

At \*RST, this value is set to OFF.

#### **18.24.1.7 :RESolution <numeric\_value>**

SENSe:VOLTage:AC:RESolution

Specifies the absolute resolution of the measurement. For example, if a measurement of 10 Volts is specified with a resolution of .01 Volts, the measurement is returned with a resolution of .01 Volts (that is, 10.00).

RANGe is not coupled to RESolution. Setting RANGe shall not cause RESolution to change. Setting RESolution shall not cause RANGe to change.

At \*RST, this value is device-dependent.

#### **18.24.1.7.1 :AUTO <Boolean>|ONCE**

SENSe:VOLTage:AC:RESolution:AUTO

Allows the system to determine the best resolution for the other measurement conditions.

Explicitly setting a value for :RESolution will turn AUTO OFF.

At \*RST, this value is set to ON.

**18.24.2 :DETector INTernal | EXTernal**

SENSe:VOLTage:DETector

The DETector can be specified as internal or external.

At \*RST, this value is set to INT.

**18.25 WINDOW Subsystem**

SENSe:WINDOW

The window subsystem selects digital signal processing windowing parameters for use by FFT analyzers. Windows are time-domain weighting functions applied to input time records to condition the results of a time-domain to frequency-domain transformation. This subsystem controls windowing done to compensate for measurement artifacts. Analytical post-acquisition windowing is controlled with CALCulate:TRANSform.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:WINDOW		
[:TYPE]	RECTangular subsystem UNIForm FLATtop subsystem  HAMMING HANNing KBESsel  FORCe EXPonential	
:KBESsel	<numeric_value>	
:EXPonential	<numeric_value>	
:FORCe	<numeric_value>	

**18.25.1 [:TYPE] RECTangular|UNIForm|FLATtop|HAMMING|HANNing  
|KBESsel|FORCe|EXPonential**

SENSe:WINDOW:TYPE

TYPE allows user selection from among several well-known or standard window types. Each window has characteristics that make it particularly suitable for certain kinds of measurements.

Individual windowing algorithms are defined in “*On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform*,” F. J. Harris, Proc of the IEEE, Vol. 66-1, January 1978, pp 51-83.

At \*RST, this value is device-dependent.

**18.25.1.1 :KBESsel <numeric\_value>**

SENSe:WINDOW:TYPE:KBESsel

Enters the exponential decay time constant which characterizes the KBESsel window. The parameter to this command has units of seconds.

At \*RST, this value is device-dependent.

**18.25.1.2 :EXPonential <numeric\_value>**

SENSe:WINDOW:TYPE:EXPonential

Enters the exponential decay time constant which characterizes the EXPonential window. The parameter to thids command has units of seconds.

At \*RST, this value is device-dependent.

**18.25.1.3 :FORCe <numeric\_value>**

SENSe:WINDOW:TYPE:FORCe

Enters the time value parameter corresponding to the width of the gated portion of the input time record for FORCe windows. The parameter of this command has units of seconds.

At \*RST, this value is device-dependent.



## 19 SOURce Subsystem

The SOURce setup commands are divided into several sections. Each section or subsystem deals with controls that directly affect device-specific settings of the device, and not those related to the signal-oriented characteristics. These commands are referenced through the SOURCe node.

The SOURce node may be optional in a particular device. The reason that the SOURce is optional is to allow devices which are primarily sources to accept shorter commands. The SOURce node cannot be optional if either SENSe or ROUTe is optional, since only one node at a given level may be optional. For example, a typical power supply may elect to make the SOURce node optional, since the most frequently used commands would be under the SOURce subsystem. If the power supply also contained either source or routing functionality, these would be controlled through their respective nodes; however, the SENSe or ROUTe nodes would be required. An optional node, such as SOURce, implies that the device shall accept and process commands with or without the optional node and have the same result. That is, the device is required to accept the optional node, if sent, without error.

In some instances, a source may contain subservient source or sensor functions. In such cases, the additional subservient functionality shall be placed as SENSe or SOURce subnodes under the SOURce subsystem. Further, no optional nodes (SENSe, SOURce or ROUTe) are permitted if such subservient functionality exists in a device. Otherwise conflicts in interpreting commands would occur, for example, between SOURce and [SENSe:]SOURce if the SENSe were allowed to be optional.

CURRent, POWER, VOLTage, FREQuency, and RESistance contain several sets of commands which have complex couplings. This includes the sweep commands STARt, STOP, CENTER, and SPAN as one set, and AMPLitude, OFFSet, HIGH, and LOW as another. Sending any one of a set singly will affect two others of the set. However, if two are sent in a single message unit, then these two should be set to the values specified and the other two changed. This is in accordance with the style guidelines and *IEEE 488.2*. STARt, STOP, CENTER, and SPAN must be implemented as a set. That is, if any one is implemented, then all must be implemented.

If the requested setting is unachievable given the other instrument settings, then an error must be generated (-211 “settings conflict”). The device may resolve the conflict in a device-dependent manner (for example, change STARt, STOP, CENTER, and/or SPAN) to resolve the error. Note that when more than one of the four sweep settings are received in the same program message unit, the sweep will be determined by the last two received.

19

The coupled commands may define commands as subnodes which alter these couplings. The command description will define how the couplings are altered. However, any command which alters couplings must default to the couplings described in this section as the \*RST couplings.

When CENTER or SPAN is changed the following couplings apply:

$$\begin{array}{ll} \text{SPAN} & = \text{SPAN} \\ \text{CENTER} & = \text{CENTER} \end{array}$$

## 1999 SCPI Command Reference

STOP	=	CENTer + (SPAN/2)
STARt	=	CENTer - (SPAN/2)

When STARt or STOP is changed the following couplings apply:

STARt	=	STARt
STOP	=	STOP
CENTer	=	(STARt + STOP)/2
SPAN	=	STOP - STARt

The couplings above assume linear units in the sweep specified.

HIGH, LOW, AMPLitude, and OFFSet have the same interactions described above for STARt, STOP, CENTER, and SPAN.

When AMPLitude or OFFSet is changed the following couplings apply:

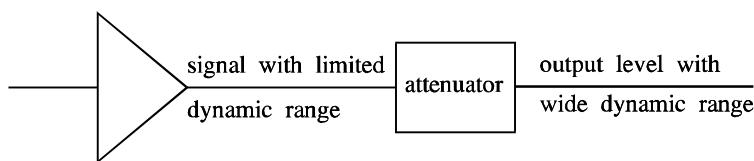
OFFSet	=	OFFSet
AMPLitude	=	AMPLitude
HIGH	=	OFFSet + (AMPLitude/2)
LOW	=	OFFSet - (AMPLitude/2)

When HIGH or LOW is changed the following couplings apply:

HIGH	=	HIGH
LOW	=	LOW
OFFSet	=	(HIGH + LOW)/2
AMPLitude	=	HIGH - LOW

For CURRent, POWER, and VOLTage when ATTenuation:AUTO is ON a coupling exists between ATTenuation and LEVel. This coupling is needed in sources which use an attenuator to extend the dynamic range of the source's level. Fig 19-1 illustrates this situation.

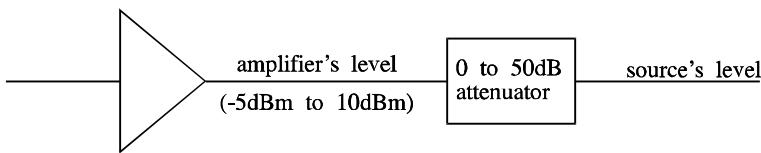
When ATTenuator:AUTO is ON, the full dynamic range of the instrument is available using the LEVel command. The instrument uses an algorithm which adjusts both the limited dynamic range signal and the attenuator to supply the requested level.



**Figure 19-1 ATTenuation - LEVel Coupling Model**

When ATTenuation:AUTO is OFF, the amount of attenuation is fixed and the range of allowed values for LEVel is reduced. The MINimum and MAXimum values for LEVel depend on the value of ATTenuation.

For example, assume an instrument has the block diagram illustrated in Fig 19-2.



**Figure 19-2 ATTenuation - LEVel Coupling Example**

Say the output of the amplifier can vary from -5dBm to 10dBm and the attenuator can range from 0 to 50dB in steps of 10dB. The source's level can thus vary from -55dBm to 10dBm.

After \*RST, POWER:ATTenuation:AUTO is ON allowing POWER:ATTenuation to automatically change depending on POWER[:LEVel]. The allowable range for POWER[:LEVel] is from -55dBm (MINimum) to 10dBm (MAXimum). If POWER[:LEVel] is set to -48dBm, POWER:ATTenuation is set to 50dB and the power level at the amplifier's output is 2dBm. If POWER[:LEVel] is -43dBm, the instrument could be in one of two states:

POWER:ATTenuation is -40dB and amplifier's level is -3dBm or  
POWER:ATTenuation is -50dB and amplifier's level is 7dBm

Which state is chosen may depend on the previous state of POWER[:LEVel] and the algorithm employed by the instrument.

The user now wants to prevent the attenuator from changing values. POWER:ATTenuation -40 is sent. Receiving a value for POWER:ATTenuation has the side effect of turning POWER:ATTenuation:AUTO OFF, see 7.4 in Syntax & Style. Because the attenuator cannot automatically change, the allowable range for POWER[:LEVel] is limited to the range -45dBm (MINimum) to -30dBm (MAXimum). Trying to program a value outside this range causes an Execution Error. The permissible range for POWER[:LEVel] can be changed by sending a new value for POWER:ATTenuation.

The actual algorithm employed in the instrument and the range of values depend on the underlying hardware implementation. SCPI does require MAXimum/MINimum queries which a user could use to discover how the bounds of the LEVel command change.

**First Level SOURce Keywords**

[ <b>:SOURce</b> ]		
<b>:ACCEleration</b>		1999
<b>:AM</b>		
<b>:COMBine</b>		1994
<b>:CORRection</b>		1992
<b>:CURREnt</b>		
<b>:DM</b>		
<b>:FM</b>		
<b>:FREQuency</b>		
<b>:FORCe</b>		1999
<b>:FUNCTION</b>		
<b>:LIST</b>		
<b>:MARKer</b>		
<b>:PHASE</b>		
<b>:PM</b>		
<b>:POWER</b>		
<b>:PULM</b>		
<b>:PULSe</b>		
<b>:RESistance</b>		
<b>:ROSCillator</b>		
<b>:SPEEd</b>		1999
<b>:SWEep</b>		
<b>:TEMPerature</b>		1993
<b>:VOLTage</b>		

## 1999 SCPI Command Reference

### 19.1 ACCELERATION Subsystem

SOURce:ACCELERation

This node controls the acceleration of the device.

KEYWORD	PARAMETER FORM	COMMENTS
:ACCELERation		1999
[:LEVel]	<numeric_value>	1999

#### 19.1.1 [:LEVel] <numeric\_value>

:SOURce:ACCELERation[:LEVel]

Sets the acceleration in m/s<sup>2</sup>.

At \*RST, this value is device dependent.

## 19.2 AM Subsystem

SOURce:AM

The Amplitude Modulation subsystem is used to set the modulation controls and the parameters associated with the modulating signal(s). The keywords to describe the carrier signal exist in other subsystems (such as, FREQuency and PHASe).

KEYWORD	PARAMETER FORM	COMMENTS
:AM		
:COUPling	AC DC GROund	
[:DEPTh]	<numeric_value>	
:EXTernal		
:COUPling	AC DC GROund	
:IMPedance	<numeric_value>	
:POLarity	NORMAl INVerted	
:INTernal		
:FREQuency	<numeric_value>	
:MODE	FIXEd LIST	1993
:POLarity	NORMAl INVerted	
:SENSitivity	<numeric_value>	
:SOURce	EXTernal INTernal{,EXTernal INTernal}	
:STATE	<Boolean>	
:TYPE	LINear LOGarithmic EXPonential	1991

## 19.2.1 :COUPLing AC|DC|GROund

SOURce:AM:COUPling

COUPLing, at this level, is used to set the coupling between the modulator and the modulating signal. The modulating signal may be the sum of several signals, either internal or external sources. If COUPLing is set to DC, then both AC and DC components of the signal pass. AC coupling passes only the AC component of the signal. The GROund parameter allows Amplitude Modulation to be turned ON without the modulating signal connected.

At \*RST, this value is device-dependent.

19

## 19.2.2 [:DEPTh] &lt;numeric\_value&gt;

SOURce:AM:DEPTh

Sets the modulation DEPTh of an AM signal. The unit for DEPTh is percent (%). This command is used where the DEPTh can be controlled independently from the modulating signal voltage level. Alternately, it is used to set a device capability that can indicate when the desired DEPTh has been achieved.

At \*RST, this value is device-dependent.

## 19.2.3 :EXTernal

SOURce:AM:EXTernal

The EXTernal subsystem is used to control the specified external signal source. Where multiple external modulation sources are available, the EXTernal keyword may have a

numeric suffix signifying the particular external signal source. If no number is given, a 1 is assumed.

#### 19.2.3.1 :COUpling AC|DC|GROund

SOURce:AM:EXTernal:COUPling

COUpling, at this level, sets the coupling for only the specified external signal source. If COUpling is set to DC, then both AC Modulation and DC components of the signal pass. AC coupling passes only the AC component of the signal. The GROund parameter allows the specified external source to be turned ON without being connected.

At \*RST, this value is device-dependent.

#### 19.2.3.2 :IMPedance <numeric\_value>

SOURce:AM:EXTernal:IMPedance

Sets the impedance of the specified external signal source. The unit for IMPedance is Ohms.

At \*RST, this value is device-dependent.

#### 19.2.3.3 :POLarity NORMAL|INVerted

SOURce:AM:EXTernal:POLarity

POLarity, at this level, sets the POLarity for only the specified external signal source.

POLarity sets the relationship between the polarity of the applied modulation voltage and the resulting direction of modulation. The definition of “normal” polarity for AM modulated signal is: a positive voltage which causes the modulation envelope or the instantaneous carrier amplitude to increase.

At \*RST, this value must be set to NORMAL.

#### 19.2.4 :INTERNAL

SOURce:AM:INTERNAL

The INTERNAL subsystem is used to control the specified internal signal source. Where multiple internal modulation sources are available, the INTERNAL keyword may have a numeric suffix signifying the particular internal signal source. If no number is given, a 1 is assumed.

#### 19.2.4.1 :FREQuency <numeric\_value>

SOURce:AM:INTERNAL:FREQuency

Sets the frequency of the specified internal signal source. Some instruments may also allow the internal signal source(s) to be controlled as a subsystem(s). The unit of FREQuency is Hz.

At \*RST, this value is device-dependent.

#### 19.2.5 :MODE

SOURce:AM:MODE

Determines which set of commands currently control the AM subsystem. The settings have the following meanings:

FIXed: Settings of individual commands are as last set by the user.

LIST: Settings of individual commands are controlled by the LIST subsystem.

At \*RST the value is FIXed.

### 19.2.6 :POLarity NORMAL|INVerted

SOURce:AM:POLarity

POLarity, at this level, is used to set the polarity between the modulator and the modulating signal. The modulating signal may be the sum of several signals, either internal or external sources. POLarity sets the relationship between the polarity of the applied modulation voltage and the resulting direction of modulation. The definition of “normal” polarity for AM modulated signal is a positive voltage, causing the modulation envelope or the instantaneous carrier amplitude to increase.

At \*RST, this value must be set to NORMAL.

### 19.2.7 :SENSitivity <numeric\_value>

SOURce:AM:SENSitivity

Controls the modulation depth by setting a sensitivity Modulation to the modulation signal voltage level. The unit for SENSitivity is percent/Volt (%/V).

At \*RST, this value is device-dependent.

### 19.2.8 :SOURce EXTERNAL|INTERNAL{,EXTernal|,INTernal}

SOURce:AM:SOURce

Selects the source for the modulating signal. It may specify a single source or a number of sources. The specified sources, in the SOURce command, are all selected and turned on. Any sources from a previous selection that are not part of the current selection list are deselected and turned off. If multiple sources are selected and the device does not support the combination requested in the SOURce command, an execution error -221 will be generated.

Where multiple internal modulation sources are available, individual INTernal sources must be distinguished by a unique number suffix. If no number suffix is provided in a command, a 1 must be assumed.

Where multiple external modulation sources are available, individual EXternal sources must be distinguished by a unique number suffix. If no number suffix is provided in a command, a 1 must be assumed.

At \*RST, this value is device-dependent.

### 19.2.9 :STATe <Boolean>

SOURce:AM:STATe

This command is used to turn Amplitude Modulation ON or OFF. Turning AM modulation ON will not automatically turn OFF any other types of modulation. Turning any or all modulations types ON or OFF must be done explicitly. Where a type of modulation is active, turning ON another type of modulation will add to the set of active modulations if that combination of capabilities are legal in the device; otherwise, an execution error -221 is generated, and the instrument state is device-dependent.

At \*RST, this value must be set to OFF.

### 19.2.10 :TYPE

SOURce:AM:TYPE

Controls the type of AM shaping

- LINear – The output voltage is directly proportional to Vmod.
- LOGarithmic – The output voltage is proportional to  $\log(Vmod)$ .
- EXPonential – The output voltage is proportional to  $10^{(Vmod)}$

At \*RST the selection will be LINear.

## 1999 SCPI Command Reference

### 19.3 COMBine Subsystem

SOURce:COMBine

This subsystem is used to combine two or more signals or data streams into one.

KEYWORD	PARAMETER FORM	COMMENTS
:COMBine		1994
:FEED	<data_handle>	1994

#### 19.3.1 :FEED <data\_handle>

Sets or queries the data flow into the COMBine block. The number of FEED in a :COMBine block is device dependent.

At \*RST, FEED is device dependent.

## 19.4 CORRection Subsystem

SOURce:CORRection

The correction subsystem provides for known external losses or gains. In contrast to CALibration, these losses usually are dependent on individual setups.

Correction data may be entered and stored in several ways. A mechanism is provided to automatically measure the signal and store the resulting “correction set”. This data can be selected at any time as the active “correction set”. The data could also be generated by an external device and transferred to the instrument over a computer interface. Commands are also provided to enter individual losses, gains, offsets, delays, etc. that the instrument should use in correcting its operation. An instrument is not required to implement all of the methods.

KEYWORD	PARAMETER FORM	COMMENTS
:CORRection		1992
[:STATe]	<Boolean>	1992
:COLLect		1992
[:ACQuire]		1992
:METHod	PMETer	1992
:SAVE	<name>	1992
:CSET		1992
[:SELEct]	<name>	1992
:STATe	<Boolean>	1992
:OFFSet		1992
[:MAGNitude]	<numeric_value>	1992
:PHASe	<numeric_value>	1992
:STATe	<Boolean>	1992
:LOSS GAIN SLOPe		1992
:STATe	<Boolean>	1992
[:OUTPut]		1992
[:MAGNitude]	<numeric_value>	1992
:PHASe	<numeric_value>	1992
:EDELay		1992
[:TIME]	<numeric_value>	1992
:DISTance	<numeric_value>	1992
:STATe	<Boolean>	1992
:RVELocity		1992
:MEDIum	COAX WAVeguide	1992
:COAX	<numeric_value>	1992
:WAVeguide	<numeric_value>	1992
:FCUToff	<numeric_value>	1992
:STATe	<Boolean>	1992

### 19.4.1 [:STATe] <Boolean>

SOURce:CORRection:STATe

Determines whether the correction data defined in this section is applied to the signal.

At \*RST, this function is OFF.

**19.4.2 :COLLect**

SOURce:CORRection:COLLect

Deals with the collection of the correction data by the device. Details may be device specific.

**19.4.2.1 [:ACQUIRE]**

SOURce:CORRection:COLLect:ACQUIRE

A signal will be generated, measured, and the results saved as data for the chosen correction method.

**19.4.2.2 :METHod PMETer**

SOURce:CORRection:COLLect:METHod

Selects the correction method to be used for the correction that is about to be performed. If only one correction method exists in the instrument then this command is not required.

The various settings of the parameters have the following meanings:

- PMETer Use external Power METer to take readings.

At \*RST, this value is device dependent.

**19.4.2.3 :SAVE [<name>]**

SOURce:CORRection:COLLect:SAVE

Calculates the correction data using the current method selection and then saves the correction data. If only one correction array exists then it is saved there. Otherwise it is saved in the set specified by &lt;name&gt;.

<name> may be either a <trace\_name>, or a <table\_name> which must be defined and exist in the instrument. Note this command will not create either <trace\_name>, or a <table\_name>. Trace definition is done in the TRACe subsystem. Table definition is done in the MEMory subsystem.

**19.4.3 :CSET**

SOURce:CORRection:CSET

The Correction SET subsystem is used to select the active correction set.

**19.4.3.1 [:SELect] <name>**

SOURce:CORRection:CSET:SELect

Specifies the active correction set. This is the set that is used when CORR:CSET:STAT ON is specified.

Note that the parameter is a <name> that must be defined and exist in the instrument. <trace\_name>, or a <table\_name> which must be defined and exist in the instrument. Note this command will not create either <trace\_name>, or a <table\_name>. Trace definition is done in the TRACe subsystem. Table definition is done in the MEMory subsystem.

**19.4.3.2 :STATE <Boolean>**

SOURce:CORRection:CSET:STATE

Determines whether the correction data defined in the selected set is applied to the output.

At \*RST, this function is OFF. OFF is chosen because all of the correction sets may be empty. The instrument may need to be supplied with data before correction can be applied.

**19.4.4 :OFFSet**

SOURce:CORRection:OFFSet

This node defines a single value that is added to the signal supplied by the instrument. This command shall be used with the currently selected units. Offset effects the signal values as additive (or subtractive) for linear units such as VOLT.

**19.4.4.1 [:MAGNitude] <numeric\_value>**

SOURce:CORRection:OFFSet:MAGNitude

This is the magnitude value of the correction offset data. Magnitude is always added linearly, as in volts.

At \*RST, this value is set to zero.

**19.4.4.2 [:PHASe] <numeric\_value>**

SOURce:CORRection:OFFSet:PHASe

This is the phase value of the correction offset data. The units associated with this are the currently selected angle units.

At \*RST, this value is set to zero.

**19.4.4.3 :STATe <Boolean>**

SOURce:CORRection:OFFSet:STATe

This function is used to enable or disable the offset correction.

At \*RST, this function is set to OFF.

**19.4.5 :LOSS[:GAIN]:SLOPe**

SOURce:CORRection:LOSS

These commands are used either to provide a mechanism to apply simple correction or to correct external factors that have not been accounted for in the correction set.

Loss, gain and slope affect the signal amplitude as a multiplier for linear units such as VOLT and as additive (or subtractive) for logarithmic units such as dB.

**:LOSS** This command is used to set the anticipated loss in the system, excluding those factors covered by the active correction set. The device then adjusts the signal by this factor to compensate for the loss. Loss is coupled to GAIN by the equation  $\text{LOSS} = 1/\text{GAIN}$  when the default unit is linear and  $\text{LOSS} = -\text{GAIN}$  when the default unit is logarithmic.

**:GAIN** This command is used to set the anticipated gain in the system, excluding those factors covered by the active correction set. The device then adjusts the signal by this factor to compensate for the gain. Gain is coupled to LOSS by the equation  $\text{GAIN} = 1/\text{LOSS}$  when the default unit is linear and  $\text{GAIN} = -\text{LOSS}$  when the default unit is logarithmic.

**:SLOPe** This node defines an increasing or decreasing value that is added to the signal. This correction value is zero at a zero value of the x axis and increase or decreases from there.

**19.4.5.1 :STATe <Boolean>**

SOURce:CORRection:LOSS:STATe

This function is used to enable or disable the correction for LOSS or GAIN.

At \*RST, this function is set to OFF.

**19.4.5.2 [:OUTPut]**

SOURce:CORRection:LOSS:OUTPut

The OUTPut node is used to reference measurements made at the instruments output.

**19.4.5.2.1 [:MAGNitude] <numeric\_value>**

SOURce:CORRection:LOSS:OUTPut:MAGNitude

This is the magnitude value of the correction for LOSS, GAIN. The units associated with LOSS and GAIN are the current relative amplitude units.

At \*RST, this value is set to zero.

**19.4.5.2.2 :PHASe <numeric\_value>**

SOURce:CORRection:LOSS:OUTPut:PHASe

This is the phase value of the correction for LOSS or GAIN. The units associated with this are the currently selected angle units.

At \*RST, this value is set to zero.

**19.4.6 :EDELay**

SOURce:CORRection:EDELay

The Electrical DELay is used to compensate for delays in the signal path.

This command is used to correct external factors that have not been accounted for in the correction set.

TIME and DISTance are coupled by the equation:

$$\text{TIME} = \text{RVELOCITY} * \text{DISTANCE}$$

RVELOCITY is not coupled to either TIME or DISTANCE.

**19.4.6.1 [:TIME] <numeric\_value>**

SOURce:CORRection:EDELay:TIME

This command will set the electrical delay with the time parameter given. The units for this command are seconds. A value accepted here will modify the DISTANCE setting because they are the same parameter in different forms.

At \*RST, the value of this setting is zero.

**19.4.6.2 :DISTance <numeric\_value>**

SOURce:CORRection:EDELay:DISTance

This command will set the electrical delay with the distance parameter given. The effective delay will be computed using this parameter and the relative velocity set in the instrument. The units for this command are meters. A value accepted here will modify the time setting because they are the same parameter in different forms. Implementation of this command requires implementation of CORRection:EDELay[:TIME].

At \*RST, the value of this setting is zero.

#### 19.4.6.3 :STATe <Boolean>

SOURce:CORRection:EDELay:STATe

This function is used to enable or disable the correction for electrical delay.

At \*RST, this function is set to OFF.

#### 19.4.7 :RVELocity

SOURce:CORRection:RVELocity

This command corrects for the relative velocity of the medium connecting to the Device Under Test. This factor will be used in any delay corrections performed by the instrument.

This command is used to correct external factors that have not been accounted for in the correction set.

TIME and DISTance are coupled by the equation:

$$\text{TIME} = \text{RVELocity} * \text{DISTance}$$

RVELocity is not coupled to either TIME or DISTance.

#### 19.4.7.1 :MEDIUM COAX|WAVeguide

SOURce:CORRection:RVELocity:MEDIUM

This command selects the method of correction for any operation that uses relative velocity. For example, for COAX a linear phase shift will be applied, but for WAVeguide (a dispersive medium) a different phase shift is calculated.

At \*RST, this value is device dependent.

#### 19.4.7.2 :COAX <numeric\_value>

SOURce:CORRection:RVELocity:COAX

This command sets the relative velocity for coaxial transmission lines. The parameter is unitless.

At \*RST, the value of this setting is one.

#### 19.4.7.3 :WAVeguide <numeric\_value>

SOURce:CORRection:RVELocity:WAVeguide

This command sets the relative velocity factor for waveguide. The parameter is unitless.

At \*RST, the value of this setting is one.

#### 19.4.7.3.1 :FCUToff <numeric\_value>

SOURce:CORRection:RVELocity:WAVeguide:FCUToff

This command specifies the frequency cutoff of the waveguide medium. The units are Hertz.

At \*RST, the value of this setting is device dependent

#### 19.4.7.4 :STATe <Boolean>

SOURce:CORRection:RVELocity:STATe

This function is used to enable or disable the relative velocity correction.

## **1999 SCPI Command Reference**

At \*RST, the value of this setting is OFF.

## 19.5

**CURRent Subsystem**

SOURce:CURRent

This subsection controls the signal amplitude characteristics of the source.

There is only one subsystem for each of the main amplitude characteristics of the signal. Thus, AC voltage and DC voltage are both placed under the VOLTage subsystem. A device which implemented both simultaneously would implement them as two separate sources, one AC and one DC.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>COMMENTS</b>
:CURRent		
:ATTenuation	<numeric_value>	
:AUTO	<Boolean> ONCE	
:ALC		
[:STATe]	<Boolean>	
:SEARch	<Boolean> ONCE	
:SOURce	INTernal DIODe PMETer MMHead	
:BANDwidth	<numeric_value>	
[:BWIDth]		
:AUTO	<Boolean> ONCE	
:CENTer	<numeric_value>	
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	<numeric_value>	
:AUTO	<Boolean> ONCE	1991
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:TRIGgered		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:LIMIT		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:STATe	<Boolean>	
:MANual	<numeric_value>	
:MODE	FIXed SWEep LIST	
:PROTection		
[:LEVel]	<numeric_value>	
:STATe		
:TRIPped?		[query only]

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>COMMENTS</b>
:CLEar	<numeric_value>	[no query]
:RANGE	<Boolean> ONCE	
:AUTO		
:REFERENCE	<numeric_value>	
:STATE	<Boolean>	
:SLEW	<numeric_value>	
:SPAN	<numeric_value>	
:HOLD	<Boolean>	
:LINK	CENTER STARt STOP	
:FULL		[no query]
:STARt	<numeric_value>	
:STOP	<numeric_value>	

### 19.5.1 :ATTenuation <numeric\_value>

SOURce:CURREnt:ATTenuation

Sets the ATTenuation level. Note that when increasing the level by 10 dB the magnitude of the outgoing signal as well as the LEVel will be decreased by 10 dB.

Default units are as determined in the UNIT system. This is coupled to LEVel.

#### 19.5.1.1 :AUTO <Boolean>

SOURce:CURREnt:ATTenuation:AUTO

Couples the attenuator to LEVel.

Programming a specified attenuation sets AUTO OFF.

See SOURce subsystem introduction for additional explanation.

At \*RST, AUTO is set to ON.

### 19.5.2 :ALC

SOURce:CURREnt:ALC

This subsystem command controls the automatic leveling control of the source.

#### 19.5.2.1 [:STATe] <Boolean>

SOURce:CURREnt:ALC:STATe

Controls whether the ALC loop controls the output level.

At \*RST, this value is device-dependent.

#### 19.5.2.2 :SEARch <Boolean>|ONCE

SOURce:CURREnt:ALC:SEARch

SEARch enables a form of leveling where the output level is calibrated by momentarily closing the leveling loop. Once the correct amount of level adjustment has been determined to produce the desired output level, the leveling loop is opened.

When ON, the search routine will be done anytime the output level requires change (for example, when LEVel is changed).

Selecting SEARch ONCE will have the effect of setting SEARch to ON and then OFF. A modulator setting is determined by the system which will not be changed until the output level is explicitly reprogrammed.

At \*RST, this value is set to OFF.

### 19.5.2.3 :SOURce INTERNAL|DIODe|PMETer|MMHead

SOURce:CURREnt:ALC:SOURce

Selects the source of the feedback signal for ALC. The parameters have the following meanings:

- INTERNAL — The ALC feedback signal is measured from a point inside the source.
- DIODe — The ALC feedback is being fed from an external diode detector.
- PMETer — The ALC signal is coming from a power meter.
- MMHead — A millimeter head, with built in leveling, is being used to supply the ALC signal.

At \*RST, this value is INTERNAL.

### 19.5.2.4 :BANDwidth|:BWIDth <numeric\_value>

SOURce:CURREnt:ALC:BANDwidth

Controls the bandwidth of the ALC feedback signal. This parameter is in units of Hz.

At \*RST, this value is device-dependent.

#### 19.5.2.4.1 :AUTO <Boolean>|ONCE

SOURce:CURREnt:ALC:BANDwidth:AUTO

Couples the bandwidth of the ALC feedback signal to instrument-dependent parameters. When AUTO is ON, the best bandwidth is chosen to reflect the current instrument state.

Explicitly selecting a value for ALC:BANDwidth sets AUTO OFF.

At \*RST, this value is set to ON.

### 19.5.3 :CENTer <numeric\_value>

SOURce:CURREnt:CENTer

Sets the center amplitude.

At \*RST, this value is set to MINimum.

### 19.5.4 [:LEVel]

SOURce:CURREnt:LEVel

This subsystem is used to control the signal amplitude when the source is operating in a continuous or FIXed MODE.

#### 19.5.4.1 [:IMMediate]

SOURce:CURREnt:LEVel:IMMediate

Indicates that the subsequent specification of a new signal amplitude is to be processed by the device without waiting for further commands.

**19.5.4.1.1 [:AMPLitude] <numeric\_value>**

SOURce:CURRent:LEVel:IMMEDIATE:AMPLitude

Sets the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal.

Note the optional nodes exist to enable CURRent to be directly followed by a <numeric\_value>. This is useful for programming simple devices in a straightforward manner.

At \*RST, the signal being sourced should be set to a “safe” condition. This is generally achieved by setting the amplitude to its minimum value in conjunction with OUTPut:STATe subsystem.

**19.5.4.1.1.1 :AUTO <Boolean>|ONCE**

SOURce:CURRent:LEVel:IMMEDIATE:AMPLitude:AUTO

If AUTO ON is selected, the voltage setting will be adjusted automatically when a new current setting (with the existing voltage setting) exceeds the maximum power limit. If AUTO OFF is selected a new current setting will cause an execution error -221 (Settings conflict), and the instrument state is device dependent when the maximum power limit is exceeded.

\*RST condition: AUTO OFF

**19.5.4.1.2 :OFFSet <numeric\_value>**

SOURce:CURRent:LEVel:IMMEDIATE:OFFSet

Sets the non-time varying component of the signal that is added to the time varying signal specified in AMPLitude, in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. When SOURce:FUNCtion is DC, the effect of the OFFSet is device-dependent.

At \*RST, the signal being sourced should be set to a “safe” condition. This is generally achieved by setting the offset to its minimum value in conjunction with OUTPut:STATe subsystem.

**19.5.4.1.3 :HIGH <numeric\_value>**

SOURce:CURRent:LEVel:IMMEDIATE:HIGH

Sets the more positive peak of a time varying signal. It is used in conjunction with LOW.

At \*RST, the signal being sourced should be set to a “safe” condition. This is achieved by setting this to a minimum value in conjunction with OUTPut:STATe subsystem.

**19.5.4.1.4 :LOW <numeric\_value>**

SOURce:CURRent:LEVel:IMMEDIATE:LOW

Sets the more negative peak of a time varying signal, it is used in conjunction with HIGH.

At \*RST, the signal being sourced should be set to a “safe” condition. This is achieved by setting this to a minimum value in conjunction with OUTPut:STATe subsystem.

**19.5.4.2 :TRIGgered**

SOURce:CURRent:LEVel:TRIGgered

Indicates that subsequent specification of a new signal amplitude is to be transferred to the IMMEDIATE value upon receipt of a trigger signal.

At \*RST, after the receipt of an ABORT command or upon being triggered, the value in the TRIGGERED subsystem tracks the values in the IMMEDIATE subsystem until a command in the TRIGGERED subsystem is received. The purpose of tracking is so that spurious triggers do not change the value of LEVel unexpectedly.

**19.5.4.2.1 [:AMPLitude] <numeric\_value>**

SOURce:CURRent:LEVel:TRIGgered:AMPLitude

Sets the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal.

**19.5.4.2.2 :OFFSet <numeric\_value>**

SOURce:CURRent:LEVel:TRIGgered:OFFSet

Sets the non-time varying component of the signal that is added to the time varying signal specified in AMPLitude. The units are set to the default value, or to a different value under the UNIT subsystem. When SOURce:FUNCTION is DC, the effect of OFFSet is device-dependent.

**19.5.4.2.3 :HIGH <numeric\_value>**

SOURce:CURRent:LEVel:TRIGgered:HIGH

This command is used to set the more positive peak of a time varying signal. It is used in conjunction with LOW.

**19.5.4.2.4 :LOW <numeric\_value>**

SOURce:CURRent:LEVel:TRIGgered:LOW

This command is used to set the more negative peak of a time varying signal. It is used in conjunction with HIGH.

**19.5.5 :LIMIT**

SOURce:CURRent:LIMit

Sets the maximum bounds on the output value. Setting a larger value will cause the output level to be clamped to the LIMIT value.

**19.5.5.1 [:AMPLitude] <numeric\_value>**

SOURce:CURRent:LIMit:AMPLitude

Sets the limit on the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal. OFFSet may only be used with AMPLitude, where AMPLitude is used to specify a time varying load.

## 1999 SCPI Command Reference

Explicit reference to the AMPLitude node is optional. Any device that implements this subsystem must accept and process commands, with or without the AMPLitude node, and respond to them in the same way.

At \*RST, this value is device-dependent.

### 19.5.5.2 :OFFSet <numeric\_value>

SOURce:CURRent:LIMit:OFFSet

Sets a non-time varying component limit of signal that is added to the time varying signal specified in AMPLitude. The units are set to the default value, alternately to a different value under the UNIT subsystem.

At \*RST, this value is device-dependent.

### 19.5.5.3 :HIGH <numeric\_value>

SOURce:CURRent:LIMit:HIGH

Sets the more positive peak limit of a time varying signal, it is used in conjunction with LOW.

At \*RST, this value is device-dependent.

### 19.5.5.4 :LOW <numeric\_value>

SOURce:CURRent:LIMit:LOW

Sets the more negative peak limit of a time varying signal, it is used in conjunction with HIGH.

At \*RST, this value is device-dependent.

### 19.5.5.5 :STATe <Boolean>

SOURce:CURRent:LIMit:STATe

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

### 19.5.6 :MANual <numeric\_value>

SOURce:CURRent:MANual

Allows manual adjustment of the amplitude between the sweep limits. The actual amplitude level is determined by the parameter only if SWEep:MODE is set to MANual and the amplitude MODE is set to SWEep. If the sweep limits are changed such that the manual amplitude value would be outside the limits, the manual amplitude value will be set to the nearest limit.

At \*RST, this value is device-dependent.

### 19.5.7 :MODE FIXed|SWEep|LIST

SOURce:CURRent:MODE

Determines which set of commands control the amplitude subsystem. If FIXed is selected, the amplitude is determined by the LEVel command under the appropriate subsystem. If SWEep is selected, the source is in the swept mode and amplitude is determined by START, STOP, CENTER, SPAN, and MANual commands under the appropriate subsystem. If LIST is

selected, the amplitude values are determined by the appropriate amplitude list (such as LIST:).

At \*RST, this value is set to FIXed.

#### 19.5.8 :PROtection

SOURce:CURRent:PROtection

Controls the protection circuits.

##### 19.5.8.1 [:LEVel] <numeric\_value>

SOURce:CURRent:PROtection:LEVel

This command sets the output level at which the output protection circuit will trip.

##### 19.5.8.2 :STATE <Boolean>

SOURce:CURRent:PROtection:STATE

Controls whether the output protection circuit is enabled.

At \*RST, this value is set to ON.

##### 19.5.8.3 :TRIPPed?

SOURce:CURRent:PROtection:TRIPPed?

This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

TRIPPed only has a query form and thus has no associated \*RST condition.

##### 19.5.8.4 :CLEAR

SOURce:CURRent:PROtection:CLEAR

Causes the protection circuit to be cleared.

This command is an event and has no associated \*RST condition.

#### 19.5.9 :RANGE <numeric\_value>

SOURce:CURRent:RANGE

Sets a range for the output amplitude. This command is very hardware-specific.

##### 19.5.9.1 :AUTO <Boolean>|ONCE

SOURce:CURRent:RANGE:AUTO

Couples the RANGE to an instrument-determined value. When AUTO is ON, the best range is chosen to reflect the current instrument state.

Explicitly selecting a value for RANGE sets AUTO OFF.

At \*RST, this value is set to ON.

##### 19.5.10 :REFERENCE <numeric\_value>

SOURce:CURRent:REFerence

Sets a reference value which, if STATE is ON, allows all amplitude parameters to be queried/set as relative to the reference value.

At \*RST, this value is device-dependent.

**19.5.10.1 :STATe <Boolean>**

SOURce:CURRent:REFerence:STATe

Determines whether amplitude is measured/output in absolute or relative mode. If STATe is ON, then amplitude is referenced to the value set in REFerence.

At \*RST, this value is set to OFF.

**19.5.11 :SLEW <numeric\_value>**

SOURce:CURRent:SLEW

Sets the slew rate of the output change when a new output level is programmed. The units are in (the currently active) amplitude unit/sec.

At \*RST, this value is device-dependent.

**19.5.12 :SPAN <numeric\_value>**

SOURce:CURRent:SPAN

Sets the amplitude span. If the current amplitude unit is logarithmic (dBm, dBuV, etc), then the unit of SPAN is dB. Otherwise SPAN is programmed in the current amplitude unit.

At \*RST, this value is set to 0.

**19.5.12.1 :FULL**

SOURce:CURRent:SPAN:FULL

When this command is received, STARt amplitude is set to its minimum value and STOP amplitude is set to its maximum value. CENTER amplitude and SPAN are set to their coupled values.

This command is an event rather than a state. Therefore it has no associated query, or meaning at \*RST.

**19.5.12.2 :HOLD <Boolean>**

SOURce:CURRent:SPAN:HOLD

Provides a mechanism to prevent the SPAN from being changed implicitly by the defined coupling between STARt, STOP, CENTER and SPAN. When HOLD is set to ON, SPAN shall only be changed by explicitly sending the SPAN command.

At \*RST, this value is set to OFF.

**19.5.12.3 :LINK CENTER|STARt|STOP**

SOURce:CURRent:SPAN:LINK

Allows the default couplings for SPAN to be overridden. LINK selects the parameter, either CENTER, STARt or STOP, that shall not be changed when SPAN's value is changed. For example, if LINK is set to STARt then changing SPAN shall cause CENTER and STOP, not STARt to change.

At \*RST, this value is set to CENTER, to be compatible with the default definition for the couplings.

**19.5.13 :STARt <numeric\_value>**

SOURce:CURRent:STARt

Sets STARt amplitude.

## 1999 SCPI Command Reference

At \*RST, this value is set to MINimum.

### 19.5.14 :STOP <numeric\_value>

SOURce:CURREnt:STOP

Sets STOP amplitude.

At \*RST, this value is set to MINimum.

## 19.6 DM Subsystem

SOURce:DM

The Digital Modulation subsystem is used to set the modulation controls and the parameters associated with the modulating signal(s). This modulation embraces a number of digital modulation techniques, which are described in the general terms of I and Q components. The keywords to describe the carrier signal exist in other subsystems (such as, FREQuency and PHASe).

KEYWORD	PARAMETER FORM	COMMENTS
:DM		
:FORMAT	BPSK PSK2 QPSK PSK4 PSK8 QAM16 QAM64 QAM256 QAM1024 PRS9 PRS25 PRS49 PRS81	
:STATE	<Boolean>	
:SOURce	EXTernal PRBS CALibrate	
:FILTter		
[:SOURce]	INTernal EXTernal	
:ICORrection	<numeric_value>	
:QCORrection	<numeric_value>	
:IQRatio		
:STATE	<Boolean>	
[:MAGNitude]	<numeric_value>	
:LEAKage		
:STATE	<Boolean>	
[:MAGNitude]	<numeric_value>	
:ANGLE	<numeric_value>	
:QUADrature		
:STATE	<Boolean>	
:ANGLE	<numeric_value>	
:COUPling		
[:ALL]	AC DC GROund	
:DATA	AC DC GROund	
:CLOCK	AC DC GROund	
:THRehold		
[:ALL]	<numeric_value>	
:DATA	<numeric_value>	
:CLOCK	<numeric_value>	
:DMODE	SERial PARallel	
:FRAMe		
:SOURce	INTernal EXTernal	
:POLarity		
[:ALL]	NORMal INVerted	
:I<n>	NORMal INVerted	
:Q<n>	NORMal INVerted	
:IClock	NORMal INVerted	
:QClock	NORMal INVerted	

KEYWORD	PARAMETER FORM	COMMENTS
	:CLOCk :SOURce	NONE EXTernal
19.6.1	<b>:FORMat &lt;modulation format&gt;</b> SOURce:DM:FORMat	Sets the format type to be employed. It is expandable to any PRS<n>, QAM<n>, QPR<n> or PSK<n>. QPSK is an alias for PSK4 and BPSK is an alias for PSK2.  At *RST, this value is device-dependent.
19.6.2	<b>:STATe &lt;Boolean&gt;</b> SOURce:DM:STATe	Turns the digital modulation subsystem ON or OFF. Turning digital modulation ON will not automatically turn OFF any other types of modulation. Turning any or all modulations types ON or OFF must be done explicitly. Where a type of modulation is active, turning ON another type of modulation will add to the set of active modulations if that combination of capabilities are legal in the device. Otherwise, an execution error -221 is generated and the instrument state is left undefined.  At *RST, this value must be set to OFF.
19.6.3	<b>:SOURce EXTernal PRBS CALibrate</b> SOURce:DM:SOURce	Sets the source of digital modulation data. EXTernal means the digital data comes in externally. The data can be also generated internally with PRBS (a pseudo-random bit sequencer). CALibrate is a service feature for calibrating the baseband of the device.  At *RST, this value must be set to EXTernal.
19.6.4	<b>:FILTer</b> SOURce:DM:FILTter	This subsystem is used to control the I and Q filters.
19.6.4.1	<b>[:SOURce] INTernal EXTernal</b> SOURce:DM:FILTter:SOURce	This mnemonic allows a choice between INTernal or EXTernal I and Q filters.  At *RST, this value must be set to INTernal.
19.6.4.2	<b>:ICORrection &lt;numeric_value&gt;</b> SOURce:DM:FILTter:ICORrection	This mnemonic allows the entry of gain information about the EXTernal I filter. This has no effect on the signal when INTernal filter source is selected.  AT *RST, this value must be set to 0.0DB.
19.6.4.3	<b>:QCORrection &lt;numeric_value&gt;</b> SOURce:DM:FILTter:QCORrection	This mnemonic allows the entry of gain information about the EXTernal Q filter. This has no effect on the signal when INTernal filter source is selected.

## 1999 SCPI Command Reference

At \*RST, this value must be set to 0.0DB.

### 19.6.5 :IQRatio

SOURce:DM:IQRatio

This subsystem controls the gain imbalance between the I and Q channels.

#### 19.6.5.1 :STATe <Boolean>

SOURce:DM:IQRatio:STATe

Turns ON and OFF the gain imbalance correction.

At \*RST, this value must be set to OFF.

#### 19.6.5.2 [:MAGNitude] <numeric\_value>

SOURce:DM:IQRatio:MAGNitude

Specifies the correction for the gain imbalance between the I and Q channels. This value is equal to I/Q.

At \*RST, this value must be set to 0.0DB.

### 19.6.6 :LEAKage

SOURce:DM:LEAKage

Carrier leakage is an offset added to the modulation pattern. Carrier leakage is measured as the magnitude relative to the full scale of the original signal. Hence a carrier leakage magnitude of 0dB would mean that the offset in the signal would be 1 full scale. This subsystem allows specification of the carrier leakage in angle-magnitude notation.

#### 19.6.6.1 :STATe <Boolean>

SOURce:DM:LEAKage:STATe

This command turns carrier leakage ON and off.

At \*RST, this value must be set to OFF.

#### 19.6.6.2 [:MAGNitude] <numeric\_value>

SOURce:DM:LEAKage:MAGNitude

Specifies the magnitude of carrier leakage relative to the full scale of the signal. 0.0dB means add 1 full scale of carrier leakage. 40dB would mean add (full scale - 40) amount of dBm.

At \*RST, this value is device-dependent.

#### 19.6.6.3 :ANGLE <numeric\_value>

SOURce:DM:LEAKage:ANGLE

Specifies the angle of carrier leakage.

At \*RST, this value is device-dependent.

### 19.6.7 :QUADrature

SOURce:DM:QUADrature

This subsystem controls quadrature correction.

**19.6.7.1 :STATE <Boolean>**

SOURce:DM:QUADrature:STATe

Turns ON or OFF the quadrature angle correction.

At \*RST, this value must be set to OFF.

**19.6.7.2 :ANGLE <numeric\_value>**

SOURce:DM:QUADrature:ANGLE

Specifies the angle of quadrature correction.

At \*RST, this value must be set to 0.0DEG.

**19.6.8 :COUPling**

SOURce:DM:COUPling

This subsystem controls the input coupling for the digital modulation inputs.

**19.6.8.1 [:ALL] AC|DC|GROund**

SOURce:DM:COUPling:ALL

Sets the coupling of both the clock and data inputs to the instrument. This command is an event, and therefore does not have an associated \*RST condition

**19.6.8.2 :DATA AC|DC|GROund**

SOURce:DM:COUPling:DATA

Sets the coupling of the data inputs to the instrument.

At \*RST, this value is device-dependent.

**19.6.8.3 :CLOCK AC|DC|GROund**

SOURce:DM:COUPling:CLOCK

Sets the coupling of the clock inputs to the instrument.

At \*RST, this value is device-dependent.

**19.6.9 :THreshold**

SOURce:DM:THreshold

This subsystem controls the detector thresholds for the digital modulation inputs.

**19.6.9.1 [:ALL] <numeric\_value>**

SOURce:DM:THreshold:ALL

Sets the threshold of both the clock and data inputs to the instrument.

**19.6.9.2 :DATA <numeric\_value>**

SOURce:DM:THreshold:DATA

Sets the threshold of the data inputs to the instrument.

At \*RST, this value is device-dependent.

**19.6.9.3 :CLOCK <numeric\_value>**

SOURce:DM:THreshold:CLOCK

Sets the threshold of the clock inputs to the instrument.

At \*RST, this value is device-dependent.

**19.6.10 :DMODE SERial|PARallel**

SOURce:DM:DMODE

This command sets the data transfer mode. SERial means data is transferred serially and PARallel means the data is transferred in a parallel fashion.

At \*RST, this value is device-dependent.

**19.6.11 :FRAMe**

SOURce:DM:FRAMe

This subsystem controls the framing for serialized digital modulation.

**19.6.11.1 :SOURce INTernal|EXTernal**

SOURce:DM:FRAMe:SOURce

Selects the source frame when data is transferred serially. INTernal selects autoframing, while EXTernal selects user-supplied framing.

**19.6.12 :POLarity [:ALL] NORMal|INVerted**

SOURce:DM:POLarity

Sets the polarity of all the clock and data inputs to the instrument.

**19.6.12.1 :I<n> NORMal|INVerted**

SOURce:DM:POLarity NORMal:I&lt;n&gt;

Selects the polarity of data inputs for the I channel. <n> selects which data input where n can be any number from 0-1 (largest DM format log4). For example, the range for n is 0-4 when the most complex mod format allowed is QAM1024.

At \*RST, all I polarities must be set to NORM.

**19.6.12.2 :Q<n> NORMal|INVerted**

SOURce:DM:POLarity NORMal:Q&lt;n&gt;

Selects the polarity of data inputs for the Q channel. <n> selects which data input where n can be any number from 0-1 (largest DM format log4). For example, the range for n is 0-4 when the most complex mod format allowed is QAM1024.

At \*RST, all Q polarities must be set to NORM.

**19.6.12.3 :IClock NORMal|INVerted**

SOURce:DM:POLarity NORMal:IClock

Selects the polarity of the I channel clock input.

At \*RST, IClock polarity must be set to NORM.

**19.6.12.4 :QClock NORMal|INVerted**

SOURce:DM:POLarity NORMal:QClock

Selects the polarity of the Q channel clock input.

At \*RST, QClock polarity must be set to NORM.

**19.6.13 :CLOCK**

SOURce:DM:CLOCK

This subsystem controls the clock for digital modulation.

## 1999 SCPI Command Reference

### 19.6.13.1 :SOURce NONE|INTernal|EXTernal

SOURce:DM:CLOCK:SOURce

Selects either no clock (asynchronous operation) or external clocking.

At \*RST, this value is device-dependent.

**19.7 FM Subsystem**

SOURce:FM

The Frequency Modulation subsystem is used to set the modulation controls and the parameters associated with the modulating signal(s). The keywords to describe the carrier signal exist in other subsystems (such as FREQuency and PHASe).

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>COMMENTS</b>
:FM		
:COUPling	AC DC GROund	
[:DEViation]	<numeric_value>	
:EXTernal		
:COUPling	AC DC GROund	
:IMPedance	<numeric_value>	
:POLarity	NORMAl INVerted	
:INTernal		
:FREQuency	<numeric_value>	
:MODE	LOCKed UNLocked	
:POLarity	NORMAl INVerted	
:SENSitivity	<numeric_value>	
:SOURce	EXTernal INTernal{ ,EXTernal ,INTernal}	
:STATe	<Boolean>	

**19.7.1 :COUPling AC|DC|GROund**

SOURce:FM:COUPling

COUPling at this level is used to set the coupling between the modulator and the modulating signal. The modulating signal may be the sum of several signals, either as internal or external sources. If COUPling is set to DC, then both AC and DC components of the signal pass. AC coupling passes only the AC component of the signal. The GROund parameter allows the modulation to be turned ON without the modulating signal connected.

At \*RST, this value is device-dependent.

**19.7.2 [:DEViation] <numeric\_value>**

SOURce:FM:DEViation

Sets the modulation DEViation of an FM signal. The unit for DEViation is Hertz (Hz). This command is used where the DEViation can be controlled independently from the modulating signal voltage level. Alternatively, it is used to set a device capability that can indicate when the desired DEViation has been achieved.

At \*RST, this value is device-dependent.

**19.7.3 :EXTernal**

SOURce:FMwEXTernal

Controls the specified external signal source. Where multiple external modulation sources are available, the EXTernal keyword may have a numeric suffix signifying the particular external signal source. If no number is given, a 1 is assumed.

**19.7.3.1 :COUPLing AC|DC|GROund**

SOURce:FM:EXTernal:COUpling

COUPLing at this level sets the coupling for only the specified external signal source. If COUPLing is set to DC, then both AC and DC Modulation components of the signal pass. AC coupling passes only the AC component of the signal. The GROund parameter allows the specified external source to be turned ON without being connected.

At \*RST, this value is device-dependent.

**19.7.3.2 :IMPedance <numeric\_value>**

SOURce:FM:EXTernal:IMPedance

Sets the impedance of the specified external signal source. The units for IMPedance is Ohms.

At \*RST, this value is device-dependent.

**19.7.3.3 :POLarity NORMAL|INVerted**

SOURce:FM:EXTernal:POLarity

POLarity at this level sets the POLarity for only the specified external signal source.

POLarity sets the relationship between the polarity of the applied modulation voltage and the resulting “direction” of modulation. The definition of “normal” polarity for FM modulated signal is a positive voltage which causes the instantaneous output frequency to increase.

At \*RST, this value must be set to NORMAL on.

**19.7.4 :INTernal**

SOURce:FM:INTernal

The INTernal subsystem is used to control the specified internal signal source. Where multiple internal modulation sources are available, the INTernal keyword may have a numeric suffix signifying the particular internal signal source. If no number is given, a 1 is assumed.

**19.7.4.1 :FREQuency <numeric\_value>**

SOURce:FM:INTernal:FREQuency

Sets the frequency of the specified internal signal source. Some instruments may also allow the internal signal source(s) to be controlled as a subsystem(s). The unit of FREQuency is Hertz (Hz).

At \*RST, this value is device-dependent.

**19.7.5 :MODE LOCKed|UNLocked**

SOURce:FM:MODE

Sets the synthesis mode employed in generating the FM signal. If MODE is set to LOCKed then a PLL is used, in synthesizing the FM signal, to prevent frequency drift. If UNLocked the phase locked loop is inactive.

At \*RST, this value is device-dependent.

**19.7.6 :POLarity NORMAL|INVerted**

SOURce:FM:POLarity

POLarity, at this level, is used to set the polarity between the modulator and the modulating signal. The modulating signal may be the sum of several signals, either as internal or external sources. POLarity sets the relationship between the polarity of the applied modulation voltage and the resulting “direction” of modulation. The definition of “normal” polarity for FM modulated signal is that a positive voltage causes the instantaneous output frequency to increase.

At \*RST, this value must be set to NORMAL.

**19.7.7 :SENSitivity <numeric\_value>**

SOURce:FM:SENSitivity

Controls the modulation deviation by setting a sensitivity Modulation to the modulation signal voltage level. The unit for SENSitivity is Hertz/Volt (Hz/V).

At \*RST, this value is device-dependent.

**19.7.8 :SOURce EXTERNAL|INTERNAL{,EXTERNAL|,INTERNAL}**

SOURce:FM:SOURce

Selects the source for the modulating signal, and it may specify a single source or a number of sources. The specified sources, in the SOURce command, are all selected and turned on. Any sources from a previous selection that are not in the current selection list are deselected and turned off. If multiple sources are selected and the device does not support the combination requested in the SOURce command, an execution error -221 must be generated.

Where multiple internal modulation sources are available, individual INTERNAL sources must be distinguished by a unique number suffix. If no number suffix is provided in a command, a 1 must be assumed.

Where multiple external modulation sources are available, individual EXTERNAL sources must be distinguished by a unique number suffix. If no number suffix is provided in a command, a 1 must be assumed.

At \*RST, this value is device-dependent.

**19.7.9 :STATE <Boolean>**

SOURce:FM:STATe

Turns frequency modulation ON or OFF. Turning FM ON will not automatically turn OFF any other types of modulation. Turning any or all modulations types ON or OFF must be done explicitly. Where a type of modulation is active turning ON another type of modulation, will add to the set of active modulations if that combination of capabilities are legal in the device. Otherwise, an execution error -221 is generated, and the instrument state is device-dependent.

At \*RST, this value must be set to OFF.

## 19.8 FORCe Subsystem

SOURce:FORCe

This node controls the force of the device.

KEYWORD	PARAMETER FORM	COMMENTS
:FORCe		1999
:CDOWn		1999
:INITiate		[event; no query]1999
:NRUNs	<numeric_value>	1999
:RLDeviation		1999
:FACCeptance	<numeric_value>	1999
:INITiate		[event; no query]1999
:RMAXimum	<numeric_value>	1999
:RVERify	<numeric_value>	1999
:SOFFset	<numeric_value>	1999
:CONFigure		1999
:ABRake		1999
:GAIN	<numeric_value>	1999
[:STATe]	<Boolean>	1999
:THReShold	<numeric_value>	1999
:GRADE		1999
:LEVel	<numeric_value>	1999
:SOURce	INTERNAL EXTERNAL	1999
[:STATe]	<Boolean>	1999
[:VEHicle]		1999
:DCoefficient	<numeric_value>{,<numeric_value>}	1999
:DINertia	<numeric_value>	1999
[:STATe]	<Boolean>	1999
:TCoefficient	<numeric_value>{,<numeric_value>}	1999
:TINertia	<numeric_value>	1999
:WEIGht	<numeric_value>	1999
:INITiate		[event; no query]1999
[:LEVel]	<numeric_value>	1999
:RLSimulation		[event; no query]1999
:INITiate		[event; no query]1999

## 19.8.1 :CDOWn

:SOURce:FORCe:CDOWn

Coast Down

This node contains the commands required to run a Dynamometer Coast Down procedure. A coast down measures the force, power, and elapsed time over multiple speed intervals as a vehicle or dynamometer roll coasts in road load simulation from a starting speed to a final speed. This procedure uses the values set in the :SOURce:FORCe:CONFigure node and the CDOWnSpeed table. The procedure results are stored in the CDOWnResult table. Measured coefficients are placed in the memory table CDOWnMCoeff and are determined after each

coast down run. CDownDCoeff should be calculated after coast down run according to SAE J2264, but do not adjust the dyno-setting coefficients.

### 19.8.1.1 :INITiate

:SOURce:FORCe:CDOWN:INITiate

This is an overlapped command. This command initiates the coast down using the current parameters from the :CONFIGure node and CDownSpeed Table. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Coast Down Procedure bit indicating that a Coast down is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Enter the theoretical data for the defined speed interval and configured parameters in CDownResult table.
- Accelerate dynamometer to :SOURce:FORCe:CDOWN:SOFFset +(plus) the starting speed for the first coast down interval, at the rate in :SOURce:ACCELERation . If the dynamometer determines this speed is not appropriate for the defined coast down, set the DYNO questionable condition register Coastdown Speed bit and issue an error message, but continue on with the procedure.
- Coast down over the speed intervals defined in the CDownSpeed table. The first interval that has 0 for start and stop speeds ends the measurement. For each interval defined in CDownSpeed, place the measured force, power, and time CDownResult table for the coast down being executed. Remaining entries should be set to 0.
- Perform a curve fit on the CDownResult data and place the coefficients in the memory table CDownMCoeff.
- Repeat the coast down sequence for :SOURce:FORCe:CDOWN:NRUNS
- After the last measurement, issue the process complete message and wait for a command.
- When commanded out of this procedure, clear the DYNO operation condition register Coastdown Procedure bit indicating that the Coast down is not executing.

### 19.8.1.2 :SOFFset <numeric\_value>

:SOURce:FORCe:CDOWN:SOFFset

Speed Offset - m/s

This command sets how much speed offset above the top speed in the CDownSpeed Table to which the dynamometer should be accelerated before beginning the coast down. This offset allows the dynamometer response to settle before coasting into the first speed segment.

At \*RST, this value is set to 5.

**19.8.1.3 :NRUNs <numeric\_value>**

:SOURce:FORCe:CDOWn:NRUNs

This command sets the number of coast downs to run.

At \*RST, this value is set to 3.

**19.8.1.4 :RLDerivation**

:SOURce:FORCe:CDOWn:RLDerivation

Road Load Derivation

This node contains the commands required to run a Dynamometer Road Load Derivation. This procedure will run a series of coast downs with the vehicle on the dyno to determine a dyno-setting that will match the results from the track. The coast down procedure is defined by the SAE publication J2264 (See Appendix A - References for information on this publication.). Additional input for this procedure comes from the

:SOURce:FORCe:CONFigure node, CDownSpeed table, CDownDCoeff, CDownMCoeff, and CDownResult table. The initial dynamometer coefficients ("best guess") will be in the first row of CDownDCoeff prior to starting this procedure.

**19.8.1.4.1 :FACCeptance <numeric\_value>**

:SOURce:FORCe:CDOWn:RLDerivation:FACCeptance

Force Acceptance - N

This command sets the maximum acceptable force difference between the force vs. speed curve derived from the dynamometer target and measured coefficients. This is the maximum difference found at any of the specified coast down speed points.

At \*RST, this value is set to 10.

**19.8.1.4.2 :INITiate**

:SOURce:FORCe:CDOWn:RLDerivation:INITiate

This is an overlapped command. This command initiates the coast down simulation using the current parameters in the :CONFigure node and CDownSpeed Table. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Road Load Simulation Procedure bit indicating that a Road Load Derivation is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Enter the theoretical data for the defined speed interval and configured parameters in CDownResult table based on the target values in :SOURce:FORCe:CONFigure:VEHicle:TCoefficient and :TINertia.
- Accelerate dynamometer to :SOURce:FORCe:CDOWn:SOFFset +(plus) the starting speed for the first coast down interval, at the rate in :SOURce:ACCELERation . If the

dynamometer determines this speed is not appropriate for the defined coast down, set the DYNO questionable condition register Coast down Trigger Speed bit and issue an error message, but continue on with the procedure.

- Coast down over the speed intervals defined in the CDownSpeed table. The first interval that has 0 for start and stop speeds ends the measurement for this coast down.
- For each coast down, calculate the dyno setting coefficients and measured coefficients. Measured coefficients are placed in CDownMCoef, and dyno setting coefficients are placed in CDownDCoeff. Also update :SOURce:FORCe:CONFigure:VEHicle:DCoefficient.
- For the present coast down, if the absolute difference between the measured and target force limit is less than the value :SOURce:FORCe:CDOWn:RLDerivation:FACCeptance, begin verification coast downs. If not, adjust the dyno set coefficients as outlined in SAE J2264, and perform another coast down. For verification coast downs, perform up to :SOURce:FORCe:CDOWn:RLDerivation:RVERification coast downs.
- If the number of coast downs performed is equal to :SOURce:FORCe:CDOWn:RLDerivation:RMAXimum, or any of the verification runs is not within :SOURce:FORCe:CDOWn:RLDerivation:FACCeptance, the procedure failed to converge. If the procedure fails to converge set the DYNO questionable condition register bit 8 and issue an error message.
- After the last coast down measurement, issue the process complete message. Clear the DYNO operation condition register bit 8 indicating that the Road Load Derivation is not executing.
- Execute the Idle Procedure.

### 19.8.1.4.3 :RMAXimum <numeric\_value>

:SOURce:FORCe:CDOWn:RLDerivation:RMAXimum

Runs Maximum

This command sets the maximum number of coast down runs the Road Load Derivation test will run without convergence.

At \*RST, this value is set to 15.

### 19.8.1.4.4 :RVERify <numeric\_value>

:SOURce:FORCe:CDOWn:RLDerivation:RVERify

Runs to Verify

This command sets the maximum number of verification coast down runs after convergence of the Road Load Derivation procedure.

At \*RST, this value is set to 2.

**19.8.2 :CONFigure**

:SOURce:FORCe:CONFigure

This node configures the force for various procedures. For dynamometers, these procedures may be Road Load Simulation, Coast downs, etc.

**19.8.2.1 :ABRake**

:SOURce:FORCe:CONFigure:ABRake

Augmented Brake

This node contains the commands to setup the Augmented Brake. This system utilizes the dynamometer to use regenerative braking of the rolls to simulate the brakes of the non-driven wheels.

**19.8.2.1.1 :GAIN <numeric\_value>**

:SOURce:FORCe:CONFigure:ABRake:GAIN

The augmented brake gain is used to adjust how much simulated braking assistance force is required.

At \*RST, this value is set to 0.

**19.8.2.1.2 [:STATe] <Boolean>**

:SOURce:FORCe:CONFigure:ABRake[:STATe]

This command sets the augmented brake simulation state.

At \*RST, this value is set to OFF.

**19.8.2.1.3 :THRehold <numeric\_value>**

:SOURce:FORCe:CONFigure:ABRake:THRehold

This command sets the vehicle force absorption level in N to initiate a simulated braking force for the non-rotating wheel brakes.

At \*RST, this value is set to 500.

**19.8.2.2 :GRADe**

:SOURce:FORCe:CONFigure:GRADe

This node contains the commands to setup the grade input for the Road Load Simulation and [if desired] for Coast downs.

19

**19.8.2.2.1 :LEVel <numeric\_value>**

:SOURce:FORCe:CONFigure:GRADe:LEVel

This command sets the grade force in units of Percent Grade. For example, a 45 degree slope is a 100% grade. The resulting force is given by:

$$\text{FORCE} = \text{WEIGht} * \sin(\arctan(\text{GRADE}/100))$$

At \*RST, this value is set to 0.

**19.8.2.2.2 :SOURce <INTernal|EXTernal>**

:SOURce:FORCe:CONFigure:GRADe:SOURce

This command sets the source for grade. The grade set by

:SOURce:FORCe:CONFigure:GRADe:LEVel is used when the source is “INTernal”. The

host computer can vary the grade at any time during the simulation by sending a new LEVel command. If set to “EXTernal”, then the simulation will include the grade from an external analog input (Full Scale volts = 100% = 45%). “EXTernal” is used when a dynamometer is equipped with an analog input to control the grade.

At \*RST, this value is set to INTernal.

### 19.8.2.2.3 [:STATe] <Boolean>

:SOURce:FORCe:CONFigure:GRADe[:STATe]

This command sets the grade simulation state. If the state is OFF, the simulation does not include a grade force. If the state is ON, the simulation includes the grade force from an internal or external source.

At \*RST, this value is set to OFF.

### 19.8.2.3 [:VEHicle]

:SOURce:FORCe:CONFigure[:VEHicle]

This node configures the dynamometer vehicle parameters. These parameters are the dynamometer operating set-points [e.g., inertia, road-load, etc.].

#### 19.8.2.3.1 :DCoefficient<numeric\_value>,<numeric\_value>,<numeric\_value> [,<numeric\_value>]

:SOURce:FORCe:CONFigure:VEHicle:DCoefficient

Dynamometer-setting Coefficient(s) - N, N/(m/s), N/(m/s)<sup>2</sup>[,N/(m/s)<sup>3</sup> ]

This command sets the coefficients of the road-load polynomial for the force to be simulated. The query form of this command should return the active dynamometer coefficients, which may have been determined from a Road Load Derivation procedure. When the fourth parameter is omitted, it is assumed 0.

At \*RST, this value is set to 0,0,0,0

#### 19.8.2.3.2 :DINertia <numeric\_value>

:SOURce:FORCe:CONFigure:VEHicle:DINertia

Dynamometer-setting Inertia - kg

This command sets the “Inertia” (in kg) to be simulated by the dynamometer. This is the total vehicle inertia (vehicle weight plus “inertia weight” of the rotating components) minus the inertia of the driven wheels and drive train (which will be spinning when the vehicle is driven on the dynamometer). When default is specified the dynamometer-setting inertia will be the mechanical base inertia.

At \*RST, this value is set to the Mechanical Base Inertia.

#### 19.8.2.3.3 [:STATe] <Boolean>

:SOURce:FORCe:CONFigure:VEHicle[:STATe]

This command tells the dynamometer that a vehicle is on the rolls.

At \*RST, this value is set to OFF.

**19.8.2.3.4 TCOefficient<numeric\_value>,<numeric\_value>,<numeric\_value>  
[,<numeric\_value>]**

:SOURce:FORCe:CONFigure:VEHicle:TCOefficient

Target-setting Coefficient(s) - N, N/(m/s), N/(m/s)<sup>2</sup> [,N/(m/s)<sup>3</sup> ]

This command sets the Target Road-Load Coefficients for the road load derivation or coast down procedure. These are derived from test track data for a vehicle. The host will update these values prior to executing a coast down or road load derivation. Target coefficients are never used to control the dynamometer. When the fourth parameter is omitted, it is assumed 0.

At \*RST, this value is set to 0,0,0,0

For road load derivation, TCOefficient are used to:

- calculate theoretical data in CDownResult memory table
- compare against measured coast down coefficients as defined in J2264

For coast downs with or without a vehicle, TCOefficient are used to:

- calculate theoretical data in CDownResult memory table

**19.8.2.3.5 :TINertia <numeric\_value>**

:SOURce:FORCe:CONFigure:VEHicle:TINertia

Target-setting Inertia - kg

This command sets the target inertia (kg) for a road load derivation procedure. Target inertia is defined in SAE J2264.

At \*RST, this value is set to 0.

**19.8.2.3.6 :WEIGht <numeric\_value>**

:SOURce:FORCe:CONFigure:VEHicle:WEIGht

Weight - kg

This command sets the (kg) weight of vehicle, not including the “inertia weight” of the wheels and drive-train. The value is used to calculate grade and augmented braking weight ratio (when simulating braking force of the non-driven wheels).

At \*RST, this value is set to 0.

**19.8.3 :INITiate**

:SOURce:FORCe:INITiate

This command initiates force. This is an overlapped command. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.

- Set the DYNO operation condition register Go To Force Procedure bit indicating that Go To Force Procedure is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Apply the force set in :SOURce:FORCe:LEVel.
- Maintain the force and wait for a command.
- When commanded out of this procedure, clear the DYNO operation condition register Go To Force Procedure bit indicating the Go To Force Procedure is not executing.
- Issue the process complete message.

### 19.8.4 [:LEVel] <numeric\_value>

:SOURce:FORCe[:LEVel]

Sets the force set-point in N.

At \*RST, this value is device dependent.

### 19.8.5 :RLSimulation

:SOURce:FORCe:RLSimulation

Road Load Simulation

This node sets the dynamometer into the procedure to apply a load to simulate vehicle operation on the road. These commands use the values set in the :SOURce:FORCe:CONFigure node.

#### 19.8.5.1 :INITiate

:SOURce:FORCe:RLSimulation:INITiate

This is an overlapped command. This command initiates the Road Load Simulation using the present parameters in the :CONFigure node. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Road Load Simulation Procedure bit indicating that Road Load Simulation is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Begin simulating the load defined by  
:SOURce:FORCe:CONFigure:VEHicle:DINertia;DCoefficient;GRADE;ABRake
- When commanded out of this procedure, clear the DYNO operation condition register bit 4 indicating that Road Load Simulation is not executing.
- Issue the process complete message.

## 19.9 FREQuency Subsystem

SOURce:FREQuency

The FREQuency subsystem controls the frequency characteristics of the instrument. Two types of controls are included in this subsystem. The first is frequency sensor controls, such as RANGE. The second is frequency sweep controls such as START, or tuning controls such as CW.

KEYWORD	PARAMETER FORM	COMMENTS
:FREQuency		
:CENTER	<numeric_value>	
[:CW]:FIXed]	<numeric_value>	
:AUTO	<Boolean> ONCE	
:MANual	<numeric_value>	
:MODE	CW FIXed SWEep LIST SENSe	
:MULTiplier	<numeric_value>	
:OFFSet	<numeric_value>	
:RESolution	<numeric_value>	1992
:AUTO	<Boolean>   ONCE	1992
:SPAN	<numeric_value>	
:FULL		[no query]
:HOLD	<Boolean>	
:LINK	CENTER STARt STOP	
:STARt	<numeric_value>	
:STOP	<numeric_value>	

## 19.9.1 :CENTER &lt;numeric\_value&gt;

SOURce:FREQuency:CENTER

Sets the CENTER frequency.

At \*RST, this function will be set to (MINimum+MAX)/2.

## 19.9.2 [:CW]:FIXed] &lt;numeric\_value&gt;

SOURce:FREQuency:CW

The Continuous Wave or FIXed node is used to select a frequency of a non-swept signal. This node is optional. An optional node, such as [:CW]:FIXed], implies that the device shall accept and process commands with or without the optional node and have the same result. That is, the device is required to accept the optional node, if sent, without error. A device must accept both :CW and :FIXed if it implements this node.

At \*RST, this value is device-dependent.

## 19.9.2.1 :AUTO &lt;Boolean&gt;|ONCE

SOURce:FREQuency:CW:AUTO

Couples the CW frequency to center frequency. Explicitly setting a value for FREQuency:CW sets AUTO OFF.

At \*RST, this value is set to ON.

**19.9.3 :MANual <numeric\_value>**

SOURce:FREQuency:MANual

Same as CW except limited by START and STOP. Receipt of this command will only change the frequency if SWEep:MODE MANual subsystem and FREQuency:MODE SWEep is selected. If the sweep limits are changed such that manual frequency would be outside the sweep limits, then the manual frequency will be set to the nearest end point of the sweep.

At \*RST, FREQuency:MAN is device-dependent.

**19.9.4 :MODE CW|FIXed|SWEep|LIST|SENSe**

SOURce:FREQuency:MODE

Determines which set of commands control the frequency subsystem. The settings have the following meanings:

- CW|FIXed: The frequency is determined by FREQuency[:CW] or FREQuency[:FIXed]. CW and FIXed are aliases. Both must be implemented, if either is implemented.
- SWEep: The source is in the swept mode and frequency is determined by FREQuency:STARt, STOP, CENT, SPAN, and MAN.
- LIST: The source is in LIST sequence mode and frequency is determined by LIST:FREQuency.
- SENSe: The source is coupled to the sensor. All frequency controls are determined by the SENSe:FREQuency block, except for OFFSet and MULTiplier. These two commands remain active under SOURce to allow for offset and harmonic tracking.

At \*RST, this value is set to CW.

**19.9.5 :MULTiplier <numeric\_value>**

SOURce:FREQuency:MULTiplier

Sets a reference multiplier for all other frequency settings in the instrument. All entered and displayed frequencies are affected by this command. The coupling equation is:

$$\text{Entered/Displayed frequency} = (\text{Hardware frequency} * \text{multiplier}) + \text{offset}$$

This command affects the values of all functions which affect generation of the signal. This includes such things as CW frequency and the sweep frequency controls. It does not affect things like input filter bandwidths. It does affect the settings of relative frequency parameters such as delta markers or FM deviation. Also note that this command does not affect the hardware settings of the instrument, but only the entered and displayed frequencies.

At \*RST, this value is set to 1.

**19.9.6 :OFFSet <numeric\_value>**

SOURce:FREQuency:OFFSet

Sets a reference frequency for all other absolute frequency settings in the instrument. This command affects the values of all functions which affect generation of the signal. This includes such things as CW frequency and the sweep frequency controls. It does not affect

things like input filter bandwidths. It also does not affect the settings of relative frequency parameters such as delta markers or FM deviation. Also note that this command does not affect the hardware settings of the instrument, but only the entered and displayed frequencies. The coupling equation is:

$$\text{Entered/Displayed frequency} = (\text{Hardware frequency} * \text{multiplier}) + \text{offset}$$

At \*RST, this value is set to 0.

#### 19.9.7 :RESolution <numeric\_value>

SOURce:FREQuency:RESolution

RESolution sets or queries the absolute increment/decrement, at which the frequency of the output signal can be changed. The default unit for <numeric\_value> is Hertz.

At \*RST, this value is device-dependent.

##### 19.9.7.1 AUTO <Boolean> | ONCE

SOURce:FREQuency:RESolution:AUTO

AUTO couples the RESolution to an instrument-determined value. When AUTO is ON, the best resolution is chosen to reflect the current instrument state.

Explicitly selecting a value for RESolution will set AUTO OFF.

At \*RST, this value is set to ON.

#### 19.9.8 :SPAN <numeric\_value>

SOURce:FREQuency:SPAN

Sets the frequency SPAN.

At \*RST, this function will be set to MAXimum.

##### 19.9.8.1 :FULL

SOURce:FREQuency:SPAN:FULL

When this command is received, STARt frequency is set to its minimum value, and STOP frequency is set to its maximum value. CENTer frequency and SPAN are set to their coupled values.

This command is an event, rather than a state. Therefore it has no associated query, or meaning at \*RST.

19

##### 19.9.8.2 :HOLD <Boolean>

SOURce:FREQuency:SPAN:HOLD

The HOLD command provides a mechanism to prevent the SPAN from being changed implicitly by the defined coupling between STARt, STOP, CENTer, and SPAN. When HOLD is set to ON, SPAN shall only be changed by explicitly sending the SPAN command.

At \*RST, this value is set to OFF.

##### 19.9.8.3 :LINK CENTer|STARt|STOP

SOURce:FREQuency:SPAN:LINK

This command allows the default couplings for SPAN to be overridden. LINK selects the parameter, either CENTer subsystem, STARt or STOP, that shall not be changed when

## 1999 SCPI Command Reference

SPAN's value is changed. For example, if LINK is set to STARt then changing SPAN shall cause CENTER and STOP, not STARt to change.

At \*RST, this value is set to CENTER, to be compatible with the default definition for the couplings.

### 19.9.9 :STARt <numeric\_value>

SOURce:FREQuency:STARt

Sets the STARting frequency. It also sets the lower limit of a manual sweep adjustment. If the instrument does not support negative frequency spans, having the STARt frequency greater than the STOP frequency will cause an error to be generated.

At \*RST, this value will be set to MINimum.

### 19.9.10 :STOP <numeric\_value>

SOURce:FREQuency:STOP

Sets the STOP frequency. It also sets the upper limit of a manual sweep adjustment. If the instrument does not support negative frequency spans, having the STOP frequency less than the STARt frequency will cause an error to be generated.

At \*RST, this value will be set to MAXimum.

## 19.10 FUNCtion Subsystem

SOURce:FUNCtion

The function subsystem controls the shape and attributes of the output signal. The switch settings provided by this function are not horizontally compatible and represent what the source can be configured to generate directly.

KEYWORD	PARAMETER FORM	COMMENTS
:FUNCtion		
[:SHAPe]	<source_shape>	
:MODE	<source_mode>	

### 19.10.1 [:SHAPe] <source\_shape>

SOURce:FUNCtion:SHAPe

Selects the shape of the output signal.

At \*RST, the value of FUNCtion is device-dependent.

The following SOURce functions are available for instruments:

- DC — An unvarying signal with respect to time.
- IMPulse — IMPulse is an approximation of a zero width pulse
- PCHirp — PeriodicCHirp supplies a sine sweep over the current frequency span which periodically repeats.
- PRNoise — PeriodicRandomNoise provides a randomly varying signal with respect to time which periodically repeats.
- PULSe — A pulse, or string of pulses is generated.
- RANDom — provides a randomly varying signal with respect to time.
- SINusoid — A sinusoidal signal is generated.
- SQUare — A square wave signal is generated.
- TRIangle — A triangular wave signal is generated.
- USER — The USER function allows the user to specify a periodic or arbitrary waveform definition.

### 19.10.2 :MODE <source\_mode>

SOURce:FUNCtion:MODE

Determines which signal characteristic is being controlled.

At \*RST, the value of this function is device-dependent.

The function may be set to the following values:

- VOLtage.
- CURRent.
- POWER.

19.11 **LIST Subsystem**

SOURce:LIST

The list subsystem controls automatic sequencing through associated lists of specified signal values. It is also used to specify concurrent or simultaneous operation of the specified signal in the associated lists. The various values are specified in the LIST subsystem commands. The individual points in each of the list configuration commands are combined to make one signal configuration.

The LIST subsystem controls a characteristic of the instrument when that subsystem's mode is set to LIST (such as :FREQuency:MODE LIST). Note that is permissible to have one characteristic controlled by the list subsystem, and the others controlled by a different subsystem (such as :FREQuency:MODE LIST or :POWER:MODE SWEEP).

When employing multiple lists, all lists must be of the same length. If they are not all of the same length, an error is generated at sequencing time. An exception is made for the case where the shorter list is of length 1. In this case, the particular list is treated as though it were a list of equal length, with all values being the same.

Lists are not affected by \*RST.

KEYWORD	PARAMETER FORM	COMMENTS
:LIST		
:AM		1993
[:DEPTH]	<numeric_value>{,<numeric_value>}	1993
:POINts?		[query only] 1993
:APRobe	<numeric_list>{,<numeric_list>}	1993
:POINts?		[query only] 1993
:CONCurrent	<numeric_value>{,<numeric_value>}	
:AUTO	<Boolean> ONCE	
:POINts?		[query only]
:CONTrol		1993
:APOWer	<Boolean>{,<Boolean>}	1993
:POINts?		[query only] 1993
:BLOWer	<Boolean>{,<Boolean>}	1993
:POINts?		[query only] 1993
:COMPressor	<Boolean>{,<Boolean>}	1993
:POINts?		[query only] 1993
:COUNt	<numeric_value>	
:CURRent	<numeric_value>{,<numeric_value>}	
:POINts?		[query only]
:DIRection	UP DOWN	
:DWELL	<numeric_value>{,<numeric_value>}	
:POINts?		[query only ]
:FREQuency	<numeric_value>{,<numeric_value>}	
:POINts?		[query only]
:GENeration	DSEQUence SEQUence DCONcurrent  CONCurrent	

KEYWORD	PARAMETER FORM	COMMENTS
:POWer	<numeric_value>{,<numeric_value>}	[query only] 1993
:POINts?		1993
:PULM		
[:STATe]	<Boolean>{,<Boolean>}	[query only] 1993
:POINts?		1993
:RESistance	<numeric_value>{,<numeric_value>}	
:POINts?		
:RTIMe	<numeric_value>{,<numeric_value>}	[query only] 1993
:POINts?		1993
:SEQUence	<numeric_value>{,<numeric_value>}	
:AUTO	<Boolean> ONCE	
:POINts?		
:TEMPerature	<numeric_value>{,<numeric_value>}	[query only] 1993
:POINts?		1993
:VOLTage	<numeric_value>{,<numeric_value>}	
:POINts?		[query only]

**19.11.1 :AM**

SOURce:LIST:AM

Subsystem for setting up the AM list values.

**19.11.1.1 :DEPTh <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:AM:DEPTh

Specifies the AM DEPTh points of the list.

**19.11.1.1.1 :POINts?**

SOURce:LIST:AM:DEPTh:POINTs?

Returns the number of points currently in the AM DEPtH list.

**19.11.2 :APRobe <numeric\_list>{,<numeric\_list>}**

SOURce:LIST:APRobe

APRobe selects the specified probe/s to control the temperature.

**19.11.2.1 :POINts?**

SOURce:LIST:APRobe:POINTs?

Returns the number of points currently in the APRobe list.

**19.11.3 :CONCurrent <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:CONCurrent

Defines a list which indicated those elements of the signal list which shall be active, when CONCurrent operation is selected. The points specified are indexes into the lists. For example, if 3 and 5 were selected, the third and fifth points in the signal list would be generated simultaneously. All entries in the CONCurrent list must be unique; otherwise a parameter error (-220) will be generated. The concurrent list is separate from the other lists and is not sorted, nor are the individual points associated with the other arrays except as described above.

### 19.11.3.1 :AUTO <Boolean>|ONCE

SOURce:LIST:CONCurrent:AUTO

When on, the CONCurrent list is set to 1-N, where N is the longest list.

At \*RST, AUTO ON is selected.

### 19.11.3.2 :POINts?

SOURce:LIST:CONCurrent:POINts?

Returns the number of points currently in the CONCurrent list.

### 19.11.4 :CONTrol

SOURce:LIST:CONTrol

Allows to turn on/off the devices

#### 19.11.4.1 :APOWer <Boolean>{,<Boolean>}

SOURce:LIST:CONTrol:APoWer

Turns the APoWer on or off.

##### 19.11.4.1.1 :POINts?

SOURce:LIST:CONTrol:APoWer:POINts?

Returns the number of points currently in the APoWer list.

#### 19.11.4.2 :BLOWer <Boolean>{,<Boolean>}

SOURce:LIST:CONTrol:BLOWer

Turns the BLOWer on or off.

##### 19.11.4.2.1 :POINTs?

SOURce:LIST:CONTrol:BLOWer:POINTs?

Returns the number of points currently in the BLOWer list.

#### 19.11.4.3 :COMPressor <Boolean>{,<Boolean>}

SOURce:LIST:CONTrol:COMPressor

Turns the COMPressor on or off.

##### 19.11.4.3.1 :POINTs?

SOURce:LIST:CONTrol:COMPressor:POINTs?

Returns the number of points currently in the COMPressor list.

### 19.11.5 :COUNt <numeric\_value>

SOURce:LIST:COUNt

Controls the number of times the list is sequenced when a trigger is received.

At \*RST, this value is set to 1.

### 19.11.6 :CURRent <numeric\_value>{,<numeric\_value>}

SOURce:LIST:CURRent

Specifies the current points of the lists.

### 19.11.6.1 :POINts?

SOURce:LIST:CURRent:POINts?

Returns the number of points currently in the CURRent list.

**19.11.7 :DIRection UP|DOWN**

SOURce:LIST:DIRection

Specifies the direction that the sequence list is scanned. If UP is selected, the list is scanned from the first to the last item in the sequence list. If DOWN is selected, the list is scanned from the last item to the first item in the sequence list.

At \*RST, this value is set to UP.

**19.11.8 :DWELI <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:DWELI

Specifies the dwell time points of the lists.

**19.11.8.1 :POINts?**

SOURce:LIST:DWELI:POINts?

Returns the number of points currently in the DWELI list.

**19.11.9 :FREQuency <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:FREQuency

Specifies the frequency points of the list set.

**19.11.9.1 :POINts?**

SOURce:LIST:FREQuency:POINts?

Returns the number of points currently in the FREQuency list.

Since this is only a query, there is no associated \*RST state.

**19.11.10 :GENeration DSEQUence|SEQUence|DCONcurrent|CONCurrent**

SOURce:LIST:GENeration

The GENeration command is used to select how the defined lists are applied in a particular instrument. An instrument may support one or more of the generation modes. Selecting Default SEQuence (DSEQUence) causes the instrument to cycle sequentially through the complete list in order. When SEQuence is selected, the sequence in which the list is used is specified by the list given in the SEQuence command. Selecting Default CONcurrent causes the device to simultaneously generate all items in the list. When CONCurrent is selected, the list specified by the CONCurrent command defines which elements in the list are active simultaneously.

At \*RST, GENERation shall be set to DSEQUence.

**19.11.11 :PULM**

SOURce:LIST:PULM

Subsystem for setting up the PULM list values.

**19.11.11.1 :STATE <Boolean>{,<Boolean>}**

SOURce:LIST:PULM:STATe

Specifies the PULM STATe points of the list.

**19.11.11.1.1 :POINts?**

SOURce:LIST:PULM:STATe:POINts?

Returns the number of points currently in the PULM STATe list.

**19.11.12 :POWer <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:POWer

Specifies the power/amplitude points of the lists.

**19.11.12.1 :POINts?**

SOURce:LIST:POWer:POINts?

Returns the number of points currently in the POWer list.

**19.11.13 :RESistance <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:RESistance

Specifies the resistance points of the lists.

**19.11.13.1 :POINts?**

SOURce:LIST:RESistance:POINts?

Returns the number of points currently in the RESistance list.

**19.11.14 :RTIMe <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:RTIMe

Ramp TIMe controls the time required to reach the target temperature.

**19.11.14.1 :POINTs?**

SOURce:LIST:RTIMe:POINTs?

Returns the number of points currently in the RTIMe list.

**19.11.15 :SEQUence <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:SEQUence

Defines a sequence for stepping through the list, when SEQuence operation is selected. Individual points may be specified as many times as desired in a single sequence. The points specified are indexes into the lists. For example, if 3 was selected, the third point in the frequency, dwell, and power/amplitude lists would be sequenced.

The sequence list is separate from the other lists and is not sorted, nor are the individual points associated with the other arrays except as described above.

**19.11.15.1 :AUTO <Boolean>|ONCE**

SOURce:LIST:SEQUence:AUTO

When on, the sequence list is set to 1 through N, where N is the longest list.

At \*RST, AUTO ON is selected.

**19.11.15.2 :POINts?**

SOURce:LIST:SEQUence:POINts?

Returns the number of points currently in the SEQuence list.

**19.11.16 :TEMPerature <numeric\_value>{,<numeric\_value>}**

SOURce:LIST:TEMPerature

TEMPerature is the target temperature.

**19.11.16.1 :POINTs?**

SOURce:LIST:TEMPerature:POINTs?

Returns the number of points currently in the TEMPerature list.

## 1999 SCPI Command Reference

### 19.11.17 :VOLTage<numeric\_value>{,<numeric\_value>}

SOURce:LIST:VOLTage

Specifies the voltage points of the lists.

#### 19.11.17.1 :POINts?

SOURce:LIST:VOLTage:POINts?

Returns the number of points currently in the VOLT list.

## 19.12 MARKer Subsystem

SOURce:MARKer

Selects between different markers, and adjusts the marker settings. The suffix with MARKer selects the marker with the same number associated with it. If the suffix is omitted, then marker #1 is assumed.

KEYWORD	PARAMETER FORM	COMMENTS
:MARKer		
:AMPLitude	ON OFF	
:AOFF		[no query]
:FREQuency	<numeric_value>	
:MODE	FREQuency POSIon DELTa	
:POINt	<numeric_value>	
:REFerence	<numeric_value>	
[:STATE]	<Boolean>	
:TIME	<numeric_value>	

## 19.12.1 :AMPLitude &lt;Boolean&gt;

SOURce:MARKer:AMPLitude

Controls whether the marker affects the signal. If ON, the signal is modified when the marker frequency equals the output frequency. If OFF, only the marker output port is affected.

At \*RST, this value is set to OFF.

## 19.12.2 :AOFF

SOURce:MARKer:AOFF

Turns OFF all markers. This command is an event, and therefore has no query form. All markers shall be turned OFF at \*RST.

## 19.12.3 :FREQuency &lt;numeric\_value&gt;

SOURce:MARKer:FREQuency

Controls the absolute frequency of the specified marker when MARKer:MODE is FREQuency. When MARKer:MODE is DELTa, this command controls relative frequency.

At \*RST, this value is set to MINimum.

## 19.12.4 :MODE FREQuency|POSIon|DELTa

SOURce:MARKer:MODE

This command controls whether the marker is tied to a frequency, referenced to an absolute trace point (position), or referenced to another marker (delta). If delta is selected, MARKer:REFerence determines which marker the current marker is referenced to.

At \*RST, this value is FREQuency.

## 19.12.5 :POINt &lt;numeric\_value&gt;

SOURce:MARKer:POINT

Sets the marker to the specified sweep point.

At \*RST, this value is set to MINimum.

## 1999 SCPI Command Reference

### 19.12.6 :REFerence <numeric\_value>

SOURce:MARKer:REFerence

MARK:REFerence establishes a reference marker for delta markers.

At \*RST, this value is device-dependent.

### 19.12.7 [:STATe] <Boolean>

SOURce:MARKer:STATe

Turns ON and OFF the marker specified by the MARKer command header suffix.

At \*RST, this value is set to OFF.

## 19.13 PHASe Subsystem

SOURce:PHASe

This subsystem allows control of the phase of the output signal against some reference.

KEYWORD	PARAMETER FORM	COMMENTS
:PHASe		
[:ADJust]	<numeric_value>	
:STEP	<numeric_value>	
:SOURce	INTernal[<n>] EXTernal[<n>]	
:REFerence		[no query]

## 19.13.1 [:ADJust] &lt;numeric\_value&gt;

SOURce:PHASe:ADJust

Controls the phase offset value relative to the reference. A negative value for ADJust shall cause the output signal to lag the reference.

The parameter has units of radians. The command shall accept a DEGree suffix.

At \*RST, this value is 0.

## 19.13.1.1 :STEP &lt;numeric\_value&gt;

SOURce:PHASe:ADJust:STEP

Controls the step size in radians. The command shall accept a DEGree suffix.

At \*RST, this value is device-dependent.

## 19.13.2 :SOURce INTernal|EXTernal

SOURce:PHASe:SOURce

Determines whether the reference source is an external signal or if it is the internal signal itself marked at some reference time.

At \*RST, this setting is device-dependent.

## 19.13.3 :REFerence

SOURce:PHASe:REFerence

This is an event which sets the current phase to be the reference for future phase adjustments.

This function is nonqueryable.

**19.14 PM Subsystem**

SOURce:PM

The Phase Modulation subsystem is used to set the modulation controls and the parameters associated with the modulating signal(s). The keywords to describe the carrier signal exist in other subsystems (such as FREQuency and PHASe).

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>COMMENTS</b>
:PM		
[:DEViation]	<numeric_value>	
:SENSitivity	<numeric_value>	
:MODE	LOCKed UNLocked	
:STATE	<Boolean>	
:SOURce	EXTernal INTernal{ ,EXTernal ,INTernal}	
:COUPLing	AC DC GROund	
:POLarity	NORMal INVersed	
:INTernal		
:FREQuency	<numeric_value>	
:EXTernal		
:IMPedance	<numeric_value>	
:COUPLing	AC DC GROund	
:POLarity	NORMal INVersed	

**19.14.1 [:DEViation] <numeric\_value>**

SOURce:PM:DEViation

Sets the modulation DEViation of a PM signal. The unit for DEViation is radians. Where PM:DEViation is implemented, degree units must also be accepted. Radians must always be returned on a query. This command is used where the DEViation can be controlled independently from the modulating signal voltage level. Alternatively, it is used to set a device capability that can indicate when the desired DEViation has been achieved.

At \*RST, this value is device-dependent.

**19.14.2 :SENSitivity <numeric\_value>**

SOURce:PM:SENSitivity

Controls the modulation deviation by setting a sensitivity Modulation to the modulation signal voltage level. The unit for SENSitivity is radians/Volt (radians/V). Where PM:SENSitivity is implemented, degree/Volt (degree/V) units must also be accepted. Radians/Volt must always be returned on a query.

At \*RST, this value is device-dependent.

**19.14.3 :MODE LOCKed|UNLocked**

SOURce:PM:MODE

Sets the synthesis mode employed in generating the PM signal. If MODE is set to LOCKed then a PLL is used in synthesizing the PM signal, to prevent frequency drift. If UNLocked the phase locked loop is inactive.

At \*RST, this value is device-dependent.

**19.14.4 :STATe <Boolean>**

SOURce:PM:STATe

Turns phase modulation ON or OFF. Turning PM ON will not automatically turn OFF any other types of modulation. Turning any or all modulations types ON or OFF must be done explicitly. Where a type of modulation is active, turning ON another type of modulation will add to the set of active modulations if that combination of capabilities are legal in the device. Otherwise, an execution error -221 is generated and the instrument state is left undefined.

At \*RST, this value must be set to OFF.

**19.14.5 :SOURce EXTernal|INTernal{,EXTernal|,INTernal}**

SOURce:PM:SOURce

Selects the source for the modulating signal, and it may specify a single source or a number of sources. The specified sources in the SOURce command are all selected and turned on. Any sources from a previous selection that are not part of the current selection list are deselected and turned off. If multiple sources are selected and the device does not support the combination requested in the SOURce command, an execution error -221 must be generated.

Where multiple internal modulation sources are available, individual INTernal sources must be distinguished by a unique number suffix. If no number suffix is provided in a command, a 1 must be assumed.

Where multiple external modulation sources are available, individual EXTERNAL sources must be distinguished by a unique number suffix. If no number suffix is provided in a command, a 1 must be assumed.

At \*RST, this value is device-dependent.

**19.14.6 :COUPling AC|DC|GROund**

SOURce:PM:COUPling

COUPling at this level is used to set the coupling between the modulator and the modulating signal. The modulating signal may be the sum of several signals, either internal or external sources. If COUPling is set to DC, then both AC and DC components of the signal pass. AC coupling passes only the AC component of the signal. The GROund parameter allows the modulation to be turned ON without the modulating signal connected.

At \*RST, this value is device-dependent.

**19.14.7 :POLarity NORMAL|INVerted**

SOURce:PM:POLarity

POLarity, at this level, is used to set the polarity between the modulator and the modulating signal. The modulating signal may be the sum of several signals, either internal or external sources. POLarity sets the relationship between the polarity of the applied modulation voltage and the resulting “direction” of modulation. The definition of “normal” polarity for PM modulated signal is that a positive voltage causes the instantaneous output phase to increase.

At \*RST, this value must be set to NORMAL.

**19.14.8 :INTernal**

SOURce:PM:INTernal

The INTernal subsystem is used to control the specified internal signal source. Where multiple internal modulation sources are available, the INTernal keyword may have a numeric suffix signifying the particular internal signal source. If no number is given, a 1 is assumed.

**19.14.8.1 :FREQuency <numeric\_value>**

SOURce:PM:INTernal:FREQuency

Sets the frequency of the specified internal signal source. Some instruments may also allow the internal signal source(s) to be controlled as a subsystem(s). The unit of FREQuency is Hertz (Hz).

At \*RST, this value is device-dependent.

**19.14.9 :EXTernal**

SOURce:PM:EXTernal

The EXTernal subsystem is used to control the specified external signal source. Where multiple external modulation sources are available, the EXTernal keyword may have a numeric suffix signifying the particular external signal source. If no number is given, a 1 is assumed.

**19.14.9.1 :IMPedance <numeric\_value>**

SOURce:PM:EXTernal:IMPedance

Sets the impedance of the specified external signal source. The units for IMPedance is Ohms.

At \*RST, this value is device-dependent.

**19.14.9.2 :COUPLing AC|DC|GROund**

SOURce:PM:EXTernal:COUPLing

COUPLing, at this level, sets the coupling for only the specified external signal source. If COUPLing is set to DC, then both AC and DC components of the signal pass. AC coupling passes only the AC component of the signal. The GROund parameter allows the specified external source to be turned ON without being connected.

At \*RST, this value is device-dependent.

**19.14.9.3 :POLarity NORMAL|INVerted**

SOURce:PM:EXTernal:POLarity

POLarity, at this level, sets the POLarity for only the specified external signal source. POLarity sets the relationship between the polarity of the applied modulation voltage and the resulting “direction” of modulation. The definition of “normal” polarity for PM modulated signal is a positive voltage that causes the instantaneous output phase to increase.

At \*RST, this value must be set to NORMAL.

19.15 **POWER Subsystem**

SOURce:POWer

This subsection controls the signal amplitude characteristics of the source.

There is only one subsystem for each of the main amplitude characteristics of the signal. Thus, AC voltage and DC voltage are both placed under the VOLtage subsystem. A device which implemented both simultaneously would implement them as two separate sources, one AC and one DC.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>COMMENTS</b>
:POWer		
:ATTenuation	<numeric_value>	
:AUTO	<Boolean> ONCE	
:ALC		
[:STATE]	<Boolean>	
:SEARch	<Boolean> ONCE	
:SOURce	INTERNAL DIODe PMETer MMHead	
:BANDwidth	<numeric_value>	
[:BWIDth]		
:AUTO	<Boolean> ONCE	
:CENTer	<numeric_value>	
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:TRIGgered		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:LIMit		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:STATE	<Boolean>	
:MANual	<numeric_value>	
:MODE	FIXed SWEep LIST	
:PROTection		
[:LEVel]	<numeric_value>	
:STATE		
:TRIPped?		[query only]
:CLEar		[no query]

KEYWORD	PARAMETER FORM	COMMENTS
:RANGE	<numeric_value>	
:AUTO	<Boolean> ONCE	
:REFERENCE	<numeric_value>	
:STATE	<Boolean>	
:SLEW	<numeric_value>	
:SPAN	<numeric_value>	
:HOLD	<Boolean>	
:LINK	CENTER STARt STOP	
:FULL		
:STARt	<numeric_value>	
:STOP	<numeric_value>	

#### 19.15.1 :ATTenuation <numeric\_value>

SOURce:POWER:ATTenuation

Sets the ATTenuation level. Note that when increasing the level by 10 dB the magnitude of the outgoing signal as well as the LEVel will be decreased by 10 dB.

Default units are as determined in the UNIT system. This is coupled to LEVel.

#### 19.15.1.1 :AUTO <Boolean>

SOURce:POWER:ATTenuation:AUTO

Couples the attenuator to LEVel.

Programming a specified attenuation sets AUTO OFF.

See SOURce subsystem introduction for additional explanation.

At \*RST, AUTO is set to ON.

#### 19.15.2 :ALC

SOURce:POWER:ALC

This subsystem command controls the automatic leveling control of the source.

#### 19.15.2.1 [:STATe] <Boolean>

SOURce:POWER:ALC:STATe

Controls whether the ALC loop controls the output level.

At \*RST, this value is device-dependent.

#### 19.15.2.2 :SEARch <Boolean>|ONCE

SOURce:POWER:ALC:SEARch

SEARch enables a form of leveling where the output level is calibrated by momentarily closing the leveling loop. Once the correct amount of level adjustment has been determined to produce the desired output level, the leveling loop is opened.

When ON, the search routine will be done anytime the output level requires change (for example, when LEVel is changed).

Selecting SEARch ONCE will have the effect of setting SEARch to ON and then OFF. A modulator setting is determined by the system which will not be changed until the output level is explicitly reprogrammed.

At \*RST, this value is set to OFF.

### 19.15.2.3 :SOURce INTERNAL|DIODE|PMETER|MMHead

SOURce:POWer:ALC:SOURce

Selects the source of the feedback signal for ALC. The parameters have the following meanings:

- INTERNAL — The ALC feedback signal is measured from a point inside the source.
- DIODE — The ALC feedback is being fed from an external diode detector.
- PMETER — The ALC signal is coming from a power meter.
- MMHead — A millimeter head, with built in leveling, is being used to supply the ALC signal.

At \*RST, this value is INTERNAL.

### 19.15.2.4 :BANDwidth|BWIDth <numeric\_value>

SOURce:POWer:ALC:BANDwidth

Controls the bandwidth of the ALC feedback signal. This parameter is in units of Hz.

At \*RST, this value is device-dependent.

### 19.15.2.4.1 :AUTO <Boolean>|ONCE

SOURce:POWer:ALC:BANDwidth:AUTO

Couples the bandwidth of the ALC feedback signal to instrument-dependent parameters. When AUTO is ON, the best bandwidth is chosen to reflect the current instrument state.

Explicitly selecting a value for ALC:BANDwidth sets AUTO OFF.

At \*RST, this value is set to ON.

### 19.15.3 :CENTer <numeric\_value>

SOURce:POWer:CENTER

Sets the center amplitude.

At \*RST, this value is set to MINimum.

### 19.15.4 [:LEVel]

SOURce:POWer:LEVel

This subsystem is used to control the signal amplitude when the source is operating in a continuous or FIXed MODE.

### 19.15.4.1 [:IMMEDIATE]

SOURce:POWer:LEVel:IMMEDIATE

Indicates that the subsequent specification of a new signal amplitude is to be processed by the device without waiting for further commands.

**19.15.4.1.1 [:AMPLitude] <numeric\_value>**

SOURce:POWer:LEVel:IMMEDIATE:AMPLitude

Sets the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal.

Note the optional nodes exist to enable POWer to be directly followed by a <numeric\_value>. This is useful for programming simple devices in a straightforward manner.

At \*RST, the signal being sourced should be set to a “safe” condition. This is generally achieved by setting the amplitude to its minimum value in conjunction with OUTPut:STATe subsystem.

**19.15.4.1.2 :OFFSet <numeric\_value>**

SOURce:POWer:LEVel:IMMEDIATE:OFFSet

Sets the non-time varying component of the signal that is added to the time varying signal specified in AMPLitude, in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. When SOURce:FUNCTION is DC, the effect of the OFFSet is device-dependent.

At \*RST, the signal being sourced should be set to a “safe” condition. This is generally achieved by setting the offset to its minimum value in conjunction with OUTPut:STATe subsystem.

**19.15.4.1.3 :HIGH <numeric\_value>**

SOURce:POWer:LEVel:IMMEDIATE:HIGH

Sets the more positive peak of a time varying signal. It is used in conjunction with LOW.

At \*RST, the signal being sourced should be set to a “safe” condition. This is achieved by setting this to a minimum value in conjunction with OUTPut:STATe subsystem.

**19.15.4.1.4 :LOW <numeric\_value>**

SOURce:POWer:LEVel:IMMEDIATE:LOW

Sets the more negative peak of a time varying signal, it is used in conjunction with HIGH.

At \*RST, the signal being sourced should be set to a “safe” condition. This is achieved by setting this to a minimum value in conjunction with OUTPut:STATe subsystem.

**19.15.4.2 :TRIGgered**

SOURce:POWer:LEVel:TRIGgered

Indicates that subsequent specification of a new signal amplitude is to be transferred to the IMMEDIATE value upon receipt of a trigger signal.

At \*RST, after the receipt of an ABORt command or upon being triggered, the value in the TRIGgered subsystem tracks the values in the IMMEDIATE subsystem until a command in the TRIGgered subsystem is received. The purpose of tracking is so that spurious triggers do not change the value of LEVel unexpectedly.

**19.15.4.2.1 [:AMPLitude] <numeric\_value>**

SOURce:POWer:LEVel:TRIGgered:AMPLitude

Sets the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal.

**19.15.4.2.2 :OFFSet <numeric\_value>**

SOURce:POWer:LEVel:TRIGgered:OFFSet

Sets the non-time varying component of the signal that is added to the time varying signal specified in AMPLitude. The units are set to the default value, or to a different value under the UNIT subsystem. When SOURce:FUNCTION is DC, the effect of OFFSet is device-dependent.

**19.15.4.2.3 :HIGH <numeric\_value>**

SOURce:POWer:LEVel:TRIGgered:HIGH

This command is used to set the more positive peak of a time varying signal. It is used in conjunction with LOW.

**19.15.4.2.4 :LOW <numeric\_value>**

SOURce:POWer:LEVel:TRIGgered:LOW

This command is used to set the more negative peak of a time varying signal. It is used in conjunction with HIGH.

**19.15.5 :LIMit**

SOURce:POWer:LIMit

Sets the maximum bounds on the output value. Setting a larger value will cause the output level to be clamped to the LIMit value.

**19.15.5.1 [:AMPLitude] <numeric\_value>**

SOURce:POWer:LIMit:AMPLitude

Sets the limit on the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal. OFFSet may only be used with AMPLitude, where AMPLitude is used to specify a time varying load.

Explicit reference to the AMPLitude node is optional. Any device that implements this subsystem must accept and process commands, with or without the AMPLitude node, and respond to them in the same way.

At \*RST, this value is device-dependent.

**19.15.5.2 :OFFSet <numeric\_value>**

SOURce:POWer:LIMit:OFFSet

Sets a non-time varying component limit of signal that is added to the time varying signal specified in AMPLitude. The units are set to the default value, alternately to a different value under the UNIT subsystem.

At \*RST, this value is device-dependent.

#### 19.15.5.3 :HIGH <numeric\_value>

SOURce:POWER:LIMit:HIGH

Sets the more positive peak limit of a time varying signal, it is used in conjunction with LOW.

At \*RST, this value is device-dependent.

#### 19.15.5.4 :LOW <numeric\_value>

SOURce:POWER:LIMit:LOW

Sets the more negative peak limit of a time varying signal, it is used in conjunction with HIGH.

At \*RST, this value is device-dependent.

#### 19.15.5.5 :STATE <Boolean>

SOURce:POWER:LIMit:STATe

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

#### 19.15.6 :MANual <numeric\_value>

SOURce:POWER:MANual

Allows manual adjustment of the amplitude between the sweep limits. The actual amplitude level is determined by the parameter only if SWEep:MODE is set to MANual and the amplitude MODE is set to SWEep. If the sweep limits are changed such that the manual amplitude value would be outside the limits, the manual amplitude value will be set to the nearest limit.

At \*RST, this value is device-dependent.

#### 19.15.7 :MODE FIXed|SWEep|LIST

SOURce:POWER:MODE

Determines which set of commands control the amplitude subsystem. If FIXed is selected, the amplitude is determined by the LEVel command under the appropriate subsystem. If SWEep is selected, the source is in the swept mode and amplitude is determined by START, STOP, CENTER, SPAN, and MANual commands under the appropriate subsystem. If LIST is selected, the amplitude values are determined by the appropriate amplitude list (such as LIST:).

At \*RST, this value is set to FIXed.

#### 19.15.8 :PROtection

SOURce:POWER:PROtection

Controls the protection circuits.

#### 19.15.8.1 [:LEVel] <numeric\_value>

SOURce:POWER:PROtection:LEVel

This command sets the output level at which the output protection circuit will trip.

**19.15.8.2 :STATe <Boolean>**

SOURce:POWer:PROTection:STATe

Controls whether the output protection circuit is enabled.

At \*RST, this value is set to ON.

**19.15.8.3 :TRIPped?**

SOURce:POWer:PROTection:TRIPped?

This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

TRIPped only has a query form and thus has no associated \*RST condition.

**19.15.8.4 :CLEar**

SOURce:POWer:PROTection:CLEar

Causes the protection circuit to be cleared.

This command is an event and has no associated \*RST condition.

**19.15.9 :RANGe <numeric\_value>**

SOURce:POWer:RANGe

Sets a range for the output amplitude. This command is very hardware-specific.

**19.15.9.1 :AUTO <Boolean>|ONCE**

SOURce:POWer:RANGe:AUTO

Couples the RANGe to an instrument-determined value. When AUTO is ON, the best range is chosen to reflect the current instrument state.

Explicitly selecting a value for RANGe sets AUTO OFF.

At \*RST, this value is set to ON.

**19.15.10 :REFerence <numeric\_value>**

SOURce:POWer:REFerence

Sets a reference value which, if STATe is ON, allows all amplitude parameters to be queried/set as relative to the reference value.

At \*RST, this value is device-dependent.

**19.15.10.1 :STATe <Boolean>**

SOURce:POWer:REFerence:STATe

Determines whether amplitude is measured/output in absolute or relative mode. If STATe is ON, then amplitude is referenced to the value set in REFerence.

At \*RST, this value is set to OFF.

**19.15.11 :SLEW <numeric\_value>**

SOURce:POWer:SLEW

Sets the slew rate of the output change when a new output level is programmed. The units are in (the currently active) amplitude unit/sec.

At \*RST, this value is device-dependent.

**19.15.12 :SPAN <numeric\_value>**

SOURce:POWer:SPAN

Sets the amplitude span. If the current amplitude unit is logarithmic (dBm, dBuV, etc), then the unit of SPAN is dB. Otherwise SPAN is programmed in the current amplitude unit.

At \*RST, this value is set to 0.

**19.15.12.1 :HOLD <Boolean>**

SOURce:POWer:SPAN:HOLD

Provides a mechanism to prevent the SPAN from being changed implicitly by the defined coupling between STARt, STOP, CENTER and SPAN. When HOLD is set to ON, SPAN shall only be changed by explicitly sending the SPAN command.

At \*RST, this value is set to OFF.

**19.15.12.2 :LINK CENTER|STARt|STOP**

SOURce:POWer:SPAN:LINK

Allows the default couplings for SPAN to be overridden. LINK selects the parameter, either CENTER, STARt or STOP, that shall not be changed when SPAN's value is changed. For example, if LINK is set to STARt then changing SPAN shall cause CENTER and STOP, not STARt to change.

At \*RST, this value is set to CENTER, to be compatible with the default definition for the couplings.

**19.15.12.3 :FULL**

SOURce:POWer:SPAN:FULL

When this command is received, STARt amplitude is set to its minimum value and STOP amplitude is set to its maximum value. CENTER amplitude and SPAN are set to their coupled values.

This command is an event rather than a state. Therefore it has no associated query, or meaning at \*RST.

**19.15.13 :STARt <numeric\_value>**

SOURce:POWer:STARt

Sets STARt amplitude.

At \*RST, this value is set to MINimum.

**19.15.14 :STOP <numeric\_value>**

SOURce:POWer:STOP

Sets STOP amplitude.

At \*RST, this value is set to MINimum.

## 19.16 PULse Modulation Subsystem

SOURce:PULM

The PULse Modulation subsystem is used to set the modulation controls and the parameters associated with the modulating signal(s). The keywords to describe the carrier signal exist in other subsystems (such as, FREQuency and PHASe).

KEYWORD	PARAMETER FORM	COMMENTS
:PULM		
:EXTernal		
:HYSTeresis	<numeric_value>	
:IMPedance	<numeric_value>	
:LEVel	<numeric_value>	
:POLarity	NORMAl INVerted	
:INTERNAL		
:FREQuency	<numeric_value>	
:MODE	FIXEd LIST	1993
:POLarity	NORMAl INVerted	
:SOURce	EXTernal INTERNAL{,EXTernal INTERNAL}	
:STATe	<Boolean>	

## 19.16.1 :EXTernal

SOURce:PULM:EXTernal

The EXTernal subsystem is used to control the specified external signal source. Where multiple external modulation sources are available, the EXTernal keyword may have a numeric suffix signifying the particular external signal source. If no number is given, a 1 is assumed.

## 19.16.1.1 :HYSTeresis &lt;numeric\_value&gt;

SOURce:PULM:EXTernal:HYSTeresis

Sets, for the specified signal source, how far the signal voltage must rise above (alternatively, fall below) the threshold LEVel before the carrier is turned ON from the OFF state (or turned OFF from the ON state). The unit for HYSTeresis is Volts.

At \*RST, this defaults to the TTL preset value.

## 19.16.1.2 :IMPedance &lt;numeric\_value&gt;

SOURce:PULM:EXTernal:IMPedance

Sets the impedance of the specified external signal source. The unit for IMPedance is Ohms.

At \*RST, this value is device-dependent.

## 19.16.1.3 :LEVel &lt;numeric\_value&gt;

SOURce:PULM:EXTernal:LEVel

For the specified signal source, this sets the threshold voltage level used to turn ON the carrier. The unit for LEVel is Volts.

At \*RST, this defaults to the TTL preset value.

**19.16.1.4 :POLarity NORMAL|INVerted**

SOURce:PULM:EXTernal:POLarity

POLarity, at this level, sets the POLarity for only the specified external signal source. POLarity sets the relationship between the polarity of the applied modulation voltage and the resulting “direction” of modulation. The definition of “normal” polarity for PULSe modulated signal is, a voltage more positive than some threshold (after applying hysteresis effects) and causes the carrier to turn on.

At \*RST, this value must be set to NORMAl.

**19.16.2 :INTernal**

SOURce:PULM:INTernal

The INTernal subsystem is used to control the specified internal signal source. Where multiple internal modulation sources are available, the INTernal keyword may have a numeric suffix signifying the particular internal signal source. If no number is given, a 1 is assumed.

**19.16.2.1 :FREQuency <numeric\_value>**

SOURce:PULM:INTernal:FREQuency

Sets the frequency of the specified internal signal source. Some instruments may also allow the internal signal source(s) to be controlled as a subsystem(s). The unit of FREQuency is Hertz (Hz).

At \*RST, this value is device-dependent.

**19.16.3 MODE**

SOURce:PULM:MODE

Determines which set of commands currently control the PULM subsystem. The settings have the following meanings:

FIXed: Settings of individual commands are as last set by the user.

LIST: Settings of individual commands are controlled by the LIST subsystem.

At \*RST the value is FIXed.

**19.16.4 :POLarity NORMAL|INVerted**

SOURce:PULM:POLarity

POLarity, at this level, is used to set the polarity between the modulator and the modulating signal. The modulating signal may be the sum of several signals, either internal or external sources. POLarity sets the relationship between the polarity of the applied modulation voltage and the resulting “direction” of modulation. The definition of “normal” polarity for PULSe modulated signal is a voltage more positive than some threshold (after applying hysteresis effects) and causes the carrier to turn on.

At \*RST, this value must be set to NORMAl.

### 19.16.5 :SOURce EXTernal|INTernal{,EXTernal|INTernal}

SOURce:PULM:SOURce

Selects the source for the modulating signal, and it may specify a single source or a number of sources. The specified sources in the SOURce command are all selected and turned on. Any sources from a previous selection are ignored. If multiple sources are selected and the device does not support the combination requested in the SOURce command, an execution error -221 must be generated.

Where multiple internal modulation sources are available, individual INTernal sources may be distinguished by a unique numeric suffix. If no numeric suffix is provided in a command, a 1 is assumed.

Where multiple external modulation sources are available, individual EXTernal sources must be distinguished by a unique number suffix. If no number suffix is provided in a command, a 1 must be assumed.

At \*RST, this value is device-dependent.

### 19.16.6 :STATe <Boolean>

SOURce:PULM:STATe

Turns Pulse Modulation ON or off. Turning ON PULse Modulation will not automatically turn OFF any other types of modulation. Turning any or all modulations types ON or OFF must be done explicitly. Where a type of modulation is active, turning ON another type of modulation will add to the set of active modulations if that combination of capabilities are legal in the device. Otherwise, an execution error -221 is generated and the instrument state is left undefined.

At \*RST, this value must be set to OFF.

**19.17 PULSe Subsystem**

SOURce:PULSe

This subsystem collects together the commands used to control pulse generation.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>COMMENTS</b>
:PULSe		
:PERiod	<numeric_value>	
:WIDTh	<numeric_value>	
:DCYCle	<numeric_value>	
:HOLD	WIDTh DCYCle	
:DELay	<numeric_value>	
:DOUble		
[:STATE]	<Boolean>	
:DELay	<numeric_value>	
:TRANsition		
:STATE	<Boolean>	
[:LEADing]	<numeric_value>	
:TRAiling	<numeric_value>	
:AUTO	<Boolean> ONCE	
:COUNt	<numeric_value>	
:POLarity	NORMal COMplement INVerted	

**19.17.1 :PERiod <numeric\_value>**

SOURce:PULSe:PERiod

Sets the period of a pulsed waveform. The fundamental units for PERiod is seconds.

At \*RST, this value is device-dependent.

**19.17.2 :WIDTh <numeric\_value>**

SOURce:PULSe:WIDTh

Sets the width or duration of the pulse. The fundamental units for WIDTh is seconds.

At \*RST, this value is device-dependent.

**19.17.3 :DCYCle <numeric\_value>**

SOURce:PULSe:DCYCle

Sets the duty cycle of a repetitive pulsed waveform. The fundamental units for DCYCle is percent (%).

At \*RST, this value is device-dependent.

**19.17.4 :HOLD WIDTh|DCYCle**

SOURce:PULSe:HOLD

Sets, for a pulsed waveform, the parameter to be held constant when the period changes.

At \*RST, this value is device-dependent.

**19.17.5 :DELay <numeric\_value>**

SOURce:PULSe:DELay

Sets the time from the start of the period to the first edge of the pulse. The fundamental units for DELay is seconds.

At \*RST, the value of this parameter is set to either 0 seconds or the nearest specified value.

**19.17.6 :DOUBole**

SOURce:PULSe:DOUBole

This subsystem contains the commands necessary to control the parameters associated with double pulse. When a double pulse is enabled, a second pulse which is identical to the primary pulse is generated at some point during the period of the pulse. This second pulse has the same characteristics as the first pulse, including width. A side effect is that since a second pulse has been added, the actual duty cycle of the signal is twice that specified in PULSe:DCYCle. Attempting to enable double pulse mode when the pulse width is greater than:

$$(PULSE:PERiod-PULSe:DOUBole:DELay)/2$$

will cause an error -221, “Settings Conflict,” to be generated.

**19.17.6.1 [:STATE] <Boolean>**

SOURce:PULSe:DOUBole:STATE

This command sets the double pulse mode to ON or OFF.

At \*RST, this parameter is set to OFF.

**19.17.6.2 :DELay <numeric\_value>**

SOURce:PULSe:DOUBole:DELay

Sets the time from the start of the period to the first edge of the second pulse. The fundamental units for DELay is seconds.

At \*RST, this value is device-dependent.

**19.17.7 :TRANsition**

SOURce:PULSe:TRANsition

Collects together the commands to control the parameters associated with characteristics of the leading and trailing edges of the pulse.

**19.17.7.1 :STATe <Boolean>**

SOURce:PULSe:TRANsition:STATe

Sets the transition mode to ON or OFF. If STATe is OFF, then the LEADING and TRAILing edges must be set to the minimum transition times available in the device. If STATe is ON, then the pulse transition times are specified by values for LEADING and TRAILing.

At \*RST, this parameter is set to OFF.

**19.17.7.2 [:LEADING] <numeric\_value>**

SOURce:PULSe:TRANsition:LEADING

This command sets the transition time for the LEADING edge. The fundamental units for LEADING is seconds.

At \*RST, this value is device-dependent.

### 19.17.7.3 :TRAiling <numeric\_value>

SOURce:PULSe:TRANSition:TRAiling

Sets the transition time for the TRAiling edge. The fundamental units for TRAiling is seconds.

At \*RST, this value device-dependent.

#### 19.17.7.3.1 :AUTO <Boolean>|ONCE

SOURce:PULSe:TRANSition:TRAiling:AUTO

Couples the value of TRAiling to LEADING, when AUTO is set to ON.

At \*RST, this parameter is set to ON.

### 19.17.8 :COUNt <numeric\_value>

SOURce:PULSe:COUNt

Sets the number of times pulse (or double pulse) must be repeated for a single trigger event.

At \*RST, the value of this parameter is set to 1.

### 19.17.9 :POLarity NORMAL|COMplement|INVersed

SOURce:PULSe:POLarity

Sets the polarity of the pulse. For NORMAL operation the second nominal state is more positive than the first. COMplement and INVersed are aliases. For either position, the second nominal transition is more negative than the first.

At \*RST, this parameter is set to NORMAL.

## 19.18 RESistance Subsystem

SOURce:RESistance

This subsystem controls the characteristics of a resistance source or load.

KEYWORD	PARAMETER FORM	COMMENTS
:RESistance		
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
[:TRIGgered]		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
[:LIMit]		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
[:PROTection]		
[:LEVel]	<numeric_value>	
:STATE	<Boolean>	
:TRIPped?		[query only]
:CLEar		[no query]
[:SLEW]	<numeric_value>	
[:CENTer]	<numeric_value>	
[:SPAN]	<numeric_value>	
:HOLD	<Boolean>	
:LINK	CENTer STARt STOP	
:FULL		[no query]
[:STARt]	<numeric_value>	
[:STOP]	<numeric_value>	
[:MANual]	<numeric_value>	
[:MODE]	FIXed SWEep LIST	
[:REference]	<numeric_value>	
:STATE	<Boolean>	
[:RANGE]	<numeric_value>	
:AUTO	<boolean> ONCE	

**19.18.1 [:LEVel]**

SOURce:RESistance:LEVel

This subsystem is used to control the resistance when the source is operating in a continuous or FIXed MODE. That is, as a constant load.

**19.18.1.1 [:IMMEDIATE]**

SOURce:RESistance:LEVel:IMMEDIATE

Indicates that the subsequent specification of a new resistance is to be processed by the device without waiting for further commands.

**19.18.1.1.1 [:AMPLitude] <numeric\_value>**

SOURce:RESistance:LEVel:IMMEDIATE:AMPLITUDE

Sets the actual magnitude of the unswept resistance. AMPLITUDE may be used to specify the resistance level for either a time varying or non-time varying load.

At \*RST, this value is device-dependent.

**19.18.1.1.2 :OFFSet <numeric\_value>**

SOURce:RESistance:LEVel:IMMEDIATE:OFFSET

Sets the non-time varying component of the resistance that is added to the time varying resistance signal specified in AMPLITUDE.

At \*RST, this value is device-dependent.

**19.18.1.1.3 :HIGH <numeric\_value>**

SOURce:RESistance:LEVel:IMMEDIATE:HIGh

Sets the more positive peak of a time varying resistance signal, it is used in conjunction with LOW.

At \*RST, this value is device-dependent.

**19.18.1.1.4 :LOW <numeric\_value>**

SOURce:RESistance:LEVel:IMMEDIATE:LOW

Sets the more negative peak of a time varying resistance signal, it is used in conjunction with HIGH.

At \*RST, this value is device-dependent.

**19.18.1.2 :TRIGgered**

SOURce:RESistance:LEVel:TRIGgered

Indicates that subsequent specification of a new resistance settings are to be transferred to the IMMEDIATE value upon receipt of a trigger signal.

At \*RST, after the receipt of an ABORt command, or upon being triggered, the values in the TRIGgered subsystems track the values in the IMMEDIATE subsystem until a command in the TRIGgered subsystem is received. The purpose of tracking is so that spurious triggers do not change the value of LEVel unexpectedly.

**19.18.1.2.1 [:AMPLitude] <numeric\_value>**

SOURce:RESistance:LEVel:TRIGgered:AMPLitude

Sets the actual magnitude of the unswept output signal. AMPLitude may be used to specify the resistance level for either a time varying or non-time varying load.

**19.18.1.2.2 :OFFSet <numeric\_value>**

SOURce:RESistance:LEVel:TRIGgered:OFFSet

Sets the non-time varying component of the resistance that is added to the time varying resistance signal specified in AMPLitude.

**19.18.1.2.3 :HIGH <numeric\_value>**

SOURce:RESistance:LEVel:TRIGgered:HIGH

Sets the more positive peak of a time varying resistance signal, and it is used in conjunction with LOW.

**19.18.1.2.4 :LOW <numeric\_value>**

SOURce:RESistance:LEVel:TRIGgered:LOW

Sets the more negative peak of a time varying resistance signal, and it is used in conjunction with HIGH.

**19.18.2 :LIMit**

SOURce:RESistance:LIMit

Sets the bounds on the resistance value. Setting a larger value will cause the resistance to be clamped to the LIMit value.

**19.18.2.1 [:AMPLitude] <numeric\_value>**

SOURce:RESistance:LIMit:AMPLitude

Sets the limit on the actual magnitude of the unswept resistance. AMPLitude may be used to specify the resistance level for either a time varying or non-time varying load. OFFSet may only be used with AMPLitude, where AMPLitude is used to specify a time varying load.

Explicit reference to the AMPLitude node is optional. Any device that implements this subsystem shall accept and process commands, with or without the AMPLitude node, and respond to them in the same way.

At \*RST, this value is device-dependent.

**19.18.2.2 :OFFSet <numeric\_value>**

SOURce:RESistance:LIMit:OFFSet

Sets the limit for the non-time varying component of resistance that is added to the time varying resistance signal specified in AMPLitude.

At \*RST, this value is device-dependent.

**19.18.2.3 :HIGH <numeric\_value>**

SOURce:RESistance:LIMit:HIGH

Sets the more positive peak limit of a time varying resistance signal, it is used in conjunction with LOW.

At \*RST, this value is device-dependent.

**19.18.2.4 :LOW <numeric\_value>**

SOURce:RESistance:LIMit:LOW

Sets the more negative peak limit of a time varying resistance signal, and it is used in conjunction with HIGH.

At \*RST, this value is device-dependent.

**19.18.3 :PROTection**

SOURce:RESistance:PROTection

Controls the resistance protection circuit

**19.18.3.1 [:LEVel] <numeric\_value>**

SOURce:RESistance:PROTection:LEVel

Sets the output resistance level at which the output protection circuit will trip.

**19.18.3.2 :STATe <Boolean>**

SOURce:RESistance:PROTection:STATe

Controls whether the output protection circuit is enabled.

At \*RST, this value is set to ON.

**19.18.3.3 :TRIPped?**

SOURce:RESistance:PROTection:TRIPped?

Returns a 1 if the protection circuit is tripped and a 0 if it is untripped. TRIPped only has a query form and thus has no associated \*RST condition.

**19.18.3.4 :CLEar**

SOURce:RESistance:PROTection:CLEar

Causes the protection circuit to be cleared. This command is an event and has no associated \*RST condition.

**19.18.4 :SLEW <numeric\_value>**

SOURce:RESistance:SLEW

Sets the slew rate of the resistance change when a new resistance level is programmed. The units are in Ohm/sec.

At \*RST, this value is device-dependent.

**19.18.5 :CENTer <numeric\_value>**

SOURce:RESistance:CENTER

Sets the CENTER RESistance. At \*RST, this value is set to MINimum.

**19.18.6 :SPAN <numeric\_value>**

SOURce:RESistance:SPAN

Sets the resistance SPAN.

At \*RST, this value is set to 0.

**19.18.6.1 :HOLD <Boolean>**

SOURce:RESistance:SPAN:HOLD

Provides a mechanism to prevent the SPAN from being changed implicitly by the defined coupling between STARt, STOP, CENTER and SPAN. When HOLD is set to ON, SPAN shall only be changed by explicitly sending the SPAN command.

At \*RST, this value is set to OFF.

**19.18.6.2 :LINK CENTER|STARt|STOP**

SOURce:RESistance:SPAN:LINK

Allows the default couplings for SPAN to be overridden. LINK selects the parameter, either CENTER, STARt or STOP, that shall not be changed when SPAN's value is changed. For example, if LINK is set to STARt then changing SPAN shall cause CENTER and STOP, not STARt to change.

At \*RST, this value is set to CENTER, to be compatible with the default definition for the couplings.

**19.18.6.3 :FULL**

SOURce:RESistance:SPAN:FULL

When this command is received, STARt RESistance is set to MINimum, and STOP RESistance is set to MAXimum. CENTER RESistance and SPAN are set to their coupled values.

This command is an event, rather than a state. Therefore it has no associated query, or meaning at \*RST.

**19.18.7 :STARt <numeric\_value>**

SOURce:RESistance:STARt

Sets STARt resistance.

At \*RST, this value is set to MINimum.

**19.18.8 :STOP <numeric\_value>**

SOURce:RESistance:STOP

Sets STOP resistance.

At \*RST, this value is set to MINimum.

**19.18.9 :MANual <numeric\_value>**

SOURce:RESistance:MANual

Allows manual adjustment of the RESistance between the sweep limits. The actual RESistance level is determined by the parameter only if SWEep:MODE is set to MANual and RESistance:MODE is set to SWEep. If the sweep limits are changed such that the manual RESistance value would be outside the limits, the manual RESistance value will be set to the nearest limit.

At \*RST, this value is device-dependent.

### 19.18.10 :MODE FIXed|SWEep|LIST

SOURce:RESistance:MODE

Determines which set of commands control the resistance subsystem. If FIXed is selected, the RESistance is determined by the LEVel command. If SWEep is selected, the source is in the swept mode and RESistance is determined by START, STOP, CENTER, SPAN, and MANUAL commands. If LIST is selected, the RESistance values are determined by LIST:RESistance.

At \*RST, this value is set to FIXed.

### 19.18.11 :REFerence <numeric\_value>

SOURce:RESistance:REFerence

Sets a reference value which, if STATe is ON, allows all RESistance parameters to be queried/set as a relative to the reference value.

At \*RST, this value is device-dependent.

#### 19.18.11.1 :STATe <Boolean>

SOURce:RESistance:REFerence:STATe

Determines whether RESistance is in absolute or relative mode. If STATe is ON, then RESistance is referenced to the value set in REFerence.

At \*RST, this value is set to OFF.

### 19.18.12 :RANGe <numeric\_value>

SOURce:RESistance:RANGE

Sets a range for the output RESistance.

#### 19.18.12.1 :AUTO <Boolean>|ONCE

SOURce:RESistance:RANGE:AUTO

Couples the RANGE to an instrument-determined value. When AUTO POWER and VOLTage is ON, the best range is chosen to reflect the current instrument state.

Explicitly selecting a value for RANGE sets AUTO OFF.

At \*RST, this value is set to ON.

19.19 **ROSCillator Subsystem**

SOURce:ROSCillator

This subsystem controls the reference oscillator.

KEYWORD	PARAMETER FORM	NOTES
:ROSCillator		
[:INTERNAL]		1992
:FREQuency	<numeric_value>	
[:EXTERNAL]		1992
:FREQuency	<numeric_value>	1992
:SOURce	INTERNAL EXTERNAL NONE	
:AUTO	<Boolean> ONCE	

## 19.19.1 [:INTERNAL]

SOURce:ROSCillator:INTERNAL

This subsystem configures the internal reference oscillator(s).

## 19.19.1.1 :FREQuency &lt;numeric\_value&gt;

SOURce:ROSCillator:INTERNAL:FREQuency

Specifies the frequency of the internal reference oscillator. The default units are Hz.

At \*RST, this value is device-dependent.

## 19.19.2 :EXTERNAL

SOURce:ROSCillator:EXTERNAL

This subsystem configures the external reference oscillator(s).

## 19.19.2.1 :FREQuency &lt;numeric\_value&gt;

SOURce:ROSCillator:EXTERNAL:FREQuency

Specifies the frequency of the external reference oscillator. The default units are Hz.

At \*RST, this value is device-dependent.

## 19.19.3 :SOURce INTERNAL|EXTERNAL|NONE

SOURce:ROSCillator:SOURce

Controls the selection of the reference oscillator source. This typically refers to a precision, stabilized time base. The parameter values have the following meanings:

- INTERNAL — The reference frequency is derived from an internal precision oscillator.
  - EXTERNAL — The reference frequency is derived from an external signal supplied to the device.
  - NONE — The reference frequency is not derived from any precision oscillator.
- At \*RST, this value is device-dependent.

## 19.19.3.1 :AUTO &lt;Boolean&gt;|ONCE

SOURce:ROSCillator:SOURce:AUTO

When AUTO is ON, the system automatically selects which oscillator will be used by the instrument.

Explicitly selecting a reference oscillator turns AUTO OFF.

## **1999 SCPI Command Reference**

At \*RST, AUTO is set to ON.

**19.20 SPEed Subsystem**

SOURce:SPEed

This node controls the speed of the device.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>COMMENTS</b>
:SPEed		1999
:INITiate		[event; no query]1999
[:LEVel]	<numeric_value>	1999
:SSDLoss		1999
:INITiate		[event; no query]1999
:LATime	<numeric_value>	1999
:STIMe	<numeric_value>	1999

**19.20.1 :INITiate**

:SOURce:SPEed:INITiate

This command initiates speed. This is an overlapped command. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for chassis dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Go To Speed Procedure bit indicating Go To Speed Procedure is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Accelerate dynamometer to the :SOURce:SPEed:LEVel speed point at the rate in :SOURce:ACCELERation. The dynamometer determines when the speed point has stabilized.
- Maintain speed and wait for a command.
- When commanded out of this procedure, clear the DYNO operation condition register Go To Speed Procedure bit indicating the Go To Speed Procedure is not executing.
- Issue the process complete message.

**19.20.2 [:LEVel] <numeric\_value>**

:SOURce:SPEed:LEVel

Set the desired speed set-point in m/s.

At \*RST, this value is device dependent.

**19.20.3 :SSDLoss**

SOURce:SPEed:SSDLoss

Steady State Driveline Loss

This node describes the procedure to measure the driveline loss forces as a function of speed. The Steady State Driveline Procedure measures the vehicle losses due to vehicle drive train

components. The vehicle losses are measured by operating the dynamometer at various selected speeds and measuring the force required to maintain the constant speed over a specified time interval. The results are stored in the DrvLLoss pre-defined table. This table has two columns, SPEed, the speeds at which the losses are to be measured (input) and LOSS, the measured losses at each speed (result). Coefficients for vehicle driveline losses are determined at the end of the procedure and reside in DrvLCoeff.

#### 19.20.3.1 :INITiate

:SOURce:SPEEd:SSDLoss:INITiate

This is an overlapped command. The command uses the speed points defined in the :MEMORY:TABLE:SELECT DrvLLoss. This command describes an event and therefore has no associated \*RST condition.

When this procedure is initiated for dynamometers, the dynamometer must:

- Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this command, issue appropriate warning messages, and execute the Idle Procedure.
- Set the DYNO operation condition register Driveline Loss Procedure bit indicating the driveline loss is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
- Accelerate dynamometer to the first speed point defined in the DrvLLoss memory table at the rate in :SOURce:ACCELERation.
- Stabilize at this speed for :SOURce:SPEEd:SSDLoss:STIMe. If the speed does not stabilize, set Failed to Stabilize bit of the DYNO questionable condition register and continue with the procedure.
- Average the force at the roll surface required to maintain the speed set point over the loss average time :SOURce:SPEEd:SSDLoss:LATime. The average force value is placed in the DrvLLoss memory table for the specified speed point. Repeat steps 3 to 5 for each point in the DrvLLoss table. The first zero speed in the DrvLLoss table ends the procedure.
- Perform a curve fit on the DrvLLoss data and place the coefficients in the memory table DrvLCoeff.
- Clear the DYNO operation condition register bit 7 indicating the driveline loss is not executing.
- Issue the process complete message.
- Execute the Idle Procedure.

#### 19.20.3.2 :LATime <numeric\_value>

:SOURce:SPEEd:SSDLoss:LATime

Loss Averaging Time - sec

## 1999 SCPI Command Reference

This command sets the time after stabilization that the losses are to be averaged before moving to the next speed point.

At \*RST, this value is set to 5.

### 19.20.3.3 :STIMe <numeric\_value>

:SOURce:SPEed:SSDLoss:STIMe

Stabilization Time - sec

Time for stabilization prior to LATime after a change in speed.

At \*RST, this value is set to 5.

**19.21 SWEep Subsystem**

SOURce:SWEep

This subsystem controls the sweep for those instruments which generate signals as a function of time. Other parameters may also be swept by specifying SWEep as their MODE (for example, FREQuency:MODE SWEep). Units for this subsystem are seconds unless otherwise noted.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>COMMENTS</b>
:SWEep		
:TIME	<numeric_value>	
:AUTO	<Boolean> ONCE	
:LLIMit	<numeric_value>	
:DWEll	<numeric_value>	
:AUTO	<Boolean> ONCE	
:DIRection	UP DOWN	
:MODE	AUTO MANual	
:SPACing	LINear LOGarithmic	
:GENeration	STEPped ANALog	
:STEP	<numeric_value>	
:POINTs	<numeric_value>	
:COUNt	<numeric_value>	

**19.21.1 :TIME <numeric\_value>**

SOURce:SWEep:TIME

Sets the duration of the sweep. Note that this does not turn sweeping on. Setting this value automatically turns TIME:AUTO OFF.

**19.21.1.1 :AUTO <Boolean>|ONCE**

SOURce:SWEep:TIME:AUTO

When enabled, the sweep time is calculated internally and is dependent on the span of the sweep.

At \*RST, AUTO ON is selected.

**19.21.1.2 :LLIMit <numeric\_value>**

SOURce:SWEep:TIME:LLIMit

Defines a lower limit for sweep time. This lower limit restricts the sweep time value set either explicitly or automatically.

At \*RST, LLIMit is set to a device-dependent value.

**19.21.2 :DWEll <numeric\_value>**

SOURce:SWEep:DWEll

Controls the amount of time spent at each point during a sweep. Note that DWEll cannot exceed TIME/POINTs. Trying to set it to a greater value will cause an error to result. The points will not change, but time may be changed as a device-dependent decision.

At \*RST, this value is device-dependent.

**19.21.2.1 :AUTO <Boolean>|ONCE**

SOURce:SWEep:DWELl:AUTO

When AUTO ON is selected, the dwell time is coupled to the sweep time and number of points. The coupling equation is:

$$DWELL = (TIME/POINTS) - (\text{device-dependent stepping time})$$

Setting a value for dwell turns AUTO OFF.

At \*RST, AUTO is set to ON.

**19.21.3 :DIRECTION UP|DOWN**

SOURce:SWEep:DIRECTION

Controls the direction of the sweep. If UP is selected, the sweep is carried out in ascending order, from START to STOP. If DOWN is selected, the sweep is carried out in descending order, from STOP to START.

At \*RST, the value of this function is UP.

**19.21.4 :MODE AUTO|MANual**

SOURce:SWEep:MODE

Selects the sweep conditions for the sweep. The various conditions are:

- AUTO: The sweep is controlled by the internal sweep generator.
- MANual: The sweep is controlled by front panel controls or by the manual command, such as, FREQuency:MANual. This setting is valid only if the :MANual capability has been implemented.

At \*RST, this function is set to AUTO.

**19.21.5 :SPACING LINear|LOGarithmic**

SOURce:SWEep:SPACING

Determines the swept entity versus time characteristics of the sweep. The various settings have the following meanings:

- LINear: For stepped sweeps, the swept entity is incremented (decremented) by the step size until the sweep limit is reached. For analog sweeps, a linear ramp is generated.
- LOGarithmic: For non-linear sweeps, step size is determined by a logarithmic curve fitted between the start and stop frequency. Stepping is determined by the SWEep:POINTs. For analog sweeps, a logarithmic ramp is generated.

At \*RST, this value is set to LINear.

**19.21.6 :GENERATION STEPped|ANALog**

SOURce:SWEep:GENERATION

Selects between an analog or stepped sweep.

- STEPped: A stepped sweep is selected, and the points are controlled by the SWEep:POINTs and SWEep:STEP settings.
- ANALog: The sweep is controlled by an analog signal.

At \*RST, this value is device-dependent.

#### 19.21.7 :STEP <numeric\_value>

SOURce:SWEep:STEP

Controls the swept entity step size for a stepped linear sweep. This parameter is not used if GENERation is ANALog or if SPACing is LOGarithmic.

This value is hard coupled to SPAN and POINts by the equation:

$$STEP = SPAN / (POINts - 1)$$

1991

Changing the value of STEP will change POINts, but not SPAN.

The value is instrument-dependent at \*RST, but is coupled to the reset value of POINts and SPAN.

#### 19.21.8 :POINts <numeric\_value>

SOURce:SWEep:POINts

The number of points in a stepped sweep. This parameter is not used if GENERation is ANALog. In a linear sweep, this value is coupled to sweep step by the equation:

$$STEP = SPAN / (POINts - 1)$$

1991

If POINts are changed, STEP will also be changed, but not SPAN. In a logarithmic sweep, POINts will determine the number of points/decade of sweep by the equation:

$$POINts/DECade = (POINts - 1) / SPAN \text{ (in decades)}$$

1991

Note that style rules on resolution do not apply to this command. If the exact number of points specified is not available, an error is generated, and the value remains unchanged.

At \*RST, this value is set to MAXimum.

#### 19.21.9 :COUNt <numeric\_value>

SOURce:SWEep:COUNt

This command determines the number of sweeps which are enabled by a single trigger event.

At \*RST, this function is set to 1.

## 1999 SCPI Command Reference

### 19.22 TEMPeratureSubSystem

SOURce:TEMPerature

This subsystem controls the temperature of an environmental chamber.

KEYWORD	PARAMETER FORM	COMMENTS
:TEMPerature		1993
:APRobe	<numeric_list>	1993
:DWELl	<numeric_value>	1993
:LCONstants		1993
:DERivative	<numeric_value>	1993
[:GAIN]	<numeric_value>	1993
:INTEGRal	<numeric_value>	1993
:MODE	FIXed LIST PROGram	1993
:PROTection		1993
[:HIGH]		1993
:CLEar		[no query] 1993
[:LEVel]	<numeric_value>	1993
:STATe	<Boolean>	1993
:TOUT	<numeric_value>	1993
:TRIPped?		[query only] 1993
[:LOW]		1993
:CLEar		[no query] 1993
[:LEVel]	<numeric_value>	1993
:STATe	<Boolean>	1993
:TOUT	<numeric_value>	1993
:TRIPped?		[query only] 1993
:RTIMe	<numeric_value>	1993
[:SPOint]	<numeric_value>	1993

#### 19.22.1 :APRobe <numeric\_list>

SOURce:TEMPerature:APRobe

Active PRobe selects one or more probes to become part of the feedback loop to control the temperature.

A temperature controller can have one or more probes to measure temperature at various places. If more than one probe is active, the signals from the probes are averaged to control the temperature.

AT \*RST which probes are active is device dependent.

#### 19.22.2 :DWELl <numeric\_value>

SOURce:TEMPerature:DWELl

The DWELl command specifies the amount of time the temperature chamber will stay at the target temperature. At the end of the DWELL period the temperature will return to the ambient temperature. The DWELl time does not include the Ramp TIMe.

AT \*RST the value is device dependent.

**19.22.3 :LCONstants**

SOURce:TEMPerature:LCONstants

A temperature chamber is a closed loop thermal system. The Loop CONstants of such a system determine its performance. These loop constants are popularly known as pid.

The p is the proportional or GAIN control and can be imagined as a coarse control. The i is the integral (lead) compensator and d is the derivative (lag) compensator. These two compensators are additional controls to fine tune the system. The INTegral value improves the steady-state error and the DERivative value improves the transient response of a feedback control system.

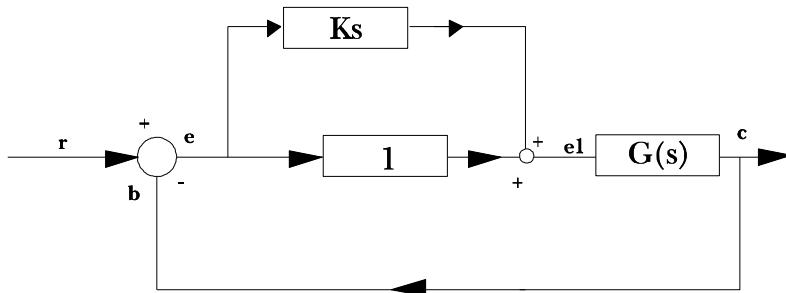
For further information on feedback control systems please refer to Vol 1: Syntax and Style - Chapter 2, Tuning a Control Loop Performance

**19.22.3.1 :DERivative <numeric\_value>**

SOURce:TEMPerature:LCONstants:DERivative

The DERivative, which is also called a lead compensator, is used to improve the transient response of the system. The unit for DERivative is (currently selected units) degrees per second.

With both GAIN and DERivative are specified, the output (heat/cold) is proportional to the input signal plus its derivative. This is shown in the figure below using Laplace transform notation.



At \*RST the value is device dependent.

**19.22.3.2 [:GAIN] <numeric\_value>**

SOURce:TEMPerature:LCONstants:GAIN

19

The value of :GAIN determines the time required to reach the target temperature. The higher the value of gain, the sooner the target temperature is achieved. The GAIN is a unitless quantity.

At \*RST the value is device dependent.

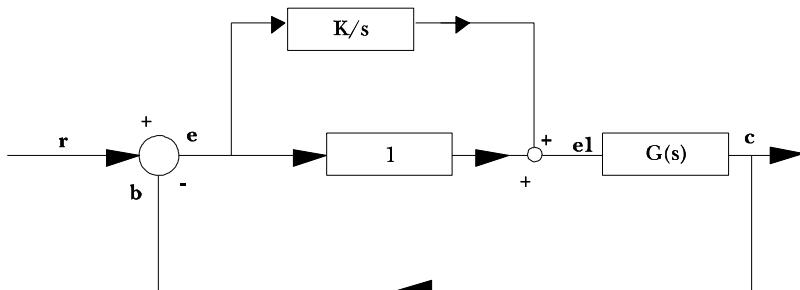
**19.22.3.3 :INTegral <numeric\_value>**

SOURce:TEMPerature:LCONstants:INTegral

The INTegral, which is also called a lag compensator, is used to minimize the steady-state error.

When the steady-state error is too large, it is possible to minimize this value by adjusting the value of INTegral. The units for INTegral are degree\*seconds.

With both GAIN and INTegral are specified, the output (heat/cold) is proportional to the input signal plus its integral. This is shown in the figure below using Laplace transform notation.



At \*RST the value is device dependent.

### 19.22.4 :MODE FIXed|LIST|PROGram

SOURce:TEMPerature:MODE

Determines which set of commands control the temperature subsystem. If FIXed is selected, the temperature is determined by the SPOint command. If LIST is selected, the temperature values are determined by the appropriate temperature list (such as LIST:). If PROGram is selected the temperature is controlled by the currently selected program specified under PROGram subsystem.

At \*RST this value is set to FIXed.

### 19.22.5 :PROTection

SOURce:TEMPerature:PROTection

Protects the temperature chamber as well as the unit under test.

#### 19.22.5.1 [:HIGH]

SOURce:TEMPerature:PROTection:HIGH

The :HIGH subsystem controls the protection. The high value must be exceeded for the protection action to be taken.

##### 19.22.5.1.1 :CLEar

SOURce:TEMPerature:PROTection:HIGH:CLEar

Causes the protection circuit to be cleared.

This command is an event and has no associated \*RST condition.

##### 19.22.5.1.2 [:LEVel] <numeric\_value>

SOURce:TEMPerature:PROTection:HIGH:LEVel

Specifies the high temperature trip level. The device will check the current value of the temperature. If the temperature exceeds the programmed protection level, the chamber will trip the protection mechanism.

At \*RST the value is device dependent.

**19.22.5.1.3 :STATe <Boolean>**

SOURce:TEMPerature:PROTection:HIGH:STATE

Enables or disables the protection mechanism.

At \*RST this value is set to ON.

**19.22.5.1.4 :TOUT <numeric\_value>**

SOURce:TEMPerature:PROTection:HIGH:TOUT

Specifies the time out in the currently specified unit of time. The device will start the time out counter anytime the difference between the controlled temperature and the SetPOint is greater than a predetermined quantity (for example 2 degrees C). If the temperature has not reached the target value and if the time out has occurred the chamber will trip the protection mechanism.

The time out counter will be reset as soon as the temperature has reached the target value within the predetermined tolerance limits.

AT \*RST the value is device dependent.

**19.22.5.1.5 :TRIPped?**

SOURce:TEMPerature:PROTection:HIGH:TRIPped?

This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

**19.22.5.2 :LOW**

SOURce:TEMPerature:PROTection:LOW

The :LOW subsystem controls the protection. The temperature must fall below the lowest level for the protection action to be taken.

**19.22.5.2.1 :CLEar**

SOURce:TEMPerature:PROTection:LOW:CLEar

Causes the protection circuit to be cleared.

This command is an event and has no associated \*RST condition.

**19.22.5.2.2 [:LEVel] <numeric\_value>**

SOURce:TEMPerature:PROTection:LOW:LEVel

Specifies the low temperature trip level. The device will check the current value of the temperature. If the temperature falls below the programmed protection level, the chamber will trip the protection mechanism.

AT \*RST the value is device dependent.

**19.22.5.2.3 :STATe <Boolean>**

SOURce:TEMPerature:PROTection:LOW:STATE

Enables or disables the protection mechanism.

At \*RST this value is set to ON.

### 19.22.5.2.4 :TOUT <numeric\_value>

SOURce:TEMPerature:PROTection:LOW:TOUT

Specifies the time out in the currently specified unit of time. The device will start the time out counter anytime the difference between the controlled temperature and the SetPOint is greater than a predetermined quantity (for example 2 degrees C). If the temperature has not reached the target value and if the time out has occurred the chamber will trip the protection mechanism.

The time out counter will be reset as soon as the temperature has reached the target value within the predetermined tolerance limits.

AT \*RST the value is device dependent.

### 19.22.5.2.5 :TRIPPed?

SOURce:TEMPerature:PROTection:LOW:TRIPPed?

This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

### 19.22.6 :RTIMe <numeric\_value>

SOURce:TEMPerature:RTIMe

The Ramp TIMe command instructs the environmental chamber the amount of time it should take to reach the target temperature.

The unit for RTIMe time is based on the currently selected unit.

AT \*RST the value is device dependent.

### 19.22.7 [:SPOint] <numeric\_value>

SOURce:TEMPerature:SPOint

The SetPOint command sets the target temperature of the environmental chamber. The units are the current temperature units.

19.23 **VOLTage Subsystem**

SOURce:VOLTage

This subsection controls the signal amplitude characteristics of the source.

There is only one subsystem for each of the main amplitude characteristics of the signal. Thus, AC voltage and DC voltage are both placed under the VOLTage subsystem. A device which implemented both simultaneously would implement them as two separate sources, one AC and one DC.

KEYWORD	PARAMETER FORM	COMMENTS
:VOLTage		
:ATTenuation	<numeric_value>	
:AUTO	<Boolean> ONCE	
:ALC		
[:STATe]	<Boolean>	
:SEARch	<Boolean> ONCE	
:SOURce	INTernal DIODe PMETer MMHead	
:BANDwidth	<numeric_value>	
[:BWIDth]		
:AUTO	<Boolean> ONCE	
:CENTer	<numeric_value>	
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	<numeric_value>	
:AUTO	<Boolean> ONCE	1991
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:TRIGgered		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:LIMIT		
[:AMPLitude]	<numeric_value>	
:OFFSet	<numeric_value>	
:HIGH	<numeric_value>	
:LOW	<numeric_value>	
:STATe	<Boolean>	
:MANual	<numeric_value>	
:MODE	FIXed SWEep LIST	
:PROTection		
[:LEVel]	<numeric_value>	
:STATE		
:TRIPped?		[query only]
:CLEar		[no query]

KEYWORD	PARAMETER FORM	COMMENTS
:RANGE	<numeric_value>	
:AUTO	<Boolean> ONCE	
:REFERENCE	<numeric_value>	
:STATE	<Boolean>	
:SLEW	<numeric_value>	
:SPAN	<numeric_value>	
:HOLD	<Boolean>	
:LINK	CENTER STARt STOP	
:FULL		
:STARt	<numeric_value>	
:STOP	<numeric_value>	[no query]

**19.23.1 :ATTenuation <numeric\_value>**

SOURce:VOLTage:ATTenuation

Sets the ATTenuation level. Note that when increasing the level by 10 dB the magnitude of the outgoing signal as well as the LEVel will be decreased by 10 dB.

Default units are as determined in the UNIT system. This is coupled to LEVel.

**19.23.1.1 :AUTO <Boolean>**

SOURce:VOLTage:ATTenuation:AUTO

Couples the attenuator to LEVel.

Programming a specified attenuation sets AUTO OFF.

See SOURce subsystem introduction for additional explanation.

At \*RST, AUTO is set to ON.

**19.23.2 :ALC**

SOURce:VOLTage:ALC

This subsystem command controls the automatic leveling control of the source.

**19.23.2.1 [:STATe] <Boolean>**

SOURce:VOLTage:ALC:STATe

Controls whether the ALC loop controls the output level.

At \*RST, this value is device-dependent.

**19.23.2.2 :SEARch <Boolean>|ONCE**

SOURce:VOLTage:ALC:SEARch

SEARch enables a form of leveling where the output level is calibrated by momentarily closing the leveling loop. Once the correct amount of level adjustment has been determined to produce the desired output level, the leveling loop is opened.

When ON, the search routine will be done anytime the output level requires change (for example, when LEVel is changed).

Selecting SEARch ONCE will have the effect of setting SEARch to ON and then OFF. A modulator setting is determined by the system which will not be changed until the output level is explicitly reprogrammed.

At \*RST, this value is set to OFF.

### 19.23.2.3 :SOURce INTERNAL|DIODe|PMETer|MMHead

SOURce:VOLTage:ALC:SOURce

Selects the source of the feedback signal for ALC. The parameters have the following meanings:

- INTERNAL — The ALC feedback signal is measured from a point inside the source.
- DIODe — The ALC feedback is being fed from an external diode detector.
- PMETer — The ALC signal is coming from a power meter.
- MMHead — A millimeter head, with built in leveling, is being used to supply the ALC signal.

At \*RST, this value is INTERNAL.

### 19.23.2.4 :BANDwidth|BWIDth <numeric\_value>

SOURce:VOLTage:ALC:BANDwidth

Controls the bandwidth of the ALC feedback signal. This parameter is in units of Hz.

At \*RST, this value is device-dependent.

#### 19.23.2.4.1 :AUTO <Boolean>|ONCE

SOURce:VOLTage:ALC:BANDwidth:AUTO

Couples the bandwidth of the ALC feedback signal to instrument-dependent parameters. When AUTO is ON, the best bandwidth is chosen to reflect the current instrument state.

Explicitly selecting a value for ALC:BANDwidth sets AUTO OFF.

At \*RST, this value is set to ON.

### 19.23.3 :CENTer <numeric\_value>

SOURce:VOLTage:CENTer

Sets the center amplitude.

At \*RST, this value is set to MINimum.

### 19.23.4 [:LEVel]

SOURce:VOLTage:LEVel

This subsystem is used to control the signal amplitude when the source is operating in a continuous or FIXed MODE.

#### 19.23.4.1 [:IMMEDIATE]

SOURce:VOLTage:LEVel:IMMEDIATE

Indicates that the subsequent specification of a new signal amplitude is to be processed by the device without waiting for further commands.

**19.23.4.1.1 [:AMPLitude] <numeric\_value>**

SOURce:VOLTage:LEVel:IMMEDIATE:AMPLitude

Sets the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal.

Note the optional nodes exist to enable VOLTage to be directly followed by a <numeric\_value>. This is useful for programming simple devices in a straightforward manner.

At \*RST, the signal being sourced should be set to a “safe” condition. This is generally achieved by setting the amplitude to its minimum value in conjunction with OUTPut:STATe subsystem.

**19.23.4.1.1.1 :AUTO <Boolean>|ONCE**

SOURce:VOLTage:LEVel:IMMEDIATE:AMPLitude:AUTO

If AUTO ON is selected, the current setting will be adjusted automatically when a new voltage setting (with the existing current setting) exceeds the maximum power limit. If AUTO OFF is selected a new voltage setting will cause an execution error -221 (Settings conflict), and the instrument state is device dependent when the maximum power limit is exceeded.

\*RST condition: AUTO OFF

**19.23.4.1.2 :OFFSet <numeric\_value>**

SOURce:VOLTage:LEVel:IMMEDIATE:OFFSet

Sets the non-time varying component of the signal that is added to the time varying signal specified in AMPLitude, in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. When SOURce:FUNction is DC, the effect of the OFFSet is device-dependent.

At \*RST, the signal being sourced should be set to a “safe” condition. This is generally achieved by setting the offset to its minimum value in conjunction with OUTPut:STATe subsystem.

**19.23.4.1.3 :HIGH <numeric\_value>**

SOURce:VOLTage:LEVel:IMMEDIATE:HIGH

Sets the more positive peak of a time varying signal. It is used in conjunction with LOW.

At \*RST, the signal being sourced should be set to a “safe” condition. This is achieved by setting this to a minimum value in conjunction with OUTPut:STATe subsystem.

**19.23.4.1.4 :LOW <numeric\_value>**

SOURce:VOLTage:LEVel:IMMEDIATE:LOW

Sets the more negative peak of a time varying signal, it is used in conjunction with HIGH.

At \*RST, the signal being sourced should be set to a “safe” condition. This is achieved by setting this to a minimum value in conjunction with OUTPut:STATe subsystem.

**19.23.4.2 :TRIGgered**

SOURce:VOLTage:LEVel:TRIGgered

Indicates that subsequent specification of a new signal amplitude is to be transferred to the IMMEDIATE value upon receipt of a trigger signal.

At \*RST, after the receipt of an ABORT command or upon being triggered, the value in the TRIGGERED subsystem tracks the values in the IMMEDIATE subsystem until a command in the TRIGGERED subsystem is received. The purpose of tracking is so that spurious triggers do not change the value of LEVel unexpectedly.

**19.23.4.2.1 [:AMPLitude] <numeric\_value>**

SOURce:VOLTage:LEVel:TRIGgered:AMPLitude

Sets the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal.

**19.23.4.2.2 :OFFSet <numeric\_value>**

SOURce:VOLTage:LEVel:TRIGgered:OFFSet

Sets the non-time varying component of the signal that is added to the time varying signal specified in AMPLitude. The units are set to the default value, or to a different value under the UNIT subsystem. When SOURce:FUNCTION is DC, the effect of OFFSet is device-dependent.

**19.23.4.2.3 :HIGH <numeric\_value>**

SOURce:VOLTage:LEVel:TRIGgered:HIGH

This command is used to set the more positive peak of a time varying signal. It is used in conjunction with LOW.

**19.23.4.2.4 :LOW <numeric\_value>**

SOURce:VOLTage:LEVel:TRIGgered:LOW

This command is used to set the more negative peak of a time varying signal. It is used in conjunction with HIGH.

**19.23.5 :LIMIT**

SOURce:VOLTage:LIMit

Sets the maximum bounds on the output value. Setting a larger value will cause the output level to be clamped to the LIMIT value.

**19.23.5.1 [:AMPLitude] <numeric\_value>**

SOURce:VOLTage:LIMit:AMPLitude

Sets the limit on the actual magnitude of the unswept output signal in terms of the current operating units. The units are set to the default value, or alternately to a different value under the UNIT subsystem. AMPLitude may be used to specify the level for either a time varying or non-time varying signal. OFFSet may only be used with AMPLitude, where AMPLitude is used to specify a time varying load.

## 1999 SCPI Command Reference

Explicit reference to the AMPLitude node is optional. Any device that implements this subsystem must accept and process commands, with or without the AMPLitude node, and respond to them in the same way.

At \*RST, this value is device-dependent.

### 19.23.5.2 :OFFSet <numeric\_value>

SOURce:VOLTage:LIMit:OFFSet

Sets a non-time varying component limit of signal that is added to the time varying signal specified in AMPLitude. The units are set to the default value, alternately to a different value under the UNIT subsystem.

At \*RST, this value is device-dependent.

### 19.23.5.3 :HIGH <numeric\_value>

SOURce:VOLTage:LIMit:HIGH

Sets the more positive peak limit of a time varying signal, it is used in conjunction with LOW.

At \*RST, this value is device-dependent.

### 19.23.5.4 :LOW <numeric\_value>

SOURce:VOLTage:LIMit:LOW

Sets the more negative peak limit of a time varying signal, it is used in conjunction with HIGH.

At \*RST, this value is device-dependent.

### 19.23.5.5 :STATe <Boolean>

SOURce:VOLTage:LIMit:STATe

Controls whether the LIMit is enabled.

At \*RST, this value is set to ON.

### 19.23.6 :MANual <numeric\_value>

SOURce:VOLTage:MANual

Allows manual adjustment of the amplitude between the sweep limits. The actual amplitude level is determined by the parameter only if SWEep:MODE is set to MANual and the amplitude MODE is set to SWEep. If the sweep limits are changed such that the manual amplitude value would be outside the limits, the manual amplitude value will be set to the nearest limit.

At \*RST, this value is device-dependent.

### 19.23.7 :MODE FIXed|SWEep|LIST

SOURce:VOLTage:MODE

Determines which set of commands control the amplitude subsystem. If FIXed is selected, the amplitude is determined by the LEVel command under the appropriate subsystem. If SWEep is selected, the source is in the swept mode and amplitude is determined by START, STOP, CENTER, SPAN, and MANual commands under the appropriate subsystem. If LIST is

selected, the amplitude values are determined by the appropriate amplitude list (such as LIST:).

At \*RST, this value is set to FIXed.

#### 19.23.8 :PROtection

SOURce:VOLTage:PROtection

Controls the protection circuits.

##### 19.23.8.1 [:LEVel] <numeric\_value>

SOURce:VOLTage:PROtection:LEVel

This command sets the output level at which the output protection circuit will trip.

##### 19.23.8.2 :STATE <Boolean>

SOURce:VOLTage:PROtection:STATE

Controls whether the output protection circuit is enabled.

At \*RST, this value is set to ON.

##### 19.23.8.3 :TRIPPed?

SOURce:VOLTage:PROtection:TRIPPed?

This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

TRIPPed only has a query form and thus has no associated \*RST condition.

##### 19.23.8.4 :CLEAR

SOURce:VOLTage:PROtection:CLEAR

Causes the protection circuit to be cleared.

This command is an event and has no associated \*RST condition.

#### 19.23.9 :RANGE <numeric\_value>

SOURce:VOLTage:RANGE

Sets a range for the output amplitude. This command is very hardware-specific.

##### 19.23.9.1 :AUTO <Boolean>|ONCE

SOURce:VOLTage:RANGE:AUTO

Couples the RANGE to an instrument-determined value. When AUTO is ON, the best range is chosen to reflect the current instrument state.

Explicitly selecting a value for RANGE sets AUTO OFF.

At \*RST, this value is set to ON.

##### 19.23.10 :REFERENCE <numeric\_value>

SOURce:VOLTage:REFerence

Sets a reference value which, if STATE is ON, allows all amplitude parameters to be queried/set as relative to the reference value.

At \*RST, this value is device-dependent.

### 19.23.10.1 :STATe <Boolean>

SOURce:VOLTage:REFerence:STATe

Determines whether amplitude is measured/output in absolute or relative mode. If STATe is ON, then amplitude is referenced to the value set in REFerence.

At \*RST, this value is set to OFF.

### 19.23.11 :SLEW <numeric\_value>

SOURce:VOLTage:SLEW

Sets the slew rate of the output change when a new output level is programmed. The units are in (the currently active) amplitude unit/sec.

At \*RST, this value is device-dependent.

### 19.23.12 :SPAN <numeric\_value>

SOURce:VOLTage:SPAN

Sets the amplitude span. If the current amplitude unit is logarithmic (dBm, dBuV, etc), then the unit of SPAN is dB. Otherwise SPAN is programmed in the current amplitude unit.

At \*RST, this value is set to 0.

### 19.23.12.1 :HOLD <Boolean>

SOURce:VOLTage:SPAN:HOLD

Provides a mechanism to prevent the SPAN from being changed implicitly by the defined coupling between STARt, STOP, CENTER and SPAN. When HOLD is set to ON, SPAN shall only be changed by explicitly sending the SPAN command.

At \*RST, this value is set to OFF.

### 19.23.12.2 :LINK CENTER|STARt|STOP

SOURce:VOLTage:SPAN:LINK

Allows the default couplings for SPAN to be overridden. LINK selects the parameter, either CENTER, STARt or STOP, that shall not be changed when SPAN's value is changed. For example, if LINK is set to STARt then changing SPAN shall cause CENTER and STOP, not STARt to change.

At \*RST, this value is set to CENTER, to be compatible with the default definition for the couplings.

### 19.23.12.3 :FULL

SOURce:VOLTage:SPAN:FULL

When this command is received, STARt amplitude is set to its minimum value and STOP amplitude is set to its maximum value. CENTER amplitude and SPAN are set to their coupled values.

This command is an event rather than a state. Therefore it has no associated query, or meaning at \*RST.

### 19.23.13 :STARt <numeric\_value>

SOURce:VOLTage:STARt

Sets STARt amplitude.

## 1999 SCPI Command Reference

At \*RST, this value is set to MINimum.

### 19.23.14 :STOP <numeric\_value>

SOURce:VOLTage:STOP

Sets STOP amplitude.

At \*RST, this value is set to MINimum.

## **1999 SCPI Command Reference**

## 20 STATus Subsystem

This subsystem controls the SCPI-defined status-reporting structures. SCPI defines, in addition to those in *IEEE 488.2*, QUEStionable, OPERation, Instrument SUMmary and INSTrument registers. These registers conform to the *IEEE 488.2* specification and each may be comprised of a condition register, an event register, an enable register, and negative and positive transition filters. The purpose and definition of the SCPI-defined registers is described in “Volume 1: Syntax and Style”.

SCPI also defines an *IEEE 488.2* queue for status. The queue provides a human readable record of instrument events. The application programmer may individually enable events into the queue. STATus:PRESet enables errors and disables all other events. If the summary of the queue is reported, it shall be reported in bit 2 of the status byte register. A subset of error/event numbers is defined by SCPI. Additional error/event numbers will be defined at a later date.

KEYWORD	PARAMETER FORM	NOTES
STATus		
:OPERation		
:BIT<n>	n=8-12	1999
:CONDITION?		[query only] 1999
:ENABLE	<NRr>   <non-decimal_numeric>	1999
[:EVENT]?		[query only] 1999
:NTRansition	<NRr>   <non-decimal_numeric>	1999
:PTRansition	<NRr>   <non-decimal_numeric>	1999
:CONDition?		[query only]
:ENABLE	<NRf>   <non-decimal numeric>	1995
[:EVENT]?		[query only]
:INSTrument		
:CONDITION?		[query only]
:ENABLE	<NRf>   <non-decimal numeric>	1995
[:EVENT]?		[query only]
:ISUMmary<n>		
:CONDITION?		[query only]
:ENABLE	<NRf>	
[:EVENT]?		[query only]
:NTRansition	<NRf>	
:PTRansition	<NRf>	
:NTRansition	<NRf>   <non-decimal numeric>	1995
:PTRansition	<NRf>   <non-decimal numeric>	1995
:MAP	<NRf>,<NRf>	
:NTRansition	<NRf>   <non-decimal numeric>	1995
:PTRansition	<NRf>   <non-decimal numeric>	1995
:PRESet		[no query]
:QUEStionable		
:BIT<n>	n=9-12	1999

## 1999 SCPI Command Reference

KEYWORD	PARAMETER FORM	NOTES
:CONDition?		[query only] 1999
:ENABle	<NRr>   <non-decimal numeric>	1999
[:EVENT?]		[query only] 1999
:NTRansition	<NRr>   <non-decimal numeric>	1999
:PTRansition	<NRr>   <non-decimal numeric>	1999
:CONDition?		[query only] 1999
:ENABle	<NRf>   <non-decimal numeric>	1995
[:EVENT?]		[query only] 1991
:INSTRument		1991
:CONDition?		[query only] 1991
:ENABle	<NRf>   <non-decimal numeric>	1991,5
[:EVENT?]		[query only] 1991
:ISUMmary<n>		1991
:CONDition?		[query only] 1991
:ENABle	<NRf>	1991
[:EVENT?]		[query only] 1991
:NTRansition	<NRf>	1991
:PTRansition	<NRf>	1991
:NTRansition	<NRf>   <non-decimal numeric>	1991,5
:PTRansition	<NRf>   <non-decimal numeric>	1991,5
:MAP	<NRf>,<NRf>	
:NTRansition	<NRf>   <non-decimal numeric>	1995
:PTRansition	<NRf>   <non-decimal numeric>	1995
:VOLTage		
:CURRent		
:TIME		
:POWER		
:TEMPerature		
:FREQuency		
:PHASe		
:MODulation		
:CALibration		
:CONDition?		[query only] 1995
:ENABle	<NRf>   <non-decimal numeric>	1995
[:EVENT?]		[query only] 1995
:NTRansition	<NRf>   <non-decimal numeric>	1995
:PTRansition	<NRf>   <non-decimal numeric>	1995

The status reporting commands are orthogonal. Status register bits are classified as either terminal (reporting a single class of events) or summary (reporting several classes of events). Summary bits have the same register structure behind them. This is called register **fan-out**. The commands to access each register are always the same and are described below. The name of each subsystem follows the name of the associated bit in the next higher status

structure. Therefore, the event register associated with the frequency error bit of the questionable signal/data register set would be accessed by the command:

```
STATus:QUESTIONable:FREQuency:EVENT?
```

Certain commands in this subsystem are required to be implemented by all SCPI instruments. These commands are described in section 4.2.1 of “Volume 1: Syntax and Style.”

The parameters to the commands are described as <NRf> as defined in *IEEE 488.2* and not as SCPI’s <numeric\_value>. This means that UP, DOWN, MINimum, and MAXimum are not accepted for these commands. When a STATus command is queried, the return form shall always be an <NR1> value.

The following commands can be applied to all SCPI registers by prefixing the command with the node(s) that represent the particular register to be controlled.

## 20.1

### **:OPERation**

STATus:OPERation

The OPERation status register contains conditions which are part of the instrument’s normal operation.

See Volume 1 Section 9.3 for more detailed information.

### 20.1.1

#### **:BIT<n>**

STATus:OPERation:BIT<n>

This command accesses the user-definable bits in the OPERation register set. The value of <n> is restricted from 8 to 12 and represents bits 8 through 12 in the :STATus:OPERation status register. This allows standardized access to user-definable terminal and summary bits.

### 20.1.2

#### **:CONDITION?**

STATus:OPERation:CONDition?

Returns the contents of the condition register associated with the status structure defined in the command. Reading the condition register is nondestructive.

The response is (NR1 NUMERIC RESPONSE DATA) (range: 0 through 32767) unless changed by the :FORMAT:SREGister command.

### 20.1.3

#### **:ENABLE <NRf> | <non-decimal numeric>**

STATus:OPERation:ENABLE

Sets the enable mask which allows true conditions in the event register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the associated summary bit.

The command accepts parameter values of either format in the range 0 through 65535 (decimal) without error.

The query response format is <NR1> unless changed by the :FORMAT:SREGister command. Note that 32767 is the maximum value returned as the most-significant bit of the register cannot be set true.

**20.1.4 [:EVENT?]**

STATus:OPERation:EVENT?

This query returns the contents of the event register associated with the status structure defined in the command.

The response is (NR1 NUMERIC RESPONSE DATA) (range: 0 through 32767) unless changed by the :FORMat:SREGister command.

Note that reading the event register clears it.

**20.1.5 :MAP <NRf>,<NRf>**

STATus:OPERation:MAP

Maps an error from the list of possible events the instrument can generate into a specified bit in the OPERation or QUESTionable register. The first parameter is the specified bit location in the OPERation or QUESTionable register. The second parameter is the event number. For example, to map event number 678 into bit 9 of the QUESTionable status register, the programmer would send:

```
STATus:QUESTionable:MAP 9,678
```

**20.1.6 :NTRansition <NRf> | <non-decimal numeric>**

STATus:OPERation:NTRansition

Sets the negative transition filter. Setting a bit in the negative transition filter shall cause a 1 to 0 transition in the corresponding bit of the associated condition register to cause a 1 to be written in the associated bit of the corresponding event register.

The command accepts parameter values of either format in the range 0 through 65535 (decimal) without error.

The query response format is <NR1> unless changed by the :FORMat:SREGister command. Note that 32767 is the maximum value returned as the most-significant bit of the register cannot be set true.

**20.1.7 :PTRansition <NRf> | <non-decimal numeric>**

STATus:OPERation:PTRansition

Sets the positive transition filter. Setting a bit in the positive transition filter shall cause a 0 to 1 transition in the corresponding bit of the associated condition register to cause a 1 to be written in the associated bit of the corresponding event register.

The command accepts parameter values of either format in the range 0 through 65535 (decimal) without error.

The query response format is <NR1> unless changed by the :FORMat:SREGister command. Note that 32767 is the maximum value returned as the most-significant bit of the register cannot be set true.

**20.2 :PRESet**

STATus:PRESet

The PRESet command is an event that configures the SCPI and device-dependent status data structures such that device-dependent events are reported at a higher level through the

## 1999 SCPI Command Reference

mandatory part of the status-reporting mechanism. Device-dependent events are summarized in the mandatory structures. The mandatory structure is defined in part by *IEEE 488.2*; SCPI-required structures compose the rest. The mandatory part of the status-reporting mechanism provides a device-independent interface for determining the gross status of a device.

The PRESet command affects only the enable register, the transition filter registers, and queue enabling for the SCPI-mandated and device-dependent status data structures. PRESet does not affect either the “status byte” or the “standard event status” as defined by IEEE 488.2. PRESet does not clear any of the event registers or any item from the error/event queue. The \*CLS command is used to clear all event registers and queues in the device status-reporting mechanism.

For the device-dependent status data structures, the PRESet commands shall set the enable register to all 1’s and the transition filter register to a device-dependent value. The device-dependent value shall be either PTR, NTR, or both, such that the device-dependent event shall be propagated to the structure that summarizes that event. For the SCPI mandatory status data structures, the PRESet command shall set the transition filter registers to recognize only positive transitions and set the enable register to 0’s. The PREset command shall also set error/event queue enabling to report only errors.

An application may select the events that shall cause a service request at the mandatory SCPI status data structure level, rather than excluding those that are of no interest to the application. Further, there is no requirement on the application to set up the device-dependent structures for normal device operation.

For this concept to be extended to encompass the *IEEE 488.2*-mandated status data structures, the following sequence of commands is required when initializing the device:

- \*CLS – Clears all event status registers and queues
- \*SRE 0 – Clears the *IEEE 488.2*-mandated service request enable register
- \*ESE 0 – Clears the *IEEE 488.2*-mandated standard event status enable register
- STATus:PRESet – Presets all other registers and queues

## 1999 SCPI Command Reference

The table below indicates the effects of various commands upon the status data structures in a device.

	SCPI Transition Filters	SCPI Enable Registers	SCPI Event Registers	SCPI Error/Event Queue Enable	SCPI Error/Event Queue	IEEE 488.2 Registers ESE SRE	IEEE 488.2 Registers SESR STB
*RST	none	none	none	none	none	none	none
*CLS	none	none	clear	none	clear	none	clear
power-on	preset <sup>#</sup>	preset <sup>#</sup>	clear <sup>#</sup>	preset <sup>#</sup>	clear <sup>#</sup>	clear <sup>#</sup>	clear <sup>#</sup>
STATus:PRESet	preset	preset	none	preset	none	none	none

#Occurs if the power-on state clear flag is true. Effect changes to none if the power-on state clear flag is false.

The following table defines the effect of STATus:PRESet

Register	Filter/enable	PRESet value =
OPERational	ENABLE	0's
	PTR	1's
	NTR	0's
QUEstionable	ENABLE	0's
	PTR	1's
	NTR	0's
ISUMmary	ENABLE	0's
	PTR	1's
	NTR	0's
INSTrument	ENABLE	1's
	PTR	device-dependent#
	NTR	device-dependent#
All others	ENABLE	1's
	PTR	device-dependent#
	NTR	device-dependent#

# Requires either PTR, NTR, or both to be set.

**20.3 :QUESTIONable**

STATus:QUESTIONable?

The QUESTIONable status register set contains bits which give an indication of the quality of various aspects of the signal.

A bit set in the condition register indicates that the data currently being acquired or generated is of questionable quality due to some condition affecting the parameter associated with that bit. For example, if the FREQ bit were set, this would mean that the frequency accuracy of the signal was of questionable quality.

See Volume 1 Section 9.4 for more detailed information.

**20.3.1 :BIT<n>**

STATus:QUESTIONable:BIT&lt;n&gt;

This command accesses the user-definable bits in the QUESTIONable register set. The value of <n> is restricted from 9 to 12 and represents bits 9 through 12 in the :STATus:QUESTIONable status register. This allows standardized access to user-definable terminal and summary bits.

Note that this definition is different from that of :STATus:OPERation:BIT in that the bit number is restricted to the range of 9 through 12, while the range for :STATus:OPERation:BIT is restricted to 8 through 12.

**20.3.2 :CONDITION?**

STATus:QUESTIONable:CONDition?

Defined the same as STATus:OPERation:CONDITION. See Section 20.1.2 for details.

**20.3.3 :ENABLE <NRf> | <non-decimal numeric>**

STATus:QUESTIONable:ENABLE

Defined the same as STATus:OPERation:ENABLE. See Section 20.1.3 for details.

**20.3.4 [:EVENT?]**

STATus:QUESTIONable:EVENT?

Defined the same as STATus:OPERation:EVENt. See Section 20.1.4 for details.

**20.3.5 :MAP <NRf>,<NRf>**

STATus:QUESTIONable:MAP

Defined the same as STATus:OPERation:MAP. See Section 20.1.5 for details.

20

**20.3.6 :NTRansition <NRf> | <non-decimal numeric>**

STATus:QUESTIONable:NTRansition

Defined the same as STATus:OPERation:NTRansition. See Section 20.1.6 for details.

**20.3.7 :PTRansition <NRf> | <non-decimal numeric>**

STATus:QUESTIONable:PTRansition

Defined the same as STATus:OPERation:PTRansition. See Section 20.1.7 for details.



## 21 SYSTem Subsystem

The SYSTem subsystem collects the functions that are not related to instrument performance. Examples include functions for performing general housekeeping and functions related to setting global configurations, such as TIME or SECurity.

KEYWORD	PARAMETER FORM	NOTES
SYSTem		
:ALTerate	<numeric_value>	
:STATe	<Boolean>	
:BEEPer		
:FREQuency	<numeric_value>	
[:IMMEDIATE]	[<frequency>[,<time>[,<volume>]]]	
:STATe	<Boolean>	
:TIME	<numeric_value>	
:VOLume	<numeric_value>	
:CAPability?		[query only] 1994
:COMMUnicate		
:CENTronics		1993
:FEED	<data_handle>	1993
:GPIB		
RDEVice		1993
:ADDResS	<numeric_value>[,<numeric_value>]	1993
:FEED	<data_handle>	1993
[:SELF]		1993
:ADDResS	<numeric_value>[,<numeric_value>]	1993
:SERial		
:CONTrol		1991
:DTR	ON OFF STANdard IBFull	1991
:RTS	ON OFF STANdard IBFull RFR	1991
:FEED	<data_handle>	1993
[:RECeive]		
:BAUD	<numeric_value>	
:BITS	<numeric_value>	
:PACE	XON ACK NONE	
:THReShold		1991
:STARt	<numeric_value>	1991
:STOP	<numeric_value>	1991
:PARity		
:CHECk	<Boolean>	
[:TYPE]	EVEN ODD ZERO ONE NONE	
:SBITS	<numeric_value>	
:TRANsmitt		
:AUTO	<Boolean>	
:BAUD	<numeric_value>	

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:BITS	<numeric_value>	
:DELay	<numeric_value>	
:PACE	XON ACK NONE	
:PARity		
[:TYPE]	EVEN ODD ZERO ONE NONE	
:SBITS	<numeric_value>	
:SOCKet	<n>	1999
:ADDReSS	<string> (IPaddress Host name)	1999
:CONNect		1999
:DISConnect		1999
:FEED	<data_handle>{,<data_handle>}	1999
:OCONDition	<event_handle>	1999
:SCONDition	<event_handle>	1999
:LISTen		1999
:PORT	<numeric_value> (port #)	1999
:TYPE	TCP   UDP	1999
:CPON	<card_destination> ALL	[event; no query]
:CTYPe?	<card_destination>	[query only]
:DATE	<numeric_value>,<numeric_value>,<numeric_value>	
:ERRor		
:ALL?		[query only] 1996
:CODE		1996
:ALL?		[query only] 1996
[:NEXT]?		[query only] 1996
:COUNT?		[query only] 1996
:ENABLE		1999
:ADD	<numeric_list>	[no query] 1999
:DELeTe	<numeric_list>	[no query] 1999
[:LIST]	<numeric_list>	1999
[:NEXT]?		[query only] 1996
:HELP		1992
:HEADers?		[query only] 1995
:SYNTax?	<command_header>	[query only] 1992
:KEY	<numeric_value>	
:CATalog		
:DEFIne	<numeric_value>,<block>[,<string>]	
:DELeTe		
:KLOCK	<Boolean>	
:LANGuage	<string>	
:LFREquency	<numeric_value>	1992
:AUTO	<Boolean>   ONCE	1992
:LOCK		1999
:OWNer?		[query only] 1999
:RELEASE		[no query] 1999

KEYWORD	PARAMETER FORM	NOTES
:REQuest?		[event; query only] 1999 1994
:PASSword		
:CDISable	<password>	[event; no query] 1994
[:CENable]	<password>	[event; no query] 1994
:STATE?		[query only] 1994
:NEW	<current_password>,<new_password>	[event; no query] 1994
:PRESet		[event; no query]
:SECurity		
:IMMEDIATE		[event; no query] 1994
[:STATe]	<Boolean>	
:SET	<block data>	
:TIME	<numeric_value>,<numeric_value>,<numeric_value>	
:TImer		1999
:COUNT	<numeric_value>	1999
[:STATe]	<Boolean>	1999
:TZONe	<numeric_value>[,<numeric_value>]	
:VERSion?		

**21.1 :ALTernate <numeric\_value>**

SYSTem:ALTerNate

Selects an alternate instrument state, which is swapped with the current state at the end of some specified action. The <numeric\_value> selects which of the states will be used. The states are created with the \*SAV command. The action is generally sweep, list sequencing or some programmable delay, depending on the device configuration.

At \*RST, this value is device-dependent.

**21.1.1 :STATe <Boolean>**

SYSTem:ALTerNate:STATe

Controls whether the alternate state mode is enabled. While it is ON, the device shall swap the active state with the one stored in the register specified in ALTerNate.

At \*RST, this parameter is set to OFF.

**21.2 :BEEPer**

SYSTem:BEEPer

This subsystem controls the audible beeper of the instrument.

21

**21.2.1 :FREQuency <numeric\_value>**

SYSTem:BEEPer:FREQuency

Programs the frequency of audible tones. It also sets the default frequency for the [:IMMEDIATE] command. The parameter is specified in Hz. The range of valid parameters is instrument-dependent.

At \*RST, this value is device-dependent.

### 21.2.2 [:IMMEDIATE][<frequency>[,<time>[,<volume>]]]

SYSTem:BEEPer:IMMEDIATE

The receipt of this command causes an audible tone to be generated by the instrument. Note that this command generates an event and therefore it has no associated \*RST state or query form.

The allowable values of the parameters <frequency>, <time> and <volume> are instrument-dependent. The frequency parameter is specified in Hz. The duration time parameter is specified in seconds. The volume parameter range is from 0 to 1, with 1 indicating the maximum volume that the instrument can produce and 0 indicating minimum volume.

### 21.2.3 :STATE <Boolean>

SYSTem:BEEPer:STATE

Enables/disables the beeper. When STATE OFF is selected, no instrument condition, except the :SYSTem:BEEPer:IMMEDIATE command, shall cause an audible beep to be emitted.

At \*RST, this value is device-dependent.

### 21.2.4 :TIME <numeric\_value>

SYSTem:BEEPer:TIME

Programs the duration of audible tones. It also sets the default duration for the [:IMMEDIATE] command. The parameter is specified in seconds. The range of valid parameters is instrument-dependent.

At \*RST, this value is device-dependent.

### 21.2.5 :VOLUME <numeric\_value>

SYSTem:BEEPer:VOLUME

Programs the volume of audible tones. It also sets the default volume for the [:IMMEDIATE] command. The parameter range is from 0 to 1, with 1 indicating the maximum volume that the instrument can produce and 0 indicating minimum volume. Actual sound pressure levels generated by any particular volume setting are instrument-dependent.

At \*RST, this value is device-dependent.

### 21.3 :CAPABILITY?

SYSTem:CAPABILITY?

This query returns an <instrumentSpecifier>. See the Compliance section in the introduction to Instrument Class Applications.

### 21.4 :COMMUNICATE

SYSTem:COMMUNICATE

The COMMUNICATE subsystem collects together the configuration of various control/communication interfaces.

### 21.4.1 :CENTRONICS

SYSTem:COMMUNICATE:CENTRONICS

The CENTRONICS subsystem sets the physical configuration of a centronics port.

**21.4.1.1 :FEED <data\_handle>**

SYSTem:COMMunicate:CENTronics:FEED

Sets or queries what data is used to FEED the remote device. Data may be streamed to an external device without sending it through the HCOPy subsystem, for example by directly feeding SENSe or CALCulate data, where no page formatting commands are needed. That is, the SYSTem:COMMunicate subsystem shall not add any page formatting to the output stream, where this is required HCOPy should be used.

At \*RST, FEED is device dependent.

**21.4.2 :GPIB**

SYSTem:COMMunicate:GPIB

The GPIB subsystem controls the physical configuration of a GPIB (*IEEE 488*) port.

**21.4.2.1 :RDEvice**

SYSTem:COMMunicate:GPIB:RDEvice

The Remote DEVice subsystem configures how the instrument uses a peripheral device, such as a printer or plotter, connected to a GPIB port.

**21.4.2.1.1 :ADDRess <numeric\_value>[,<numeric\_value>]**

SYSTem:COMMunicate:GPIB:RDEvice:ADDRess

Sets or queries the bus address of the peripheral device. The first parameter is the primary address and the optional second parameter is the secondary address. Changing the address with this does not affect the address of the peripheral device. It does affect the address to which data is sent by the instrument.

At \*RST, the value of this parameter is device dependent.

**21.4.2.1.2 :FEED <data\_handle>**

SYSTem:COMMunicate:GPIB:RDEvice:FEED

Sets or queries what data is used to FEED the peripheral device. Data may be streamed to an external device without sending it through the HCOPy subsystem, for example by directly feeding SENSe or CALCulate data, where no page formatting commands are needed. That is, the SYSTem:COMMunicate subsystem shall not add any page formatting to the output stream. Where page formatting is required, HCOPy should be used.

At \*RST, FEED is device dependent.

**21.4.2.2 [:SELF]**

SYSTem:COMMunicate:GPIB[:SELF]

The SELF subsystem sets the configuration relating to the GPIB port on the instrument itself.

**21.4.2.2.1 :ADDRess <numeric\_value>[,<numeric\_value>]**

SYSTem:COMMunicate:GPIB[:SELF]:ADDRess

Sets the GPIB address of the instrument. The first parameter is the primary address and the optional second parameter is the secondary address.

\*RST has no effect on the value of these parameters.

### 21.4.3 :SERial

SYSTem:COMMunicate:SERial

Controls the physical configuration of a serial port, such as RS-232-D, RS-422, or EIA/TIA-562. Any command that changes the configuration of the port takes effect immediately upon receipt of the Program Message Terminator.

\*RST has no effect on the value of these parameters.

#### 21.4.3.1 :CONTrol

SYSTem:COMMunicate:SERial:CONTrol

Add command which independently sets the mode in which DTR and RTS pacing operate. Since hardware handshaking can be configured to pertain to either transmit, receive, or both, a CONTrol branch separate from the transmit and receive branches was added.

##### 21.4.3.1.1 :DTR ON|OFF|STANDARD|IBFull

SYSTem:COMMunicate:SERial:CONTrol:DTR

Sets the hardware pacing scheme.

The ON parameter indicates that the DTR line should always be asserted (held in the ON condition).

The OFF parameter indicates that the DTR line should always be unasserted (held in the OFF condition).

When the STANDARD option is set for DTR, the DTR line behaves as specified in the EIA standard RS-232-D section 4.4. As stated in the standard:

“Signals on this circuit are used to control the switching of the DCE to the communication channel. The ON condition prepares the DCE to be connected to the communication channel and maintains the connection established by external means.”

A common use of the DTR line that has evolved is that of pacing. The IBFull parameter sets the DTR line to indicate when the device is ready to receive. When the number of received bytes in the input buffer of the device reaches the STOP threshold the device will de-assert the DTR line. When the number of bytes has been reduced to the START threshold, the device will assert DTR indicating that it can receive input again. The device will also monitor the state of DSR and CTS and will stop transmission if either of those lines becomes de-asserted.

\*RST has no effect on the value of this parameter.

##### 21.4.3.1.2 :RTS ON|OFF|STANDARD|IBFull|RFR

SYSTem:COMMunicate:SERial:CONTrol:RTS

Sets the hardware pacing (hand-shaking) scheme.

The ON parameter indicates that the RTS line should always be asserted.

The OFF parameter indicates that the RTS line should always be unasserted.

When the STANDARD option is set for RTS, the RTS line behaves as specified in the EIA standard RS-232-D section 4.4. As stated in the standard:

“This circuit is used to condition the local DCE for data transmission and on a half-duplex channel, to control the direction of data transmission of the local DCE.”

A common use of the RTS line that has evolved is that of pacing. Many high speed asynchronous modems use this line (paired with CTS) as receive/transmit pacing. The IBFull or its alias RFR (Ready For Receiving) parameter sets the RTS line to indicate when the device is ready to receive. When the number of received bytes in the input buffer of the device reaches the STOP threshold the device will de-assert the RTS line. When the number of bytes has been reduced to the STARt threshold, the device will assert RTS, indicating that it can receive input again. The device will also monitor the state of CTS and will stop transmission if that line becomes de-asserted.

The IBFull (RFR) mode follows the CCITT Recommendations V.42, Section 7.3.1, “Flow Control across the DTE/DCE interface, Option a) Using Circuit 133 and 106.”

\*RST has no effect on the value of this parameter.

#### **21.4.3.2 :FEED <data\_handle>**

SYSTem:COMMunicate:SERial:FEED

Sets or queries what data is used to FEED the peripheral device connected to the serial port. Data may be streamed to an external device without sending it through the HCOPy subsystem, for example by directly feeding SENSe or CALCulate data, where no page formatting commands are needed. That is, the SYSTem:COMMunicate subsystem shall not add any page formatting to the output stream, where this is required HCOPy should be used.

At \*RST, FEED is device dependent.

#### **21.4.3.3 [:RECeive]**

SYSTem:COMMunicate:SERial:RECeive

The optional node RECeive is used to affect the parameters associated with reception.

##### **21.4.3.3.1 :BAUD <numeric\_value>**

SYSTem:COMMunicate:SERial:RECeive:BAUD

Sets the baud rate.

\*RST has no effect on the value of this parameter.

##### **21.4.3.3.2 :BITS <numeric\_value>**

SYSTem:COMMunicate:SERial:RECeive:BITS

Sets the number of data bits, typically 7 or 8. The value shall be rounded to the nearest integer.

\*RST has no effect on the value of this parameter.

##### **21.4.3.3.3 :PACE XON|ACK|NONE**

SYSTem:COMMunicate:SERial:RECeive:PACE

This command sets the software pacing scheme.

The XON pacing mode follows the CCITT Recommendations V.42, Section 7.3.1, “Flow Control across the DTE/DCE interface, Option b) Using DC1/DC3 characters (XON/XOFF functions).”

\*RST has no effect on the value of this parameter.

### 21.4.3.3.3.1 :THreshold

SYSTem:COMMunicate:SERial:RECeive:PACE:THreshold

This command sets the receive pacing threshold.

#### 21.4.3.3.3.1.1 :STARt <numeric\_value>

SYSTem:COMMunicate:SERial:RECeive:PACE:THreshold:STARt

This command indicates that when the number of characters in the device's input buffer goes to or below the indicated amount and some form of pacing has been set, the device should indicate that it is ready to receive.

\*RST has no effect on the value of this parameter.

#### 21.4.3.3.3.1.2 :STOP <numeric\_value>

SYSTem:COMMunicate:SERial:RECeive:PACE:THreshold:STOP

This command indicates that when the number of characters in the device's input buffer goes to or above the indicated amount and some form of pacing has been set, the device should indicate that it is not ready to receive.

\*RST has no effect on the value of this parameter.

### 21.4.3.3.4 :PARity

SYSTem:COMMunicate:SERial:RECeive:PARity

This subsystem controls the parity of the receiving channel

#### 21.4.3.3.4.1 :CHECK <Boolean>

SYSTem:COMMunicate:SERial:RECeive:PARity:CHECK

When CHECK ON is selected, the parity shall be checked in accordance with the selected PARity:TYPE.

When CHECK is OFF, the parity TYPE setting (NONE or "other than NONE") still determines the expectation of not receiving or receiving a parity bit.

\*RST has no effect on the value of this parameter.

#### 21.4.3.3.4.2 [:TYPE] EVEN|ODD|ZERO|ONE|NONE|IGNore

SYSTem:COMMunicate:SERial:RECeive:PARity:TYPE

Sets the parity scheme that is to be employed for reception.

\*RST has no effect on the value of this parameter.

#### 21.4.3.3.5 :SBITs <numeric\_value>

SYSTem:COMMunicate:SERial:RECeive:SBITS

Sets the number of stop bits, typically 1, 1.5 or 2 bits.

\*RST has no effect on the value of this parameter.

### 21.4.3.4 :TRANsmi~~t~~

SYSTem:COMMunicate:SERial:TRANsmi~~t~~

Affects the parameters associated with transmission.

**21.4.3.4.1 :AUTO <Boolean>**

SYSTem:COMMUnicatE:SERial:TRANsmiT:AUTo

Couples the TRANsmiT parameter values to the RECeive parameter values when AUTO ON is selected.

At \*RST, AUTO ON is selected.

**21.4.3.4.2 :BAUD <numeric\_value>**

SYSTem:COMMUnicatE:SERial:TRANsmiT:BAUD

This command sets the baud rate.

\*RST has no effect on the value of this parameter.

**21.4.3.4.3 :BITS <numeric\_value>**

SYSTem:COMMUnicatE:SERial:TRANsmiT:BITs

Sets the number of data bits, typically 7 or 8. The value shall be rounded to the nearest integer.

\*RST has no effect on the value of this parameter.

**21.4.3.4.4 :DELay <numeric\_value>**

SYSTem:COMMUnicatE:SERial:TRANsmiT:DELay

Sets the minimum delay in seconds that the device must wait after receipt of a Program Message Terminator before issuing a response message. The delay does not apply to software pacing or to other low level protocols.

\*RST has no effect on the value of this parameter.

**21.4.3.4.5 :PACE XON|ACK|NONE**

SYSTem:COMMUnicatE:SERial:TRANsmiT:PACE

Sets the software pacing scheme.

The XON pacing mode follows the CCITT Recommendations V.42, Section 7.3.1, “Flow Control across the DTE/DCE interface, Option b) Using DC1/DC3 characters (XON/XOFF functions).”

\*RST has no effect on the value of this parameter.

**21.4.3.4.6 :PARity**

SYSTem:COMMUnicatE:SERial:TRANsmiT:PARity

This subsystem controls the parity of the transmitting channel

21

**21.4.3.4.6.1 [:TYPE] EVEN|ODD|ZERO|ONE|NONE**

SYSTem:COMMUnicatE:SERial:TRANsmiT:PARity:TYPE

Sets the parity scheme that is to be employed for transmission.

\*RST has no effect on the value of this parameter.

**21.4.3.4.7 :SBITs <numeric\_value>**

SYSTem:COMMUnicatE:SERial:TRANsmiT:SBITs

Sets the number of stop bits, typically 1, 1.5, or 2 bits.

\*RST has no effect on the value of this parameter.

#### 21.4.4 :SOCKet <n>

:SYSTem:COMMUnicatE:SOCKet<n>

This command identifies the socket to be opened. The syntax of the numeric suffix is as specified in 6.2.5.2 in Volume 1, Syntax and Style.

##### 21.4.4.1 :ADDResS <string>

:SYSTem:COMMUnicatE:SOCKet:ADDResS

This command allows the user to enter the IP address or host name of the remote device. If it is an IP address, it is specified using IP dot notation (e.g. 123.45.67.89). If it is a host name, it is specified by using the host name string (e.g., my\_host).

##### 21.4.4.2 :CONNect

:SYSTem:COMMUnicatE:SOCKet:CONNect

This command implements the socket connection.

##### 21.4.4.3 :DISConnect

:SYSTem:COMMUnicatE:SOCKet:DISConnect

The Disconnect command breaks the socket connection.

##### 21.4.4.4 :FEED <n> <data\_handle>{,<data\_handle>}

:SYSTem:COMMUnicatE:SOCKet:FEED

Sets or queries what data is used to FEED the communications socket.

Note: There can be multiple FEEDs for the same socket.

##### 21.4.4.4.1 :OCONDition <event\_handle>

:SYSTem:COMMUnicatE:SOCKet:FEED:OCONDition

The On Condition command, :OCONDiton, defines the rate at which data will be collected and sent to the socket FEED.

##### 21.4.4.4.2 :SCONDition <event\_handle>

:SYSTem:COMMUnicatE:SOCKet:FEED:SCONDition

The Send Condition, SCONDition, defines the rate at which the data will be transmitted on the socket.

##### 21.4.4.5 :LISTen

:SYSTem:COMMUnicatE:SOCKet:LISTen

The LISTen command instructs the device (instrument) to listen for a connection on the specified socket. This will allow data channels from one instrument to another instrument to be opened.

##### 21.4.4.6 :PORT <numeric\_value>

:SYSTem:COMMUnicatE:SOCKet:PORT

This command allows the user to enter the port number. This parameter is a 16 bit integer.

**21.4.4.7 :TYPE TCP|UDP**

SYSTem:COMMunicate:SOCKet:TYPE

This command allows the user to select the type of Ethernet port to be opened, either TCP or UDP. The query form will return either TCP or UDP.

**21.5 :CPON <card\_destination>|ALL**

SYSTem:CPON

Card Power ON resets a card within a cardcage to its power-on state. The parameter can either be a module number or a user-defined card name. If the parameter is ALL, then all addressable cards are reset to their power-on state.

SYSTem:CPON is an event; no query is allowed.

**21.6 :CTYPe? <card\_destination>**

SYSTem:CTYPe?

The Card TYPe query identifies a card within a cardcage. The response is identical in form to that defined for the “\*IDN?” query in *IEEE 488.2*.

**21.7 :DATE <year>,<month>,<day>**

SYSTem:DATE

Any instrument which has an internal calendar shall implement the following command to set the calendar:

<year>: Must be <numeric\_value>. It is rounded to the nearest integer. Its range is limited by the capability of the instrument. The year shall be entered as a four-digit number, including century and millennium information.

<month>: Must be <numeric\_value>. It is rounded to the nearest integer. Its range is 1 to 12 inclusive. The number 1 corresponds to the month January, 2 to February, and so on.

<day>: Must be <numeric\_value>. It is rounded to the nearest integer. Its range is 1 to number of days in the month from the previous parameter. A device implementing this command shall keep track of the number of days in each month, accounting for leap years through the range of years that it accepts.

The response message shall consist of three fields separated by commas:

<year>,<month>,<day>

<year>: NR1 format. Range is device-dependent.

<month>: NR1 format. Range is 1 to 12.

<day>: NR1 format. Range is 1 to 31.

This shall not be affected by a \*RST command.

21

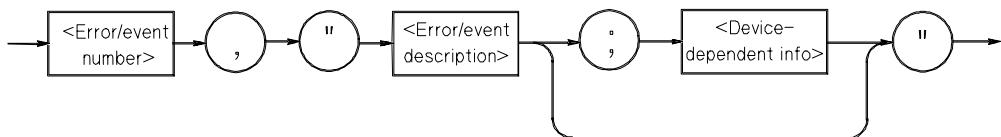
**21.8 :ERRor Subsystem**

SYSTem:ERRor

This subsystem collects commands and queries relating to the reading and control of the

error/event queue.

The error/event queue contains items that include a numerical and textual description of the error or event. Querying for the full queue item (for example, with SYSTem:ERROr[:NEXT]?) will return a response with the following syntax:



The <Error/event\_number> is a unique integer in the range [-32768, 32767]. All positive numbers are instrument-dependent. All negative numbers are reserved by the SCPI standard with certain standard error/event codes described in this document. The value, zero, is also reserved to indicate that no error or event has occurred.

The second parameter of the full response is a quoted string containing an <Error/event\_description> followed by optional <Device-dependent info> text. Each <Error/event\_number> has a unique and fixed <Error/event\_description> associated with it.

The maximum string length of <Error/event\_description> plus <Device-dependent\_info> is 255 characters. For standard defined error/event codes, the <Error/event\_description> shall be sent exactly as indicated in this document including case.

The <Device-dependent\_info> text is optional but may be useful in some instruments to provide additional information allowing the user to determine the exact error or event and its context. If used, it is set off from the <Error/event\_description> portion of the string with a semicolon. For example:

```
-131, "Invalid suffix;FREQuency:CENT 2.0E+5 dBuV"
```

Device-dependent information may be defined in greater detail for specific instrument classes in volume 4.

Where Date and Time are desired to be returned with the error message, they should be appended to the <device\_dependent\_information> separated by a semi-colon using the following format:

<date><space><time>

date - yyyy/mm/dd - year/month/day  
time - hh:mm:ss.sss - 24 hour time

Example:

```
-200, "Execution error;Torque Safety Limit
Exceeded;1996/08/15 13:22:51.01"
```

### 21.8.1 The Error/Event Queue

As errors and events are detected, they are placed in a queue. This queue is first in, first out. If the queue overflows, the last error/event in the queue is replaced with error

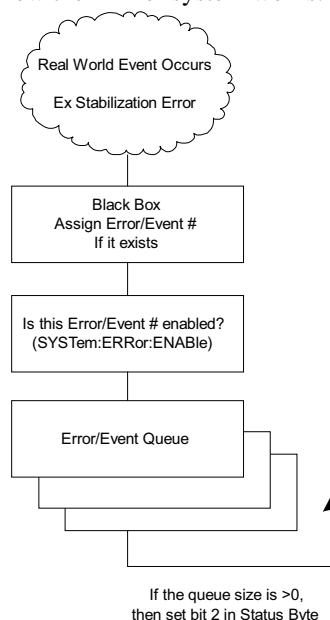
-350, "Queue overflow"

Any time the queue overflows, the least recent errors/events remain in the queue, and the most recent error/event is discarded. The minimum length of the error/event queue is 2, one position for the first error/event, and one for the "Queue overflow" message. Reading an error/event from the head of the queue removes that error/event from the queue, and opens a position at the tail of the queue for a new error/event, if one is subsequently detected.

When all errors/events have been read from the queue, further error/event queries shall return

0, "No error"

After a STATus:PRESet, the masking shall be changed so that only errors are reported. Below is an overview of how the ERRor system works.



**Figure 21.1 System Error Format**

The error/event queue shall be cleared when any of the following occur (IEEE 488.2, section 11.4.3.4):

- Upon receipt of a \*CLS command.
- Upon reading the last item from the queue.

**21.8.2 Error/Event numbers**

The system-defined error/event numbers are chosen on an enumerated (“1 of N”) basis. The SCPI-defined error/event numbers and the <error/event\_description> portions of the full queue item are listed here. The first error/event described in each class (for example, -100, -200, -300, -400) is a “generic” error. In selecting the proper Error/event number to report, more specific error/event codes are preferred, and the generic error/event is used only if the others are inappropriate.

Note the organization of the following tables. A “simple-minded” parser might implement only the XX0 errors, and a smarter one might implement all of them. A “smart and friendly” parser might use the instrument-dependent part of the error/event message string to point out the offending part of the command.

**21.8.3 No Error**

This message indicates that the device has no errors or events to report.

<b>Error Number</b>	<b>Error Description [description/explanation/examples]</b>
0	No error [The queue is completely empty. Every error/event in the queue has been read or the queue was purposely cleared by power-on, *CLS, etc.]

**21.8.4 ALL?**

SYSTem:ERRor:ALL? queries the error/event queue for all the unread items and removes them from the queue. The response is a comma separated list of number, string pairs in FIFO order. If the queue is empty, the response is 0, “No error”

Note: If the queue is not empty, the 0, “No error” is not sent.

**21.8.5 CODE****21.8.5.1 ALL?**

SYSTem:ERRor:CODE:ALL? queries the error/event queue for all the unread items and removes them from the queue. The response returns a comma separated list of only the error/event code numbers in FIFO order. If the queue is empty, the response is 0.

Note: If the queue is not empty, the 0 is not sent.

**21.8.5.2 [NEXT]?**

SYSTem:ERRor:CODE[:NEXT]? queries the error/event queue for the next item and removes it from the queue. The response returns only the error/event code number omitting the string. Except for the shortened response, the query operates identically to SYSTem:ERRor[:NEXT]?

**21.8.6 COUNT?**

SYSTem:ERRor:COUNt? queries the error/event queue for the number of unread items. As errors and events may occur at any time, more items may be present in the queue at the time it is actually read.

Note: If the queue is empty, the response is 0.

**21.8.7 :ENABLE**

SYSTem:ERRor:ENABLE

The ENABLe subsystem sets the errors and events that will be allowed into the queue.

At \*RST, this list is not changed.

**21.8.7.1 :ADD <numeric list>**

SYSTem:ERRor:ENABLE:ADD

The events included in the numeric list are added to the previous ENABLe list. The syntax of <numeric list> is the same as SYSTem:ERRor:ENABLE:LIST.

This command is an event and cannot be queried.

**21.8.7.2 :DELete <numeric list>**

SYSTem:ERRor:ENABLE:DElete

The events included in the numeric list are deleted from the previous ENABLe. The syntax of <numeric list> is the same as SYSTem:ERRor:ENABLE:LIST.

This command is an event and cannot be queried.

**21.8.7.3 [:LIST] <numeric list>**

SYSTem:ERRor:ENABLE[:LIST]

The list indicates which errors/events are allowed to enter the queue. The list allows a comma separated list or a range of numbers. In the case of a range, the end points are inclusive. For example, SYSTem:ERRor:ENABLE (-123,225:227) will allow events numbered -123, 225, 226, and 227 to enter the queue.

At \*RST, this list is not changed.

**21.8.8 [NEXT]?**

SYSTem:ERRor[:NEXT]

SYSTem:ERRor[:NEXT]? queries the error/event queue for the next item and removes it from the queue. The response returns the full queue item consisting of an integer and a string as described in the introduction to the SYSTem:ERRor subsystem.

This command is required of all SCPI implementations (See Syntax and Style, 4.2.1).

**21.8.9 Command Error**

An <error/event number> in the range [ -199 , -100 ] indicates that an *IEEE 488.2* syntax error has been detected by the instrument's parser. The occurrence of any error in this class shall cause the command error bit (bit 5) in the event status register (*IEEE 488.2*, section 11.5.1) to be set. One of the following events has occurred:

- An *IEEE 488.2* syntax error has been detected by the parser. That is, a controller-to-device message was received which is in violation of the *IEEE 488.2* standard. Possible violations include a data element which violates the device listening formats or whose type is unacceptable to the device.
- An unrecognized header was received. Unrecognized headers include incorrect device-specific headers and incorrect or unimplemented *IEEE 488.2* common commands.

## 1999 SCPI Command Reference

- A Group Execute Trigger (GET) was entered into the input buffer inside of an *IEEE 488.2 <PROGRAM MESSAGE>*.

Events that generate command errors shall not generate execution errors, device-specific errors, or query errors; see the other error definitions in this chapter.

Error Number	Error Description [description/explanation/examples]
-100	Command error [This is the generic syntax error for <b>devices</b> that cannot detect more specific errors. This code indicates only that a Command Error as defined in <i>IEEE 488.2, 11.5.1.1.4</i> has occurred.]
-101	Invalid character [A syntactic element contains a character which is invalid for that type; for example, a header containing an ampersand, SETUP&. This error might be used in place of errors -114, -121, -141, and perhaps some others.]
-102	Syntax error [An unrecognized command or data type was encountered; for example, a string was received when the <b>device</b> does not accept strings.]
-103	Invalid separator [The parser was expecting a separator and encountered an illegal character; for example, the semicolon was omitted after a program message unit, *EMC 1:CH1:VOLTS 5.]
-104	Data type error [The parser recognized a data element different than one allowed; for example, numeric or string data was expected but block data was encountered.]
-105	GET not allowed [A Group Execute Trigger was received within a program message (see <i>IEEE 488.2, 7.7</i> ).]
-108	Parameter not allowed [More parameters were received than expected for the header; for example, the *EMC common command only accepts one parameter, so receiving *EMC 0,1 is not allowed.]
-109	Missing parameter [Fewer parameters were received than required for the header; for example, the *EMC common command requires one parameter, so receiving *EMC is not allowed.]
-110	Command header error [An error was detected in the header. This error message should be used when the <b>device</b> cannot detect the more specific errors described for errors -111 through -119.]

## 1999 SCPI Command Reference

- 111 Header separator error  
[A character which is not a legal header separator was encountered while parsing the header; for example, no white space followed the header, thus \*GMC"MACRO" is an error.]
- 112 Program mnemonic too long  
[The header contains more than twelve characters (see *IEEE 488.2, 7.6.1.4.1*.)]
- 113 Undefined header  
[The header is syntactically correct, but it is undefined for this specific **device**; for example, \*XYZ is not defined for any **device**.]
- 114 Header suffix out of range  
[The value of a numeric suffix attached to a program mnemonic, see Syntax and Style section 6.2.5.2, makes the header invalid.]
- 115 Unexpected number of parameters  
[The number of parameters received does not correspond to the number of parameters expected. This is typically due an inconsistency with the number of instruments in the selected group (see section on INSTRument:DEFine:GROup).]
- 120 Numeric data error  
[This error, as well as errors -121 through -129, are generated when parsing a data element which appears to be numeric, including the nondecimal numeric types. This particular error message should be used if the **device** cannot detect a more specific error.]
- 121 Invalid character in number  
[An invalid character for the data type being parsed was encountered; for example, an alpha in a decimal numeric or a "9" in octal data.]
- 123 Exponent too large  
[The magnitude of the exponent was larger than 32000 (see *IEEE 488.2, 7.7.2.4.1*).]
- 124 Too many digits  
[The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros (see *IEEE 488.2, 7.7.2.4.1*).]
- 128 Numeric data not allowed  
[A legal numeric data element was received, but the **device** does not accept one in this position for the header.]
- 130 Suffix error  
[This error, as well as errors -131 through -139, are generated when parsing a suffix. This particular error message should be used if the **device** cannot detect a more specific error.]

## 1999 SCPI Command Reference

- 131 Invalid suffix  
[The suffix does not follow the syntax described in *IEEE 488.2, 7.7.3.2*, or the suffix is inappropriate for this **device**.]
- 134 Suffix too long  
[The suffix contained more than 12 characters (see *IEEE 488.2, 7.7.3.4*).]
- 138 Suffix not allowed  
[A suffix was encountered after a numeric element which does not allow suffixes.]
- 140 Character data error  
[This error, as well as errors -141 through -149, are generated when parsing a character data element. This particular error message should be used if the **device** cannot detect a more specific error.]
- 141 Invalid character data  
[Either the character data element contains an invalid character or the particular element received is not valid for the header.]
- 144 Character data too long  
[The character data element contains more than twelve characters (see *IEEE 488.2, 7.7.1.4*).]
- 148 Character data not allowed  
[A legal character data element was encountered where prohibited by the **device**.]
- 150 String data error  
[This error, as well as errors -151 through -159, are generated when parsing a string data element. This particular error message should be used if the **device** cannot detect a more specific error.]
- 151 Invalid string data  
[A string data element was expected, but was invalid for some reason (see *IEEE 488.2, 7.7.5.2*); for example, an END message was received before the terminal quote character.]
- 158 String data not allowed  
[A string data element was encountered but was not allowed by the **device** at this point in parsing.]
- 160 Block data error  
[This error, as well as errors -161 through -169, are generated when parsing a block data element. This particular error message should be used if the **device** cannot detect a more specific error.]
- 161 Invalid block data  
[A block data element was expected, but was invalid for some reason (see *IEEE 488.2, 7.7.6.2*); for example, an END message was received before the length was satisfied.]

- 168 Block data not allowed  
[A legal block data element was encountered but was not allowed by the **device** at this point in parsing.]
- 170 Expression error  
[This error, as well as errors -171 through -179, are generated when parsing an expression data element. This particular error message should be used if the **device** cannot detect a more specific error.]
- 171 Invalid expression  
[The expression data element was invalid (see *IEEE 488.2, 7.7.7.2*); for example, unmatched parentheses or an illegal character.]
- 178 Expression data not allowed  
[A legal expression data was encountered but was not allowed by the **device** at this point in parsing.]
- 180 Macro error  
[This error, as well as errors -181 through -189, are generated when defining a macro or executing a macro. This particular error message should be used if the **device** cannot detect a more specific error.]
- 181 Invalid outside macro definition  
[Indicates that a macro parameter placeholder (\$<number>) was encountered outside of a macro definition.]
- 183 Invalid inside macro definition  
[Indicates that the program message unit sequence, sent with a \*DDT or \*DMC command, is syntactically invalid (see *IEEE 488.2, 10.7.6.3*).]
- 184 Macro parameter error  
[Indicates that a command inside the macro definition had the wrong number or type of parameters.]

#### 21.8.10 Execution Error

An <error/event number> in the range [ -299 , -200 ] indicates that an error has been detected by the instrument's execution control block. The occurrence of any error in this class shall cause the execution error bit (bit 4) in the event status register (*IEEE 488.2, section 11.5.1*) to be set. One of the following events has occurred:

- A <PROGRAM DATA> element following a header was evaluated by the device as outside of its legal input range or is otherwise inconsistent with the device's capabilities.
- A valid program message could not be properly executed due to some device condition.

Execution errors shall be reported by the device after rounding and expression evaluation operations have taken place. Rounding a numeric data element, for example, shall not be reported as an execution error. Events that generate execution errors shall not generate

## 1999 SCPI Command Reference

Command Errors, device-specific errors, or Query Errors; see the other error definitions in this section.

Error Number	Error String [description/explanation/examples]
-200	Execution error [This is the generic syntax error for <b>devices</b> that cannot detect more specific errors. This code indicates only that an Execution Error as defined in <i>IEEE 488.2, 11.5.1.1.5</i> has occurred.]
-201	Invalid while in local [Indicates that a command is not executable while the <b>device</b> is in local due to a hard local control (see <i>IEEE 488.2, 5.6.1.5</i> ); for example, a <b>device</b> with a rotary switch receives a message which would change the switches state, but the <b>device</b> is in local so the message can not be executed.]
-202	Settings lost due to rtl [Indicates that a setting associated with a hard local control (see <i>IEEE 488.2, 5.6.1.5</i> ) was lost when the <b>device</b> changed to LOCS from REMS or to LWLS from RWLS.]
-203	Command protected [Indicates that a legal password-protected program command or query could not be executed because the command was disabled.]
-210	Trigger error
-211	Trigger ignored [Indicates that a GET, *TRG, or triggering signal was received and recognized by the device but was ignored because of device timing considerations; for example, the device was not ready to respond. Note: a DT0 device always ignores GET and treats *TRG as a Command Error.]
-212	Arm ignored [Indicates that an arming signal was received and recognized by the <b>device</b> but was ignored.]
-213	Init ignored [Indicates that a request for a measurement initiation was ignored as another measurement was already in progress.]
-214	Trigger deadlock [Indicates that the trigger source for the initiation of a measurement is set to GET and subsequent measurement query is received. The measurement cannot be started until a GET is received, but the GET would cause an INTERRUPTED error.]
-215	Arm deadlock [Indicates that the arm source for the initiation of a measurement is set to GET and subsequent measurement query is received. The measurement cannot be

started until a GET is received, but the GET would cause an INTERRUPTED error.]

- 220 Parameter error
  - [Indicates that a program data element related error occurred. This error message should be used when the **device** cannot detect the more specific errors described for errors -221 through -229.]
- 221 Settings conflict
  - [Indicates that a legal program data element was parsed but could not be executed due to the current device state (see *IEEE 488.2, 6.4.5.3* and *11.5.1.1.5.*)]
- 222 Data out of range
  - [Indicates that a legal program data element was parsed but could not be executed because the interpreted value was outside the legal range as defined by the **device** (see *IEEE 488.2, 11.5.1.1.5.*)]
- 223 Too much data
  - [Indicates that a legal program data element of block, expression, or string type was received that contained more data than the device could handle due to memory or related device-specific requirements.]
- 224 Illegal parameter value
  - [Used where exact value, from a list of possibles, was expected.]
- 225 Out of memory. [The device has insufficient memory to perform the requested operation.]
- 226 Lists not same length. [Attempted to use LIST structure having individual LIST's of unequal lengths.]
- 230 Data corrupt or stale
  - [Possibly invalid data; new reading started but not completed since last access.]
- 231 Data questionable
  - [Indicates that measurement accuracy is suspect.]
- 232 Invalid format
  - [Indicates that a legal program data element was parsed but could not be executed because the data format or structure is inappropriate. For example when loading memory tables or when sending a SYSTem:SET parameter from an unknown instrument.]
- 233 Invalid version
  - [Indicates that a legal program data element was parsed but could not be executed because the version of the data is incorrect to the device. This particular error should be used when file or block data formats are recognized by the instrument but cannot be executed for reasons of version incompatibility. For example, a not supported file version, a not supported instrument version]

## 1999 SCPI Command Reference

- 240 Hardware error  
[Indicates that a legal program command or query could not be executed because of a hardware problem in the **device**. Definition of what constitutes a hardware problem is completely device-specific. This error message should be used when the **device** cannot detect the more specific errors described for errors -241 through -249.]
- 241 Hardware missing  
[Indicates that a legal program command or query could not be executed because of missing **device** hardware; for example, an option was not installed. Definition of what constitutes missing hardware is completely device-specific.]
- 250 Mass storage error  
[Indicates that a mass storage error occurred. This error message should be used when the **device** cannot detect the more specific errors described for errors -251 through -259.]
- 251 Missing mass storage  
[Indicates that a legal program command or query could not be executed because of missing mass storage; for example, an option that was not installed. Definition of what constitutes missing mass storage is device-specific.]
- 252 Missing media  
[Indicates that a legal program command or query could not be executed because of a missing media; for example, no disk. The definition of what constitutes missing media is device-specific.]
- 253 Corrupt media  
[Indicates that a legal program command or query could not be executed because of corrupt media; for example, bad disk or wrong format. The definition of what constitutes corrupt media is device-specific.]
- 254 Media full  
[Indicates that a legal program command or query could not be executed because the media was full; for example, there is no room on the disk. The definition of what constitutes a full media is device-specific.]
- 255 Directory full  
[Indicates that a legal program command or query could not be executed because the media directory was full. The definition of what constitutes a full media directory is device-specific.]
- 256 File name not found  
[Indicates that a legal program command or query could not be executed because the file name on the device media was not found; for example, an attempt was made to read or copy a nonexistent file. The definition of what constitutes a file not being found is device-specific.]

## 1999 SCPI Command Reference

- 257 File name error
  - [Indicates that a legal program command or query could not be executed because the file name on the device media was in error; for example, an attempt was made to copy to a duplicate file name. The definition of what constitutes a file name error is device-specific.]
- 258 Media protected
  - [Indicates that a legal program command or query could not be executed because the media was protected; for example, the write-protect tab on a disk was present. The definition of what constitutes protected media is device-specific.]
- 260 Expression error
  - [Indicates that a expression program data element related error occurred. This error message should be used when the **device** cannot detect the more specific errors described for errors -261 through -269.]
- 261 Math error in expression
  - [Indicates that a syntactically legal expression program data element could not be executed due to a math error; for example, a divide-by-zero was attempted. The definition of math error is device-specific.]
- 270 Macro error
  - [Indicates that a macro-related execution error occurred. This error message should be used when the **device** cannot detect the more specific errors described for errors -271 through -279.]
- 271 Macro syntax error
  - [Indicates that that a syntactically legal macro program data sequence, according to *IEEE 488.2, 10.7.2*, could not be executed due to a syntax error within the macro definition (see *IEEE 488.2, 10.7.6.3.*)]
- 272 Macro execution error
  - [Indicates that a syntactically legal macro program data sequence could not be executed due to some error in the macro definition (see *IEEE 488.2, 10.7.6.3.*)]
- 273 Illegal macro label
  - [Indicates that the macro label defined in the \*DMC command was a legal string syntax, but could not be accepted by the **device** (see *IEEE 488.2, 10.7.3* and *10.7.6.2*); for example, the label was too long, the same as a common command header, or contained invalid header syntax.]
- 274 Macro parameter error
  - [Indicates that the macro definition improperly used a macro parameter placeholder (see *IEEE 488.2, 10.7.3.*)]
- 275 Macro definition too long
  - [Indicates that a syntactically legal macro program data sequence could not be executed because the string or block contents were too long for the device to handle (see *IEEE 488.2, 10.7.6.1.*)]

## 1999 SCPI Command Reference

- 276 Macro recursion error  
[Indicates that a syntactically legal macro program data sequence could not be executed because the device found it to be recursive (see *IEEE 488.2, 10.7.6.6.*.)]
- 277 Macro redefinition not allowed  
[Indicates that a syntactically legal macro label in the \*DMC command could not be executed because the macro label was already defined (see *IEEE 488.2, 10.7.6.4.*.)]
- 278 Macro header not found  
[Indicates that a syntactically legal macro label in the \*GMC? query could not be executed because the header was not previously defined.]
- 280 Program error  
[Indicates that a downloaded program-related execution error occurred. This error message should be used when the **device** cannot detect the more specific errors described for errors -281 through -289. A downloaded program is used to add algorithmic capability to a **device**. The syntax used in the program and the mechanism for downloading a program is device-specific.]
- 281 Cannot create program  
[Indicates that an attempt to create a program was unsuccessful. A reason for the failure might include not enough memory.]
- 282 Illegal program name  
[The name used to reference a program was invalid; for example, redefining an existing program, deleting a nonexistent program, or in general, referencing a nonexistent program.]
- 283 Illegal variable name  
[An attempt was made to reference a nonexistent variable in a program.]
- 284 Program currently running  
[Certain operations dealing with programs may be illegal while the program is running; for example, deleting a running program might not be possible.]
- 285 Program syntax error  
[Indicates that a syntax error appears in a downloaded program. The syntax used when parsing the downloaded program is device-specific.]
- 286 Program runtime error
- 290 Memory use error  
[Indicates that a user request has directly or indirectly caused an error related to memory or <data\_handle>s, this is not the same as “bad” memory.]
- 291 Out of memory
- 292 Referenced name does not exist
- 293 Referenced name already exists

- 294 Incompatible type  
 [Indicates that the type or structure of a memory item is inadequate]

### 21.8.11 Device-Specific Error

An <error/event number> in the range [ -399 , -300 ] or [ 1 , 32767 ] indicates that the instrument has detected an error which is not a command error, a query error, or an execution error; some device operations did not properly complete, possibly due to an abnormal hardware or firmware condition. These codes are also used for self-test response errors. The occurrence of any error in this class should cause the device-specific error bit (bit 3) in the event status register (*IEEE 488.2*, section 11.5.1) to be set. The meaning of positive error codes is device-dependent and may be enumerated or bit mapped; the <error message> string for positive error codes is not defined by SCPI and available to the device designer. Note that the string is not optional; if the designer does not wish to implement a string for a particular error, the null string should be sent (for example, 42,""). The occurrence of any error in this class should cause the device-specific error bit (bit 3) in the event status register (*IEEE 488.2*, section 11.5.1) to be set. Events that generate device-specific errors shall not generate command errors, execution errors, or query errors; see the other error definitions in this section.

Error Number	Error String [description/explanation/examples]
-300	Device-specific error [This is the generic device-dependent error for <b>devices</b> that cannot detect more specific errors. This code indicates only that a Device-Dependent Error as defined in <i>IEEE 488.2</i> , 11.5.1.1.6 has occurred.]
-310	System error [Indicates that some error, termed “system error” by the device, has occurred. This code is device-dependent.]
-311	Memory error [Indicates some physical fault in the <b>device</b> ’s memory, such as parity error.]
-312	PUD memory lost [Indicates that the protected user data saved by the *PUD command has been lost.]
-313	Calibration memory lost [Indicates that nonvolatile calibration data used by the *CAL? command has been lost.]
-314	Save/recall memory lost [Indicates that the nonvolatile data saved by the *SAV? command has been lost.]
-315	Configuration memory lost [Indicates that nonvolatile configuration data saved by the <b>device</b> has been lost. The meaning of this error is device-specific.]

## 1999 SCPI Command Reference

- 320 Storage fault
  - [Indicates that the firmware detected a fault when using data storage. This error is not an indication of physical damage or failure of any mass storage element.]
- 321 Out of memory
  - [An internal operation needed more memory than was available.]
- 330 Self-test failed
- 340 Calibration failed
- 350 Queue overflow
  - [A specific code entered into the queue in lieu of the code that caused the error. This code indicates that there is no room in the queue and an error occurred but was not recorded.]
- 360 Communication error
  - [This is the generic communication error for devices that cannot detect the more specific errors described for errors -361 through -363.]
- 361 Parity error in program message
  - [Parity bit not correct when data received for example, on a serial port.]
- 362 Framing error in program message
  - [A stop bit was not detected when data was received for example, on a serial port (for example, a baud rate mismatch).]
- 363 Input buffer overrun
  - [Software or hardware input buffer on serial port overflows with data caused by improper or nonexistent pacing.]
- 365 Time out error
  - [This is a generic device-dependent error.]

### 21.8.12 Query Error

An <error/event number> in the range [ -499 , -400 ] indicates that the output queue control of the instrument has detected a problem with the message exchange protocol described in *IEEE 488.2*, chapter 6. The occurrence of any error in this class shall cause the query error bit (bit 2) in the event status register (*IEEE 488.2*, section 11.5.1) to be set. These errors correspond to message exchange protocol errors described in *IEEE 488.2*, section 6.5. One of the following is true:

- An attempt is being made to read data from the output queue when no output is either present or pending;
- Data in the output queue has been lost.

Events that generate query errors shall not generate command errors, execution errors, or device-specific errors; see the other error definitions in this section.

Error Number	Error String [description/explanation/examples]
--------------	---

-400	Query error [This is the generic query error for <b>devices</b> that cannot detect more specific errors. This code indicates only that a Query Error as defined in <i>IEEE 488.2, 11.5.1.1.7</i> and 6.3 has occurred.]
-410	Query INTERRUPTED [Indicates that a condition causing an INTERRUPTED Query error occurred (see <i>IEEE 488.2, 6.3.2.3</i> ); for example, a query followed by DAB or GET before a response was completely sent.]
-420	Query UNTERMINATED [Indicates that a condition causing an UNTERMINATED Query error occurred (see <i>IEEE 488.2, 6.3.2.2</i> ); for example, the <b>device</b> was addressed to talk and an incomplete program message was received.]
-430	Query DEADLOCKED [Indicates that a condition causing an DEADLOCKED Query error occurred (see <i>IEEE 488.2, 6.3.1.7</i> ); for example, both input buffer and output buffer are full and the device cannot continue.]
-440	Query UNTERMINATED after indefinite response [Indicates that a query was received in the same program message after an query requesting an indefinite response was executed (see <i>IEEE 488.2, 6.5.7.5</i> ).]

### 21.8.13 Power On Event

An <error/event number> in the range [-599:-500] is used when the instrument wishes to report a 488.2 power on event to the event/error queue. This event occurs when the instrument detects an off to on transition in its power supply. This event also sets the power on bit, (bit 7) of the Standard Event Status Register. See IEEE 488.2, section 11.5.1.

Event Number	Event String [description/explanation/examples]
-500	Power on [The instrument has detected an off to on transition in its power supply.]

### 21.8.14 User Request Event

An <error/event number> in the range [-699:-600] is used when the instrument wishes to report a 488.2 user request event. This event occurs when the instrument detects the activation of a user request local control. This event also sets the user request bit (bit 6) of the Standard Event Status Register. See IEEE 488.2, section 11.5.1.

Event Number	Event String [description/explanation/examples]
-600	User request [The instrument has detected the activation of a user request local control]

**21.8.15 Request Control Event**

An <error/event number> in the range [-799:-700] is used when the instrument wishes to report a 488.2 request control event to the error/event queue. This event occurs when the instrument requests to become the active IEEE 488.1 controller-in-charge. This event also sets request control bit (bit 1) of the Standard Event Status Register. See IEEE 488.2, section 11.5.1.

**Event      Event String [description/explanation/examples]****Number**

-700      Request control

[The instrument requested to become the active IEEE 488.1 controller-in-charge]

**21.8.16 Operation Complete Event**

An <error/event number> in the range [-899:-800] is used when the instrument wishes to report a 488.2 operation complete event to the error/event queue. This event occurs when instrument's synchronization protocol, having been enabled by an \*OPC command, completes all selected pending operations. This protocol is described in IEEE 488.2, section 12.5.2. This event also sets the operation complete bit (bit 0) of the Standard Event Status Register. See IEEE 488.2, section 11.5.1. Note: \*OPC? does not set bit 0 nor does it enter any event in the error/event queue.

**Event      Event String [description/explanation/examples]****Number**

-800      Operation complete

[The instrument has completed all selected pending operations in accordance with the IEEE 488.2, 12.5.2 synchronization protocol]

**21.9****:HELP**

SYSTem:HELP

The HELP subsystem contains the instrument's help utilities that are provided to the programmer.

**21.9.1****:HEADers?**

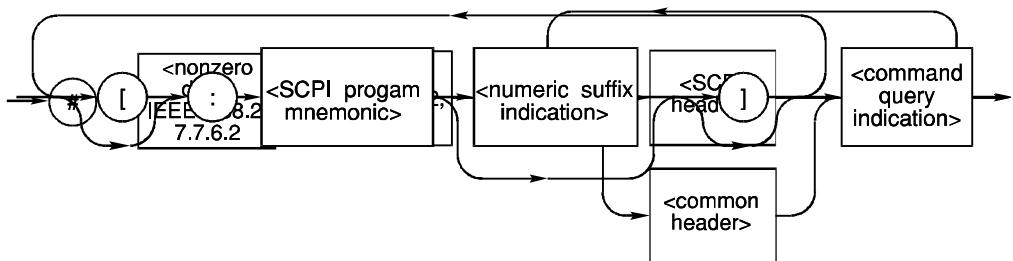
SYSTem:HELP:HEADers?

The HEADers? query shall return all SCPI commands and queries and IEEE 488.2 common commands and common queries implemented by the instrument. The response shall be a <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> element, see IEEE 488.2 section 8.7.9. The full path for every command and query shall be returned separated by linefeeds. The syntax of the response is defined as:

The <nonzero digit> and sequence of <digit> follow the rules in IEEE 488.2, Section 8.7.9.

An <SCPI header> is defined as:

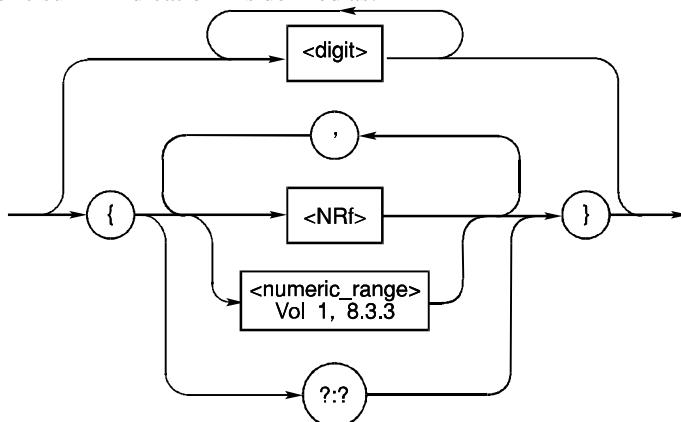
It shall contain all the nodes from the root. The <SCPI program mnemonic> contains the node in standard SCPI format. The shortform shall use uppercase characters while the additional characters for the longform shall be in lowercase characters. Default nodes shall be surrounded by square brackets ([]).



The entire <SCPI header> shall have an indication of the header type, using a <command query indication>.

A node which has a numeric suffix, described in Volume 1, section 6.2.5, is indicated by including a <numeric suffix indication>.

A <numeric suffix indication> is defined as:



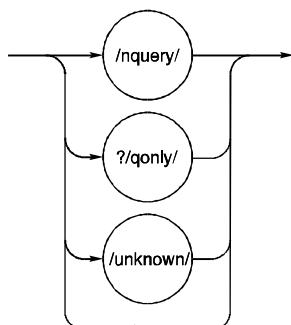
where:

nothing	indicates	no suffix is allowed
<numeric_list>	indicates	list of suffixes allowed
<digit>(s)	indicates	this specific suffix is allowed
{?:?}	indicates	range of suffixes is unknown, or can't tell if suffix is allowed

Numbers between the curly braces shall indicate valid values for the numeric suffix. For example, {1:4} means the values one through four may be appended to that node. If the instrument cannot determine the legal values for the numeric suffix, it may use the special form, {?:?}, to show that a numeric suffix may be allowed even though the range is unknown. The use of a <numeric\_list> is especially useful with a large number of suffixes. For headers which accept only a small number of numeric suffixes, specific values are preferred.

A <command query indication> is defined as:

## 1999 SCPI Command Reference



If the instrument is unable to determine if the header is an event, query only, or a command/query pair, it shall append /unknown/ to the header. Otherwise, a normal command/query pair shall have nothing additional following the path, a query only header shall have ?/qonly/ appended, and a command which is an event with no query form shall have /nquery/ appended.

A <common header> is defined as:



A <common response header> contains only upper case letters.

Headers may appear in any order.

The response shall not include any macro labels. The \*LMC? common query is used to return macro labels.

For example, an instrument which implemented the required commands listed in Syntax & Style, Volume 1 section 4.2.1, this query, and the required IEEE 488.2 common commands and queries might return:

```
#3425
:SYSTem:ERRor?/qonly/
:SYSTem:HELP:HEADers?/qonly/
:SYSTem:VERSion?/qonly/
:STATus:OPERation[:EVENT]?/qonly/
:STATus:OPERation:CONDition?/qonly/
:STATus:OPERation:ENABLE
:STATus:QUESTIONable[:EVENT]?/qonly/
:STATus:QUESTIONable:CONDition?/qonly/
:STATus:QUESTIONable:ENABLE
:STATus:PRESet/nquery/
*IDN?/qonly/
*RST/nquery/
*TST?/qonly/
*OPC/nquery/
*OPC?/qonly/
```

\*WAI/nquery/  
 \*CLS/nquery/  
 \*ESE  
 \*ESR?/qonly/  
 \*SRE  
 \*STB?/qonly/

\*RST shall have no direct effect on the response to this query, but changing the instrument state may change the list of valid commands and queries.

### 21.9.2 :SYNTax? <command\_header>

SYSTem:HELP:SYNTax?

The SYNTax? query causes the device to return a string containing the syntax specification of the command associated with the <command\_header>. The <command\_header> parameter is a string containing any implemented command or query header that starts from the root. Default nodes and suffixes may appear in the <command\_header> parameter.

Numeric suffixes may be explicitly specified in the <command\_header>. Only the header and parameter syntax of the specified <command\_header> is returned. If no numeric suffix is specified, default 1 is assumed. In that case, devices that implement a numeric suffix shall show the suffix 1 as an optional element in their response.

The part of the returned string that reflects the syntax of the parameter part of the command is separated from the header syntax part by one or more ASCII space characters (decimal 32). The parameter part shall contain all required and allowed parameters. Parameter notation shall conform to notations of the Command Tables, being used throughout Volume 2 and specified in chapter 5.1 of Volume 1. Optionality is reflected by placing the optional items in between square brackets. Optional nodes are reflected in the same way.

The Long Form shall be used to represent mnemonics in the response to the :SYNTax? query. The letter case notation habit, as specified in chapter 5.1 of “Volume 1: Syntax and Style,” shall be used for mnemonics to differentiate between Long- and Short form.

Any <command\_header> that is not a valid command header being recognized by the device, shall cause the device to return a null string (""). E.g. if the <command\_header> contains only a part of the header, contains an illegal numeric suffix, etc.

For example, if the command tree were

```
[ :ALPHA<n> ]
  :BETA
    [ :GAMMA ]
      :DELTa<n> ON|OFF|TRIGger
```

where the <n> indicates a range of suffix values is accepted for ALPHA and DELTa only.

Then SYSTem:HELP:SYNTax? “:BETA:GAMMA:DELTa” will return

```
“[ALPHA[1]]:BETA[:GAMMA]:DELTa[1] ON|OFF|TRIGger”
```

SYNTax? is a query only and therefore does not have an associated \*RST value.

**21.10 :KEY <numeric\_value>**

SYSTem:KEY

The KEY node is a limited implementation of the naming conventions subsystem described in “Volume 1: Syntax and Style”. The limitation is that keys can only be referred to numerically, and the numbers allowed by a particular instrument (key codes) are fixed.

The KEY? query returns the key code for the last key pressed. Instruments may implement a queue of keys (possibly of length 1) and return a value of -1 to indicate an empty queue. The queue is emptied at \*RST.

If the KEY command is sent with a parameter, that simulates the pressing of that key. The keycode determined by the parameter is entirely device-dependent but must match the corresponding parameter returned for the query form. The KEY command may or may not put an entry into the KEY? queue.

The queue of keys is cleared by \*RST.

**21.10.1 :CATalog**

SYSTem:KEY:CATalog

This query returns a list of the defined keys in the subsystem. The response is a comma separated list of NR1 elements corresponding to each key that has been defined.

**21.10.2 :DEFine <numeric\_value>,<block>[,<string>]**

SYSTem:KEY:DEFine

The DEFine command shall redefine a key on the instrument’s keyboard so that pressing the key will execute a particular function. The first parameter is the key code for the key to be remapped, and the second parameter contains the command sequence to be executed. The optional third parameter is a key label for the redefined key.

The particular keycodes are device-dependent.

The softkey label parameter is optional, but it must be accepted by the implementation. If softkeys are not implemented, the third parameter should be ignored.

The command sequence is a block containing any sequence of SCPI commands. Implementations have the option of limiting this block to contain a single command. It is device-dependent when the block has its syntax checked.

The query form of the command returns a list of all redefined keys in the form: <NR1>,<block>,<string>. If the key is unlabeled, the third parameter shall be an empty string.

Key definitions are not affected by \*RST.

**21.10.3 :DELete <numeric\_value>**

SYSTem:KEY:DELete

The DELete command removes the definition of the key referred to by <numeric\_value> set by a previous DEFine command. The key may return to its default definition, if any.

**21.11 :KLOCK <Boolean>**

SYSTem:KLOCK

This command locks the local controls of an instrument. This includes any front panel, keyboard, or other local interfaces. The SYSTem:KLOCK setting does not affect the 488.1 Remote/Local state, if implemented.

This value cannot be reset to OFF, unless SYSTem:SECurity:STATe is OFF.

If SYSTem:SECurity:STATe is OFF, the KLOCK value is set to OFF at \*RST. If SYSTem:SECurity:STATe is ON, \*RST has no effect.

**21.12 :LANGuage <string>**

SYSTem:LANGuage

The receipt of this command shall cause the instrument to perform a language switch to another language. The contents of the string are device dependent, however, the following contents are suggested:

- COMPatibility — The language is a backwards compatible set for the product line.
- CIIL — Selects the CIIL language of the instrument.

All devices which implement alternate languages shall also implement this command. An instrument which is currently executing SCPI will not leave SCPI to enter an alternate language under any other conditions, except possibly power-on. There is no query corresponding to this command.

**21.13 :LFREquency<numeric\_value>**

SYSTem:LFREquency

The numeric value is used as the reference frequency. If the instrument is capable of using only discrete reference values (e.g. 50 or 60 Hz), it will use the specified value to select the nearest valid setting.

\*RST does not affect this setting.

**21.13.1 :AUTO <Boolean> | ONCE**

SYSTem:LFREquency:AUTO

If AUTO ONCE is specified, the instrument will measure the line frequency internally and use the result (or the nearest valid setting, if necessary) as its reference. After the measurement is complete, this setting reverts to OFF. If AUTO ON is specified, the instrument will continuously monitor the line frequency (through repetitive measurements, phase locking, etc.).

Behavior at power-on is device-dependent; an instrument capable of measuring the line frequency may choose to do so when power is applied. Other options are (1) to preserve the value last specified with LFREquency, (2) to choose a default setting, or (3) to use hard controls to determine the setting.

\*RST does not affect this setting.

**21.14 :LOCK**

SYSTem:LOCK

This subsystem controls the availability of an instrument. It allows the communication channel that these SCPI commands arrived on (session) to attempt to attain exclusive use of this instrument by attaining a lock.

The behavior of the device when receiving SCPI commands from a session that does not have the lock is device dependent.

On \*RST the ownership of the lock is device dependent.

**21.14.1 :OWNer?**

:SYSTem:LOCK:OWNer?

This query returns what session currently has the lock on this device. If no session has a lock then “NONE” shall be returned.

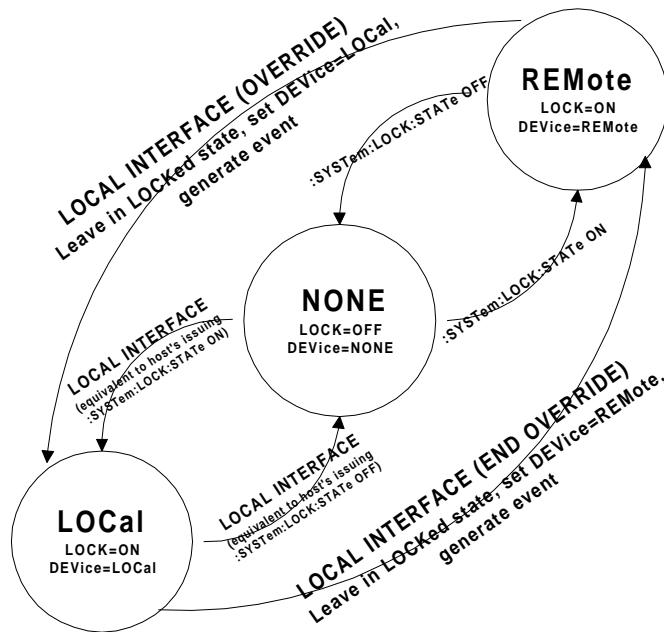


Figure 21.2 System Lock Format

21

**21.14.2 :RELEASE**

:SYSTem:LOCK:RELEASE

This event releases the lock if owned by this session. If this session does not have the lock, this command has no effect. This command has no query form.

**21.14.3 :REQUEST?**

:SYSTem:LOCK:REQUEST?

This event shall only be implemented as a query. It attempts to attain the lock on this device and returns 1 if successful and 0 if it fails.

**21.15 :PASWord**

SYSTem:PASWord

This subsystem provides for the enabling, disabling, and changing of passwords to control access to selected SCPI commands and queries. The instrument's documentation shall indicate which commands are under the control of SYSTem:PASWord.

Password protection shall be locally defeatable. The device shall have a local means to:

- reset the password to the default value, "DEFAULT".
- enable all protected commands.

The means of protection for this local resetting and enabling is entirely device dependent.

The device shall report error -203, (command protected), if a password-protected command is received that is currently disabled.

Neither the current password nor command protection state shall be affected by \*RST.

**21.15.1 :CDISable <password>**

SYSTem:PASWord:CDISable

Sending :CDISable (Command DISable) disables protected commands. The <password> data is of type string and the password is case sensitive. There is no query form.

Sending an invalid password generates error -221, (settings conflict).

**21.15.2 [:CENable] <password>**

SYSTem:PASWord:CENable

Sending :CENable (Command ENable) enables protected commands to function. The <password> data is of type string and the password is case sensitive. There is no query form.

Sending an invalid password generates error -221, (settings conflict).

**21.15.2.1 :STATe?**

SYSTem:PASWord:CENable:STATE?

STATE? returns a 0 if password protected commands are disabled and a 1 if they are enabled. There is no command form.

**21.15.3 :NEW <current password>,<new password>**

SYSTem:PASWord:NEW

This command changes the password from the default or current value to the new value. Both parameters are strings. The <current password> must be correct for the command to execute. Password strings are case sensitive. There is no query form.

Sending an invalid current password generates error -221, (settings conflict).

**21.16 :PRESet**

SYSTem:PRESet

Sets the device to the same state as the front-panel reset key. This command is similar to the \*RST command, but differs in that it duplicates the action of the front panel reset key.

## 1999 SCPI Command Reference

Normally this state will be optimized for front panel operation of the device instead of bus operation.

Other PRESet states may be accessed with this instruction by using a numeric suffix. For instance, PRESet2 could provide a high speed configuration.

The state of the device after PRESet is not defined by SCPI. There is no query corresponding to this command.

### 21.17 :SECurity

SYSTem:SECurity

Controls the security of the instrument.

#### 21.17.1 :IMMEDIATE

SYSTem:SECurity:IMMEDIATE

This command is an event which immediately destroys all measurement data and instrument settings. Current settings are initialized to their \*RST values. All macros, user variables, and \*SAV/\*RCL registers are initialized. Critical calibration constants are retained.

This command can be under control of the SYSTem:PASSword subsystem. The device shall report error -203, (command protected), if SECurity:IMMEDIATE is received while password protected.

This command is an event, and therefore has no \*RST value associated with it.

#### 21.17.2 [:STATe] <Boolean>

SYSTem:SECurity:STATe

STATe controls the security mode of the instrument. For example, when in secure mode, any display annunciators which have been disabled cannot be reenabled without destroying certain stored information.

The value is not affected by \*RST or \*RCL.

When this value changes from ON to OFF, everything except possibly some CALibration data must be initialized or destroyed. For example, the following shall be initialized:

- Instrument state (perform \*RST).
- All macros (perform \*PMC).
- All user variables.
- All state registers (perform \*SAV into all registers after \*RST).

### 21.18 :SET <block data>

SYSTem:SET

Sets the instrument to the complete or partial state determined by the contents of the parameter. The form and contents of the block data is device-dependent.

This command shall be required for instruments that return a block of binary settings in response to the \*LRN? query.

Instruments that implement the \*LRN? query and return a block of binary settings as response to \*LRN?, shall implement also the SYSTem:SET command. In this case the response format of \*LRN? shall contain SYST:SET <block data>.

Note that IEEE 488.2 requires the response to \*LRN? to be a legal program message. Since the SYSTem:SET command is the SCPI command which programs the instrument settings by binary block data, this must be the response to the \*LRN? query when it returns this data format.

The implementation of the SYSTem:SET command is optional if the \*LRN? query response are the SCPI commands that are normally used for programming the instruments.

The query form of this command will return a definite block data element which contains the instrument's current state. The SET command may be implemented without the associated query.

## 21.19 :TIME <hour>,<minute>,<second>

SYSTem:TIME

Any instrument which has an internal clock shall implement this command to set the clock. The three parameters for this command are:

- <hour>: Must be <numeric\_value>. It is always rounded to the nearest integer. Its range is 0 to 23 inclusive. All instruments implementing the TIME function shall report and accept hour information in 24 hour format.
- <minute>: Must be <numeric\_value>. It is always rounded to the nearest integer. Its range is 0 to 59 inclusive.
- <second>: Must be <numeric\_value>. It is rounded to the resolution of the clock. Its range is 0 to 60. 60 is allowed since rounding may cause a number greater than 59.5 to be rounded to 60. When this element is rounded to 60 it shall be set to 0 and the minute value incremented. Any other carries shall be rippled through the date.

The query response message shall consist of three fields separated by commas:

<hour>,<minute>,<second>

The three fields will contain the following information:

- <hour>: NR1 format ranging from 0 to 23.
- <minute>: NR1 format ranging from 0 to 59.
- <second>: NR1 or NR2 format. If the resolution of the clock is no better than seconds, an NR1 format shall be used. If the resolution is better than seconds, an NR2 format shall be used. Range is 0 to 59.999999...

This setting shall not be affected by a \*RST command.

### 21.19.1 :TMR

SYSTem:TIME:TMR

This subsystem contains commands that control an internal timer/counter.

In some instruments the value of this timer may be used to timestamp internal readings.

#### 21.19.1.1 :COUNt <numeric\_value>

SYSTem:TIME:TMR:COUNt

COUNt sets or queries the current value of the timer.

Sending this command shall not affect TMR:STATE. When the TMR:STATE is ON the count will increase.

This command sets or queries the current count for the timer in seconds.

The default value of this parameter is 0.

At \*RST this value be set to 0.

#### 21.19.1.2 [:STATe] <Boolean>

SYSTem:TIME:TMR:STATe

This node sets or queries the TMR:STATE. If the STATE is ON the TMR is running.

At \*RST the TMR:STATE is OFF

### 21.20 :TZONe <hour> [,<minute>]

SYSTem:TZONe

An instrument which has an internal clock and can keep track of the time zone shall implement this command to set the time zone.

The fields have the following meanings:

<hour>: Must be a <numeric\_value>. The value shall be rounded to the nearest integer. Range -12 to +12.

<minute>: Must be a <numeric\_value>. The value shall be rounded to the nearest integer. Range -59 to 59.

When each field is subtracted from the value of the TIME command, the result is the correct universal coordinated time (also known as UCT, Zulu, Greenwich Mean Time).

The default value of <minute> is zero.

This shall not be changed by the receipt of a \*RST.

### 21.21 :VERSion?

SYSTem:VERSion?

This query returns an <NR2> formatted numeric value corresponding to the SCPI version number for which the instrument complies. The response shall have the form *YYYY.V* where the Ys represent the year-version (i.e. 1990) and the V represents an approved revision number for that year. If no approved revisions are claimed, then this extension shall be 0.

## **1999 SCPI Command Reference**

An instrument known to claim SCPI compatibility that does not respond to the VERSion query shall be assumed to conform to version 1990.0.

This query shall be implemented by all instruments claiming conformance to a SCPI version greater than 1990.0.

VERSion? is a query only and therefore does not have an associated \*RST state.

## 1999 SCPI Command Reference

## **22 TEST Subsystem**

The TEST subsystem provides a section to extend standard instrument self-test procedures beyond the *IEEE 488.2 \*TST* command.

## 1999 SCPI Command Reference

## 23 TRACe | DATA

A TRACe or a DATA area is a named entity stored in instrument memory. TRACe | DATA areas may also be used to store other types of data, such as constant arrays for use in trace arithmetic or corrections, or displayed waveforms. Alternatively, TRACe | DATA areas may be used for equivalent scalar (single point) purposes.

Certain TRACe format and capabilities are the focus of other emerging standards. To prevent diluting the effect of SCPI or any other standards related to the manipulation of TRACes, SCPI has left such capability undefined at this time with the view to adopting an industry standard when one becomes available. The undefined capabilities include the following:

- Trace math operations for specifying arithmetic relationships between traces, including the use of user-defined constants and functions.
- Trace activation and routing for specifying the destination for measurement results.

KEYWORD	PARAMETER FORM	NOTES
TRACe DATA		1991
:CATalog?		[query only]
:COPY	<trace_name>, (<trace_name> <data_handle>)	[event; no query] 1991,2
[:DATA]	<trace_name>,(<block> <dif_expression>  (<numeric_value>{,<numeric_value>}))	1993
:LINE	<trace_name>,<numeric_value>,<numeric_value>, <numeric_value>,<numeric_value>	1992
:PREamble?	<trace_name>	[query only] 1994
:VALue	<trace_name>,<numeric_value>,<numeric_value>	1992
:DEFine	<trace_name>[,(<numeric_value> <trace_name>)]	
:DELetE		
[:NAME]	<trace_name>	[event; no query]
:ALL		[event; no query]
:FEED	<trace_name>,(<data_handle> NONE)	1991
:CONTrol	<trace_name>, (ALWays OCONDition NEXT NEVer)	1991
:OCONDition	<trace_name>,<condition_expr>	1991
:FREE?		[query only]
:POINTs	<trace_name>[,<numeric_value>]	1993
:AUTO	<trace_name>,(<Boolean> ONCE)	[event; no query]

Many TRACe subsystem commands are of the following general form:

TRACe :COMMAND <destination trace\_name> ,<source trace data>

The <trace\_name> is <CHARACTER PROGRAM DATA> (IEEE 488.2, section 7.7.1). There is no difference between uppercase and lowercase letters used in <trace\_name>. There

is no short form abbreviation for <trace\_name>. The destination trace is always the first parameter in the list. The <source trace data> parameter may be an existing <trace\_name>, or a block, or expression which is evaluated as trace data.

\*RST has no effect on this subsystem.

23.1

### :CATalog?

TRACe:CATalog?

The CATalog? query returns a comma-separated list of strings which contains the names of all traces. If there are no <trace\_name> defined, a single empty string is returned. Some instruments may have one or more intrinsic, permanent <trace\_name>. These traces must be listed along with any user-named traces.

23.2

### :COPY <trace\_name>, ( <trace\_name> | <data\_handle> )

TRACe:COPY

The COPY command set the data values in the destination trace, from internal data stores in the instrument. There is no query form of this command, it is an event only.

If the data source is a <trace name>, the data is copied into the destination trace. For example, COPY REF,TRACE1 will copy the data currently in TRACE1 into the trace named REF.

If the data source is a <data\_handle>, the data shall be copied directly from the data pool specified. For example, COPY REF, "CALCulate1" will copy the data emitted by the CALCulate1 block. When the data is copied depends on the type of memory associated with the data flow. If the memory is transient, as described in section 2.7.5, then the data shall be copied when data becomes available, for example when a new acquisition takes place. If the memory is static the data shall be copied immediately, as there is memory that always has some value in it associated with the specified <data\_handle>, typically the result from an earlier acquisition.

23.3

### [:DATA]<trace\_name>,(<block>|<dif\_expression>| <numeric\_value>{,<numeric\_value>}))

TRACe:DATA

The DATA command sets the data values in the destination trace, from information provided by the controller. If the data source is a block of data, then the block is directly loaded into the destination trace. In this case the data source is block data, the data format is determined by the FORMAT Subsystem. Note that the data source may also be a Data Interchange Format expression.

If the data source is a single <numeric\_value>, every element of the destination trace is set to the constant value represented by the evaluation of <numeric value>. For example, DATA REF,0.22361 sets all points in the REF trace to 0.22361. If a <numeric\_value> is sent for each of the points in the allocated <trace\_name>, then the <numeric\_value>s are mapped into the <trace\_name>. If an insufficient number of <numeric\_value>s are sent, that is less than the allocated points, then the result is device dependent. If too many <numeric\_value>s are sent the device shall generate error -223 (too much data), and the result shall be device dependent.

The DATA? query <trace name> returns the data values for <trace name>, according to the format determined by commands in the FORMat subsystem.

Note: All parentheses in the syntax for this command are used for grouping purposes only.

### 23.3.1 :LINE <trace\_name>,<numeric\_value>,<numeric\_value>,<numeric\_value>,<numeric\_value>

TRACe:DATA:LINE

The LINE command is an event which defines a line segment (a series of points) within the boundaries of the trace whose name is specified by <trace\_name>. The first <numeric\_value> determines the index of the starting point in the trace. The second <numeric\_value> is the value of the trace at the starting point. The third and fourth <numeric\_value> parameters set the end of the line segment index and value. A line segment is drawn connecting the start point and end point, inclusive. This command has no query form. The index of the first point in a trace is 1.

### 23.3.2 :PREamble? <trace\_name>

TRACe:DATA:PREamble?

PREamble? returns the preamble information supporting the DATA(CURVe(VALues)) but omits the following data:

- DIF blocks of waveform, measurement, and delta.
- DIF curve block keywords VALUes and CSUM and their parameters.

The DIF difid block must contain the keyword SCOPe with the parameter of PREamble.

While the PREamble? query is designed to efficiently transmit data relating directly to the curve, the various notes and identification parameters may be sent if absolutely required.

The <trace\_name> parameter functions the same as it does in TRACe:DATA?.

### 23.3.3 :VALue <trace\_name>,<numeric\_value>,<numeric\_value>

TRACe:DATA:VALue

The VALue command sets or queries the value of an individual point in a trace. The first <numeric\_value> is the index of the addressed point in the trace defined by <trace\_name>. The second <numeric\_value> is the value to be set for the point. The index of the first point in a trace is 1.

The VALue? <trace\_name>,<numeric\_value> query returns the data value of the trace <trace\_name> at the <numeric\_value> indexed point.

### 23.4 :DEFine <trace\_name>[,(<numeric\_value>|<trace\_name>)]

TRACe:DEFine

The DEFine command allocates and initializes a new trace. The first parameter is <CHARACTER DATA>, and it specifies the new <trace\_name>. If the second parameter is a <numeric\_value>, a new trace is allocated with the specified number of data elements. The trace shall be initialized to instrument-specified default values. If the second parameter is the name of an existing trace, the new trace becomes a copy of the existing trace in all respects, including size and contents. The data in the existing trace is not affected. When the second

## 1999 SCPI Command Reference

parameter is omitted, the new trace shall have an instrument-specified default size and initial value.

Limits on the number of named traces are instrument-specific.

### 23.5 :DELETED

TRACe:DELETED

The DELETED commands allow the user to dissociate <trace\_name> from trace memory. Instrument intrinsic <trace\_name> cannot be deleted. There is no query form for the DELETED commands.

#### 23.5.1 [:NAME] <trace\_name>

TRACe:DELETED:NAME

The NAME command dissociates a user-created <trace\_name> from its trace memory. The NAME node is optional in a program message. That is, an instrument is required to accept both forms of this command. If the instrument supports dynamic trace memory allocation, then the memory allocated for <trace\_name> is freed.

#### 23.5.2 :ALL

TRACe:DELETED:ALL

The ALL command dissociates all user-created <trace\_name> from their trace memory units. If the instrument supports dynamic trace memory allocation, then all memory allocated for user-created traces is freed.

### 23.6 :FEED <trace\_name>, ( <data\_handle> | NONE )

TRACe:FEED

Sets or queries which data flow is fed into the specified TRACe|DATA memory. The first parameter specifies the name of the TRACe|DATA memory. The second parameter (used only in the command form) selects the data flow to be fed into the memory.

A <data\_handle> is the name of a point in the data flow described by the Instrument Model (found at the beginning of this volume). By setting the <data\_handle> to a null string ("’"), no data shall be fed into the specified TRACe|DATA memory. When a TRACe|DATA memory is created with the DEFine command, the FEED <data\_handle> for that memory shall be set to null ("’").

The query form of this command requires that a valid <trace\_name> shall be specified. If the TRACe|DATA memory specified in <trace\_name> does not exist then an error -224, "Illegal parameter value" shall be generated. The query shall return a string that represents the <data\_handle> selected. If no feed is selected, then the query shall return a null string ("’").

At \*RST, FEED shall be set to NONE.

#### 23.6.1 :CONTrol <trace\_name>, ALways | OCONdition | NEXT | NEVer

TRACe:FEED:CONTrol

Sets or queries how often the specified TRACe | DATA area accepts new data. This control has no effect if the FEED <data\_handle> is set to null ("’"). The first parameter specifies the TRACe | DATA area, the second parameter selects how often new data shall be accepted.

By selecting ALWays, whenever new data is available it shall be fed into the specified TRACe | DATA area. On CONdition defines that data shall be gated by the OCONDition command. NEXT is a one-shot feed, it always waits for new data, such as an new acquistion, and ignores any existing data. Once NEXT has completed its feed of data, CONTrol reverts to NEVer. NEVER shall cause no data to be fed.

At \*RST, CONTrol is set to a device dependent value.

### 23.6.2 :OCONDition <trace\_name>, <condition\_expr>

TRACe:FEED:OCONDition

The On CONDITION command sets or queries the condition used to gate data flow into the specified TRACe | DATA area. The first parameter specifies the TRACe | DATA area. The second parameter is EXPRESSION PROGRAM DATA, which defines the condition, which when evaluated to be True shall allow data to be fed through. The production

<condition\_expr> is of the general form:

(' <operand\_str> <equiv\_op> <operand\_str> ') or  
(' <event\_handle> ')

The productions <operand\_str> and <equiv\_op>, the operand string and the equivalence operator respectively, shall be defined later.

### 23.7 :FREE?

TRACe:FREE?

The FREE? query returns the amount of user memory space available for traces, in the following form:

<bytes available>, <bytes in use>

where the returned data for both shall be in <NR1> format.

### 23.8 :POINts <trace\_name>[,<numeric\_value>]

TRACe:POINts

The POINts command sets the number of measurement data points available in the specified trace memory. The first parameter specifies the name of the trace to resize. The optional second parameter specifies the number of data points which the <trace\_name> must accommodate. If this parameter is omitted, an instrument specific default value will be used. If more than one <numeric\_value> is associated with each trace data point (as would be the case with a trace consisting of real and imaginary pairs), then the second parameter specifies the number of N-tuples in the trace.

The query POINts? <trace\_name> returns the number of measurement data points in the specified trace. Note that each data point may have several numeric data items associated with it.

### 23.8.1 :AUTO <trace\_name>,(<Boolean>|ONCE)

TRACe:POINts:AUTO

The AUTO command turns trace autosizing ON/OFF. When it is enabled, <trace\_name> will automatically be resize as necessary to accommodate any new data block assigned to it. In this mode of operation, changes in measurement setup will not require adjustments to the

## 1999 SCPI Command Reference

size of defined traces. If POINts:AUTO <trace\_name>, ONCE is specified, and the trace sizing occurs once (on the next measurement). Limits on trace sizes are instrument-specific.

## 24 TRIGger Subsystem

The trigger subsystem is used to synchronize device action(s) with events. A device action might be the acquisition of a measurement or the application of a stimulus. The trigger subsystem consists of the expanded capability model which is capable of describing very complex device trigger systems. It also makes provision, through the ARM-TRIGger model, for simple descriptions of less complicated trigger systems. These two models are consistent and compatible with each other. The ARM-TRIGger model represents a subset of the capability available in the expanded capability model.

The figures in this section that represent the trigger model adhere to the following nomenclature. A box identifies a state of a transition diagram and is referred to as a layer. A sequence is a set of vertically connected layers. A solid line defines flow of control between states. A dashed line defines how an event is propagated to other parts of the trigger model and the instrument. Events generated by control flowing from one part of the trigger model to another part are called “sequence events.”

### 24.1 ARM-TRIGger Model

The ARM-TRIGger model represents a level of capability that is often found in a device and is shown in figure 24-1. The ARM-TRIGger model shows two independent levels of event detection, one in each of the ARM and the TRIGger layers. For a given device the model

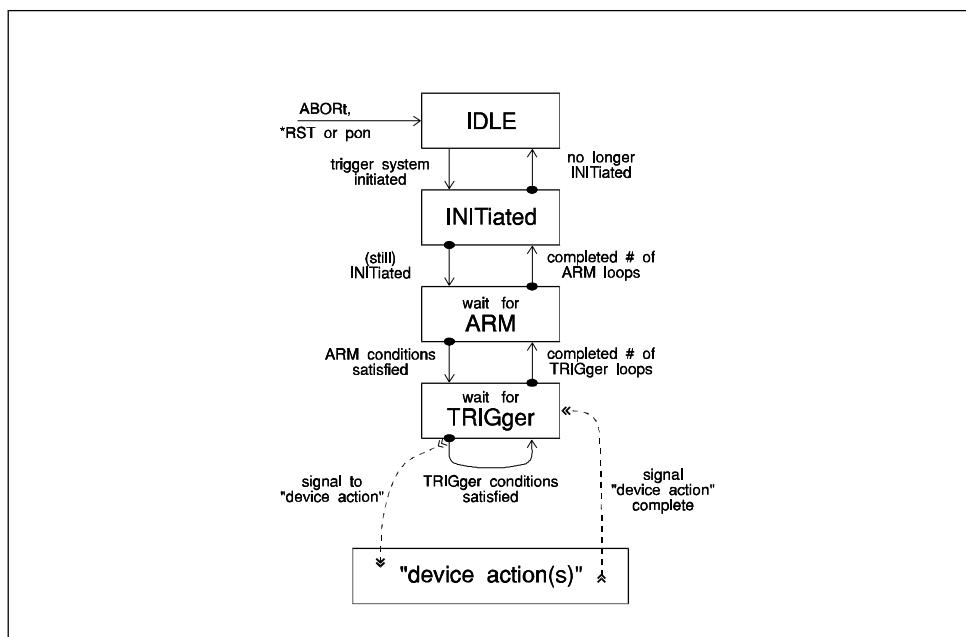


Figure 24-1 ARM-TRIGger Model

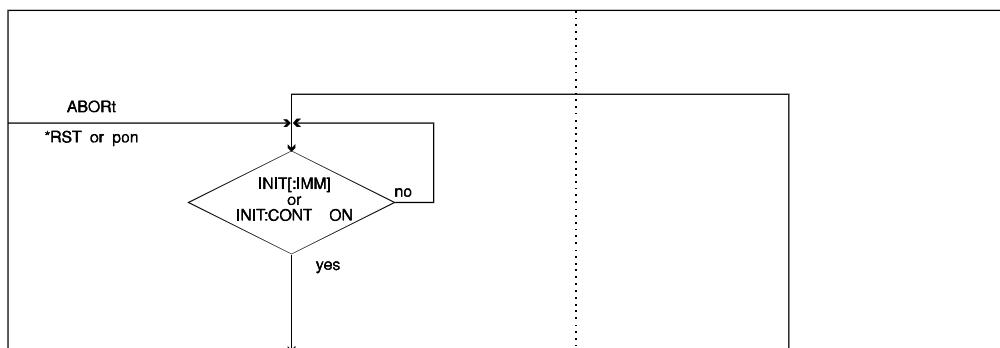
may be simplified further by the omission of the ARM layer and any inappropriate commands for that device.

## 24.2 Model Layers

The following figures detail each of the various types of layers that may exist in a sequence. Each layer is shown with a dotted line, which is used to divide the upward and downward traverses through the layer. The different traverses are referenced in describing the operation of each layer.

### 24.2.1 IDLE State

On receipt of either \*RST or ABORT, the trigger subsystem shall enter the IDLE state. Receiving the IEEE 488.2 **dcas** message may also cause a transition to IDLE. Devices which cannot process commands when not in IDLE must enter IDLE when **dcas** is received to meet the IEEE 488.2 requirements for device clear. The downward traverse from IDLE state, “trigger system initiated,” is affected by either the INITiate[:IMMEDIATE] command or by setting INITiate:CONTinuous to ON.



**Figure 24-2 IDLE State**

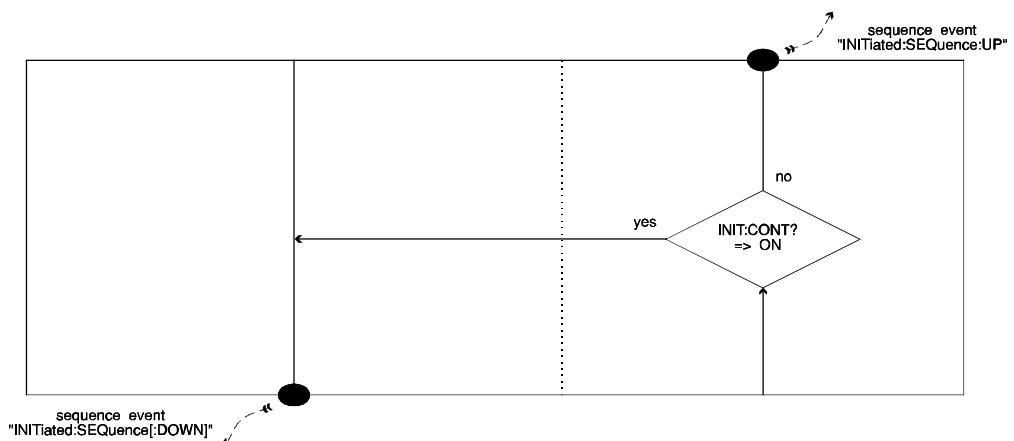
If the INITiate commands are implemented as overlapped, exiting the IDLE state shall cause the pending operation flag (as defined by IEEE 488.2) associated with the initiated action to be set true. Entering the IDLE state shall cause the pending operation flag to be set false.

### 24.2.2 Initiated

Once the trigger sequence is initiated from the IDLE state, control passes through the INITiated state, shown in Fig 24-3, immediately making the downward traverse to the highest ARM:LAYER state. The upward traverse is dependent on the setting of CONTinuous. If CONTinuous is set to OFF then the upward transition to the IDLE state shall be made. Otherwise, the downward traverse to the highest ARM:LAYER state shall be made, so that the trigger sequence remains initiated. If the sequence does not contain any ARM layers, the downward transition goes to a TRIGger state.

The INITiated state is necessary to avoid having the pending operation flag change state every time the entire trigger sequence is completed when CONTinuous is ON. On exit from the INITiated state, in both directions, a “sequence event” is generated. These events may be used to start actions in the trigger subsystem or actions in other parts of the instrument.

Fig 24-3 shows the names of these events. An instrument shall implement the initiated state even if it implements the INITiate commands as sequential.



**Figure 24-3 INITiated State**

#### 24.2.3 Event Detection Layer

The ARM and TRIGger layers are both event detection layers with the TRIGger layer being subservient to the ARM layer. Each layer provides one level of event detection.

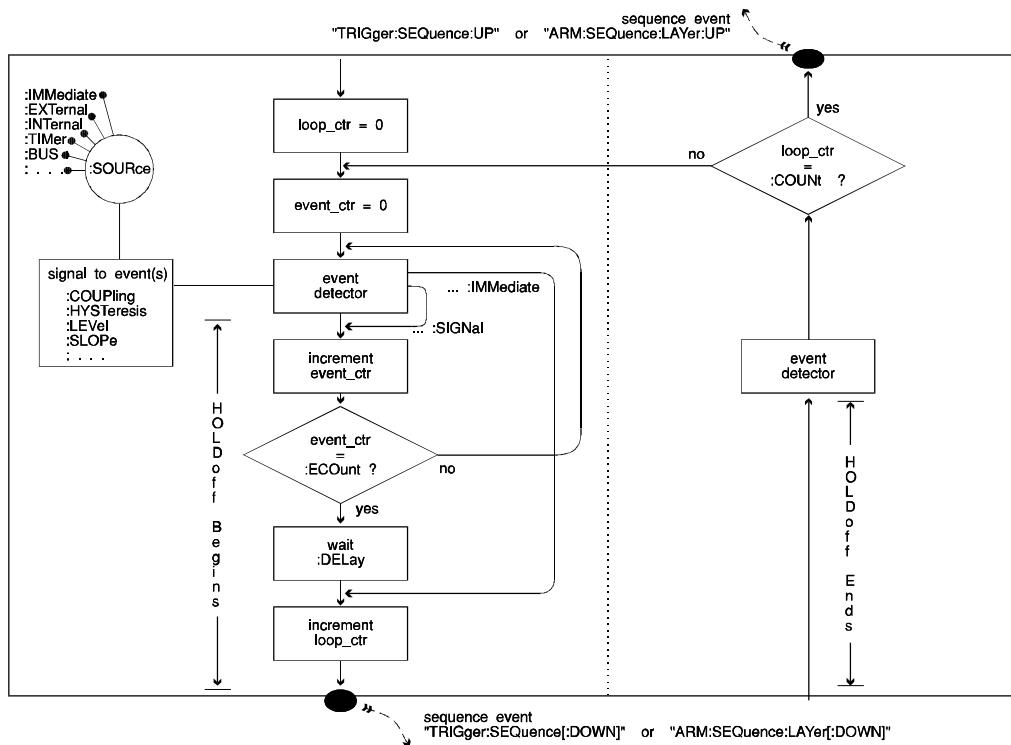
The downward traverse through an event detection layer depends on the sourced event being detected the specified number of times. Typically, the first event detected after entering the event detection layer is all that is required to proceed. However, a particular number of occurrences of the same event may be specified using ECOunt; for example, wait for the tenth positive edge of a signal. The downward traverse is also subject to a time delay if one is specified.

Two commands override the normal process of the downward traverse. The layer name followed by :IMMEDIATE causes the event detection and the subsequent delay to be bypassed. The layer name followed by :SIGNAl causes the device to proceed as though a single event had occurred.

On exit from the ARM or TRIGger state, in either direction, a “sequence event” is generated. These events may be used to start actions in the trigger subsystem or in other parts of the instrument. An action may not be associated with every “sequence event.”

The upward traverse is dependent upon the event detector being satisfied and the value of COUNT for the given layer. The event detector in the upward traverse is used to detect when an action started by the exit from this layer on the downward traverse has completed. This event detector has no commands associated with it, and it is only included in the model to prevent the trigger system from looping around until the “device action” associated with the given layer has completed.

When COUNT is greater than one, all of the subservient layers are cycled repeatedly COUNT times. For example, to make five measurements, each qualified by the same combination of ARM and TRIGGER events, the COUNT in the ARM layer should be set to 5. Each time the ARM layer is entered from below, the flow is redirected to follow the downward traverse. After the fifth cycle, when COUNT is satisfied, the upward traverse to the initiated state is made.



**Figure 24-4 Event Detection Layer**

### 24.3 Sequence Event Use

In the ARM-TRIGGER model shown in Fig 24-1, the “sequence event” generated by the TRIGGER layer is shown starting a “device action.” The “device action” is dependent on the settings of the SENSe or SOURce subsystems, for example by the FUNCtion selected. The settings of the SENSe and SOURce subsystems are in general device dependent and

therefore, the “device action” tied to a “sequence event” is also device dependent. For example, in one instrument the event may be used to generate a signal and in another instrument it may be used to make a measurement.

Further, the completion of a device action itself is device dependent. For example, the device action could be defined as executing a complete sweep, or it could be defined as initiating a sweep. These device actions would signal their completion at different times and consequently, would permit the upward transition through the layer at different times. Where an instrument permits control of the characteristics of a device action, the commands for that control shall appear in the subsystem employing the event. In the earlier example, a command to differentiate between initiating a sweep and executing a complete sweep would exist with the commands that control the other characteristics of the sweep.

The name of a “sequence event” reflects where it was generated. Its form is

“(INITiate|TRIGger)([:SEQUence[1]]]:SEQUence<n>[:<sequence\_name>)([:DOWN]:UP)”  
or  
“ARM([:SEQUence[1]]]:SEQUence<n>[:<sequence\_name>)([:LAYer[1]]]:LAYer<n>)  
(:DOWN]:UP)”

Note: Parentheses used here are for grouping and not literals. The syntax of a sequence event is the same as the syntax of the instrument-control header.

It includes the direction of traverse, the layer, and the sequence. The use of defaults, however, allows some short aliases. “ARM” is equivalent to  
“ARM:SEQUence1:LAYer1:DOWN”.

### 24.4

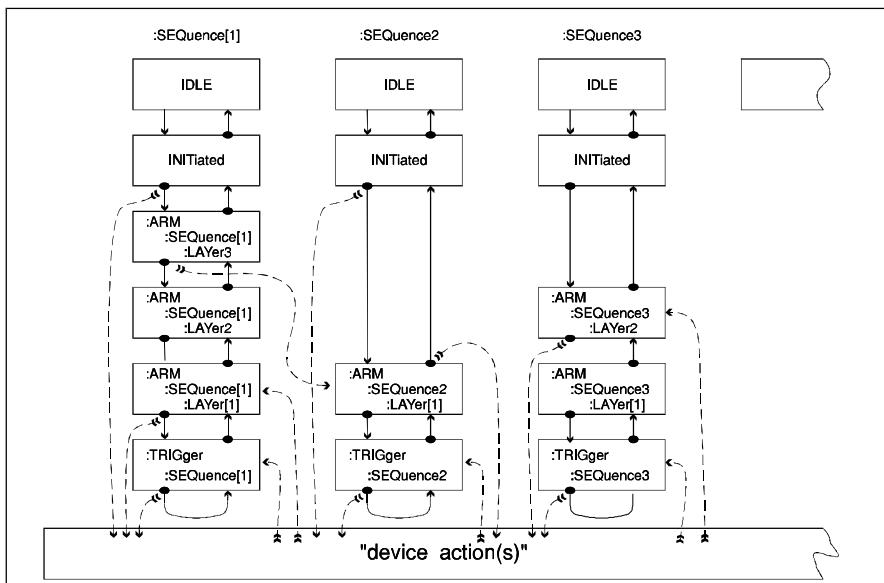
#### Expanded Capability Trigger Model

Fig 24-5 is an example of an expanded capability trigger model. An instrument could implement a more extensive or simpler trigger system. The purpose of the figure is to illustrate concepts. A picture of all possible configurations is not possible.

The expanded capability trigger model expands upon the ARM-TRIGger model to meet more complex needs through additional “Event Detection Layers” as shown in Fig 24-4. These additional layers give extra levels of event detection which provide more qualification to a particular action, or establish more “sequence events” to synchronize many actions within an instrument. Any number of “Event Detection Layers” may be included as additional ARM layers, as shown in Fig 24-5. A single trigger layer still exists in the expanded capability model.

Multiple sequences may exist in the expanded capability model to describe concurrently operating trigger conditions, also shown in Fig 24-5. Any number of sequences are permitted. Sequences may have the same or a different number of layers. Some of the sequences may have aliases. These aliases are used where the sequence is performing a standard trigger operation. Multiple sequences may be synchronized by linking “sequence events” of one sequence to the event detector of another sequence. Some of the standard sequences interact in a pre-defined manner which saves creating unnecessary links and complicating the model for the standard sequences.

An IDLE state is shown with each sequence. The commands \*RST and ABORt place all sequences immediately in their IDLE states. A device clear shall put the sequences to their IDLE states if the device is unable to parse another command without aborting the trigger subsystem (non-overlapped operation). Not aborting the initiated sequences is preferable if the parser can be cleared by a device clear so it can process a command.



**Figure 24-5 Expanded Capability Trigger Model**

### 24.4.1 LAYER Nomenclature

Each state in the trigger subsystem is made unique by its position determined by its row and column. In the trigger subsystem each column is called a SEQuence. The rows are labelled in a different manner to maintain compatibility with the ARM-TRIGger model. The lowest row is called the TRIGger layer. The next one up is called the first ARM layer. The third layer from the bottom is called the second ARM layer.

The <layer\_name> for a particular state is determined from its SEQuence, whether it is an ARM or a TRIGger layer, and finally the LAYer number if it is an ARM layer. In Figure 24-5, the layer in the lower left most position is TRIGger[:SEQUence[1]]. Because it is in SEQuence1, SEQuence is optional and may be omitted. For this layer, TRIGger is sufficient, and is the name used in the ARM-TRIGger model. SEQuence[1] also has the alias STARt. This layer's name is also TRIGger:STARt.

The SEQuence number increases from left to right, and other SEQuences may have aliases as well. The layer to the right of TRIGger:SEQUence1 is TRIGger:SEQUence2, which may also have the sequence alias TRIGger:STOP. The layer above TRIGger:STARt is ARM[:SEQUence[1]][:LAYer1]], which may be abbreviated to just ARM as LAYer1, as

well as SEQuence1, is the default. The next layer up is ARM[:SEQuence[1]]:LAYer2. The LAYer number increases from the lowest ARM layer to the top.

The “sequence event” names reflects the <layer\_name> and whether it occurred on the up or downward traverse. Thus, the event used in the ARM-TRIGger model to start the “device action” is TRIGger[:SEQuence[1]][:DOWN]. The events related to the INITiated state are INITiated[:DOWN] and INITiated:UP.

#### 24.4.2 Standard SEQuences

SCPI describes the behavior of SEQuences which have specific names. An instrument is not required to implement any of these sequences, but any sequence given one of these names shall behave as described.

STARt	The STARt sequence exists to provide pre-qualification to a “device action.” This sequence shall leave IDLE when an INITiate:IMMEDIATE:ALL or INITiate:CONTinuous:ALL ON is given. If the STARt is implemented then it shall exist as an alias to SEQuence[1].
STOP	The STOP sequence allows the user to force the currently initiated sequences to IDLE on a particular event or series of events. All SEQuences shall go to IDLE upon “INITiated:STOP:UP”.

The STOP sequence shall not leave IDLE when an INITiate:IMMEDIATE:ALL or INITiate:CONTinuous:ALL ON is given, but shall leave IDLE upon “INITiate:STARt:DOWN”. This means that STOP is subservient to STARt in the way the STOP sequence leaves IDLE. All sequences, however, are subservient to STOP because the sequence event from STOP affects all other sequences.

For those instruments that implement STARt as an alias for SEQuence[1], or implement STARt as an alias for SEQuence[1] and STOP as an alias for SEQuence2, the ARM and TRIGger SEQuence:DEFine command and query are not required. Those instruments that employ a different mapping, whether the mapping is fixed or user definable, or use other aliases shall implement the DEFine command and query for all sequences that exist.

#### 24.4.3 Subservient Sequences

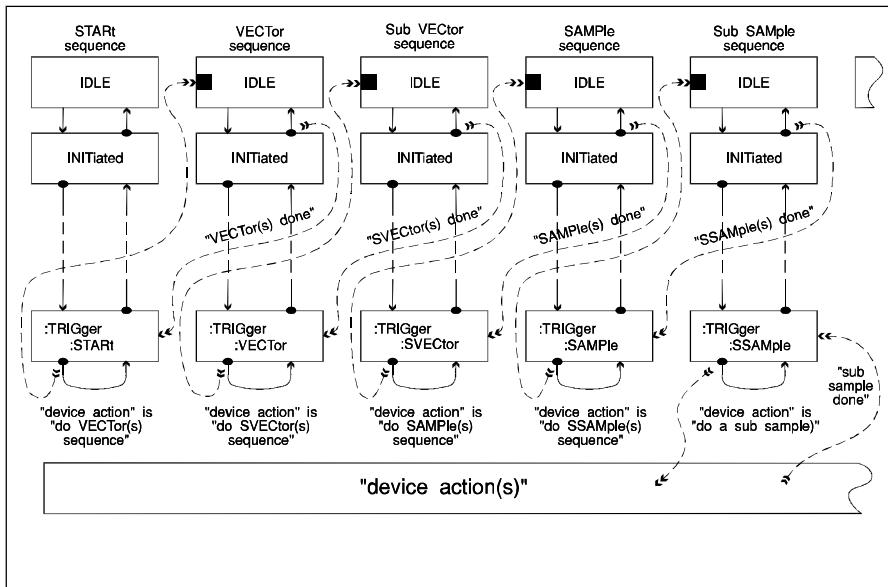
When an event from one sequence controls the initiation of another sequence, the second sequence is subservient to the first. The “device action” of a sequence initiates another sequence.

An instrument may implement any number of sequences depending on its needs and capabilities. The names for the sequences should reflect the application of an instrument. Naming sequences requires implementing the SEQuence:DEFine command.

Each subservient sequence has IDLE and INITiated states like other sequences. The INITiated state of a subservient sequence does not, however, check the setting of INITiate:CONTinuous. It merely exits out the top. This state is included so the event, INITiate:SEQuence:UP, is available for all sequences. The IDLE state of a subservient sequence does not check either the setting of INITiate:CONTinuous or for the arrival of

## 1999 SCPI Command Reference

INITiate[:IMMediate]. It does, however, check for an event from a connected sequence as shown in Figure 24-6.



**Figure 24-6 Subservient Sequences**

Fig 24-6 illustrates how several sequences could be organized in a subservient manner. An actual instrument would probably have more or fewer than four. It might also use different sequence names for the same device actions being controlled.

**VECTor** In this example, the VECTor sequence is used to control the pacing of whole vectors such as when generating or acquiring a waveform. Multiple VECTor sequences could exist, each distinguished by a numeric suffix. Several VECTor sequences would be used for multiple nested sweeps to generate a surface. For example, sweeping the power level at each point in a frequency sweep. The controls of the VECTor sequence often are coupled to either the SENSe:SWEep or SOURce:SWEep subsystems. Changing SWEep:DWEll and SWEep:TIME might change TIMER and DELay values.

**SVECtor** This Sub VECTor sequence is used to control the pacing of collections of samples used to create a VECTor. Sub VECTors are useful in acquisition instruments where whole vectors are averaged. The settings in such a case would be coupled to the SENSe:AVERage subsystem.

**SAMPLE** The SAMPLE sequence is used to control the pacing of each sample. A SAMPLE is a single point in the SVECtor.

## 1999 SCPI Command Reference

**SSAMple**      The Sub SAMple sequence is used to control the pacing of steps needed to create a sample. Sub SAMple is used where a SAMple point is composed of more than one acquisition on generation point.

SSAMple is subservient to SAMPle; SSAMple exits IDLE upon TRIGger:SAMPLE:DOWN. The upward event detector in TRIGger:SAMPLE is linked to INITiated:SSAMple:UP. Similarly, SAMPle is subservient to SVECtor and SVECtor is subservient to VECTor. Further, VECTor is subservient to VECTor2. This relationship is shown in Fig 24-6. The names and number of subservient sequences may vary between instruments, but the structure is the same.

## 1999 SCPI Command Reference

- ◆ The keywords ABORt, ARM, INITiate, and TRIGger are at the root level in the command hierarchy. They are grouped here because of their close functional relationships. ◆

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
ABORT		[no query] 1991
ARM		1991
[SEQuence]		1994
:DEFine	<sequence_name>	1994
:MGRules	<Boolean>	1994
[LAYer]		
:COUNt	<numeric_value>	
:COUpling	AC DC	
:DELay	<numeric_value>	
:AUTO	<Boolean> ONCE	
:ECL		[no query]
:ECOut	<numeric_value>	
:FILTer		1991
:HPASs		1991
:FREQuency	<numeric_value>	1991
[:STATE]	<Boolean>	1991
[LPASs]		1991
:FREQuency	<numeric_value>	1991
[:STATE]	<Boolean>	1991
:HYSTeresis	<numeric_value>	
[IMMEDIATE]		[no query]
:LEVel	<numeric_value>	
:AUTO	<Boolean>   ONCE	1994
:LINK	<event_handle>	
:PROTocol		1994
:VXI	SYNChronous SSYNchronous ASYNchronous	1994
:SIGNal		[no query]
:SLOPe	POSitive NEGative EITHER	
:SOURce	AINternal BUS ECLTrg<n> EXTernal HOLD  IMMEDIATE INTernal LINE LINK  MANual OUTPut TIMer TTLTrg<n>	1995
:TImer	<numeric_value>	
:TTL		[no query]
:TYPE	EDGE   VIDeo	1994
:VIDeo		1994
:FIELD		1994
[:NUMBER]	<numeric_value>	1994
:SELect	ODD   EVEN   ALL   NUMBER	1994
:FORMAT		1994
:LPFRame	<numeric_value>	1994
:LINE		1994,5

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
[:NUMBER]	<numeric_value>	1995
.SELect	ALL   NUMBER	1995
:SIGnal		1994
[:POLarity]	POSitive   NEGative	1994
INITiate		1991
:CONTinuous	<Boolean>	
[:ALL]	<Boolean>	1994
:NAME	<sequence_name>,<Boolean>	1994
:SEQUence	<Boolean>	1994
[:IMMEDIATE]		[no query]
[:ALL]		[no query]
:NAME	<sequence_name>	[no query]
:SEQUence		[no query]
:POFLag	INCLude   EXCLude	1995
TRIGger		1991
[:SEQUence]		1994
:ATRigger		1994
[:STATe]	<Boolean>	1994
:COUNT	<numeric_value>	
:COUPLing	AC DC	
:DEFine	<sequence_name>	1994
:MGRules	<Boolean>	1994
:DELAY	<numeric_value>	
:AUTO	<Boolean> ONCE	
:ECL		[no query]
:ECOunt	<numeric_value>	
:FILTter		1991
:HPASs		1991
:FREQuency	<numeric_value>	1991
[:STATe]	<Boolean>	1991
[:LPASs]		1991
:FREQuency	<numeric_value>	1991
[:STATe]	<Boolean>	1991
:HOLDoff	<numeric_value>	1994
:HYSTERESIS	<numeric_value>	
[:IMMEDIATE]		[no query]
:LEVel	<numeric_value>	
:AUTO	<Boolean>   ONCE	1994
:LINK	<event_handle>	
:PROTocol		1994
:VXI	SYNChronous SSYNchronous ASYNchronous	1994
:SIGNal		[no query]
:SLOPe	POSitive NEGative EITHER	

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:SOURce	AINternal BUS ECLTrg<n> EXTernal HOLD  IMMEDIATE INTernal LINE LINK  MANual OUTPut TITMer TTLTrg<n>	1995
:TITMer	<numeric_value>	
:TTL		[no query]
:TYPE	EDGE   VIDeo	1994
:VIDeo		1994
:FIELD		1994
[:NUMBER]	<numeric_value>	1994
:SElect	ODD   EVEN   ALL   NUMBER	1994
:FORMAT		1994
:LPFFrame	<numeric_value>	1994
:LINE		1994,5
[:NUMBER]	<numeric_value>	1995
:SElect	ALL   NUMBER	1995
:SSIGNAL		1994
[:POLarity]	POSitive   NEGative	1994

### 24.5 ABORt

ABORt

The ABORt command resets the trigger system and places all trigger sequences in the IDLE state. Any actions related to the trigger system that are in progress, such as a sweep or acquiring a measurement, shall also be aborted as quickly as possible. The ABORt command shall not be considered complete until all trigger sequences are in the IDLE state. The execution of an ABORt command shall set false the pending operation flags that were set by the initiation of the trigger system.

This command is an event and has no associated \*RST condition or query form.

### 24.6 ARM

ARM

The purpose of the ARM subsystem is to qualify a single event or a sequence of events before enabling the TRIGger. The ARM subsystem for a given sequence may be omitted if TRIGger does not require additional qualification by other events.

#### 24.6.1 [:SEQUENCE]

ARM:SEQUENCE

SEQUENCE is used in the expanded capability model to identify a particular sequence of layers in the trigger system. A <sequence\_name>, see 24.4.2, may also be used here.

##### 24.6.1.1 :DEFIne <sequence\_name>

ARM:SEQUENCE:DEFIne

Sets or queries the SEQUENCE alias. This command is required where aliases are employed and they do not correspond to the standard mapping. The numeric suffix on SEQUENCE corresponds to the sequence number to which the alias is to be applied.

This command is an alias to TRIGger:SEQUence:DEFine. Changing the value of ARM:SEQUence:DEFine changes the value of TRIGger:SEQUence:DEFine to the same value.

The <sequence\_name> is character data.

The effect of <sequence\_name> is modified by DEFine:MGRules as follows:

- Regardless of the state or presence of DEFine:MGRules, there is no difference between uppercase and lowercase letters used in <sequence\_name>.
- If the command DEFine:MGRules is absent or set to OFF, the full<sequence\_name> as written is the only form allowed.

For example, if the <sequence\_name> were SAMPLE1, then SAMPLE1 is the only alias accepted.

- If the command DEFine:MGRules is present and set to ON, the <sequence\_name> is interpreted as <program\_mnemonic>[<numeric\_suffix>] where all the contiguous trailing digits form the <numeric\_suffix>. Mnemonic generation rules as described in Syntax and Style, Section 6.2.1. shall apply to the <program\_mnemonic> and the <numeric suffix>.

For example, if the <sequence\_name> were defined as VECTOR or VECTOR1, then VECT, VECT1, VECTOR, and VECTOR1 would all be acceptable aliases.

If the <sequence\_name> is identical to a keyword under ARM or TRIGger subsystem, the device shall report execution error -224. For example ARM:SEQUence2:DEFine FILTER will cause an error. This command shall accept any other valid character program data for <sequence\_name>.

The query returns a string which contains the <sequence\_name>, so that a null string can indicate that nothing is defined.

If the command DEFine:MGRules is present and set to ON, the returned <sequence\_name> will follow the upper/lower case convention described in Syntax and Style, section 5.1, Interpreting Command Tables. An omitted <numeric\_suffix> or a <numeric\_suffix> of 1 will be represented as [1].

For example, if <sequence\_name> is VECTOR1 then the response is “VECTor[1]”; if it is DELAYED2, the response is “DELayed2”; if it is SAMPLE, the response is “SAMPlE[1]”.

By executing this command the previously defined name is overwritten.

\*RST has no effect on the defined name.

### 24.6.1.1.1 MGRules <Boolean>

ARM:SEQUence:DEFine:MGRules

This command sets or queries the state of MGRules, Mnemonic Generation Rules.

At \*RST, the value of this parameter is OFF.

### 24.6.1.2 [:LAYer]

ARM:SEQuence:LAYer

LAYer is used in the expanded capability model to identify a particular ARM LAYer in the trigger system.

#### 24.6.1.2.1 :COUNt <numeric\_value>

ARM:SEQuence:LAYer:COUNt

COUNt controls the path of the trigger system in the upward traverse of the event detection layer, shown in figure 24-4, “Event Detection Layer.” The trigger system is directed either to the next layer up or to loop back through the downward traverse. Moving to the next layer up occurs when the number of times that the trigger system has passed through the downward traverse (given by the value of loop\_ctr) is equal to the specified COUNt. In the simple case, COUNt has the value 1, a single downward traverse through the layer occurs, followed by the actions dictated by subservient layers, and finally a single upward traverse to next layer up. COUnt shall be set to a value of 1 or greater.

At \*RST, this value is set to 1.

#### 24.6.1.2.2 :COUPling AC|DC

ARM:SEQuence:LAYer:COUPling

COUPling only has effect if the source for the event detector is an analog electrical signal, such as INTernal, LINE, or EXTernal. It selects AC or DC coupling for the SOURced signal. DC coupling allows the signal’s AC and DC components to pass. AC coupling passes only the signal’s AC component.

At \*RST, this value is device-dependent.

#### 24.6.1.2.3 :DELay <numeric\_value>

ARM:SEQuence:LAYer:DELay

DELay sets the time duration between the recognition of an event(s) and the downward exit of the specified layer. DELay must be either zero or a positive value. An attempt to set it to a negative value shall cause an error -222 to be generated.

At \*RST, this value is set to 0 or the smallest available positive value. The units for DELay are seconds.

#### 24.6.1.2.3.1 :AUTO <Boolean>|ONCE

ARM:SEQuence:LAYer:DELay:AUTO

This command is used for sensor devices with internal settling delays. If ON, the delay is set to the minimum necessary to ensure that a valid measurement can be acquired. This value may be coupled to other instrument settings such as function, range, resolution, and input bandwidth.

At \*RST, AUTO is set ON.

**24.6.1.2.4 :ECL**

ARM:SEQUence:LAYer:ECL

This command is an event which presets LEVel, HYSTeresis, COUPling, and DELay to values appropriate for a ECL signal. This command cannot be queried.

**24.6.1.2.5 :ECOut <numeric\_value>**

ARM:SEQUence:LAYer:ECOut

ECOut specifies a particular number of occurrences of the same event that must be recognized. For example where it is necessary to wait for the tenth positive edge of a signal, ECOut must be set to 10. ECOut must be a positive value of 1 or greater. An attempt to set it differently shall cause an error -222 to be generated.

At \*RST, this value is set to 1.

**24.6.1.2.6 :FILT**

ARM:SEQUence:LAYer:FILT

This function allows a filter to be inserted between the source switch and event detector. FILTer only has effect if :SOURce is an analog switch such as INTernal, EXTernal, or LINE.

**24.6.1.2.6.1 :HPAS**

ARM:SEQUence:LAYer:FILT:HPAS

Controls the high pass filter.

**24.6.1.2.6.1.1 :FREQuency <numeric\_value>**

ARM:SEQUence:LAYer:FILT:HPAS:FREQuency

Determines the cutoff frequency of the high pass filter. Default units are Hertz.

At \*RST, this value is device-dependent.

**24.6.1.2.6.1.2 [:STATe] <Boolean>**

ARM:SEQUence:LAYer:FILT:HPAS:STATe

Turns the high pass filter ON or OFF.

At \*RST, this value is device-dependent.

**24.6.1.2.6.2 [:LPAS]**

ARM:SEQUence:LAYer:FILT:LPAS

Controls the low pass filter.

**24.6.1.2.6.2.1 :FREQuency <numeric\_value>**

ARM:SEQUence:LAYer:FILT:LPAS:FREQuency

Determines the cutoff frequency of the low pass filter. Default units are Hertz.

At \*RST, this value is device-dependent.

**24.6.1.2.6.2.2 [:STATe] <Boolean>**

ARM:SEQUence:LAYer:FILT:LPAS:STATe

Turns the low pass filter ON or OFF.

At \*RST, this value is device-dependent.

**24.6.1.2.7 :HYSTerisis <numeric\_value>**

ARM:SEQUence:LAYer:HYSTerisis

HYSTerisis is a qualifier of LEVel. It sets how far a signal must fall below LEVel before a rising edge can again be detected, and how far a signal must rise above LEVel before a falling edge can again be detected. Its function is to eliminate false events caused by signal noise. Units for the parameter default to the current amplitude units.

In instruments that have only two discrete values for hysteresis (such as Noise Rejection On and Noise Rejection Off), the device designer should pick two numeric values that approximate the actual hysteresis settings. The :HYSTerisis command has no :STATe function because confusion would occur when turning hysteresis off. This might imply to a user that the actual hysteresis had been set to 0. This is rarely the case in instruments that have only two states.

At \*RST, this value is device-dependent.

**24.6.1.2.8 [:IMMEDIATE]**

ARM:SEQUence:LAYer:IMMEDIATE

This command provides a onetime override of the normal process of the downward traverse of the event detection layer. This command shall cause the immediate exit of the specified event detection layer if the trigger system is in the specified layer when the IMMEDIATE is received. Otherwise, the IMMEDIATE command shall be ignored and an error -212 shall be generated.

The actual event detection and delay shall be skipped; however, normal processing of the loop\_ctr as shown in figure 24-4, “Event Detection Layer” shall occur. This command has no effect on any other settings in this subsystem.

This command is an event and has no \*RST condition and cannot be queried.

**24.6.1.2.9 :LEVel <numeric\_value>**

ARM:SEQUence:LAYer:LEVel

LEVel qualifies the characteristic of the selected SOURce signal that generates an event. LEVel only has effect if the source for the event detector is an analog electrical signal, such as INTERNAL, LINE, or EXTERNAL. Units for the parameter default to the current amplitude unit.

At \*RST, this value is instrument-dependent.

**24.6.1.2.9.1 :AUTO <Boolean> | ONCE**

ARM:SEQUence:LAYer:LEVel:AUTO

When AUTO is ON, the device dynamically selects the best arm level based on a device-dependent algorithm applied to the arm signal.

When AUTO is OFF, the arm LEVel sets the arm level.

Setting AUTO to ONCE turns AUTO ON, adjusts the arm level once and then resets AUTO to OFF.

Setting the arm LEVel turns AUTO OFF.

At \*RST, the value of this parameter is OFF.

#### 24.6.1.2.10 :LINK <event\_handle>

ARM:SEQuence:LAYer:LINK

Sets or queries the internal event that is LINKed to the event detector in the ARM layer. Internal events occur throughout all of the Instrument Model, a particular implementation will dictate which events are available to this LINK command. The <event\_handle> is STRING PROGRAM DATA and it is defined further in the Instrument Model, Chapter 2.

An Example of events that are commonly used, are the events that occur when a particular layer is exited, these handles are of the form:

"ARM[ :SEQuence[ :LAYer ]]" or "TRIGger[ :SEQuence ]"

Other common examples include completion of a process in a block or an instrument state transition.

At \*RST, this value is device-dependent.

#### 24.6.1.2.11 PROTocol

ARM:SEQuence:LAYer:PROTocol

This subsystem controls the protocol used with the selected source.

#### 24.6.1.2.11.1 VXI SYNChronous|SSYNchroNous|ASYNchroNous

ARM:SEQuence:LAYer:PROTocol:VXI

This command selects the trigger protocol for the VXI TTLTrg or ECLTrg trigger line when used as the arm source. The protocols are specified in the VXI Bus System Specification under sections B.6.2.3 and B.6.2.4.

- SYNChronous      selects the synchronous trigger protocol.
- SSYNchroNous      selects the semi-synchronous trigger protocol.
- ASYNchroNous      selects the asynchronous trigger protocol.

At \*RST, the SYNChronous protocol is selected.

#### 24.6.1.2.12 :SIGNal

ARM:SEQuence:LAYer:SIGNal

This command provides a one-time override of the normal process of the downward traverse through the event detector in figure 24-4, “Event Detection Layer.” This command shall cause the immediate exit of the event detector block in the specified event detection layer if the trigger system is waiting for the event specified by the SOURce command. Otherwise, the SIGNal command shall be ignored and an error -212 shall be generated.

Only the event detector block is bypassed. Normal processing of the delay and the event\_ctr as shown in figure 24-4, “Event Detection Layer,” shall occur. This command has no effect on any other settings in this subsystem.

This command defines an event, and as such does not have an associated query form or \*RST condition.

### 24.6.1.2.13 :SLOPe POSitive|NEGative|EITHer

ARM:SEQUence:LAYer:SLOPe

SLOPe qualifies whether the event occurs on the rising edge, falling edge, or either edge of the signal.

At \*RST, this value is set to POS.

### 24.6.1.2.14 :SOURce <parameter>

ARM:SEQUence:LAYer:SOURce

This command selects the source for the event detector. Only one source may be specified at a time for a given event detector. The various sources are:

- AINTernal — The triggering source is selected automatically by the instrument from internal algorithms to best match the current operating mode. An Automatic INTernal event is derived from measurement functions and/or sensor capabilities in the signal conditioning block.
- BUS — The source is signal specific to the control interface. For *IEEE 488.1*, the group execute trigger, GET, would satisfy this condition. In VXI the word serial command *TRIGger* performs this function. The event detector is also satisfied, independent of the interface, when a \*TRG command is received. Note that neither GET nor \*TRG can be sent without having an effect on the message exchange protocol described in *IEEE 488.2*.
- ECLTrg — The signal source is the specified VXI P2 or P3 backplane ECLTrg TRIGGER line. Valid lines are ECLTrg0 and ECLTrg1 on P2, and ECLTrg2 through ECLTrg5 on P3.
- EXTernal — An external signal jack is selected as the source. If no suffix is specified, EXTernal1 is assumed.
- HOLD — The event detection is disabled. However, the IMMEDIATE command shall override HOLD.
- IMMEDIATE — No waiting for an event occurs.
- INTernal — An internal channel is selected as the source. An INTernal event is derived from a measurement function and/or sensor capability in the signal conditioning block. If no suffix is specified, INTernal1 is assumed.
- LINE — The source signal is determined from the AC line voltage.
- LINK — The source for the event detector is specified by the LINK command.
- MANual — The signal is user-generated, such as by pressing a front panel key.
- OUTPut — The signal source is taken from an output channel. If no channel is specified, OUTPut1 is assumed.

- **TIMer** — The source signal comes from a periodic timer. The period of the timer is specified by the **TIMer** command.
- **TTLTrg** — The signal source is the specified VXI P2 backplane TTLTrg TRIGGER line. Valid lines are TTLTrg0 through TTLTrg7.

At \*RST, IMMEDIATE shall be selected as the SOURce. This value was chosen so that devices that do not implement any ARM layers (and hence the controls for the ARM layer) behave at \*RST in the same manner as devices that do.

#### 24.6.1.2.15 :**TIMer <numeric\_value>**

ARM:SEQUENCE:LAYER:TIMER

**TIMER** sets the period of an internal periodic signal source. Its value affects the trigger system only when it is selected as the SOURce for the event detector. **TIMER** must be a positive value. An attempt to set it differently shall cause an error -222 to be generated. The synchronization of the timer's period is device-dependent.

At \*RST, this value is device-dependent value. The default units for this value is seconds.

#### 24.6.1.2.16 :**TTL**

ARM:SEQUENCE:LAYER:TTL

This command is an event which presets LEVel, HYSTeresis, COUpling, and DELay to values appropriate for a TTL signal. This command cannot be queried.

#### 24.6.1.2.17 :**TYPE EDGE | VIDEO**

ARM:SEQUENCE:LAYER:TYPE

Sets or queries the type of triggering.

At \*RST, the value of this parameter is EDGE.

#### 24.6.1.2.18 :**VIDEO**

ARM:SEQUENCE:LAYER:VIDEO

This subsystem controls parameters necessary to trigger on video signals.

#### 24.6.1.2.18.1 :**FIELD**

ARM:SEQUENCE:LAYER:VIDEO:FIELD

This subsystem selects the video field to trigger on.

#### 24.6.1.2.18.1.1 :**[:NUMBER] <numeric\_value>**

ARM:SEQUENCE:LAYER:VIDEO:FIELD:NUMBER

NUMBER sets or queries the field number to trigger on if FIELD:SELECT is set to NUMBER.

At \*RST, the value of this parameter is 1.

#### 24.6.1.2.18.1.2 :**SElect ODD | EVEN | ALL | NUMBER**

ARM:SEQUENCE:LAYER:VIDEO:FIELD:SELECT

SElect sets or queries the video field selection method to be used. Four character data parameters are defined as follows:

- ODD selects odd-numbered fields to trigger on.
- EVEN selects even-numbered fields to trigger on.
- ALL triggers on all fields regardless of number.
- NUMBER triggers on the field number specified by NUMBER.

At \*RST, the value of this parameter is ALL.

### 24.6.1.2.18.2 :FORMAT

ARM:SEQUENCE:LAYEr:VIDeo:FORMAT

FORMAT controls parameters that allow the video trigger system to respond to signals from standard video formats.

### 24.6.1.2.18.2.1 :LPFRame <numeric\_variable>

ARM:SEQUENCE:LAYEr:VIDeo:FORMAT:LPFRame

LPFRame, Lines Per FFrame, sets or queries the lines per frame associated with the signal formatting standard being used.

For example, A signal following NTSC would select 525 lines per frame while a PAL signal would select 625 lines per frame.

At \*RST, the value of this parameter is device dependent.

### 24.6.1.2.18.3 :LINE

ARM:SEQUENCE:LAYEr:VIDeo:LINE

This subsystem selects a particular line in the field to arm on.

### 24.6.1.2.18.3.1 [:NUMBER] <numeric\_value>

ARM:SEQUENCE:LAYEr:VIDeo:LINE:NUMBER

NUMBER sets or queries the line number to arm on if LINE:SElect is set to NUMBER. Setting the NUMBER value has no effect on the value of the SElect parameter.

At \*RST, the value of this parameter is device dependent.

### 24.6.1.2.18.3.2 :SElect ALL | NUMBER

ARM:SEQUENCE:LAYEr:VIDeo:LINE:SElect

SElect sets or queries the video line selection method to be used. Two character data parameters are defined as follows:

- ALL — arms on all lines regardless of number.
- NUMBER — arms on the line number specified by NUMBER.

At \*RST, the value of this parameter is ALL.

### 24.6.1.2.18.4 :SSIGnal

ARM:SEQUENCE:LAYEr:VIDeo:SSIGnal

SSIGnal, Synchronizing SIGnal, controls parameters relating to the video synchronizing signal.

**24.6.1.2.18.4.1 :POLarity POSitive | NEGative**

ARM:SEQUence:LAYer:VIDeo:SSIGnal:POLarity

SSIGnal:POLarity, sets or queries sync pulse triggering polarity.

At \*RST, the value of this parameter is NEGative.

**24.7 INITiate**

INITiate

The INITiate subsystem is used to control the initiation of the trigger subsystem. It initiates all trigger sequences as a group except those sequences that are defined otherwise by either the standard or in the instrument documentation.

Instruments which implement the INITiate command as overlapped can continue parsing and executing subsequent commands and queries while initiated. Instruments which implement the INITiate commands as sequential cannot execute subsequent commands and queries until the trigger model has returned to the IDLE state.

For this reason, devices which implement the INITiate commands as sequential shall enter IDLE when the IEEE 488.2 **dcas** message is received. They shall also execute the next message unit received after the **dcas** before leaving IDLE (if INITiate:CONTinuous is ON). **dcas** followed by \*RST or **dcas** followed by INIT:CONT OFF shall always stop the trigger model in the IDLE state.

There are advantages to implementing the INITiate commands as overlapped.

- The instrument designer decides whether or not a device clear aborts the trigger model.
- The instrument can continue parsing and executing commands and queries while the trigger model is initiated (for example, FETCh? to get the readings which are being triggered in a meter).

The disadvantage to implementing any commands as overlapped, including the INITiate commands, is that it adds complexity to the implementation of the IEEE 488.2 common commands \*OPC, \*WAI and \*OPC?

**24.7.1 :CONTinuous <Boolean>**

INITiate:CONTinuous

The CONTinuous command is used to select whether the trigger system is continuously initiated or not. With CONTinuous set to OFF, the trigger system shall remain in the IDLE state until CONTinuous is set to ON or INITiate:IMMEDIATE is received. Once CONTinuous is set to ON, the trigger system shall be initiated and shall exit the IDLE state. On completion of each trigger cycle, with CONTinuous ON, the trigger system shall immediately commence another trigger cycle without entering the IDLE state.

When INITiate:CONTinuous is set to OFF, the current trigger cycle shall be completed before entering the IDLE state. The return to IDLE shall also occur as the result of an ABORt or \*RST command.

With the trigger system set to cycle continuously, the ABORt command shall force the trigger system to the IDLE state; however, the value of INITiate:CONTinuous is unaffected. If CONTinuous was set to ON prior to receiving ABORt, it shall remain ON and the trigger system shall immediately exit the IDLE state.

This subsystem does not apply to sequences that are initiated by the INITiate:ALL command.

#### 24.7.1.1 [:ALL] <Boolean>

INITiate:CONTinuous:ALL

Sets or queries whether or not all sequences are continuously initiated.

At \*RST, this value is set to OFF.

#### 24.7.1.2 :NAME <sequence\_name>,<Boolean>

INITiate:CONTinuous:NAME

Sets or queries whether or not the SEQuence with the alias specified by <sequence\_name> is continuously initiated. The <sequence\_name> is character program data. See 24.4.2 for standard sequence names.

The <sequence\_name> is a required parameter of the query form, the query form returns 0 or 1.

At \*RST, this value is set to OFF.

#### 24.7.1.3 :SEQuence <Boolean>

INITiate:CONTinuous:SEQuence

Sets or queries whether or not the specified SEQuence is continuously initiated. The numeric suffix on SEQuence corresponds to the sequence number. If NAME is implemented, this command shall also be implemented.

At \*RST, this value is set to OFF.

#### 24.7.2 [:IMMEDIATE]

INITiate:IMMEDIATE

These commands shall cause all sequences to exit the IDLE state; they are initiated. The IMMEDIATE command shall cause the trigger system to initiate and complete one full trigger cycle, returning to IDLE on completion. If the device is not in IDLE or if INITiate:CONTinuous is set to ON, an IMMEDIATE command shall have no effect on the trigger system and an error -213 shall be generated.

INITiate[:IMMEDIATE] is an event and cannot be queried as there is no state associated with it.

#### 24.7.2.1 [:ALL]

INITiate:IMMEDIATE:ALL

This command causes all SEQuences to be INITiated, except those defined to behave otherwise.

**24.7.2.2 :NAME <sequence\_name>**

INITiate:IMMediate:NAME

This command causes the SEQuence with the alias specified by <sequence\_name> to be INITiated. The <sequence\_name> is character program data. See 24.4.2 for standard sequence names.

**24.7.2.3 :SEQuence**

INITiate:IMMediate:SEQuence

This command causes the specified SEQuence to be INITiated. The numeric suffix on SEQuence corresponds to the sequence number. If NAME is implemented, this command shall also be implemented.

**24.7.3 :POFLag INCLude | EXCLude**

INITiate:POFLag

POFLag, Pending Operation Flag, allows the Pending-Operation flag associated with the initiation of a trigger sequence to be included or excluded from the No-Operation-Pending flag set.

Setting POFLag to EXCLude means the No-Operation-Pending flag set is independent of the state of the trigger subsystem. Moving from the INITiated to IDLE trigger states and vice versa has no effect on the NOP flag in this state.

Setting POFLag to INCLude means the No-Operation-Pending flag set includes the trigger subsystem's Pending-Operation flag which goes true when the trigger subsystem moves from IDLE to INITiated state and returns to false when IDLE is re-entered.

At \*RST, the value of this parameter is INCLude.

**24.8 TRIGger**

TRIGger

The purpose of the TRIGger subsystem is to qualify a single event before enabling the triggered sequence operation, such as enabling a sweep, starting a measurement, or changing the state of the device.

**24.8.1 [:SEQuence]**

TRIGger:SEQuence

SEQuence is used in the expanded capability model to identify a particular sequence of layers in the trigger system. A <sequence\_name>, see 24.4.2, may also be used here.

**24.8.1.1 :ATRigger**

TRIGger:SEQuence:ATRigger

ATRigger, Auto TRigger, controls an event detector timer that causes control flow to exit the event detector if a valid trigger does not appear within a preset time period. The auto trigger timer is started upon entering the TRIGger layer, and after the control flow leaves the event detector. The precise timing of the timer is device dependent. See Figure 24-4.

**24.8.1.1.1 [:STATe] <Boolean>**

TRIGger:SEQuence:ATRigger:STATe

STATe sets and queries the state of the auto trigger function. STATe ON enables the function and STATe OFF disables it.

At \*RST, the value of this parameter is OFF.

**24.8.1.2 :COUNt <numeric\_value>**

TRIGger:SEQuence:COUNt

COUNt controls the path of the trigger system in the upward traverse of the event detection layer, shown in figure 24-4, “Event Detection Layer.” The trigger system is directed either to the next layer up or to loop back through the downward traverse. Moving to the next layer up occurs when the number of times that the trigger system has passed through the downward traverse (given by the value of loop\_ctr) is equal to the specified COUNt. In the simple case, COUNt has the value 1, a single downward traverse through the layer occurs, followed by the actions dictated by subservient layers and finally a single upward traverse to next layer up. COUNt shall be set to a value of 1 or greater.

At \*RST, this value is set to 1.

**24.8.1.3 :COUpling AC|DC**

TRIGger:SEQuence:COUPling

COUPling only has effect if the source for the event detector is an analog electrical signal, such as INTERNAL, LINE, or EXTERNAL. It selects AC or DC coupling for the SOURced signal. DC coupling allows the signal’s AC and DC components to pass. AC coupling passes only the signal’s AC component.

At \*RST, this value is device-dependent.

**24.8.1.4 :DEFIne <sequence\_name>**

TRIGger:SEQuence:DEFIne

Sets or queries the SEQuence alias. This command is required where aliases are employed and they do not correspond to the standard mapping. The numeric suffix on SEQuence corresponds the sequence number to which the alias is to be applied.

This command is an alias to ARM:SEQuence:LAYer:DEFIne. Changing the value of TRIGger:SEQuence:DEFIne changes the value of ARM:SEQuence:DEFIne to the same value.

The <sequence\_name> is character data.

The effect of <sequence\_name> is modified by DEFIne:MGRules as follows:

- Regardless of the state or presence of DEFIne:MGRules, there is no difference between uppercase and lowercase letters used in <sequence\_name>.
- If the command DEFIne:MGRules is absent or set to OFF, the full<sequence\_name> as written is the only form allowed.

For example, if the <sequence\_name> were SAMPLE1, then SAMPLE1 is the only alias accepted.

- If the command DEFine:MGRules is present and set to ON, the <sequence\_name> is interpreted as <program\_mnemonic>[<numeric\_suffix>] where all the contiguous trailing digits form the <numeric\_suffix>. Mnemonic generation rules as described in Syntax and Style, Section 6.2.1. shall apply to the <program\_mnemonic> and the <numeric suffix>.

For example, if the <sequence\_name> were defined as VECTOR or VECTOR1, then VECT, VECT1, VECTOR, and VECTOR1 would all be acceptable aliases.

If the <sequence\_name> is identical to a keyword under ARM or TRIGger subsystem, the device shall report execution error -224. For example ARM:SEQuence2:DEFine FILTER will cause an error. This command shall accept any other valid character program data for <sequence\_name>.

The query returns a string which contains the <sequence\_name>, so that a null string can indicate that nothing is defined.

If the command DEFine:MGRules is present and set to ON, the returned <sequence\_name> will follow the upper/lower case convention described in Syntax and Style, section 5.1, Interpreting Command Tables. An omitted <numeric\_suffix> or a <numeric\_suffix> of 1 will be represented as [1].

For example, if <sequence\_name> is VECTOR1 then the response is “VECTOr[1]”; if it is DELAYED2, the response is “DELayed2”; if it is SAMPLE, the response is “SAMPlE[1]”.

By executing this command the previously defined name is overwritten.

\*RST has no effect on the defined name.

### 24.8.1.4.1 MGRules <Boolean>

TRIGger:SEQuence:DEFine:MGRules

This command sets or queries the state of MGRules, Mnemonic Generation Rules.

At \*RST, the value of this parameter is OFF.

### 24.8.1.5 :DELay <numeric\_value>

TRIGger:SEQuence:DELay

DELay sets the time duration between the recognition of an event(s) and the downward exit of the specified layer. DELay must be either zero or a positive value. An attempt to set it to a negative value shall cause an error -222 to be generated.

At \*RST, this value is set to 0 or the smallest available positive value. The value has units of seconds.

**24.8.1.5.1 :AUTO <Boolean>|ONCE**

TRIGger:SEQUence:DELay:AUTO

This command is used for sensor devices with internal settling delays. If ON, the delay is set to the minimum necessary to ensure that a valid measurement can be acquired. This value may be coupled to other instrument settings such as function, range, resolution, and input bandwidth.

At \*RST, AUTO is set ON.

**24.8.1.6 :ECL**

TRIGger:SEQUence:ECL

This command is an event which presets LEVel, HYSTeresis, COUPling, and DELay to values appropriate for a ECL signal. This command cannot be queried.

**24.8.1.7 :ECOut <numeric\_value>**

TRIGger:SEQUence:ECOut

ECOut specifies a particular number of occurrences of the same event that must be recognized. For example, where it is necessary to wait for the tenth positive edge of a signal, ECOut must be set to 10. ECOut must be a positive value of 1 or greater. An attempt to set it differently shall cause an error -222 to be generated.

At \*RST, this value is set to 1.

**24.8.1.8 :FILTter**

TRIGger:SEQUence:FILTter

This function allows a filter to be inserted between the source switch and event detector. FILTter only has effect if :SOURce is an analog switch such as INTernal, EXTERNAL, or LINE.

**24.8.1.8.1 :HPASs**

TRIGger:SEQUence:FILTter:HPASs

Controls the high pass filter.

**24.8.1.8.1.1 :FREQuency <numeric\_value>**

TRIGger:SEQUence:FILTter:HPASs:FREQuency

Determines the cutoff frequency of the high pass filter. Default units are Hertz.

At \*RST, this value is device-dependent.

**24.8.1.8.1.2 [:STATe] <Boolean>**

TRIGger:SEQUence:FILTter:HPASs:STATe

Turns the high pass filter ON or OFF.

At \*RST, this value is device-dependent.

**24.8.1.8.2 [:LPASs]**

TRIGger:SEQUence:FILTter:LPASs

Controls the low pass filter.

**24.8.1.8.2.1 :FREQuency <numeric\_value>**

TRIGger:SEQuence:FILTer:LPASs:FREQuency

Determines the cutoff frequency of the low pass filter. Default units are Hertz.

At \*RST, this value is device-dependent.

**24.8.1.8.2.2 [:STATe] <Boolean>**

TRIGger:SEQuence:FILTer:LPASs:STATe

Turns the low pass filter ON or OFF.

At \*RST, this value is device-dependent.

**24.8.1.9 :HOLDoff <numeric\_value>**

TRIGger:SEQuence:HOLDoff

HOLDoff controls the time during which the event detector is inhibited from acting on any new trigger. The exact point in the trigger model at which the trigger holdoff timer is started is device dependent, but it shall be after the event detector on the downward traverse and before the event detector on the upward traverse. See Figure 24-4.

The range of the parameter is zero to one, where one is maximum holdoff and zero is minimum.

At \*RST, the value of the parameter is zero.

**24.8.1.10 :HYSTeresis <numeric\_value>**

TRIGger:SEQuence:HYSTeresis

HYSTeresis is a qualifier of LEVel. It sets how far a signal must fall below LEVel before a rising edge can again be detected, and how far a signal must rise above LEVel before a falling edge can again be detected. Its function is to eliminate false events caused by signal noise. Units for the parameter default to the current amplitude units.

In instruments that have only two discrete values for hysteresis (such as Noise Rejection On and Noise Rejection Off), the device designer should pick two numeric values that approximate the actual hysteresis settings. The :HYSTeresis command has no :STATe function because confusion would occur when turning hysteresis off. This might imply to a user that the actual hysteresis had been set to 0. This is rarely the case in instruments that have only two states.

At \*RST, this value is device-dependent.

**24.8.1.11 [:IMMEDIATE]**

TRIGger:SEQuence:IMMEDIATE

This command exists to provide a one-time override of the normal process of the downward traverse of the event detection layer. This command shall cause the immediate exit of the specified event detection layer if the trigger system is in the specified layer when the IMMEDIATE is received. Otherwise, the IMMEDIATE command shall be ignored and an error -211 shall be generated.

The actual event detection and delay shall be skipped; however, normal processing of the loop\_ctr as shown in figure 24-4, “Event Detection Layer,” shall occur. This command has no effect on any other settings in this subsystem.

This command is an event, has no \*RST condition, and cannot be queried.

### 24.8.1.12 :LEVel <numeric\_value>

TRIGger:SEQUence:LEVel

LEVel qualifies the characteristic of the selected SOURce signal that generates an event. LEVel only has effect if the source for the event detector is an analog electrical signal, such as INTernal, LINE, or EXTERNAL. Units for the parameter default to the current amplitude unit.

At \*RST, this value is instrument-dependent.

### 24.8.1.12.1 :AUTO <Boolean> | ONCE

TRIGger:SEQUence:LEVel:AUTO

When AUTO is ON, the device dynamically selects the best trigger level based on a device-dependent algorithm applied to the trigger signal.

When AUTO is OFF, the trigger LEVel sets the trigger level.

Setting AUTO to ONCE turns AUTO ON, adjusts the trigger level once and then resets AUTO to OFF.

Setting the trigger LEVel turns AUTO OFF.

At \*RST, the value of this parameter is OFF.

### 24.8.1.13 :LINK <event\_handle>

TRIGger:SEQUence:LINK

Sets or queries the internal event that is LINKed to the event detector in the TRIGger layer. Internal events occur throughout all of the Instrument Model, a particular implementation will dictate which events are available to this LINK command. The <event\_handle> is STRING PROGRAM DATA and it is defined further in the Instrument Model, Chapter 2.

An Example of events that are commonly used, are the events that occur when a particular layer is exited, these handles are of the form:

“ARM[ :SEQUence[ :LAYER ] ]” or “TRIGger[ :SEQUence ]”

Other common examples include completion of a process in a block or an instrument state transition.

At \*RST, this value is device-dependent.

### 24.8.1.14 :PROTocol

TRIGger:SEQUence:PROTocol

This subsystem controls the protocol used with the selected source.

**24.8.1.14.1 :VXI SYNChronous|SSYNchronous|ASYNchronous**

TRIGger:SEQuence:PROTocol:VXI

This command selects the trigger protocol for the VXI TTLTrg or ECLTrg trigger line when used as the trigger source. The protocols are specified in the VXI Bus System Specification under sections B.6.2.3 and B.6.2.4.

- SYNChronous      selects the synchronous trigger protocol.
- SSYNchronous      selects the semi-synchronous trigger protocol.
- ASYNchronous      selects the asynchronous trigger protocol.

At \*RST, the SYNChronous protocol is selected.

**24.8.1.15 :SIGNal**

TRIGger:SEQuence:SIGNAL

This command provides a one-time override of the normal process of the downward traverse through the event detector in figure 24-4, “Event Detection Layer.” This command shall cause the immediate exit of the event detector block in the specified event detection layer if the trigger system is waiting for the event specified by the SOURce command. Otherwise, the SIGNal command shall be ignored and an error -211 shall be generated.

Only the event detector block is bypassed. Normal processing of the delay and the event\_ctr as shown in figure 24-4, “Event Detection Layer,” shall occur. This command has no effect on any other settings in this subsystem.

This command defines an event, and as such does not have an associated query form or \*RST condition.

**24.8.1.16 :SLOPe POSitive|NEGative|EITHer**

TRIGger:SEQuence:SLOPe

SLOPe qualifies whether the event occurs on the rising edge, falling edge, or either edge of the signal.

At \*RST, this value is set to POS.

**24.8.1.17 :SOURce <parameter>**

TRIGger:SEQuence:SOURce

This command selects the source for the event detector. Only one source may be specified in the parameter at a time for a given event detector. The various sources are:

- AINTernal — The triggering source is selected automatically by the instrument from internal algorithms to best match the current operating mode. An Automatic INTernal event is derived from measurement functions and/or sensor capabilities in the signal conditioning block.
- BUS — The source is signal specific to the control interface. For IEEE 488.1 the group execute trigger, GET, would satisfy this condition. In VXI the word serial command *TRIGger* performs this function. The event detector is also satisfied, independent of the interface, when a \*TRG command is received. Note that

## 1999 SCPI Command Reference

neither GET nor \*TRG can be sent without having an effect on the message exchange protocol described in *IEEE 488.2*.

- ECLTrg — The signal source is the specified VXI P2 or P3 backplane ECLTrg TRIGGER line. Valid lines are ECLTrg0 and ECLTrg1 on P2, and ECLTrg2 through ECLTrg5 on P3.
- EXTernal — An external signal jack is selected as the source. If no suffix is specified, EXTernal1 is assumed.
- HOLD — The event detection is disabled. However, the IMMEDIATE command shall override HOLD.
- IMMEDIATE — No waiting for an event occurs.
- INTernal — An internal channel is selected as the source. An INTernal event is derived from a measurement function and/or sensor capability in the signal conditioning block. If no suffix is specified, INTernal1 is assumed.
- LINE — The source signal is determined from the AC line voltage.
- LINK — The source for the event detector is specified by the LINK command.
- MANual — The signal is user-generated, such as by pressing a front panel key.
- OUTPut — The signal source is taken from an output channel. If no channel is specified, OUTPut1 is assumed.
- TIMer — The source signal comes from a periodic timer. The period of the timer is specified by the TIMer command.
- TTLTrg — The signal source is the specified VXI P2 backplane TTLTrg TRIGGER line. Valid lines are TTLTrg0 through TTLTrg7.

At \*RST, IMMEDIATE shall be selected as the SOURce.

### 24.8.1.18 :TIMer <numeric\_value>

TRIGger:SEQunce:TIMer

TIMer sets the period of an internal periodic signal source. Its value affects the trigger system only when it is selected as the SOURce for the event detector. TIMer must be a positive value. An attempt to set it differently shall cause an error -222 to be generated. The synchronization of the timer's period is device-dependent.

At \*RST, this value is set to a device-dependent value.

### 24.8.1.19 :TTL

TRIGger:SEQunce:TTL

This command is an event which presets LEVel, HYSTeresis, COUPling, and DELay to values appropriate for a TTL signal. This command cannot be queried.

**24.8.1.20 :TYPE EDGE | VIDEO**

TRIGger:SEQuence:TYPE

Sets or queries the type of triggering.

At \*RST, the value of this parameter is EDGE.

**24.8.1.21 :VIDeo**

TRIGger:SEQuence:VIDeo

This subsystem controls parameters necessary to trigger on video signals.

**24.8.1.21.1 :FIELd**

TRIGger:SEQuence:VIDeo:FIELd

This subsystem selects the video field to trigger on.

**24.8.1.21.1.1 [:NUMBer] <numeric\_value>**

TRIGger:SEQuence:VIDeo:FIELd:NUMBer

NUMBer sets or queries the field number to trigger on if FIELd:SELect is set to NUMBer.

At \*RST, the value of this parameter is 1.

**24.8.1.21.1.2 :SELECT ODD | EVEN | ALL | NUMBER**

TRIGger:SEQuence:VIDeo:FIELd:SELect

SELect sets or queries the video field selection method to be used. Four character data parameters are defined as follows:

- ODD selects odd-numbered fields to trigger on.
- EVEN selects even-numbered fields to trigger on.
- ALL triggers on all fields regardless of number.
- NUMBER triggers on the field number specified by NUMBER.

At \*RST, the value of this parameter is ALL.

**24.8.1.21.2 :FORMAT**

TRIGger:SEQuence:VIDeo:FORMAT

FORMAT controls parameters that allow the video trigger system to respond to signals from standard video formats.

**24.8.1.21.2.1 :LPFRame <numeric\_variable>**

TRIGger:SEQuence:VIDeo:FORMAT:LPFRame

LPFRame, Lines Per FFrame, sets or queries the lines per frame associated with the signal formatting standard being used.

For example, A signal following NTSC would select 525 lines per frame while a PAL signal would select 625 lines per frame.

At \*RST, the value of this parameter is device dependent.

### 24.8.1.21.3 :LINE

TRIGger:SEQUence:VIDeo:LINE

This subsystem selects a particular line in the field to trigger on.

#### 24.8.1.21.3.1 [:NUMBER] <numeric\_value>

TRIGger:SEQUence:VIDeo:LINE:NUMBER

NUMBER sets or queries the line number to trigger on if LINE:SElect is set to NUMBER. Setting the NUMBER value has no effect on the value of the SElect parameter.

At \*RST, the value of this parameter is device dependent.

#### 24.8.1.21.3.2 :SElect ALL | NUMBER

TRIGger:SEQUence:VIDeo:LINE:SElect

SElect sets or queries the video line selection method to be used. Two character data parameters are defined as follows:

- ALL — triggers on all lines regardless of number.
- NUMBER — triggers on the line number specified by NUMBER.

At \*RST, the value of this parameter is ALL.

### 24.8.1.21.4 :SSIGnal

TRIGger:SEQUence:VIDeo:SSIGnal

SSIGnal, Synchronizing SIGnal, controls parameters relating to the video synchronizing signal.

#### 24.8.1.21.4.1 :POLarity POSitive | NEGative

TRIGger:SEQUence:VIDeo:SSIGnal:POLarity

SSIGnal:POLarity, sets or queries sync pulse triggering polarity.

At \*RST, the value of this parameter is NEGative.

## 25 UNIT Subsystem

Default units are defined, where applicable, for each SCPI command. The UNIT subsystem provides a mechanism to change the default values. The units selected apply to the designated command parameters for both command and response.

The UNIT command at the root level has a global effect on the selected units. The UNIT command may also be applied to lower levels in the SCPI command hierarchy to have a localized effect. When the UNIT command is applied to a node, then all the nodes below the node to which the unit command was applied shall be affected by the localized UNIT command. There is no restriction on the number of levels to which UNIT may be applied. In this way the more global units are overridden by the more local units. Units may also be overridden temporarily by attaching the desired unit as a suffix to the appropriate parameter in the command, if the instrument supports that unit.

For example, to program a source with units of Volts, the command UNIT:VOLTage VOLT would be used. To program only the modulator of that same source in dBVs, the additional command [SOURce:]MODulation:UNIT DBUV would be required.

The UNIT command cannot be applied to either SENSe, SOURce or ROUTe nodes, since the UNIT command at these nodes and at the root cannot be unambiguously recognized. This condition arises from the ability to have one of SENSe, SOURce or ROUTe nodes optional.

KEYWORD	PARAMETER FORM	NOTES
:UNIT		
:ANGLE	DEG RAD	
:CURRent	<amplitude unit>	
:POWER	<amplitude unit>	
:TEMPerature	C CEL F FAR K	
:TIME	HOUR MINute SECond	
:VOLTage	<amplitude unit>	1993

### 25.1 :ANGLE DEG|RAD

UNIT:ANGLE

Specifies the fundamental unit of angle.

At \*RST, the default unit is RAD.

### 25.2 :CURRent, :POWER, and :VOLTage

UNIT:CURRent

Selects a default unit for commands which program absolute amplitude. The default unit may be overridden when programming a function by explicitly sending a suffix. The default unit may also be overridden if a unit command exists at a lower tree level. Query values of commands which program in amplitude units shall be returned in the current amplitude unit

## 1999 SCPI Command Reference

unless overridden by a unit command at a tree level closer to the command itself (such as POWER:UNIT).

The possible values are:

- **Linear power:** W (Watt)
- **Linear voltage:** V (Volt)
- **Linear current:** A (Ampere)
- **Logarithmic power:** DBM, DB[IEEE suffix multiplier]W (such as DBUW)
- **Logarithmic voltage:** DB[IEEE suffix multiplier]V (such as DBUV)
- **Logarithmic current:** DB[IEEE suffix multiplier]A (such as DBA)

An instrument must accept all values in any class if it accepts any value in that class. However, the decision to accept a class is device-dependent. A device which accepts logarithmic units must also accept the comparable linear unit. This means that UNIT:VOLTage WATT is permissible if it makes sense for the instrument. It is also implied that an impedance is specified for the signal when converting between different fundamental quantities.

At \*RST, this value is device-dependent. However, the following restrictions apply. The UNIT subsystem must be implemented if a device chooses any value other than the fundamental unit for either the VOLTage, POWER, or CURRent subsystem , or if it implements logarithmic units.

### 25.3 :TEMPerature C|CEL|F|FAR|K

UNIT:TEMPerature

Specifies the fundamental unit of temperature as degrees Celsius, Fahrenheit, or Kelvin. The form FAR and CEL, defined by IEEE 488.2, shall be accepted for Fahrenheit and Celsius respectively, and these shall be the preferred forms that are returned for a query.

At \*RST, this value is device-dependent.

### 25.4 :TIME HOUR|MINute|SECond

UNIT:TIME

Specifies the fundamental unit of time.

At \*RST, this value is SECond.

## 26 VXI Subsystem

The VXI subsystem contains commands that control the administration functions associated with operating a VXI-based system. This section will track the work of the VXI consortium.

This section describes ASCII commands which are issued to the system from an external host. The exact internal destination device and routing methods to that device are system specific. These commands are used for access to system configuration information, and for common capabilities. The terminology used in this section is patterned after IEEE 488.2.

The following commands are the standard set of VXIbus system commands which are ASCII encoded. If a device implements one or more of these commands, the syntax of this section shall be used. The response syntax shall also follow this section.

The COMMON ASCII SYSTEM COMMANDS are organized into subsystems. If a device implements a required command in a subsystem then it shall implement all the required commands in the subsystem. If a device implements an optional command in a subsystem then it shall implement all the required commands in the subsystem. If a device implements a command from any subsystem then it shall also implement the VXI:SElect command.

KEYWORD	PARAMETER FORM	NOTES
VXI		
:CONFigure		
:DNUMber?		[query only]
:HIERarchy?		[query only] 1992
:ALL?		[query only] 1992
:VERBose?		[query only] 1992
:ALL?		[query only] 1992
:INFormation?		[query only] 1992
:ALL?		[query only] 1992
:VERBose?		[query only] 1992
:ALL?		[query only] 1992
:LADDress?		[query only] 1992
:NUMber?		[query only] 1992
:REGister		1992
:READ?	<numeric_value> <reg_name>	[query only] 1992
:VERBose?	<numeric_value> <reg_name>	[query only] 1992
:WRIte	(<numeric_value> <reg_name>),<data>	[no query] 1992
:RESET?		[query only] 1992
:VERBose?		[query only] 1992
:SELECT	<numeric_value>	1992
:WSProtocol		1992
:COMMand		1992
[:ANY]	<numeric_value>	[no query] 1992
:AHILine	<numeric_value>,<numeric_value>	[no query] 1992
:AILINE	<numeric_value>,<numeric_value>	[no query] 1992

## 1999 SCPI Command Reference

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:AMControl	<numeric_value>	[no query] 1992
:ANO		[no query] 1992
:BAVAILABLE	<Boolean>,<numeric_value>	[no query] 1992
:BNO	<Boolean>	[no query] 1992
:BRQ		[no query] 1992
:CEVENT	<Boolean>,<numeric_value>	[no query] 1992
:CLR		[no query] 1992
:CLOCK		[no query] 1992
:CRESPONSE	<numeric_value>	[no query] 1992
:ENO		[no query] 1992
:GDEVice	<numeric_value>	[no query] 1992
:ICOMmander	<numeric_value>	[no query] 1992
:RDEVice	<numeric_value>	[no query] 1992
:RHANDlers		[no query] 1992
:RHLLine	<numeric_value>	[no query] 1992
:RILLine	<numeric_value>	[no query] 1992
:RINTerrupter		[no query] 1992
:RMODid		[no query] 1992
:RPERror		[no query] 1992
:RPRotocol		[no query] 1992
:RSTB		[no query] 1992
:RSARea		[no query] 1992
:SLModid	<Boolean>,<numeric_value>	[no query] 1992
:SLOCK		[no query] 1992
:SUModid	<Boolean>,<numeric_value>	[no query] 1992
:TRIGGER		[no query] 1992
:MESSAGE		1992
:RECEIVE?	<numeric_value> <terminator>	[query only] 1992
:SEND	<message_string> [,,(END NEND)]	[no query] 1992
:QUERy		1992
[:ANY?]	<numeric_value>	[query only] 1992
:AHLLine?	<numeric_value>,<numeric_value>	[query only] 1992
:AIIline?	<numeric_value>,<numeric_value>	[query only] 1992
:AMControl?	<numeric_value>	[query only] 1992
:ANO?		[query only] 1992
:BNO?	<Boolean>	[query only] 1992
:BRQ?		[query only] 1992
:CEVENT?	<Boolean>,<numeric_value>	[query only] 1992
:CRESPONSE?	<numeric_value>	[query only] 1992
:ENO?		[query only] 1992
:RDEVice?	<numeric_value>	[query only] 1992
:RHANDlers?		[query only] 1992
:RHLLine?	<numeric_value>	[query only] 1992
:RILLine?	<numeric_value>	[query only] 1992

KEYWORD	PARAMETER FORM	NOTES
:RINTerrupter?	[query only]	1992
:RMODid?	[query only]	1992
:RPERror?	[query only]	1992
:RProtocol?	[query only]	1992
:RSTB?	[query only]	1992
:RSARea?	[query only]	1992
:SLModid? <Boolean>,<numeric_value>	[query only]	1992
:SUModid? <Boolean>,<numeric_value>	[query only]	1992
:RESPonse?	[query only]	1992

## 26.1 :CONFigure

VXI:CONFigure

Provides the necessary commands to query the configuration of a VXI system.

### 26.1.1 :DNUMber?

VXI:CONFigure:DNUMber?

This query returns the number of devices in the system, in <NR1 numeric value> format. The range of this integer numeric value is 1 to 256 inclusive.

### 26.1.2 :HIERarchy?

VXI:CONFigure:HIERarchy?

This command returns current hierarchy configuration information about the selected logical address. The individual fields of the response are comma separated. If the information about the selected logical address is not available from the destination device, (that is, the requested device is not in the servant area of the destination device), then error -224, "Parameter error" will be set and no response data will be sent. This is a required configuration command. The output fields are defined as follows:

- Logical address: a <NR1>, between -1 and 255 inclusive. A -1 indicates that the device has no logical address.
- commander's logical address: a <NR1>, between -1 and 255 inclusive. -1 indicates that this device has no commander or the commander is unknown.
- Interrupt Handlers: a comma separated list of 7 <NR1>, between 0 and 7 inclusive. Interrupt lines 1-7 are mapped to the individual return values. 0 is used to indicate that the particular interrupt handler is not configured. A set of return values of 0,0,0,5,2,0,6 would indicate that the device is configured as follows:
  - handler 4 is configured to handle interrupts on line 5
  - handler 5 is configured to handle interrupts on line 2
  - handler 7 is configured to handle interrupts on line 6
  - handlers 1,2,3,6 are not configured
- Interrupters: a comma separated list of 7 <NR1>, between 0 and 7 inclusive. Interrupt lines 1-7 are mapped to the individual return values. 0 is used to indicate that the particular interrupter is not configured. A set of return values of 0,0,0,5,2,0,6 would indicate that the device is configured as follows:

## 1999 SCPI Command Reference

interrupter 4 is configured to interrupt on line 5  
interrupter 5 is configured to interrupt on line 2  
interrupter 7 is configured to interrupt on line 6  
interrupters 1,2,3,6 are not configured

- Pass/Failed: a <NR1> which contains the pass/fail status of the specified device encoded as follows:  
FAIL=0 IFAIL=1 PASS=2 READY=3
- manufacturer specific comment: This up to 80 character quoted string contains manufacturer specific data. It is sent with a 488.2 string response data format.

### 26.1.2.1 :ALL?

VXI:CONFigure:HIERarchy?:ALL?

When issued to a resource manager, configuration information about all logical addresses. If the command is received by a device which is not the resource manager, it returns the current hierarchy configuration information about the destination device followed by all of its immediate servants. The information is returned in the order specified in VXI:CONFigure:LADDress?. (the information about multiple logical addresses will be semi-colon separated and follow the IEEE 488.2 response message format). The individual fields of the output are comma separated. This is an optional configuration command.

### 26.1.2.2 :VERBose?

VXI:CONFigure:HIERarchy?:VERBose?

This command has the same functionality as VXI:CONFigure:HIERarchy?. However it returns a quoted string indicating the device information for the selected This command is intended to allow for a human readable form of the response to VXI:CONFigure:HIERarchy?. The format of this string is manufacturer specific. This is an optional configuration command.

### 26.1.2.2.1 :ALL?

VXI:CONFigure:HIERarchy?:VERBose?:ALL?

This command has the same functionality as VXI:CONFigure:HIERarchy:ALL?. However it returns a semi- colon separated sequence of strings indicating the device information for all devices. The format of this string is manufacturer specific. This command is intended to allow for a human readable form of the response to VXI:CONFigure:HIERarchy:ALL?. This is an optional configuration command.

### 26.1.3 :INFormation?

VXI:CONFigure:INFormation?

This command returns the static information about the selected logical address. The individual fields of the response are comma separated. If the information about the selected logical address is not available from the destination device, (that is, the requested device is not in the servant area of the destination device), then error -224, “Parameter error” will be set and no response data will be sent. This is a required configuration command. The command returns the following values:

- logical address: a <NR1> between -1 and 255 inclusive. A -1 indicates that the

device has no logical address.

- manufacturer id: a <NR1>, between -1 and 4095 inclusive. -1 indicates that the device has no manufacturer id.
- model code: a <NR1>, between -1 and 65535 inclusive. -1 indicates that the device has no model code
- device class: a <NR1>, between 0 and 5 inclusive. Each value corresponds to the device's classification as follows:
  - 0: VXIbus memory device.
  - 1: VXIbus extended device.
  - 2: VXIbus message based device.
  - 3: VXIbus register based device.
  - 4: Hybrid device.
  - 5: Non-VXIbus device.
- address space: a <NR1>, between 0 and 15 inclusive, which is the sum of the binary weighted codes of the address space(s) occupied by the device. The following values are defined:
  - 1: The device has A16 registers.
  - 2: The device has A24 registers.
  - 4: The device has A32 registers.
  - 8: The device has A64 registers.
- A16 memory offset: the base address for any A16 registers (other than the VXIbus defined registers) which are present on the device. This is a <NR1> in the range of -1 to 65535. -1 is used to indicate that the device has no A16 memory.
- A24 memory offset: the base address for any A24 registers which are present on the device. This is a <NR1> in the range of -1 to 16777215. -1 is used to indicate that the device has no A24 memory.
- A32 memory offset: the base address for any A32 registers which are present on the device. This is a <NR1> value in the range of -1 to 4294967295. -1 is used to indicate that the device has no A32 memory.
- A16 memory size: the number of bytes reserved for any A16 registers (other than the VXIbus defined registers) which are present on the device. This is a <NR1> in the range of -1 to 65535. -1 is used to indicate that the device has no A16 memory.
- A24 memory size: the number of bytes reserved for any A24 registers which are present on the device. This is a <NR1> in the range of -1 to 16777215. -1 is used to indicate that the device has no A24 memory.
- A32 memory size: the number of bytes reserved for any A32 registers which are present on the device. This is a <NR1> in the range of -1 to 4294967295. -1 is used to indicate that the device has no A32 memory.
- Slot number: a <NR1>, between -1 and the number of slots which exist in the

## 1999 SCPI Command Reference

cage. -1 indicates that the slot which contains this device is unknown.

- Slot 0's logical address: a <NR1>, between -1 and 255 inclusive. -1 indicates that Slot 0 device associated with this device is unknown.
- subclass: a <NR1> representing the contents of the subclass register. -1 indicates that the subclass register is not defined for this device.
- attribute: a <NR1> representing the contents of the attribute register. -1 indicates that the attribute register is not defined for this device.
- manufacturer specific comment: This up to 80 character quoted string contains manufacturer specific data. It is sent with a 488.2 string response data format.

### 26.1.3.1 :ALL?

VXI:CONFigure:INFormation?:ALL?

When issued to a resource manager, this command returns the static information about all logical addresses. If the command is received by a device which is not the resource manager, it returns the static information about the destination device followed by all of its immediate servants. The information is returned in the order specified in the response to VXI:CONFigure:LADDress?. (The information about multiple logical addresses will be semi-colon separated and follow the IEEE 488.2 response message format). The individual fields of the output are comma separated. This is an optional configuration command.

### 26.1.3.2 :VERBose?

VXI:CONFigure:INFormation?:VERBose?

This command has the same functionality as VXI:CONFigure:INFormation?. However it returns a quoted string indicating the device information for the selected device. The format of this string is manufacturer specific. This command is intended to allow for a human readable form of the response to VXI:CONFigure:INFormation?. This is an optional configuration command.

### 26.1.3.3 :ALL?

VXI:CONFigure:INFormation?:ALL?

This command has the same functionality as VXI:CONFigure:INFormation:ALL?. However it returns a semi-colon separated sequence of quoted strings indicating the device information for all devices. The format of each string is manufacturer specific. This command is intended to allow for a human readable form of the response to VXI:CONFigure:INFormation:ALL?. The format of this string is manufacturer specific. This is an optional configuration command.

### 26.1.4 :LADDress?

VXI:CONFigure:LADDress?

When issued to a resource manager, the Logical ADDdress command returns a comma separated list of <NR1> which are the logical addresses of the devices in the system. Each <NR1> will be an integer in the range of 0 to 255 inclusive. The logical address of the device which is responding to the command will be the first item in the list. If the command is received by a device other than the resource manager, then the response to the command will contain the logical address of the destination device followed by a list of the devices

which are immediate servants to the destination device. This is a required configuration command.

### 26.1.5 :NUMBer?

VXI:CONFigure:NUMBER?

When issued to a resource manager, this command returns a <NR1> which is the number of devices in the system. If the command is received by a device which is not the resource manager, then the response to the command will contain the number of devices which are immediate servants to the destination device including the destination device. For example, a commander with 3 servants would return a value of 4, or a resource manager for a system of 4 devices would return a value of 5. The range of this <NR1> is 1 to 256 inclusive. This is a required configuration command.

## 26.2 REGister

VXI:REGister

The node collects together the commands that interact with the VXI registers.

### 26.2.1 :READ? <register>

VXI:REGister:READ?

This command returns the contents of the specified 16 bit register at the selected logical address as a <NR1>. This is a required register access command. The register is specified as the byte address of the desired register or as optionally the register name. It has values of all even numbers from 0 to 62 inclusive (as a <numeric\_value>) or the following (optional) words:

A24Low: A24 Pointer Low register (18)

A24High: A24 Pointer High register (16)

A32Low: A32 Pointer Low register (22)

A32High: A32 Pointer High register (20)

ATTRibute: Attribute register (8)

DHIGH: Data High register (12)

DLOW: Data Low register (14)

DTYPe: Device Type register (2)

ICONtrol: Interrupt Control register (28)

ID: ID register (0)

ISTatus: Interrupt Status register (26)

MODId: MODID register (8)

OFFSet: Offset register (6)

PROTocol: PROTocol register (8)

RESPonse: RESPonse register (10)

## 1999 SCPI Command Reference

SNLow: Serial Number Low register (12)

SNHigh: Serial Number High register (10)

STATus: Status register (4)

SUBClass: Subclass register (30)

VNUMber: Version Number register (14)

### 26.2.1.1 :VERBose? <register>

VXI:REGister:READ?:VERBose?

This command has the same functionality as VXI:REGister:READ?. However it returns a quoted string indicating the register contents for the selected device. The format of this string is manufacturer specific. This command is intended to allow for a human readable form of the response to VXI:REG:READ? which could include additional decoding of bits in specific registers. This is an optional register access command.

### 26.2.2 :WRITe (<numeric\_value> | <register>), <data>

VXI:REGister:WRITE

This command writes data to the specified register on the selected logical address. This is a required register access command. The Data is a 16 bit value specified as a <numeric\_value> in the range of -32768 to 32767. The register is specified as the byte address of the desired register or as optionally the register name. It has values of all even numbers from 0 to 62 inclusive (expressed as a <numeric\_value>) or the following (optional) words:

CONTrol: Control register (4)

DEXTended: Data Extended register (10)

DHIGH: Data High register (12)

DLOW: Data Low register (14)

ICONtrol: Interrupt Control register (28)

MODid: MODID register (8)

LADDress: Logical Address register (0)

OFFSet: Offset register (6)

SIGNAL: Signal register (8)

### 26.3 :RESet?

VXI:RESET?

This command resets the selected logical address. SYSFAIL generation is inhibited while the device is in the self test state. The command waits for five seconds or until the selected device has indicated passed (whichever occurs first). If the device passes its self test, then SYSFAIL generation will be re-enabled. If the device fails its self test, then SYSFAIL generation will remain inhibited. The return value from this command is the state of the selected device after it has been reset. The command returns an <NR1> which is encoded as

follows: FAIL=0, PASS=2, READY=3. The state of the A24/A32 enable bit is not altered by this command. This is a required register access command.

### 26.3.1 :VERBose?

VXI:RESet?:VERBose?

This command has the same functionality as VXI:RESet?. However it returns a quoted string indicating the state of the specified device. The format of this string is manufacturer specific. This is an optional register access command.

### 26.4 :SElect <logical\_address>

VXI:SElect

This command specifies the logical address which is to be used by all subsequent commands in the VXI subsystem. Logical\_address is specified as a <numeric\_value> in the range of 0 to 255. The query form returns the logical address as a <NR1> the value of -1 is reserved to indicate that no logical address has been selected. The \*RST default value for logical\_address is that no logical address is selected. All other commands which require a logical\_address to be selected will respond with an error -221, “Settings conflict” if no logical address is selected.

### 26.5 :WSPRrotocol

VXI:WSPRrotocol

This node collects together all the commands associated with the word serial protocol.

#### 26.5.1 :COMMAND

##### 26.5.1.1 [:ANY] <data>

VXI:WSPRrotocol:COMMAND:ANY

This command sends the specified word serial command to the selected logical address. The data field is a <numeric\_value> or a non-decimal value which specifies the command to be sent. The response of a word serial query which is sent with this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is a required Word Serial Support command.

##### 26.5.1.2 :AHLIne <hand\_id>,<line\_number>

VXI:WSPRrotocol:COMMAND:AHLIne

This command sends an Assign Handler Line command to the selected logical address. The hand\_id field has a value in the range of 1 to 7 inclusive, and line\_number field has a value in the range of 0 to 7 inclusive. They are both specified with <numeric\_value>s or as non-decimal values. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

##### 26.5.1.3 :AIIne <int\_id>,<line\_number>

VXI:WSPRrotocol:COMMAND:AIIne

This command sends an Assign Interrupter Line command to the selected logical address. The int\_id field has a value in the range of 1 to 7 inclusive, and line\_number field has a value in the range of 0 to 7 inclusive. They are both specified with numeric\_ values or as non-decimal values. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.4 :AMControl <response\_mask>

VXI:WSPRrotocol:COMMAND:AMControl

This command sends an Asynchronous Mode Control command to the selected logical address. The response mask field has a value in the range of 0 to 15 inclusive, and is defined in VXI rev. 1.3. It is specified with a <numeric\_value> or as a non-decimal value. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.5 :ANO

VXI:WSPRrotocol:COMMAND:ANO

This command sends an Abort Normal Operation command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.6 :BAvailable <Boolean>,<byte>

VXI:WSPRrotocol:COMMAND:BAvailable

This command sends a Byte Available command to the selected logical address. The Boolean field selects whether the END bit is set in the command. The byte field has a value in the range of 0 to 255 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. This is an optional Word Serial Support command.

### 26.5.1.7 :BNO <Boolean>

VXI:WSPRrotocol:COMMAND:BNO

This command sends an Begin Normal Operation command to the selected logical address. The Boolean field selects whether the Top\_level bit is set in the command. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.8 :BRQ

VXI:WSPRrotocol:COMMAND:BRQ

This command sends an Byte Request command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.9 :CEvent <Boolean>,<event\_number>

VXI:WSPRrotocol:COMMAND:CEvent

This command sends a Control Event command to the selected logical address. The Boolean field selects whether the Enable bit is set in the command. The event\_number field has a value in the range of 0 to 127 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.10 :CLR

VXI:WSPRrotocol:COMMAND:CLR

This command sends an Clear command to the selected logical address. This is an optional Word Serial Support command.

**26.5.1.11 :CLOCK**

VXI:WSPRrotocol:COMMAND:CLOCK

This command sends an Clear Lock command to the selected logical address. This is an optional Word Serial Support command.

**26.5.1.12 :CRESpone <response\_mask>**

VXI:WSPRrotocol:COMMAND:CRESpone

This command sends a Control Response command to the selected logical address. The response\_mask field has a value in the range of 0 to 127 inclusive, and is defined in VXI rev. 1.3. It is specified with a <numeric\_value> or as a non-decimal value. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

**26.5.1.13 :ENO**

VXI:WSPRrotocol:COMMAND:ENO

This command sends an End Normal Operation command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

**26.5.1.14 :GDEvice <logical\_address>**

VXI:WSPRrotocol:COMMAND:GDEvice

This command sends a Grant Device command to the selected logical address. The logical\_address field has a value in the range of 0 to 255 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. This is an optional Word Serial Support command.

**26.5.1.15 :ICOMmander <logical\_address>**

VXI:WSPRrotocol:COMMAND:ICOMmander

This command sends a Identify Commander command to the selected logical address. The logical\_address field has a value in the range of 0 to 255 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. This is an optional Word Serial Support command.

**26.5.1.16 :RDEvice <logical\_address>**

VXI:WSPRrotocol:COMMAND:RDEvice

This command sends a Release Device command to the selected logical address. The logical\_address field has a value in the range of 0 to 255 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

**26.5.1.17 :RHAndlers**

VXI:WSPRrotocol:COMMAND:RHAndlers

This command sends a Read Handlers command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

## 1999 SCPI Command Reference

### 26.5.1.18 :RHLIne <hand\_id>

VXI:WSPRrotocol:COMMAND:RHLIne

This command sends a Read Handler Line command to the selected logical address. The hand\_id field has a value in the range of 1 to 7 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.19 :RILIne <int\_id>

VXI:WSPRrotocol:COMMAND:RILIne

This command sends a Read Interrupter Line command to the selected logical address. The int\_id field has a value in the range of 1 to 7 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.20 :RINTerrupter

VXI:WSPRrotocol:COMMAND:RINTerrupter

This command sends a Read Interrupters command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.21 :RMODId

VXI:WSPRrotocol:COMMAND:RMODId

This command sends a Read MODId command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.22 :RPERror

VXI:WSPRrotocol:COMMAND:RPERrror

This command sends a Read Protocol Error command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.23 :RPRotocol

VXI:WSPRrotocol:COMMAND:RPRotocol

This command sends a Read Protocol command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

### 26.5.1.24 :RSTB

VXI:WSPRrotocol:COMMAND:RSTB

This command sends a Read Status Byte command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

**26.5.1.25 :RSARea**

VXI:WSPRrotocol:COMMAND:RSARea

This command sends a Read Servant Area command to the selected logical address. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

**26.5.1.26 :SLModid <Boolean>,<MODID 6-0>**

VXI:WSPRrotocol:COMMAND:SLModid

This command sends a Set Lower MODid command to the selected logical address. The Boolean field selects whether the Enable bit is set in the command. The MODID 6-0 field has a value in the range of 0 to 127 inclusive. It is specified with a <NR1> or as a non-decimal value. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

**26.5.1.27 :SLOCK**

VXI:WSPRrotocol:COMMAND:SLOCK

This command sends a Set Lock command to the selected logical address. This is an optional Word Serial Support command.

**26.5.1.28 :SUModid <Boolean>,<MODID 12-7>**

VXI:WSPRrotocol:COMMAND:SUModid

This command sends a Set Upper MODid command to the selected logical address. The Boolean field selects whether the Enable bit is set in the command. The MODID 12-7 field has a value in the range of 0 to 63 inclusive. It is specified with a <NR1> or as a non-decimal value. The response to this command can be read with the VXI:WSPRrotocol:RESPonse? command. This is an optional Word Serial Support command.

**26.5.1.29 :TRIGger**

VXI:WSPRrotocol:COMMAND:TRIGger

This command sends a Trigger command to the selected logical address. This is an optional Word Serial Support command.

**26.5.2 :MESSage**

VXI:WSPRrotocol:MESSAge

This node collects together all the messages (as defined by VXI) of the word serial protocol.

**26.5.2.1 :RECeive? <count>|<terminator>**

VXI:WSPRrotocol:MESSAge:RECeive?

This command receives a message from the selected logical address using both the word serial protocol and the byte transfer protocol. The command will always terminate on the End bit being set. Additional termination options are on a specified number of bytes (count), or on a match to a particular terminator, (that is, LF, CRLF, END). The response is returned as a string. This is a required Word Serial Support command.

**26.5.2.2 :SEND <message\_string> [,END|NEND]**

VXI:WSPRrotocol:MESSAge:SEND

This command sends the specified message string to the selected logical address. The string is sent using the word serial protocol with the byte transfer protocol. The last byte of the

string is sent with the End bit set unless NEND is specified. This is a required Word Serial Support command.

### 26.5.3 :QUERy

VXI:WSPRrotocol:QUERy

This node collects together all the queries (as defined by VXI) of the word serial protocol.

#### 26.5.3.1 [:ANY]? <data>

VXI:WSPRrotocol:QUERy:ANY?

This command sends the specified word serial query to the selected logical address. The data field is a <numeric\_value> or a non-decimal value which specifies the query to be sent. The returned value is the response to the word serial query and is a <NR1>. This command obeys the byte transfer protocol. This is a required Word Serial Support command.

#### 26.5.3.2 :AHLIne? <hand\_id>,<line\_number>

VXI:WSPRrotocol:QUERy:AHLIne?

This command sends an Assign Handler Line command to the selected logical address. The hand\_id field has a value in the range of 1 to 7 inclusive, and line\_number field has a value in the range of 0 to 7 inclusive. They are both specified with <numeric\_value>s or as non-decimal values. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

#### 26.5.3.3 :AIIne? <int\_id>,<line\_number>

VXI:WSPRrotocol:QUERy:AIIne?

This command sends an Assign Interrupter Line command to the selected logical address. The int\_id field has a value in the range of 1 to 7 inclusive, and line\_number field has a value in the range of 0 to 7 inclusive. They are both specified with <numeric\_value>s or as non-decimal values. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

#### 26.5.3.4 :AMControl? <response\_mask>

VXI:WSPRrotocol:QUERy:AMControl?

This command sends an Asynchronous Mode Control command to the selected logical address. The response\_mask field has a value in the range of 0 to 15 inclusive, and is defined in VXI rev. 1.3. It is specified with a <numeric\_value> or as a non-decimal value. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

#### 26.5.3.5 :ANO?

VXI:WSPRrotocol:QUERy:ANO?

This command sends an Abort Normal Operation command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

#### 26.5.3.6 :BNO? <Boolean>

VXI:WSPRrotocol:QUERy:BNO?

This command sends an Begin Normal Operation command to the selected logical address. The Boolean field selects whether the Top\_level bit is set in the command. The returned

value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.7 :BRQ?

VXI:WSPRrotocol:QUERy:BRQ?

This command sends an Byte Request command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.8 :CEVent? <Boolean>,<event\_number>

VXI:WSPRrotocol:QUERy:CEVent?

This command sends a Control Event command to the selected logical address. The Boolean field selects whether the Enable bit is set in the command. The event\_number field has a value in the range of 0 to 127 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.9 :CREsponse? <response\_mask>

VXI:WSPRrotocol:QUERy:CREsponse?

This command sends a Control Response command to the selected logical address. The response\_mask field has a value in the range of 0 to 127 inclusive, and is defined in VXI rev. 1.3. It is specified with a numeric\_value or as a non-decimal value. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.10 :ENO?

VXI:WSPRrotocol:QUERy:ENO?

This command sends an End Normal Operation command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.11 :RDEvice? <logical\_address>

VXI:WSPRrotocol:QUERy:RDEvice?

This command sends a Release Device command to the selected logical address. The logical\_address field has a value in the range of 0 to 255 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.12 :RHANDlers?

VXI:WSPRrotocol:QUERy:RHANDlers?

This command sends a Read Handlers command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.13 :RHLine? <hand\_id>

VXI:WSPRrotocol:QUERy:RHLine?

This command sends a Read Handler Line command to the selected logical address. The hand\_id field has a value in the range of 1 to 7 inclusive. It is specified with a

## 1999 SCPI Command Reference

<numeric\_value> or as a non-decimal value. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.14 :RILine? <int\_id>

VXI:WSPRrotocol:QUERy:RILine?

This command sends a Read Interrupter Line command to the selected logical address. The int\_id field has a value in the range of 1 to 7 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.15 :RINTerrupter?

VXI:WSPRrotocol:QUERy:RINTerrupter?

This command sends a Read Interrupters command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.16 :RMODid?

VXI:WSPRrotocol:QUERy:RMODid?

This command sends a Read MODid command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.17 :RPERror?

VXI:WSPRrotocol:QUERy:RPERrror?

This command sends a Read Protocol Error command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.18 :RPRotocol?

VXI:WSPRrotocol:QUERy:RPRotocol?

This command sends a Read Protocol command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.19 :RSTB?

VXI:WSPRrotocol:QUERy:RSTB?

This command sends a Read Status Byte command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.20 :RSARea?

VXI:WSPRrotocol:QUERy:RSARea?

This command sends a Read Servant Area command to the selected logical address. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.21 :SLModid? <Boolean>,<MODID 6-0>

VXI:WSPRrotocol:QUERy:SLModid?

This command sends a Set Lower MODid command to the selected logical address. The Boolean field selects whether the Enable bit is set in the command. The MODID 6-0 field has a value in the range of 0 to 127 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.3.22 :SUModid? <Boolean>,<MODID 12-7>

VXI:WSPRrotocol:QUERy:SUModid?

This command sends a Set Upper MODid command to the selected logical address. The Boolean field selects whether the Enable bit is set in the command. The MODID 12-7 field has a value in the range of 0 to 63 inclusive. It is specified with a <numeric\_value> or as a non-decimal value. The returned value is the response to the command and is a <NR1>. This is an optional Word Serial Support command.

### 26.5.4 :RESPonse?

VXI:WSPRrotocol:RESPonse?

This command returns one word of data from the data low register on the selected logical address. This command obeys the Byte transfer protocol. The data is returned as a <NR1>. This is a required Word Serial Support command.

## **1999 SCPI Command Reference**

# Index

---

## A

ABORt 2-5  
  HCOPY 10-4  
  TRIGger Subsystem 24-12  
:AC  
  INPut:BIAS:CURREnt 11-4  
  INPut:BIAS:VOLTage 11-4  
  MEASure 3-9  
  SENSe:CURREnt 18-25  
  SENSe:POWER 18-64  
  SENSe:VOLTage 18-84  
ACCeleration  
  SENSe:FUNCtion 18-50  
  SOURce 19-5  
ACCeleration Subsystem  
  SOURce 19-5  
:ACHannel  
  SENSe:POWER 18-63  
:ACQUire  
  SOURce:CORRection:COLLect 19-12  
:ACQuire STANDARD  
  SENSe:CORRection:COLLect 18-17  
:ADDResS  
  SYSTem:COMMUnicatE:GPIB 21-5  
  SYSTem:COMMUnicatE:GPIB:RDEVice 21-5  
:ADJust  
  SOURce:PHASE 19-56  
:AFC  
  SENSe 18-38  
:AHLine  
  VXI:WSPRotocol:COMMAND 26-9  
:AHLine?  
  VXI:WSPRotocol:QUERy 26-14  
:AILine  
  VXI:WSPRotocol:COMMAND 26-9  
:AILine?  
  VXI:WSPRotocol:QUERy 26-14  
:ALC  
  SOURce:CURREnt 19-18  
  SOURce:POWER 19-61  
  SOURce:VOLTage 19-94  
:ALL  
  CALCulate:MATH:EXPRESSion:DElete 4-22  
  CALibration 5-2  
  DISPlay:ANNotatIon 8-4  
  HCOPY:ITEM 10-8

MEMory:DELetE 13-7  
PROGram:SElected:DELetE 16-2  
ROUTe:MODule:DELetE 17-2  
ROUTe:OPEN 17-3  
ROUTe:PATH:DELetE 17-4  
SENSe:FUNCtion:OFF 18-44  
SENSe:FUNCtion:ON 18-45  
SOURce:DM:THReshold 19-29  
TRACe | DATA:DElete 23-4  
TRIGger Subsystem:INITiate:CONTinuous 24-22  
TRIGger Subsystem:INITiate:IMMEDIATE 24-22  
:ALL AC  
  SOURce:DM:COUpling 19-29  
:ALL DC  
  SOURce:DM:COUpling 19-29  
:ALL GROund  
  SOURce:DM:COUpling 19-29  
:ALL?  
  CALibration 5-2  
  MEMory:CATalog 13-5  
  MEMory:FREE 13-8  
  VXI:CONFigure:HIERarchy? 26-4  
  VXI:CONFigure:HIERarchy?:VERBose? 26-4  
  VXI:CONFigure:INFormation? 26-6  
:ALTernate  
  SYSTem 21-3  
AM  
  SENSe:FUNCtion 18-50  
  SOURce 19-6 - 19-9  
  SOURce:LIST 19-49  
AM Subsystem  
  SENSe 18-4 - 18-5  
  SOURce 19-6 - 19-9  
:AMControl  
  VXI:WSPRotocol:COMMAND 26-10  
:AMControl?  
  VXI:WSPRotocol:QUERy 26-14  
Amorphous SENSe Model  
  Instrument Model:Internal Routing 2-8  
:AMPLitude  
  DISPlay:ANNotatIon 8-4  
  MEASure 3-11  
  SOURce:CURREnt:LEVel:IMMEDIATE 19-20  
  SOURce:CURREnt:LEVel:TRIGgered 19-21  
  SOURce:CURREnt:LIMit 19-21  
  SOURce:MARKer 19-54  
  SOURce:POWER:LEVel:IMMEDIATE 19-63

## 1999 SCPI Command Reference

SOURce:POWER:LEVel:TRIGgered 19-64	:ASCii?
SOURce:POWER:LIMit 19-64	MEMORY:CATalog 13-5
SOURce:RESistance:LEVel:IMMEDIATE 19-75	MEMORY:FREE 13-8
SOURce:RESistance:LEVel:TRIGgered 19-76	:ATRigger TRIGGER Subsystem:TRIGger:SEQuence 24-23
SOURce:RESistance:LIMit 19-76	:ATTenuation OUTPUT Subsystem 15-3
SOURce:VOLTage:LEVel:IMMEDIATE 19-96	:ATTenuation
SOURce:VOLTage:LEVel:TRIGgered 19-97	INPut 11-3
SOURce:VOLTage:LIMit 19-97	SENSe:CURRent:AC DC 18-26
:ANGLE	SENSe:POWER:AC DC 18-64
INPut:POSITION:X 11-8	SENSe:VOLTage:AC DC 18-85
INPut:POSITION:Y 11-10	SOURce:CURRent 19-18
INPut:POSITION:Z 11-12	SOURce:POWER 19-61
OUTPut:POSITION:X 15-6	SOURce:VOLTage 19-94
OUTPut:POSITION:Y 15-8	:ATTRibutes
OUTPut:POSITION:Z 15-10	DISPlay:WINDOW:TEXT 8-9
SOURce:DM:LEAKage 19-28	:AUTO
SOURce:DM:QUADrature 19-29	CALCulate 4-7, 4-12, 4-14, 4-18, 4-21, 4-25 - 4-26, 4-28, 4-31
UNIT 25-1	CALibration 5-2, 5-7
:ANNotation	DISPlay 8-11 - 8-12, 8-14 - 8-15
DISPlay 8-4	HCOPy 10-13
HCOPy:ITEM 10-8	INPut:ATTenuation 11-3
:ANO	INPut:GAIN 11-6
VXI:WSPProtocol:COMMAND 26-10	OUTPut:FILTer 15-3
:ANO?	SENSe 18-4, 18-7, 18-10 - 18-11, 18-20, 18-26 - 18-28, 18-31, 18-36, 18-38 - 18-40, 18-59 - 18-62, 18-64 - 18-67, 18-69 - 18-70, 18-72, 18-79, 18-82, 18-85 - 18-87
VXI:WSPProtocol:QUERy 26-14	SENSe:ROSCillator:SOURce INTERNAL 19-80
:ANY	SOURce 19-18 - 19-20, 19-23, 19-43, 19-45, 19-50, 19-52, 19-61 - 19-62, 19-66, 19-73, 19-79, 19-85 - 19-86, 19-94 - 19-96, 19-99
VXI:WSPProtocol:COMMAND 26-9	SYSTem 21-9, 21-33
:ANY?	TRACe   DATA 23-5
VXI:WSPProtocol:QUERy 26-14	TRIGGER Subsystem:ARM:SEQuence
:AOFF	:LAYER:DELay 24-14
SOURce:MARKer 19-54	TRIGGER Subsystem:ARM:SEQuence
:APERture	:LAYER:LEVel 24-16
CALCulate:GDAperture 4-17	TRIGGER Subsystem:TRIGger:SEQuence
CALCulate:SMOothing 4-23	:DELAY 24-26
SENSe:CURRent:AC DC 18-25	TRIGGER Subsystem:TRIGger:SEQuence
SENSe:FREQuency 18-37	:LEVEL 24-28
SENSe:POWER:AC DC 18-64	:AVERage Subsystem
SENSe:RESistance 18-68	CALCulate 4-6 - 4-8
SENSe:SMOothing 18-73	SENSe 18-6 - 18-9
SENSe:VOLTage:AC DC 18-84	:AWEighting
:APower	INPut:FILTter 11-5
CONTrol 6-1	SENSe:FILTter 18-35
SOURce:LIST:CONTrol 19-50	:AXIS
:APRobe	DISPlay:WINDOW:TRACe:GRATicule 8-11
SOURce:LIST 19-49	
ARM 2-5	
TRIGger Subsystem 24-12 - 24-20	
:ARM-TRIGger Model	
TRIGger Subsystem 24-1	
:ARRay	
MEASure 3-7	
ASCii 9-1, 9-3	

## 1999 SCPI Command Reference

### B

:BACKground  
    DISPlay:WINDOW 8-6  
:BANDwidth  
    SOURce:POWER:ALC 19-62  
:BANDwidth  
    SENSe:DETector 18-30  
    SOURce:CURRent:ALC 19-19  
    SOURce:VOLTage:ALC 19-95  
BANDwidth Subsystem  
    SENSe 18-10 - 18-11  
:BAUD  
    SYSTem:COMM:SERial:RECeive 21-7  
    SYSTem:COMM:SERial:TRANSmitt 21-9  
:BAvailable  
    VXI:WSPRotocol:COMMAND 26-10  
:BEEPer  
    SYSTem 21-3  
:BIAS  
    INPut 11-3  
    SENSe:MIXer 18-60  
BINary 9-2 - 9-3  
:BINary?  
    MEMORY:CATalog 13-5  
    MEMORY:FREE 13-8  
:BINertia  
    CALibration 5-2 - 5-3  
:BIT  
    STATus 20-3  
:BITS  
    SYSTem:COMM:SERial:RECeive 21-7  
    SYSTem:COMM:SERial:TRANSmitt 21-9  
:BLOWER  
    CONTrol 6-2  
    SOURce:LIST:CONTrol 19-50  
:BNO  
    VXI:WSPProtocol:COMMAND 26-10  
:BNO?  
    VXI:WSPProtocol:QUERy 26-14  
:BORDer  
    FORMAT 9-1  
:BOTTom  
    DISPlay:WINDOW:TRACe:Y:SCALE 8-14  
:BRAKe  
    CONTrol 6-2  
:BRIGHtness  
    DISPlay 8-4  
:BRQ  
    VXI:WSPProtocol:COMMAND 26-10  
:BRQ?  
    VXI:WSPProtocol:QUERy 26-15

### :BURSt

    :MEASure:FREQuency 3-9

### BWIDth

    SENSe:DETector 18-30

    SOURce:CURRent:ALC 19-19

    SOURce:POWER:ALC 19-62

    SOURce:VOLTage:ALC 19-95

### :BWIDth Subsystem

    SENSe 18-10 - 18-11

### C

### CALCulate

    Instrument Model:Measurement Function 2-4

    Instrument Model:Signal Generation 2-5

### CALCulate Subsystem

    CALIBration Subsystem 5-1 - 5-10

### :CAPability?

    SYSTem 21-4

### :CARM

    SENSe:FILTTer 18-35

### :CATalog

    CALCulate:MATH:EXPReSSion 4-21

    MEMORY 13-4

    SYSTem:KEY 21-32

### :CATalog?

    MMEMory 14-2

    PROGram 16-2

    ROUTe:MODule 17-2

    ROUTe:PATH 17-3

    TRACe | DATA 23-2

### :CCIR

    SENSe:FILTTer 18-35

### :CCITt

    SENSe:FILTTer 18-34

### :CDFunction

    SENSe:FUNCTION:VOLTage 18-56

### :CDIRectory

    MMEMory 14-3

### :CENTer

    CALCulate:FILTTer:GATE:FREQuency 4-14

    CALCulate:FILTTer:GATE:TIME 4-12

    CALCulate:TRANSform:DISTance 4-28

    CALCulate:TRANSform:FREQuency 4-30

    CALCulate:TRANSform:TIME 4-26

    DISPlay:WINDOW:TRACe:X:SCALE 8-12

    SENSe:FREQuency 18-37

    SOURce:CURRent 19-19

    SOURce:FREQuency 19-43

    SOURce:POWER 19-62

    SOURce:RESistance 19-77

    SOURce:VOLTage 19-95

## 1999 SCPI Command Reference

:CENTronics	SENSe:FUNCtion:POWer 18-54
SYSTem:COMMUnicatE 21-4	
:CEVent	:COLLect
VXI:WSPRotocol:COMMAND 26-10	SENSe:CORRection 18-17
:CEVent?	SOURce:CORRection 19-12
VXI:WSPRotocol:QUERy 26-15	:COLOR
:CHECk	DISPlay:CMAP 8-4
SYSTem:COMM:SERial:RECeive 21-8	DISPlay:WINDOW:BACKground 8-6
:CLEan	DISPlay:WINDOW:GRAPHics 8-7
CONTrol 6-3	DISPlay:WINDOW:TEXT 8-9
:CLEAR	DISPlay:WINDOW:TRACe 8-10
CALCulate:AVERage 4-6	HCOPy:DEVice 10-6
CALCulate:LIMit 4-21	HCOPy:DEVice:CMAP 10-5
DISPlay:WINDOW:GRAPHics 8-7	HCOPy:ITEM:ANNotation 10-8
DISPlay:WINDOW:TEXT 8-9	HCOPy:ITEM:LABEL 10-9
MEMORY 13-6	HCOPy:ITEM:MENU 10-10
OUTPut:PROTection 15-12	HCOPy:ITEM:TDSTamp 10-11
SENSe:CURRent:AC:DC:PROTection 18-26	HCOPy:ITEM:WINDOW:TEXT 10-12
SENSe:POWER:AC:DC:PROTection 18-65	HCOPy:ITEM:WINDOW:TRACe 10-12
SENSe:VOLTage:AC:DC:PROTection 18-85	HCOPy:ITEM:WINDOW:TRACe:GRATicule 10-12
SOURce:CURRent:PROTection 19-23	:COMBine Subsystem
SOURce:POWER:PROTection 19-66	SOURce 19-10
SOURce:RESistance:PROTection 19-77	:COMMAND
SOURce:VOLTage:PROTection 19-99	VXI:WSPRotocol 26-9
:CLIMits	:Command Error
CALCulate 4-9	SYSTem:ERROr? 21-15
:CLOCK	:COMMUnicate
SOURce:DM 19-30	SYSTem 21-4 - 21-10
SOURce:DM:THreshold 19-29	:COMPressor
VXI:WSPRotocol:COMMAND 26-11	CONTrol 6-2
:CLOCK AC	SOURce:LIST:CONTrol 19-50
SOURce:DM:COUpling 19-29	:CONCentration Subsystem
:CLOCK DC	SENSe 18-12 - 18-14
SOURce:DM:COUpling 19-29	:CONCurrent
:CLOCK GRound	SENSe:FUNCtion 18-43
SOURce:DM:COUpling 19-29	SOURce:LIST 19-49
Cloned Models	:CONDITION
Instrument Model:Internal Routing 2-7	MEMory:TABLE 13-10
:CLOSE	SENSe:FUNCtion 18-51
MMEMory 14-3	:CONDITION Subsystem
ROUTe 17-1	SENSe 18-15
:CLR	:CONDITION?
VXI:WSPRotocol:COMMAND 26-10	STATus 20-3, 20-7
:CMAP	:CONFIGure
DISPlay 8-4	Measurement Instructions 3-2
HCOPy:DEVice 10-5	VXI 26-3
:CMESSage	:CONTinuous
SENSe:FILTer 18-34	TRIGger Subsystem:INITiate 24-21
:COAX	:CONTrast
SENSe:CORRection:RVELocity 18-22	DISPlay 8-5
SOURce:CORRection:RVELocity 19-15	:CONTrol
:COHerence	CALCulate:LIMit 4-19

## 1999 SCPI Command Reference

SOURce:LIST 19-50  
SYSTem:COMM:SERial 21-6  
:CONTrol , ALWays  
    TRACe | DATA:FEED 23-4  
:CONTrol NEVer  
    TRACe | DATA:FEED 23-4  
:CONTrol NEXT  
    TRACe | DATA:FEED 23-4  
:CONTrol OCOndition  
    TRACe | DATA:FEED 23-4  
CONTrol Subsystem 6-1 - 6-6  
:COPY  
    MEMory 13-6  
    MMEMory 14-3  
    TRACe | DATA 23-2  
CORRection Subsystem  
    SENSe 18-16 - 18-24  
    SOURce 19-11 - 19-16  
:COUNT  
    CALCulate:AVERage 4-7  
    CALCulate:TRANSform:HISTogram 4-24  
    SENSe:AVERage 18-7  
    SENSe:LIST 18-58  
    SENSe:SWEep 18-78  
    SOURce:LIST 19-50  
    SOURce:PULSe 19-73  
    SOURce:SWEep 19-87  
    TRIGger Subsystem:ARM:SEQUence  
        :LAYer 24-14  
    TRIGger Subsystem:TRIGger:SEQUence 24-24  
:COUNT?  
    SENSe:FUNCTION:OFF 18-44  
    SENSe:FUNCTION:ON 18-45  
:COUPLE  
    INSTrument 12-2  
:COUPLing  
    OUTPut Subsystem 15-3  
    SOURce:DM 19-29  
:COUPLing AC  
    INPut 11-4  
    SOURce:AM 19-6  
    SOURce:AM:EXTernal 19-7  
    SOURce:FM 19-32  
    SOURce:FM:EXTernal 19-33  
    SOURce:PM 19-58  
    SOURce:PM:EXTernal 19-59  
    TRIGger Subsystem:ARM:SEQUence  
        :LAYer 24-14  
    TRIGger Subsystem:TRIGger:SEQUence 24-24  
:COUPLing DC  
    INPut 11-4  
    SOURce:AM 19-6

SOURce:AM:EXTernal 19-7  
SOURce:FM 19-32  
SOURce:FM:EXTernal 19-33  
SOURce:PM 19-58  
SOURce:PM:EXTernal 19-59  
TRIGger Subsystem:ARM  
    :SEQUence:LAYer 24-14  
    TRIGger Subsystem:TRIGger:SEQUence 24-24  
:COUPLing GROund  
    INPut 11-4  
    SOURce:AM 19-6  
    SOURce:AM:EXTernal 19-7  
    SOURce:FM 19-32  
    SOURce:FM:EXTernal 19-33  
    SOURce:PM 19-58  
    SOURce:PM:EXTernal 19-59  
:COVER  
    CONTrol 6-2  
:CPOint  
    DISPLAY:WINDOW:TRACe:R:SCALE 8-15  
:CPON  
    SYSTem 21-11  
:CREference  
    CALCulate:FORMAT:UPHase 4-17  
:CREsponse  
    VXI:WSPProtocol:COMMAND 26-11  
:CREsponse?  
    VXI:WSPProtocol:QUERy 26-15  
:CROSS  
    SENSe:FUNCTION:POWER 18-54  
:CSET  
    SENSe:CORRection 18-18  
    SOURce:CORRection 19-12  
:CSIZE  
    DISPLAY:WINDOW:GRAphics 8-7  
    DISPLAY:WINDOW:TEXT 8-10  
:CTYPe?  
    SYSTem 21-11  
:CURRent  
    :MEASure 3-8  
    INPUT:BIAS 11-3  
    MEMORY:TABLE 13-11  
    SENSe:FUNCTION 18-51  
    SOURce 19-17 - 19-25  
    SOURce:LIST 19-50  
CURRENT Subsystem  
    SENSe 18-25 - 18-29  
    SOURce 19-17 - 19-25  
:CURRent, :POWER, and :VOLTage  
    UNIT 25-1  
:CUT  
    HCOPy:ITEM 10-8

## 1999 SCPI Command Reference

:CW

SENSe:FREQuency 18-38  
SOURce:FREQuency 19-43

### D

:DATA 23-1 - 23-6

CALCulate:LIMit:CONTrol 4-19

CALCulate:LIMit:LOWER 4-20

CALCulate:LIMit:UPPer 4-19

CALibration 5-4

DISPlay:WINDOW:TEXT 8-10

FORMAT 9-1

HCOPy:ITEM:TDSTamp 10-11

MEMORY 13-7

MMEMory 14-3

SOURce:DM:THReShold 19-29

TRACe | DATA 23-2

:DATA AC

SOURce:DM:COUPLing 19-29

Data and Control Flow

Instrument Model:Internal Routing 2-6

:DATA DC

SOURce:DM:COUPLing 19-29

:DATA GROund

SOURce:DM:COUPLing 19-29

DATA SubSystem

SENSe 18-42 - 18-57

:DATA?

CALCulate 4-10

CALCulate:CLIMits:FLIMits 4-9

CALCulate:LIMit:REPort 4-20

HCOPy 10-4

HCOPy:ITEM:ALL 10-8

HCOPy:ITEM:CUT 10-9

HCOPy:ITEM:FFEEd 10-9

HCOPy:ITEM:LABEL 10-10

HCOPy:ITEM:MENU 10-10

HCOPy:ITEM:WINDOW 10-11

HCOPy:ITEM:WINDOW:TEXT 10-12

HCOPy:ITEM:WINDOW:TRACe 10-12

HCOPy:ITEM:WINDOW:TRACe

:GRATicule 10-12

HCOPy:SDUMp 10-15

SENSe:FUNCTION & DATA 18-42

:DATE

SYSTem 21-11

:DC

INPut:BIAS:CURRent 11-4

INPut:BIAS:VOLTage 11-4

MEASure 3-9

SENSe:CURRent 18-25

SENSe:POWER 18-64

SENSe:VOLTage 18-84

:DCYCLE

:MEASure 3-14

SOURce:PULSe 19-71

:DEFault

DISPlay:CMAP 8-4

HCOPy:DEVice:CMAP 10-6

PROGram:SElected 16-3

:DEFine

CALCulate:MATH:EXPRESSion 4-22

MEMory Subsystem:TABLE 13-14

PROGram:EXPLicit 16-4

PROGram:SElected 16-2

ROUTE:MODule 17-2

ROUTE:PATH 17-3

SYSTem:KEY 21-32

TRACe | DATA 23-3

TRIGger Subsystem:ARM:SEQUence 24-12

TRIGger Subsystem:TRIGger:SEQUence 24-24

:DElay

OUTPut:PROTection 15-12

SOURce:PULSe 19-72

SOURce:PULSe:DOUBle 19-72

SYSTem:COMMunicate:SERial:TRANsmiT 21-9

TRIGger Subsystem:ARM

:SEQUence:LAYer 24-14

TRIGger Subsystem:TRIGger:SEQUence 24-25

:DElete

CALCulate:MATH:EXPRESSion 4-22

MEMORY 13-7

MMEMory 14-3

PROGram:EXPLicit 16-5

PROGram:SElected 16-2

ROUTE:MODule 17-2

ROUTE:PATH 17-4

SYSTem:KEY 21-32

TRACe | DATA 23-4

:DEMPhasis

SENSe:FILTer 18-34

:DEPTh

SENSe:AM 18-4

SOURce:AM 19-6

SOURce:LIST:AM 19-49

:DERivative

CALCulate 4-10

DESTination 2-3

DESTination?

HCOPy 10-4

:DETector EXTERNAL

SENSe:CURRent 18-29

SENSe:POWER 18-67

## 1999 SCPI Command Reference

SENSe:VOLTage 18-88  
:DETector INTernal  
  SENSe:CURREnt 18-29  
  SENSe:POWER 18-67  
  SENSe:VOLTage 18-88  
DETector Subsystem  
  SENSe 18-30 - 18-31  
:DEViation  
  SENSe:FM 18-36  
  SENSe:PM 18-62  
  SOURce:FM 19-32  
  SOURce:PM 19-57  
:DEvice  
  HCOPy 10-5 - 10-6  
:Device-Specific Error  
  SYSTem:ERRor? 21-25  
DIAGnostic Subsystem 7-1 - 7-2  
Digital Modulation Subsystem  
  SOURce 19-26 - 19-31  
:DIMensions  
  HCOPy:PAGE 10-13  
:DINTerchange  
  FORMat 9-2  
  MMEMory:LOAD and :STORE 14-4  
:DIRection DOWN  
  INPut:POSition:X:ANGLE 11-8  
  INPut:POSition:X:DISTance 11-9  
  INPut:POSition:Y:ANGLE 11-10  
  INPut:POSition:Y:DISTance 11-11  
  INPut:POSition:Z:ANGLE 11-12  
  INPut:POSition:Z:DISTance 11-13  
  OUTPut:POSition:X:ANGLE 15-6  
  OUTPut:POSition:X:DISTance 15-7  
  OUTPut:POSition:Y:ANGLE 15-8  
  OUTPut:POSition:Y:DISTance 15-9  
  OUTPut:POSition:Z:ANGLE 15-10  
  OUTPut:POSition:Z:DISTance 15-11  
  SENSe:CURREnt:AC|DC 18-27  
  SENSe:LIST 18-58  
  SENSe:POWER:AC|DC:RANGE:AUTO 18-66  
  SENSe:RESistance:RANGE:AUTO 18-69  
  SENSe:SWEep 18-78  
  SENSe:VOLTage:AC|DC 18-86  
  SOURce:LIST 19-51  
  SOURce:SWEep 19-86  
:DIRection EITHer  
  SENSe:CURREnt:AC|DC 18-27  
  SENSe:POWER:AC|DC:RANGE:AUTO 18-66  
  SENSe:RESistance:RANGE:AUTO 18-69  
  SENSe:VOLTage:AC|DC 18-86  
:DIRection UP  
  INPut:POSition:X:ANGLE 11-8

INPut:POSition:X:DISTance 11-9  
INPut:POSition:Y:ANGLE 11-10  
INPut:POSition:Y:DISTance 11-11  
INPut:POSition:Z:ANGLE 11-12  
INPut:POSition:Z:DISTance 11-13  
OUTPut:POSition:X:ANGLE 15-6  
OUTPut:POSition:X:DISTance 15-7  
OUTPut:POSition:Y:ANGLE 15-8  
OUTPut:POSition:Y:DISTance 15-9  
OUTPut:POSition:Z:ANGLE 15-10  
OUTPut:POSition:Z:DISTance 15-11  
SENSe:CURRent:AC|DC 18-27  
SENSe:LIST 18-58  
SENSe:POWER:AC|DC:RANGE:AUTO 18-66  
SENSe:RESistance:RANGE:AUTO 18-69  
SENSe:SWEep 18-78  
SENSe:VOLTage:AC|DC 18-86  
SOURce:LIST 19-51  
SOURce:SWEep 19-86  
DISPlay Subsystem 8-1 - 8-16  
:DISTance  
  CALCulate:TRANSform 4-27  
  INPut:POSition:X 11-9  
  INPut:POSition:Y 11-11  
  INPut:POSition:Z 11-13  
  OUTPut:POSition:X 15-7  
  OUTPut:POSition:Y 15-9  
  OUTPut:POSition:Z 15-11  
  SENSe:CORRection:EDELay 18-19  
  SENSe:FUNCtion 18-52  
  SOURce:CORRection:EDELay 19-14  
DISTance Subsystem  
  SENSe 18-32  
:DISTortion  
  SENSe:FUNCtion:POWER 18-50, 18-52 - 18-54  
DM  
  SOURce 19-26 - 19-31  
DM Subsystem  
  SOURce 19-26 - 19-31  
:DMODe PARallel  
  SOURce:DM 19-30  
:DMODe SERial  
  SOURce:DM 19-30  
:DNUMber?  
  VXI:CONFigure 26-3  
:DOUBLE  
  SOURce:PULSe 19-72  
:DRAW  
  DISPlay:WINDOW:GRAPHics 8-8  
:DTR IBFull  
  SYSTem:COMM:SERial:CONTrol 21-6  
:DTR OFF

## 1999 SCPI Command Reference

SYSTem:COMM:SERial:CONTrol 21-6  
:DTR ON  
    SYSTem:COMM:SERial:CONTrol 21-6  
:DTR STANDard  
    SYSTem:COMM:SERial:CONTrol 21-6  
:DURATION  
    CONTrol 6-3  
:DWELl  
    SENSe:LIST 18-58  
    SENSe:SWEep 18-78  
    SOURce:LIST 19-51  
    SOURce:SWEep 19-85

### E

:EBENch  
    CONTrol 6-3  
:ECL  
    TRIGger Subsystem:ARM  
        :SEQUence:LAYer 24-15  
    TRIGger Subsystem:TRIGger:SEQUence 24-26  
:ECLTrg  
    OUTPut Subsystem 15-12  
:ECOunt  
    TRIGger Subsystem:ARM  
        :SEQUence:LAYer 24-15  
    TRIGger Subsystem:TRIGger:SEQUence 24-26  
:EDElay  
    SENSe:CORRection 18-19  
    SOURce:CORRection 19-14  
:ENABLE  
    DISPlay 8-5  
    STATus 20-3, 20-7  
:ENO  
    VXI:WSPProtocol:COMMAND 26-11  
:ENO?  
    VXI:WSPProtocol:QUERy 26-15  
:Error/Event numbers  
    SYSTem:ERRor? 21-14  
    The Error/Event Queue  
        SYSTem:ERRor? 21-13  
:ERRor?  
    SYSTem 21-11 - 21-27  
:Error/Event numbers  
    SYSTem:ERRor? 21-14  
:EVEnT?  
    STATus 20-4, 20-7  
<event\_handle>  
    Instrument Model:Internal Routing 2-14  
:EXChange  
    MEMORY 13-7  
:EXECute

PROGram:EXPLicit 16-5  
PROGram:SElected 16-2  
:Execution Error  
    SYSTem:ERRor? 21-19  
:Expanded Capability Trigger Model  
    TRIGger Subsystem 24-5 - 24-11  
:EXPLICIT  
    PROGram 16-4 - 16-6  
EXPonential  
    CALCulate:AVERage:TCONTrol 4-7  
    CALCulate:FILTer:GATE:FREQuency 4-14  
    CALCulate:FILTer:GATE:TIME 4-12  
    CALCulate:TRANSform:DISTance 4-29  
    CALCulate:TRANSform:FREQuency 4-31  
    CALCulate:TRANSform:TIME 4-27  
    SENSe:WINDOW:TYPE 18-89  
:EXPRESSION  
    CALCulate:MATH 4-21  
:EXTernal  
    OUTPut:FILTter 15-3  
    SENSe:ROSCillator 18-71, 19-80  
    SOURce:AM 19-6  
    SOURce:FM 19-32  
    SOURce:PM 19-59  
    SOURce:PULM 19-68

### F

:FAIL?  
    CALCulate:CLIMits 4-9  
    CALCulate:LIMit 4-20  
:FALL  
    MEASure 3-13  
:FCOunt?  
    CALCulate:LIMit 4-20  
:FCUToff  
    SENSe:CORRection:RVELOCITY 18-22  
    SOURce:CORRection:RVELOCITY 19-15  
FEED  
    CALCulate 4-10  
    DISPLAY:WINDOW:TEXT 8-10  
    DISPLAY:WINDOW:TRACe 8-10  
    HCOPy 10-7  
    Instrument Model:Internal Routing 2-9  
    MMEMory 14-4  
    SYSTem:COMM:SERial 21-7  
    SYSTem:COMMunicate:CENTronics 21-5  
    SYSTem:COMMunicate:GPIB:RDEvice 21-5  
    TRACe | DATA 23-4  
FEED Exceptions  
    Instrument Model:Internal Routing  
    :Lamina and Cloned Models 2-8

# 1999 SCPI Command Reference

FEEDs from CALCulate Sub-Blocks  
    Instrument Model:Internal Routing 2-12

FERRor  
    SENSe:FUNCtion 18-52

FETCh  
    Measurement Instructions 3-3

:FFEed  
    HCOPy:ITEM 10-9

:FIELD  
    TRIGger Subsystem:ARM:SEQUence  
        :LAYer:VIDeo 24-19  
    TRIGger Subsystem:TRIGger:SEQUence  
        :VIDeo 24-31

:FILTer  
    CALCulate 4-11 - 4-14  
    INPUT 11-5  
    OUTPut Subsystem 15-3  
    SOURce:DM 19-27  
    TRIGger Subsystem:ARM:SEQUence  
        :LAYer 24-15  
    TRIGger Subsystem:TRIGger:SEQUence 24-26

FILTER Subsystem  
    SENSe 18-33 - 18-35

:FIXed  
    SENSe:FREQuency 18-38  
    SOURce:FREQuency 19-43

:FLIMits  
    CALCulate:CLIMits 4-9

FM  
    SENSe:FUNCtion 18-52

FM Subsystem  
    SENSe 18-36  
    SOURce 19-32 - 19-34

:FORCe  
    CALCulate:FILTer:GATE:FREQuency 4-15  
    CALCulate:FILTter:GATE:TIME 4-13  
    CALCulate:TRANSform:DISTance 4-29  
    CALCulate:TRANSform:FREQuency 4-31  
    CALCulate:TRANSform:TIME 4-27  
    SENSe:FUNCtion 18-52  
    SENSe:WINDow:TYPE 18-89

FORCe Subsystem  
    SOURce 19-35 - 19-42

FORMAT  
    CALCulate 4-15 - 4-16  
    Instrument Model 2-5  
    SOURce:DM 19-27  
    TRIGger Subsystem:ARM:SEQUence  
        :LAYer:VIDeo 24-20  
    TRIGger Subsystem:TRIGger:SEQUence  
        :VIDeo:FIELD 24-31

FORMAT Subsystem 9-1 - 9-4

:FRAMe  
    DISPlay:WINDOW:TRACe:GRATicule 8-11  
    SOURce:DM 19-30

:FREE  
    MEMory 13-8

:FREE?  
    TRACe | DATA 23-5

:FREQuency  
    :MEASure 3-9  
    CALCulate:FILTter:GATE 4-13  
    CALCulate:TRANSform 4-29  
    DISPlay:ANNotation 8-4  
    INPUT:FILTter:HPASs 11-5  
    INPUT:FILTter:LPASs 11-5  
    MEMory Subsystem:TABLE 13-15  
    OUTPUT:FILTter:HPASs 15-4  
    OUTPUT:FILTter:LPASs 15-4  
    SENSe:FILTter:HPASs 18-34  
    SENSe:FILTter:LPASs 18-33  
    SENSe:FUNCtion 18-52  
    SENSe:LIST 18-59  
    SENSe:ROSCillator:EXTernal 18-71, 19-80  
    SENSe:ROSCillator:INTERNAL 18-71, 19-80  
    SOURce:AM:INTERNAL 19-7  
    SOURce:FM:INTERNAL 19-33  
    SOURce:LIST 19-51  
    SOURce:MARKer 19-54  
    SOURce:PM:INTERNAL 19-59  
    SOURce:PULM:INTERNAL 19-69  
    SYStem:BEEPer 21-3  
    TRIGger Subsystem:ARM:SEQUence  
        :LAYer:FILTer:HPASs 24-15  
    TRIGger Subsystem:ARM:SEQUence  
        :LAYer:FILTer:LPASs 24-15  
    TRIGger Subsystem:TRIGger:SEQUence  
        :FILTter:HPASs 24-26  
    TRIGger Subsystem:TRIGger:SEQUence  
        :FILTter:LPASs 24-27

FREQuency Subsystem  
    SENSe 18-37 - 18-41  
    SOURce 19-43 - 19-46

:FRESistance  
    :MEASure 3-8  
    SENSe:FUNCtion 18-53

FRESistance Subsystem  
    SENSe 18-68 - 18-70

FRONT  
    SENSe:FUNCtion:SPEEd 18-55

:FSENsor  
    CALibration 5-7

:FTIME  
    MEASure 3-13

## 1999 SCPI Command Reference

:FULL  
SENSe:FREQuency:SPAN 18-40  
SOURce:CURRent:SPAN 19-24  
SOURce:FREQuency:SPAN 19-45  
SOURce:POWER:SPAN 19-67  
SOURce:RESistance:SPAN 19-78  
SOURce:VOLTage:SPAN 19-100

<function>  
Measurement Instructions 3-5  
SENSe 18-43  
SENSe:DETector 18-30  
FUNCTION & DATA Subsystem  
SENSe 18-42 - 18-57  
FUNCTION Subsystem  
SOURce 19-47  
<function\_name>  
SENSe:FUNCTION 18-48  
Fundamental Measurement Layer  
MEASure 3-7

### G

:GAIN  
INPut 11-6  
SENSe:CORRection 18-20  
SOURce:CORRection 19-13  
:GATE  
CALCulate:FILTER 4-11  
:GDAperture  
CALCulate 4-17  
:GDEVice  
VXI:WSPRotocol:COMMand 26-11  
:GENERATION ANALog  
SENSe:SWEep 18-79  
SOURce:SWEep 19-86  
:GENERATION CONCurrent  
SOURce:LIST 19-51  
:GENERATION DCONcurrent  
SOURce:LIST 19-51  
:GENERATION DSEQUence  
SOURce:LIST 19-51  
:GENERATION SEQuence  
SOURce:LIST 19-51  
:GENERATION STEPped  
SENSe:SWEep 18-79  
SOURce:SWEep 19-86  
:GEOMetry  
DISPLAY:WINDOW 8-6  
:GPIB  
SYSTem:COMMUnicate 21-5  
:GRAPHics  
DISPLAY:WINDOW 8-7

:GRATicule  
DISPLAY:WINDOW:TRACe 8-11  
HCOPy:ITEM:WINDOW:TRACe 10-12  
:GRID  
DISPLAY:WINDOW:TRACe:GRATicule 8-11  
:GUARD  
INPut 11-6

### H

:HARMonic  
SENSe:MIXer 18-60  
HCOPy Subsystem 10-1 - 10-16  
:HEADers?  
SYSTem:HELP 21-28  
:HELP  
SYSTem 21-28 - 21-31  
HEXadecimal 9-2 - 9-3  
:HIERarchy?  
VXI:CONFigure 26-3  
:HIGH  
INPut:POSITION:X:ANGLE:LIMit 11-8  
INPut:POSITION:X:DISTance:LIMit 11-9  
INPut:POSITION:Y:ANGLE:LIMit 11-10  
INPut:POSITION:Y:DISTance:LIMit 11-11  
INPut:POSITION:Z:ANGLE:LIMit 11-12  
INPut:POSITION:Z:DISTance:LIMit 11-13  
MEASure 3-12  
OUTPut:POSITION:X:ANGLE:LIMit 15-6  
OUTPut:POSITION:X:DISTance:LIMit 15-7  
OUTPut:POSITION:Y:ANGLE:LIMit 15-8  
OUTPut:POSITION:Y:DISTance:LIMit 15-9  
OUTPut:POSITION:Z:ANGLE:LIMit 15-10  
OUTPut:POSITION:Z:DISTance:LIMit 15-11  
SOURce:CURREnt:LEVel:IMMEDIATE 19-20  
SOURce:CURREnt:LEVel:TRIGgered 19-21  
SOURce:CURREnt:LIMit 19-22  
SOURce:POWER:LEVel:IMMEDIATE 19-63  
SOURce:POWER:LEVel:TRIGgered 19-64  
SOURce:POWER:LIMit 19-65  
SOURce:RESistance:LEVel:IMMEDIATE 19-75  
SOURce:RESistance:LEVel:TRIGgered 19-76  
SOURce:RESistance:LIMit 19-76  
SOURce:VOLTage:LEVel:IMMEDIATE 19-96  
SOURce:VOLTage:LEVel:TRIGgered 19-97  
SOURce:VOLTage:LIMit 19-98  
:HISTogram  
CALCulate:TRANsform 4-24  
SENSe:FUNCTION:VOLTage 18-56  
:HOLD  
SENSe:FREQuency:SPAN 18-40  
SOURce:CURREnt:SPAN 19-24

## 1999 SCPI Command Reference

SOURce:FREQuency:SPAN 19-45  
SOURce:POWER:SPAN 19-67  
SOURce:RESistance:SPAN 19-78  
SOURce:VOLTage:SPAN 19-100  
:HOLD DCYCle  
    SOURce:PULSe 19-71  
:HOLD WIDTh  
    SOURce:PULSe 19-71  
:HOLDoff  
    TRIGger Subsystem:TRIGger:SEQuence 24-27  
:HORizontal  
    INPUT:POLarization 11-7  
    OUTPut:POLarization 15-5  
:HPASs  
    INPUT:FILTER 11-5  
    OUTPut:FILTER 15-4  
    SENSe:FILTER 18-33  
    TRIGger Subsystem:ARM  
        :SEQuence:LAYer:FILTER 24-15  
    TRIGger Subsystem:TRIGger  
        :SEQuence:FILTER 24-26  
:HPGL  
    DISPlay:WINDOW:GRAphics 8-8  
:HSL  
    DISPlay:CMAP:COLOr 8-5  
    HCOPy:DEvice:CMAP:COLOr 10-5  
:HYSTeresis  
    SOURce:PULM:EXTernal 19-68  
    TRIGger Subsystem:ARM:SEQuence  
        :LAYer 24-16  
    TRIGger Subsystem:TRIGger:SEQuence 24-27

### I

:I INVerted  
    SOURce:DM:POLarity :ALL  
        NORMAL|INVerted 19-30  
:I<n> NORMAL  
    SOURce:DM:POLarity :ALL  
        NORMAL|INVerted 19-30  
:IClock INVerted  
    SOURce:DM:POLarity :ALL  
        NORMAL|INVerted 19-30  
:IClock NORMAL  
    SOURce:DM:POLarity :ALL  
        NORMAL|INVerted 19-30  
:ICOMmander  
    VXI:WSPProtocol:COMMAND 26-11  
:ICORrection  
    SOURce:DM:FILTER 19-27  
:IDLE  
    CONTrol 6-3

:IDLE State  
    TRIGger Subsystem:Model Layers 24-2  
:IDRaw  
    DISPlay:WINDOW:GRAphics 8-8  
:IMMediate  
    CALCulate 4-18  
    CALCulate:LIMit:CLEar 4-21  
    HCOPy 10-7 - 10-13, 10-15  
    INPUT:POSITION:X:ANGLE 11-8  
    INPUT:POSITION:X:DISTance 11-9  
    INPUT:POSITION:Y:ANGLE 11-10  
    INPUT:POSITION:Y:DISTance 11-11  
    INPUT:POSITION:Z:ANGLE 11-12  
    INPUT:POSITION:Z:DISTance 11-13  
    OUTPut:POSITION:X:ANGLE 15-6  
    OUTPut:POSITION:X:DISTance 15-7  
    OUTPut:POSITION:Y:ANGLE 15-8  
    OUTPut:POSITION:Y:DISTance 15-9  
    OUTPut:POSITION:Z:ANGLE 15-10  
    OUTPut:POSITION:Z:DISTance 15-11  
    OUTPut:TTLTrg|ECLTrg 15-12  
    SOURce:CURRent:LEVel 19-19  
    SOURce:POWER:LEVel 19-62  
    SOURce:RESistance:LEVel 19-75  
    SOURce:VOLTage:LEVel 19-95  
    SYSTem:BEEPer 21-4  
    TRIGger Subsystem:ARM:SEQuence  
        :LAYer 24-16  
    TRIGger Subsystem:INITiate 24-22  
    TRIGger Subsystem:TRIGger:SEQuence 24-27  
:IMOve  
    DISPlay:WINDOW:GRAphics 8-8  
:IMPedance  
    INPUT 11-6  
    OUTPut Subsystem 15-5  
    SENSe:CORRection 18-19  
    SOURce:AM:EXTernal 19-7  
    SOURce:FM:EXTernal 19-33  
    SOURce:PM:EXTernal 19-59  
    SOURce:PULM:EXTernal 19-68  
:INFormation?  
    VXI:CONFigure 26-4  
:INITialize  
    MMEMory 14-4  
INITiate 2-5  
    CONTrol 6-3  
        TRIGger Subsystem 24-21 - 24-22  
:Initiated  
    TRIGger Subsystem:Model Layers 24-2  
INPUT 11-1 - 11-14  
    Instrument Model:Measurement Function 2-3  
    SENSe:CORRection:IMPedance 18-19

## 1999 SCPI Command Reference

SENSe:CORRection:LOSS 18-20  
Instrument Model 2-1 - 2-14  
INSTrument Subsystem 12-1 - 12-4  
INTeger 9-2  
:INTEGRal  
    CALCulate 4-18  
:INTERNAL  
    SENSe:ROSCillator 18-71, 19-80  
    SOURce:AM 19-7  
    SOURce:FM 19-33  
    SOURce:PM 19-59  
    SOURce:PULM 19-69  
Internal Routing  
    Instrument Model 2-5 - 2-14  
:INTERpolate  
    CALCulate:LIMit 4-21  
Introduction 1-1 - 1-2  
:IQRatio  
    SOURce:DM 19-28  
:ITEM  
    HCOPy 10-7 - 10-12

### K

:KBESsel  
    CALCulate:FILTer:GATE:FREQuency 4-14  
    CALCulate:FILTer:GATE:TIME 4-12  
    CALCulate:TRANSform:DISTance 4-29  
    CALCulate:TRANSform:FREQuency 4-31  
    CALCulate:TRANSform:TIME 4-27  
    SENSe:WINDOW:TYPE 18-89  
:KEY  
    DISPlay:MENU 8-6  
    SYSTem 21-32  
:KLOCK  
    SYSTem 21-33

### L

:LABEL  
    DISPlay:WINDOW:GRAPHics 8-8  
    DISPlay:WINDOW:TRACe:R 8-15  
    DISPlay:WINDOW:TRACe:X 8-12  
    DISPlay:WINDOW:TRACe:Y 8-13  
    HCOPy:ITEM 10-9  
:LADDress?  
    VXI:CONFIGure 26-6  
Lamina and Cloned Models  
    Instrument Model:Internal Routing 2-7  
:LANGuage  
    HCOPy:DEVICE 10-6  
    SYSTem 21-33

:LAYer  
    TRIGger Subsystem:ARM:SEQUence 24-14  
:LAYer Nomenclature  
    TRIGger Subsystem  
        :Expanded Capability Trigger Model 24-6  
:LDIRection  
    DISPlay:WINDOW:GRAPHics 8-8  
:LEADING  
    SOURce:PULSe:TRANSition 19-72  
:LEAKage  
    SOURce:DM 19-28  
:LEFT  
    DISPlay:WINDOW:TRACe:X:SCALE 8-12  
:LENGTH  
    HCOPy:PAGE 10-14  
:LEVel  
    OUTPut:TTLTrg:ECLTrg 15-13  
    SENSe:CONDITION 18-15  
    SENSe:CURREnt:AC:DC:PROTection 18-26  
    SENSe:POWER:AC:DC:PROTection 18-65  
    SENSe:VOLTage:AC:DC:PROTection 18-85  
    SOURCE:CURREnt 19-19  
    SOURCE:CURREnt:PROTection 19-23  
    SOURCE:POWER 19-62  
    SOURCE:POWER:PROTection 19-65  
    SOURCE:PULM:EXTernal 19-68  
    SOURCE:RESistance 19-75  
    SOURCE:RESistance:PROTection 19-77  
    SOURCE:VOLTage 19-95  
    SOURCE:VOLTage:PROTection 19-99  
    TRIGger Subsystem:ARM:SEQUence  
        :LAYer 24-16  
    TRIGger Subsystem:TRIGger:SEQUence 24-28  
:LFREquency  
    SYSTem 21-33  
:LIFT  
    CONTrol 6-3  
:LIMIT  
    CALCulate 4-18 - 4-20  
    INPut:POSition:X:ANGLE 11-8  
    INPut:POSition:X:DISTance 11-9  
    INPut:POSition:Y:ANGLE 11-10  
    INPut:POSition:Y:DISTance 11-11  
    INPut:POSition:Z:ANGLE 11-12  
    INPut:POSition:Z:DISTance 11-13  
    OUTPut:POSition:X:ANGLE 15-6  
    OUTPut:POSition:X:DISTance 15-7  
    OUTPut:POSition:Y:ANGLE 15-8  
    OUTPut:POSition:Y:DISTance 15-9  
    OUTPut:POSition:Z:ANGLE 15-10  
    OUTPut:POSition:Z:DISTance 15-11  
    SENSe:MIXer:BIAS 18-60

## 1999 SCPI Command Reference

SOURce:CURREnt 19-21	MMEMory 14-4
SOURce:POWER 19-64	:LOCate
SOURce:RESistance 19-76	DISPlay:WINDOW:TEXT 8-10
SOURce:VOLTage 19-97	:LOSS
:LINE	MEMory:TABLE 13-16
TRACe   DATA:DATA 23-3	SENSe:CORRection 18-20
TRIGger Subsystem:ARM	SENSe:MIXer 18-61
:SEQuence:LAYer:VIDeo 24-20	SOURce:CORRection 19-13
TRIGger Subsystem:TRIGger	:LOW
:SEQuence:VIDeo 24-32	INPUT 11-6
:LINK	INPUT:POSITION:X:ANGLE:LIMit 11-8
TRIGger Subsystem:ARM	INPUT:POSITION:X:DISTance:LIMit 11-9
:SEQuence:LAYer 24-17	INPUT:POSITION:Y:ANGLE:LIMit 11-10
TRIGger Subsystem:TRIGger:SEQuence 24-28	INPUT:POSITION:Y:DISTance:LIMit 11-11
:LINK CENTer	INPUT:POSITION:Z:ANGLE:LIMit 11-12
DISPlay:WINDOW:TRACe:X:SCALE:PDIV 8-13	INPUT:POSITION:Z:DISTance:LIMit 11-13
SENSe:FREQuency:SPAN 18-40	MEASure 3-12
SOURce:CURREnt:SPAN 19-24	OUTPUT Subsystem 15-5
SOURce:FREQuency:SPAN 19-45	OUTPUT:POSITION:X:ANGLE:LIMit 15-6
SOURce:POWER:SPAN 19-67	OUTPUT:POSITION:X:DISTance:LIMit 15-7
SOURce:RESistance:SPAN 19-78	OUTPUT:POSITION:Y:ANGLE:LIMit 15-8
SOURce:VOLTage:SPAN 19-100	OUTPUT:POSITION:Y:DISTance:LIMit 15-9
:LINK LEFT	OUTPUT:POSITION:Z:ANGLE:LIMit 15-10
DISPlay:WINDOW:TRACe:X:SCALE:PDIV 8-13	OUTPUT:POSITION:Z:DISTance:LIMit 15-11
:LINK RIGHt	SOURce:CURREnt:LEVel:IMMEDIATE 19-20
DISPlay:WINDOW:TRACe:X:SCALE:PDIV 8-13	SOURce:CURREnt:LEVel:TRIGgered 19-21
:LINK STARt	SOURce:CURREnt:LIMit 19-22
SENSe:FREQuency:SPAN 18-40	SOURce:POWER:LEVel:IMMEDIATE 19-63
SOURce:CURREnt:SPAN 19-24	SOURce:POWER:LEVel:TRIGgered 19-64
SOURce:FREQuency:SPAN 19-45	SOURce:POWER:LIMit 19-65
SOURce:POWER:SPAN 19-67	SOURce:RESistance:LEVel:IMMEDIATE 19-75
SOURce:RESistance:SPAN 19-78	SOURce:RESistance:LEVel:TRIGgered 19-76
SOURce:VOLTage:SPAN 19-100	SOURce:RESistance:LIMit 19-77
:LINK STOP	SOURce:VOLTage:LEVel:IMMEDIATE 19-96
SENSe:FREQuency:SPAN 18-40	SOURce:VOLTage:LEVel:TRIGgered 19-97
SOURce:CURREnt:SPAN 19-24	SOURce:VOLTage:LIMit 19-98
SOURce:FREQuency:SPAN 19-45	:LOWER
SOURce:POWER:SPAN 19-67	CALCulate:LIMit 4-20
SOURce:RESistance:SPAN 19-78	SENSe:AM:DEPTH:RANGE 18-4
SOURce:VOLTage:SPAN 19-100	SENSe:CURREnt:AC:DC:RANGE 18-27
LIST Subsystem	SENSe:FM:DEViation:RANGE 18-36
SENSe 18-58 - 18-59	SENSe:FREQuency:RANGE 18-39
SOURce 19-48 - 19-53	SENSe:PM:DEViation:RANGE 18-62
LL	SENSe:POWER:AC:DC:RANGE 18-66
See also IDElete	SENSe:POWER:ACHannel:SPACing 18-64
:LLEFt	SENSe:RESistance:RANGE 18-69
DISPlay:WINDOW:GEOMetry 8-7	SENSe:VOLTage:AC:DC:RANGE 18-86
HCOPy:PAGE:DIMensions 10-14	:LPAs
:LLIMit	INPUT:FILTter 11-5
SENSe:SWEep:TIME 18-83	OUTPUT:FILTter 15-4
SOURce:SWEep:TIME 19-85	SENSe:FILTter 18-33
:LOAD	TRIGger Subsystem:ARM

## 1999 SCPI Command Reference

:SEQUence:LAYer:FILTer 24-15  
TRIGger Subsystem:TRIGger  
:SEQUence:FILTer 24-26  
:LPFFrame  
    TRIGger Subsystem:ARM  
    :SEQUence:LAYer:VIDeo:FORMAT 24-20  
    TRIGger Subsystem:TRIGger  
    :SEQUence:VIDeo:FIELD:FORMAT 24-31  
:LTYPe  
    DISPlay:WINDOW:GRAphics 8-8  
    HCOPy:ITEM:WINDOW:TRACe 10-13

### M

:MACRo  
    MMEMory:LOAD and :STORe 14-4  
:MACRo?  
    MEMory:CATalog 13-5  
    MEMory:FREE 13-8  
:MAGNitude  
    MEMory:TABLE:CONDition 13-11  
    SENSe:CORRection:LOSS:INPUT 18-21  
    SENSe:CORRection:OFFSet 18-21  
    SOURce:CORRection:LOSS:OUTPut 19-14  
    SOURce:CORRection:OFFSet 19-13  
    SOURce:DM:IQRatio 19-28  
    SOURce:DM:LEAKage 19-28  
:MAGNitude  
    MEMory:TABLE:CURREnt 13-11  
    MEMory:TABLE:LOSS 13-16  
    MEMory:TABLE:POWer 13-17  
    MEMory:TABLE:RESistance 13-17  
    MEMory:TABLE:VOLTage 13-19  
:MALlocate  
    PROGram:EXPLicit 16-5  
    PROGram:SELected 16-3  
:MANual  
    SENSe:FREQuency 18-38  
    SOURce:CURREnt 19-22  
    SOURce:FREQuency 19-44  
    SOURce:POWer 19-65  
    SOURce:RESistance 19-78  
    SOURce:VOLTage 19-98  
:MAP  
    STATus 20-4, 20-7  
MARKer Subsystem  
    SOURce 19-54 - 19-55  
:MATH  
    CALCulate 4-21 - 4-22  
:MAXimum  
    MEASure 3-14  
:MCOnrol

    CONTrol 6-4  
MEASure  
    Measurement Instructions 3-5  
Measurement Function  
    Instrument Model 2-3  
Measurement Function Layer 3-9  
Measurement Instructions 3-1 - 3-14  
:MEDIUM COAX  
    SENSe:CORRection:RVELocity 18-22  
    SOURce:CORRection:RVELocity 19-15  
:MEDIUM WAVeguide  
    SENSe:CORRection:RVELocity 18-22  
    SOURce:CORRection:RVELocity 19-15  
MEMORY  
    Instrument Model 2-5  
Memory Associated with Data Flow  
    Instrument Model:INTERNAL Routing 2-12  
MEMORY Subsystem 13-1 - 13-20  
:MENU  
    DISPLAY 8-6  
    HCOPy:ITEM 10-10  
:MESSage  
    VXI:WSPProtocol 26-13  
:METHOD PMETer  
    SOURce:CORRection:COLlect 19-12  
:METHOD TPORt  
    SENSe:CORRection:COLlect 18-18  
:MRRules  
    TRIGger Subsystem:ARM:SEQUence  
        :DEFine 24-13  
    TRIGger Subsystem:TRIGger:SEQUence  
        :DEFine 24-25  
:MINimum  
    MEASure 3-14  
MIXer Subsystem  
    SENSe 18-60 - 18-61  
MMEMory Subsystem 14-1 - 14-6  
:MODE  
    HCOPy:DEVice 10-7  
    SOURce:AM 19-7  
    SOURce:FUNCTION 19-47  
    SOURce:PULM 19-69  
:MODE AUTO  
    SENSe:SWEep 18-79  
    SOURce:SWEep 19-86  
:MODE CW  
    SENSe:FREQuency 18-38  
    SOURce:FREQuency 19-44  
:MODE DELTa  
    SOURce:MARKer 19-54  
:MODE FIXed  
    SENSe:FREQuency 18-38

## 1999 SCPI Command Reference

SOURce:CURREnt 19-22  
SOURce:FREQuency 19-44  
SOURce:POWer 19-65  
SOURce:RESistance 19-79  
SOURce:VOLTage 19-98  
:MODE FREQuency  
    SOURce:MARKer 19-54  
:MODE LIST  
    SENSe:FREQuency 18-38  
    SOURce:CURREnt 19-22  
    SOURce:FREQuency 19-44  
    SOURce:POWer 19-65  
    SOURce:RESistance 19-79  
    SOURce:VOLTage 19-98  
:MODE LOCKed  
    SOURce:FM 19-33  
    SOURce:PM 19-57  
:MODE MANUAL  
    SENSe:SWEep 18-79  
    SOURce:SWEep 19-86  
:MODE POSition  
    SOURce:MARKer 19-54  
:MODE SENSe  
    SOURce:FREQuency 19-44  
:MODE SOURce  
    SENSe:FREQuency 18-38  
:MODE SWEep  
    SENSe:FREQuency 18-38  
    SOURce:CURREnt 19-22  
    SOURce:FREQuency 19-44  
    SOURce:POWer 19-65  
    SOURce:RESistance 19-79  
    SOURce:VOLTage 19-98  
:MODE UNLocked  
    SOURce:FM 19-33  
    SOURce:PM 19-57  
:Model Layers  
    TRIGger Subsystem 24-2 - 24-3  
:MODULE  
    ROUTe 17-2  
:MOVE  
    DISPlay:WINDOW:GRAPHics 8-9  
    MMEMory 14-5  
MOVing  
    CALCulate:AVERage:TCONtrol 4-7  
:MSIS  
    MMEMory 14-5  
:MULTiplier  
    SENSe:FREQuency 18-39  
    SOURce:FREQuency 19-44

## N

:NAME  
    CALCulate:MATH:EXPReSSion 4-22  
    DISPlay:MENU 8-6  
    MEMory:CLEar 13-6  
    MEMory:COPY 13-6  
    MEMory:DELetE 13-7  
    MEMory:EXCHange 13-7  
    MMEMory 14-5  
    PROGram:SElected 16-3  
    ROUTe:MODule:DELetE 17-3  
    ROUTe:PATH:DELetE 17-4  
    TRACe | DATA:DELetE 23-4  
    TRIGger Subsystem:INITiate:CONTinuous 24-22  
    TRIGger Subsystem:INITiate:IMMEDIATE 24-23  
:NDUTycycle  
    MEASure 3-14  
:No Error  
    SYSTem:ERRor? 21-14  
NORMAL  
    CALCulate:AVERage:TCONtrol 4-7  
notation  
    X<sub>n</sub>\* 4-1  
:NPCLcycles  
    SENSe:CURREnt:AC|DC 18-26  
    SENSe:POWER:AC|DC 18-64  
    SENSe:RESistance 18-68  
    SENSe:VOLTage:AC|DC 18-85  
:NSElect  
    INSTrument 12-4  
:NSTates?  
    MEMORY 13-9  
:NTRansition  
    STATus 20-4, 20-7  
:NUMBER  
    PROGram:EXPLicit 16-5  
    PROGram:SElected 16-3  
    TRIGger Subsystem:ARM:SEQUence  
    :LAYer:VIDeo:FIELD 24-19  
    TRIGger Subsystem:ARM:SEQUence  
    :LAYer:VIDeo:LINE 24-20  
    TRIGger Subsystem:TRIGger:SEQUence  
    :VIDeo:FIELD 24-31  
    TRIGger Subsystem:TRIGger:SEQUence  
    :VIDeo:LINE 24-32  
:NUMBER?  
    VXI:CONFIGure 26-7  
Numeric Suffixes  
    Instrument Model:Internal Routing 2-7  
:NWIDth  
    MEASure 3-14

## 1999 SCPI Command Reference

### O

:OCOMpensated  
    SENSe:RESistance 18-68  
:OCONDition ,  
    TRACe | DATA:FEED 23-5  
OCTal 9-2 - 9-3  
:OEDGE  
    DISPlay:WINDOW:TRACe:R:SCALe 8-16  
:OFF  
    SENSe:FUNCtion 18-44  
:OFFSet  
    INPut 11-7  
:OFFSet  
    INPut:POSITION:X:ANGLE 11-8  
    INPut:POSITION:X:DISTance 11-9  
    INPut:POSITION:Y:ANGLE 11-10  
    INPut:POSITION:Y:DISTance 11-11  
    INPut:POSITION:Z:ANGLE 11-12  
    INPut:POSITION:Z:DISTance 11-13  
    OUTPut:POSITION:X:ANGLE 15-6  
    OUTPut:POSITION:X:DISTance 15-7  
    OUTPut:POSITION:Y:ANGLE 15-8  
    OUTPut:POSITION:Y:DISTance 15-9  
    OUTPut:POSITION:Z:ANGLE 15-10  
    OUTPut:POSITION:Z:DISTance 15-11  
SENSe:CORRection 18-21  
SENSe:CURREnt:AC|DC:RANGE 18-28  
SENSe:FREQuency 18-39  
SENSe:POWER:AC|DC:RANGE 18-66  
SENSe:VOLTage:AC|DC:RANGE 18-87  
SOURce:CORRection 19-13  
SOURce:CURREnt:LEVel:IMMEDIATE 19-20  
SOURce:CURREnt:LEVel:TRIGgered 19-21  
SOURce:CURREnt:LIMit 19-22  
SOURce:FREQuency 19-44  
SOURce:POWER:LEVel:IMMEDIATE 19-63  
SOURce:POWER:LEVel:TRIGgered 19-64  
SOURce:POWER:LIMit 19-64  
SOURce:RESistance:LEVel:IMMEDIATE 19-75  
SOURce:RESistance:LEVel:TRIGgered 19-76  
SOURce:RESistance:LIMit 19-76  
SOURce:VOLTage:LEVel:IMMEDIATE 19-96  
SOURce:VOLTage:LEVel:TRIGgered 19-97  
SOURce:VOLTage:LIMit 19-98  
:ON  
    SENSe:FUNCtion 18-44  
ONCE  
    CALCulate 4-7, 4-12, 4-14, 4-21, 4-26, 4-28, 4-31  
    CALibration 5-7  
    DISPlay 8-11 - 8-12, 8-14 - 8-15  
    INPut:ATTenuation 11-3

INPut:GAIN 11-6  
OUTPut:FILTer 15-3  
SENSe 18-4, 18-7, 18-10 - 18-11, 18-27 - 18-28,  
18-31, 18-36, 18-38 - 18-40, 18-59 - 18-60, 18-62,  
18-66 - 18-67, 18-69 - 18-70, 18-72, 18-79, 18-82,  
18-86 - 18-87  
SENSe:ROSCillator:SOURce INTERNAL 19-80  
SOURce 19-18 - 19-20, 19-23, 19-43, 19-45,  
19-50, 19-52, 19-61 - 19-62, 19-66, 19-73, 19-79,  
19-85 - 19-86, 19-94 - 19-96, 19-99  
SYSTem 21-33  
TRACe | DATA 23-5  
TRIGGER Subsystem:ARM  
:SEQUence:LAYer:DELay 24-14  
TRIGGER Subsystem:ARM  
:SEQUence:LAYer:LEVel 24-16  
TRIGGER Subsystem:TRIGger  
:SEQUence:DELay 24-26  
TRIGGER Subsystem:TRIGger:SEQUence  
:LEVel 24-28  
:OPEN  
    MMEMory 14-6  
    ROUTE 17-3  
:OPERation  
    STATus 20-3  
Operation Complete Event  
    SYSTem:ERRor? 21-28  
OPERation Status Register 20-3  
:ORDinate  
    CALCulate:TRANSform:HISTogram 4-24  
:ORIENTATION  
    HCOPy:PAGE 10-14  
Other Subsystems  
    Instrument Model:Internal Routing 2-9  
OUTPut  
    Instrument Model:Signal Generation 2-4  
    SENSe:CORRection:IMPedance 18-19  
    SENSe:CORRection:LOSS 18-20  
    SOURce:CORRection:LOSS 19-14  
OUTPut Subsystem 15-1 - 15-14  
:OVERshoot  
    MEASure:FALL 3-13  
    MEASure:RISE 3-12

### P

:PACE ACK  
    SYSTem:COMM:SERial:RECeive 21-7  
    SYSTem:COMM:SERial:TRANsmiT 21-9  
:PACE NONE  
    SYSTem:COMM:SERial:RECeive 21-7  
    SYSTem:COMM:SERial:TRANsmiT 21-9

## 1999 SCPI Command Reference

:PACE XON	SOURce:MARKer 19-54
SYSTem:COMM:SERial:RECeive 21-7	
SYSTem:COMM:SERial:TRANsmiT 21-9	
:PACK	:POINts
MMEMory 14-6	CALCulate 4-10
PACKed 9-2	CALCulate:FILTer:GATE:FREQuency 4-14
:PAGE	CALCulate:FILTer:GATE:TIME 4-12
DISPlay:WINDOW:TEXT 8-10	CALCulate:SMOOthing 4-23
HCOPy 10-13 - 10-14	CALCulate:TRANSform:DISTance 4-28
:PARity	CALCulate:TRANSform:FREQuency 4-31
SYSTem:COMM:SERial:RECeive 21-8	CALCulate:TRANSform:HISTogram 4-24
SYSTem:COMM:SERial:TRANsmiT 21-9	CALCulate:TRANSform:TIME 4-26
:PATH	SENSe:SMOOthing 18-73
CALCulate 4-31 - 4-32	SENSe:SWEep 18-80
ROUTe 17-3	SOURce:SWEep 19-87
:PCL	TRACe   DATA 23-5 - 23-6
DISPlay:WINDOW:GRAPHics 8-8	:POINts?
:PDFunction	CALCulate:CLIMits:FLIMits 4-9
SENSe:FUNCtion:VOLTage 18-57	CALCulate:LIMit:CONTrol 4-19
:PDIVision	CALCulate:LIMit:LOWER 4-20
DISPlay:WINDOW:TRACe:X:SCALE 8-13	CALCulate:LIMit:REPort 4-21
DISPlay:WINDOW:TRACe:Y:SCALE 8-14	CALCulate:LIMit:UPPer 4-20
:PDUtcycle	MEMORY:TABLE:CONDition:MAGNitude 13-11
:MEASure 3-14	MEMORY:TABLE:CURREnt:MAGNitude 13-11
:PERiod	MEMORY:TABLE:CURREnt:PHASE 13-12
:MEASure 3-10	MEMORY:TABLE:FREQuency 13-15
SENSe:FUNCtion 18-53	MEMORY:TABLE:LOSS:MAGNitude 13-16
SOURce:PULSe 19-71	MEMORY:TABLE:LOSS:PHASE 13-16
:PERSistence	MEMORY:TABLE:POWER:MAGNitude 13-17
DISPlay:WINDOW:TRACe 8-11	MEMORY:TABLE:RESistance:MAGNitude 13-17
:PHASe	MEMORY:TABLE:VOLTage:MAGNitude 13-19
:MEASure 3-10	MEMORY:TABLE:VOLTage:PHASE 13-20
MEMORY Subsystem:TABLE:CURREnt 13-11	SENSe:LIST:DWELl 18-58
MEMORY Subsystem:TABLE:LOSS 13-16	SENSe:LIST:FREQuency 18-59
MEMORY Subsystem:TABLE:VOLTage 13-19	SENSe:LIST:SEQUence 18-59
SENSe:CORRection:LOSS:INPUT :OUTPut 18-21	SOURce:LIST:APRobe 19-49
SENSe:CORRection:OFFSet 18-21	SOURce:LIST:CONCurrent 19-50
SENSe:FUNCtion 18-53	SOURce:LIST:CONTrol:APower 19-50
SOURce:CORRection:LOSS:OUTPut 19-14	SOURce:LIST:CONTrol:BLOWer 19-50
SOURce:CORRection:OFFSet 19-13	SOURce:LIST:CONTrol:COMPressor 19-50
PHASe Subsystem	SOURce:LIST:CURRent 19-50
SOURce 19-56	SOURce:LIST:DEPTH:AM 19-49
:PLoSs	SOURce:LIST:DWELl 19-51
CALibration 5-4 - 5-5	SOURce:LIST:FREQuency 19-51
PM	SOURce:LIST:POWER 19-52
SENSe:FUNCtion 18-53	SOURce:LIST:PULM:STATE 19-51
PM Subsystem	SOURce:LIST:RESistance 19-52
SENSe 18-62	SOURce:LIST:RTIME 19-52
SOURce 19-57 - 19-59	SOURce:LIST:SEQUence 19-52
:POFLag	SOURce:LIST:TEMPerature 19-52
TRIGger Subsystem:INITiate 24-23	SOURce:LIST:VOLTage 19-53
:POINT	:POLarity
	INPut 11-7
	OUTPut Subsystem 15-5

## 1999 SCPI Command Reference

:POLarity :ALL NORMAl SOURce:DM 19-30	POWER Subsystem SENSe 18-63 - 18-67
:POLarity COMplement SOURce:PULSe 19-73	SOURce 19-60 - 19-67
:POLarity INVerted OUTPut:TTLTrg :ECLTrg 15-13 SOURce:AM 19-8	:PREFerence CALCulate:FORMAT:UPHase 4-17
SOURce:AM:EXTernal 19-7	Presentation Layer Measurement Instructions 3-7
SOURce:DM 19-30	:<presentation_layer> SENSe:FUNCTION 18-47
SOURce:FM 19-34	:PRESet STATus 20-4
SOURce:FM:EXTernal 19-33	SYSTem 21-35
SOURce:PM 19-58	:PREShoot MEASure:FALL 3-13
SOURce:PM:EXTernal 19-59	MEASure:RISE 3-12
SOURce:PULM 19-69	:PRF :MEASure:FREQuency 3-10
SOURce:PULM:EXTernal 19-69	PROGram Subsystem 16-1 - 16-6
SOURce:PULSe 19-73	:PROTection OUTPut Subsystem 15-12
:POLarity NEGative TRIGger Subsystem:ARM:SEQUence :LAYer:VIDeo:SSIGnal 24-21	SENSe:CURREnt:AC DC 18-26
TRIGger Subsystem:TRIGger:SEQUence :VIDeo:SSIGnal 24-32	SENSe:POWER:AC DC 18-65
:POLarity NORMAl OUTPut:TTLTrg :ECLTrg 15-13	SENSe:VOLTage:AC DC 18-85
SOURce:AM 19-8	SOURce:CURREnt 19-23
SOURce:AM:EXTernal 19-7	SOURce:POWER 19-65
SOURce:FM 19-34	SOURce:RESistance 19-77
SOURce:FM:EXTernal 19-33	SOURce:VOLTage 19-99
SOURce:PM 19-58	:PROTocol TRIGger Subsystem:ARM:SEQUence
SOURce:PM:EXTernal 19-59	:LAYer 24-17
SOURce:PULM:EXTernal 19-69	TRIGger Subsystem:TRIGger:SEQUence 24-28
SOURce:PULSe 19-73	:PROTocol ASYNchronous OUTPut:TTLTrg :ECLTrg 15-13
SOURce:PULSe Modulation 19-69	:PROTocol SSYNchronous OUTPut:TTLTrg :ECLTrg 15-13
:POLarity POSitive TRIGger Subsystem:ARM:SEQUence :LAYer:VIDeo:SSIGnal 24-21	:PROTocol SYNchronous OUTPut:TTLTrg :ECLTrg 15-13
TRIGger Subsystem:TRIGger:SEQUence :VIDeo:SSIGnal 24-32	:PSDensity SENSe:FUNCTION:POWER 18-54
:POLarization INPut 11-7	:PTPeak MEASure 3-14
OUTPut Subsystem 15-5	SENSe:CURREnt:AC DC:RANGE 18-28
:POSIon OUTPut Subsystem 15-5	SENSe:POWER:AC DC:RANGE 18-66
:POSIon INPut 11-7 - 11-13	SENSe:VOLTage:AC DC:RANGE 18-87
:POWer	:PTRansition STATus 20-4, 20-7
:MEASure 3-8	:PULM SOURce:LIST 19-51
MEMory:TABLE 13-17	PULse Modulation Subsystem SOURce 19-68 - 19-70
SENSe:FUNCTION 18-53	PULse Subsystem SOURce 19-71 - 19-73
SOURce:LIST 19-52	
UNIT 25-1	
Power On Event	
SYSTem:ERRor? 21-27	

## 1999 SCPI Command Reference

:PWIDth  
  MEASure 3-13

### Q

:Q INverted  
  SOURce:DM:POLarity :ALL  
    NORMal|INVerted 19-30  
:Q<n> NRMal  
  SOURce:DM:POLarity :ALL  
    NORMal|INVerted 19-30  
:QClock INVerted  
  SOURce:DM:POLarity :ALL  
    NORMal|INVerted 19-30  
:QClock NORMal  
  SOURce:DM:POLarity :ALL  
    NORMal|INVerted 19-30  
:QCORrection  
  SOURce:DM:FILTer 19-27  
:QUADrant  
  HCOPy:PAGE:DIMensions 10-14  
:QUADrature  
  SOURce:DM 19-28  
:QUERy  
  VXI:WSPRrotocol 26-14  
:Query Error  
  SYSTem:ERRor? 21-26  
Querying the Data Flow  
  Instrument Model:Internal Routing 2-14  
:QUESTIONable  
  STATus 20-7  
QUESTIONable Data 20-7

### R

:R  
  DISPlay:WINDOW:TRACe 8-15  
:RANGe  
  CALCulate:TRANSform:HISTogram 4-24  
  SENSe:AM:DEPTH 18-4  
  SENSe:CURRent:AC:DC 18-27  
  SENSe:FM:DEViation 18-36  
  SENSe:FREQuency 18-39  
  SENSe:PM:DEViation 18-62  
  SENSe:POWER:AC:DC 18-65  
  SENSe:RESistance 18-69  
  SENSe:VOLTage:AC:DC 18-86  
  SOURce:CURRent 19-23  
  SOURce:POWER 19-66  
  SOURce:RESistance 19-79  
  SOURce:VOLTage 19-99  
:RATio

SENSe:BANDwidth|BWIDth:RESolution 18-10  
SENSe:BANDwidth|BWIDth:VIDeo 18-11  
:RDEVice  
  SYSTem:COMMunicate:GPIB 21-5  
  VXI:WSPRrotocol:COMMAND 26-11  
:RDEVice?  
  VXI:WSPRrotocol:QUERy 26-15  
READ  
  Measurement Instructions 3-4  
:READ?  
  VXI:REGister 26-7  
REAL 9-2  
:REALtime  
  SENSe:SWEep 18-81  
REAR  
  SENSe:FUNCTION:SPEEd 18-55  
:RECeive  
  SYSTem:COMM:SERial 21-7  
:RECeive?  
  VXI:WSPRrotocol:MESSAge 26-13  
:REFERENCE  
  SENSe:CURREnt:AC:DC 18-28  
  SENSe:POWER:AC:DC 18-67  
  SENSe:RESistance 18-70  
  SENSe:VOLTage:AC:DC 18-87  
  SOURce:CURREnt 19-23  
  SOURce:MARKer 19-55  
  SOURce:PHASe 19-56  
  SOURce:POWER 19-66  
  SOURce:RESistance 19-79  
  SOURce:VOLTage 19-99  
:REGister  
  VXI 26-7  
REPeat  
  CALCulate:AVERage:TCONtrol 4-7  
:REPort  
  CALCulate:LIMit 4-20  
Request Control Event  
  SYSTem:ERRor? 21-28  
:Requirements  
  Introduction 1-1  
:RESET?  
  VXI 26-8  
:RESistance  
  :MEASure 3-8  
  MEMORY:TABLE 13-17  
  SENSe:FUNCTION 18-55  
  SOURce:LIST 19-52  
RESistance Subsystem  
  SENSe 18-68 - 18-70  
  SOURce 19-74 - 19-79  
:RESolution

## 1999 SCPI Command Reference

HCOPy:DEVice 10-7  
SENSe:BANDwidth|BWIDth 18-10  
SENSe:CURREnt:AC|DC 18-28  
SENSe:FREQuency 18-40  
SENSe:POWER:AC|DC 18-67  
SENSe:RESistance 18-70  
SENSe:VOLTage:AC|DC 18-87  
SOURce:FREQuency 19-45  
:RESPonse?  
    VXI:WSPRotocol 26-17  
:RGB  
    DISPlay:CMAP:COLor 8-5  
    HCOPy:DEVice:CMAP:COLor 10-5  
:RHANdlers  
    VXI:WSPRotocol:COMMAND 26-11  
:RHANdlers?  
    VXI:WSPRotocol:QUERy 26-15  
:RHLIne  
    VXI:WSPRotocol:COMMAND 26-12  
:RHLIne?  
    VXI:WSPRotocol:QUERy 26-15  
:RIGHT  
    DISPlay:WINDOW:TRACe:X:SCALe 8-13  
:RILIne  
    VXI:WSPRotocol:COMMAND 26-12  
:RILIne?  
    VXI:WSPRotocol:QUERy 26-16  
:RINTerrupter  
    VXI:WSPRotocol:COMMAND 26-12  
:RINTerrupter?  
    VXI:WSPRotocol:QUERy 26-16  
:RISE  
    MEASure 3-12  
:RLEVel  
    DISPlay:WINDOW:TRACe:Y:SCALe 8-14  
:RLIne  
    DISPlay:WINDOW:TRACe:Y 8-13  
:RMODId  
    VXI:WSPRotocol:COMMAND 26-12  
:RMODId?  
    VXI:WSPRotocol:QUERy 26-16  
:ROSCillator  
    OUTPut Subsystem 15-12  
ROSCillator Subsystem  
    SENSe 18-71 - 18-72  
    SOURce 19-80 - 19-81  
:ROTation  
    CONTrol 6-4  
ROUTE Subsystem 17-1 - 17-6  
:RPERror  
    VXI:WSPRotocol:COMMAND 26-12  
:RPERror?

VXI:WSPRotocol:QUERy 26-16  
:RPOSition  
    DISPlay:WINDOW:TRACe:Y:SCALe 8-15  
:RPRotocol  
    VXI:WSPRotocol:COMMAND 26-12  
:RPRotocol?  
    VXI:WSPRotocol:QUERy 26-16  
:RSARea  
    VXI:WSPRotocol:COMMAND 26-13  
:RSARea?  
    VXI:WSPRotocol:QUERy 26-16  
:RSTB  
    VXI:WSPRotocol:COMMAND 26-12  
:RSTB?  
    VXI:WSPRotocol:QUERy 26-16  
:RTIMe  
    MEASure 3-12  
    SOURce:LIST 19-52  
:RTS IBFull  
    SYSTem:COMM:SERial:CONTrol 21-6  
:RTS OFF  
    SYSTem:COMM:SERial:CONTrol 21-6  
:RTS ON  
    SYSTem:COMM:SERial:CONTrol 21-6  
:RTS RFR  
    SYSTem:COMM:SERial:CONTrol 21-6  
:RTS STAndard  
    SYSTem:COMM:SERial:CONTrol 21-6  
:RVELOCITY  
    SENSe:CORRection 18-22  
    SOURce:CORRection 19-15

## S

:S11  
    SENSe:FUNCtion:POWer 18-54  
:S12  
    SENSe:FUNCtion:POWer 18-54  
:S21  
    SENSe:FUNCtion:POWer 18-54  
:S22  
    SENSe:FUNCtion:POWer 18-54  
:SAMPLE  
    ROUTe 17-4  
:SAVE  
    SENSe:CORRection:COLLect 18-18  
    SOURce:CORRection:COLLect 19-12  
:SB  
    See SENSe:FUNCtion  
:SBITS  
    SYSTem:COMM:SERial:RECeive 21-8  
    SYSTem:COMM:SERial:TRANsmiT 21-9

## 1999 SCPI Command Reference

:SCALar	SYSTem:COMMUnicatE:GPIB 21-5
MEASure 3-7	
:SCALE	:SEND
DISPlay:WINDOW:TRACe:R 8-15	VXI:WSPRotocol:MESSAge 26-13
DISPlay:WINDOW:TRACe:X 8-12	
DISPlay:WINDOW:TRACe:Y 8-13	SENSe
HCOPy:PAGe 10-14	Instrument Model:Measurement Function 2-4
:SCAN	SENSe:DATA? 18-42
ROUTe 17-6	SENSe Subsystem 18-1 - 18-90
:SDUMp	:SENSitivity
HCOPy 10-15 - 10-16	SOURce:AM 19-8
:SEArch	SOURce:FM 19-34
SOURce:CURREnt:ALC 19-18	SOURce:PM 19-57
SOURce:POWer:ALC 19-61	Sensor Function Selection
SOURce:VOLTage:ALC 19-94	Instrument Model:Internal Routing 2-9
:SECurity	:<sensor_function>
SYSTem 21-36	SENSe:FUNCtion 18-45
:SELect	:SEQUence
INSTRument 12-4	SENSe:LIST 18-59
MEMORY:TABLE 13-18	SOURce:LIST 19-52
SENSe:CORRection:CSET 18-18	TRIGger Subsystem:ARM 24-12
SOURce:CORRection:CSET 19-12	TRIGger Subsystem:INITiate:CONTinuous 24-22
TRIGger Subsystem:ARM:SEQUence	TRIGger Subsystem:INITiate:IMMEDIATE 24-23
:LAYer:VIDeo:LINE 24-20	TRIGger Subsystem:TRIGger 24-23
TRIGger Subsystem:TRIGger:SEQUence	:Sequence Event Use
:VIDeo:LINE 24-32	TRIGger Subsystem 24-4
VXI 26-9	:SERial
:SELect ALL	SYSTem:COMMUnicatE 21-6
TRIGger Subsystem:ARM:SEQUence	:SET
:LAYer:VIDeo:FIELD 24-19	SYSTem 21-36
TRIGger Subsystem:TRIGger:SEQUence	:SHApe
:VIDeo:FIELD 24-31	SOURce:FUNCtion 19-47
:SELect EVEN	:SHApe LINEar
TRIGger Subsystem:ARM:SEQUence	SENSe:DETector 18-31
:LAYer:VIDeo:FIELD 24-19	:SHApe LOGarithmic
TRIGger Subsystem:TRIGger	SENSe:DETector 18-31
:SEQUence:VIDeo:FIELD 24-31	:SIGNal
:SELect NUMBER	TRIGger Subsystem:ARM
TRIGger Subsystem:ARM	:SEQUence:LAYer 24-17
:SEQUence:LAYer:VIDeo:FIELD 24-19	TRIGger Subsystem:TRIGger:SEQUence 24-29
TRIGger Subsystem:TRIGger	Signal Generation
:SEQUence:VIDeo:FIELD 24-31	Instrument Model 2-4
:SELect ODD	Signal Routing
TRIGger Subsystem:ARM	Instrument Model 2-2
:SEQUence:LAYer:VIDeo:FIELD 24-19	Signal Status Register 20-7
TRIGger Subsystem:TRIGger	Simple Measurements
:SEQUence:VIDeo:FIELD 24-31	Measurement Function Layer 3-9
:SELected	:SIZE
CALCulate:MATH:EXPReSSion:DELeTe 4-22	DISPlay:WINDOW:GEOMetry 8-7
PROGram 16-2 - 16-3	HCOPy:PAGe 10-15
PROGram:SELected:DELeTe 16-2	:SLEW
:SELF	SOURce:CURREnt 19-24
	SOURce:POWer 19-66
	SOURce:RESistance 19-77

## 1999 SCPI Command Reference

SOURce:VOLTage 19-100	SOURce:AM 19-8
:SLModid	SOURce:DM 19-27
VXI:WSPRotocol:COMMAND 26-13	SOURce:DM:CLOCk 19-31
:SLModid?	SOURce:DM:FILTter 19-27
VXI:WSPRotocol:QUERy 26-17	SOURce:DM:FRAMe 19-30
:SLOCK	SOURce:FM 19-34
VXI:WSPRotocol:COMMAND 26-13	SOURce:PHASe 19-56
:SLOPe	SOURce:PM 19-58
SENSe:CORRection 18-20	SOURce:PULM 19-70
SOURce:CORRection 19-13	:SOURce INTernal
:SLOPe EITHer	SENSe:ROSCillator 18-71, 19-80
TRIGger Subsystem:ARM	SOURce:AM 19-8
:SEQUence:LAYer 24-18	SOURce:CURREnt:ALC 19-19
TRIGger Subsystem:TRIGger:SEQUence 24-29	SOURce:DM:CLOCk 19-31
:SLOPe NEGative	SOURce:DM:FILTter 19-27
TRIGger Subsystem:ARM	SOURce:DM:FRAMe 19-30
:SEQUence:LAYer 24-18	SOURce:PHASe 19-56
TRIGger Subsystem:TRIGger:SEQUence 24-29	SOURce:POWER:ALC 19-62
:SLOPe POSitive	SOURce:VOLTage:ALC 19-95
TRIGger Subsystem:ARM	:SOURce INTernal,EXTernal
:SEQUence:LAYer 24-18	SOURce:FM 19-34
TRIGger Subsystem:TRIGger:SEQUence 24-29	SOURce:PM 19-58
:SMOothing	SOURce:PULM 19-70
CALculate 4-23	:SOURce MMHead
SMOothing Subsystem	SOURce:CURREnt:ALC 19-19
SENSe 18-73	SOURce:POWER:ALC 19-62
:SNDRatio	SOURce:VOLTage:ALC 19-95
SENSe:FUNCTION & DATA:FUNCTION:POWER	:SOURce NONE
18-51 - 18-53, 18-55	SENSe:ROSCillator 18-71, 19-80
SOURce	SOURce:DM:CLOCk 19-31
CALibration 5-6	:SOURce PMETER
Instrument Model:Signal Generation 2-5	SOURce:CURREnt:ALC 19-19
OUTPut:TTLTrg:ECLTrg 15-13	SOURce:POWER:ALC 19-62
TRIGger Subsystem:ARM	SOURce:VOLTage:ALC 19-95
:SEQUence:LAYer 24-18	:SOURce PRBS
TRIGger Subsystem:TRIGger:SEQUence 24-29	SOURce:DM 19-27
:SOURce ,INTernal	SOURce:SubSystem 19-1 - 19-102
SOURce:FM 19-34	:SPACing
SOURce:PM 19-58	SENSe:POWER:ACHannel 18-63
SOURce:PULM 19-70	:SPACing LINear
:SOURce CALibrate	DISPLAY:WINDOW:TRACe:R 8-16
SOURce:DM 19-27	DISPLAY:WINDOW:TRACe:Y 8-15
:SOURce CLK10	SENSe:SWEep 18-81
SENSe:ROSCillator 18-71, 19-80	SOURce:SWEep 19-86
:SOURce CLK100	:SPACing LOGarithmic
SENSe:ROSCillator 18-71, 19-80	DISPLAY:WINDOW:TRACe:R 8-16
:SOURce DIODE	DISPLAY:WINDOW:TRACe:Y 8-15
SOURce:CURREnt:ALC 19-19	SENSe:SWEep 18-81
SOURce:POWER:ALC 19-62	SOURce:SWEep 19-86
SOURce:VOLTage:ALC 19-95	:SPAN
:SOURce EXTERNAL	CALCulate:FILTter:GATE:FREQuency 4-14
SENSe:ROSCillator 18-71, 19-80	CALCulate:FILTter:GATE:TIME 4-12

## 1999 SCPI Command Reference

CALCulate:GDAperture 4-17	INPut:FILTter:AWEighting 11-5
CALCulate:TRANSform:DISTance 4-28	INPut:FILTter:HPASs 11-5
CALCulate:TRANSform:FREQuency 4-30	INPut:FILTter:LPASs 11-5
CALCulate:TRANSform:TIME 4-26	INPut:GAIN 11-6
SENSe:FREQuency 18-40	INPut:OFFSet 11-7
SOURce:CURREnt 19-24	INPut:POSIon:X:ANGLE:LIMit 11-8
SOURce:FREQuency 19-45	INPut:POSIon:X:DISTance:LIMit 11-9
SOURce:POWER 19-67	INPut:POSIon:Y:ANGLE:LIMit 11-10
SOURce:RESistance 19-77	INPut:POSIon:Y:DISTance:LIMit 11-11
SOURce:VOLTage 19-100	INPut:POSIon:Z:ANGLE:LIMit 11-12
:SPeed	INPut:POSIon:Z:DISTance:LIMit 11-13
HCOPy:DEVice 10-7	INSTRument 12-4
SENSe:FUNCTION 18-55	MEMory 13-9
SPeed Subsystem	MMEMory 14-4
SOURce 19-82 - 19-84	OUTPut Subsystem 15-14
:SREGister	OUTPut:FILTter:EXTernal 15-4
FORMAT 9-3 - 9-4	OUTPut:FILTter:HPASs 15-4
SSB Subsystem	OUTPut:FILTter:LPASs 15-4
SENSe 18-74	OUTPut:POSIon:X:ANGLE:LIMit 15-6
:SSIGnal	OUTPut:POSIon:X:DISTance:LIMit 15-7
TRIGger Subsystem:ARM	OUTPut:POSIon:Y:ANGLE:LIMit 15-8
:SEQUence:LAYer:VIDeo 24-20	OUTPut:POSIon:Y:DISTance:LIMit 15-9
TRIGger Subsystem:TRIGger	OUTPut:POSIon:Z:ANGLE:LIMit 15-10
:SEQUence:VIDeo 24-32	OUTPut:POSIon:Z:DISTance:LIMit 15-11
STABilize Subsystem	OUTPut:PROtection 15-12
SENSe 18-75 - 18-76	OUTPut:ROSCillator 15-12
:Standard SEQuences	OUTPut:TTLTrg ECLTrg 15-13
TRIGger Subsystem:Expanded	PROGram 16-3, 16-5
Capability Trigger Model 24-7	SENSe 18-7, 18-18 - 18-19, 18-21 - 18-23, 18-26,
:STARt	18-28, 18-33 - 18-35, 18-65, 18-67, 18-70, 18-73,
CALCulate:FILTter:GATE:FREQuency 4-13	18-81, 18-85, 18-87
CALCulate:FILTter:GATE:TIME 4-11	SOURce 19-8, 19-11 - 19-15, 19-18, 19-22 -
CALCulate:TRANSform:DISTance 4-28	19-24, 19-27 - 19-29, 19-34, 19-51, 19-55, 19-58,
CALCulate:TRANSform:FREQuency 4-30	19-61, 19-65 - 19-66, 19-70, 19-72, 19-77, 19-79,
CALCulate:TRANSform:TIME 4-25	19-94, 19-98 - 19-100
SENSe:FREQuency 18-41	CALCulate 4-13
SOURce:CURREnt 19-24	SYSTem 21-3 - 21-4, 21-36
SOURce:FREQuency 19-46	TRIGger Subsystem:ARM
SOURce:POWER 19-67	:SEQUence:LAYer:FILTter:HPASs 24-15
SOURce:RESistance 19-78	TRIGger Subsystem:ARM
SOURce:VOLTage 19-100	:SEQUence:LAYer:FILTter:LPASs 24-15
SYSTem:COMM:SERial:RECeive 21-8	TRIGger Subsystem:TRIGger
:STATE	:SEQUence:ATRigger 24-24
CALCulate 4-7, 4-10 - 4-11, 4-18 - 4-20, 4-23,	TRIGger Subsystem:TRIGger
4-25, 4-27, 4-29	:SEQUence:FILTter:HPASs 24-26
CALibration 5-6	TRIGger Subsystem:TRIGger
CONTrol 6-1 - 6-2	:SEQUence:FILTter:LPASs 24-27
DISPlay 8-6, 8-9 - 8-11	:STATE?
HCOPy 10-8 - 10-13	MEMory 13-5, 13-9
INPut 11-14	ROUTE 17-2
INPut:ATTenuation 11-3	SENSe 18-45
INPut:BIAS 11-4	status reporting

## 1999 SCPI Command Reference

OPERation 20-3  
QUEstionable 20-7  
STATus Subsystem 20-1 - 20-8  
.STEP  
    SENSe:SWEep 18-82  
    SOURce:PHASe:ADJust 19-56  
    SOURce:SWEep 19-87  
.STIMulus IMPulse  
    CALCulate:TRANsform:DISTance 4-27  
    CALCulate:TRANsform:FREQuency 4-30  
    CALCulate:TRANsform:TIME 4-25  
.STIMulus STEP  
    CALCulate:TRANsform:DISTance 4-27  
    CALCulate:TRANsform:FREQuency 4-30  
    CALCulate:TRANsform:TIME 4-25  
.STOP  
    CALCulate:FILTER:GATE:FREQuency 4-13  
    CALCulate:FILTER:GATE:TIME 4-11  
    CALCulate:TRANsform:DISTance 4-28  
    CALCulate:TRANsform:FREQuency 4-30  
    CALCulate:TRANsform:TIME 4-26  
    SENSe:FREQuency 18-41  
    SOURce:CURREnt 19-25  
    SOURce:FREQuency 19-46  
    SOURce:POWER 19-67  
    SOURce:RESistance 19-78  
    SOURce:VOLTage 19-101  
    SYStem:COMM:SERial:RECeive 21-8  
.STORe  
    MMEMory 14-4  
.STRing  
    PROGram:EXPLicit 16-6  
    PROGram:SELected 16-4  
.Subservient Sequences  
    TRIGger Subsystem:Expanded Capability Trigger Model 24-7  
.SUModid  
    VXI:WSPRotocol:COMMAND 26-13  
.SUModid?  
    VXI:WSPRotocol:QUERy 26-17  
SWEep Subsystem  
    SENSe 18-77 - 18-83  
    SOURce 19-85 - 19-87  
.SYNTAX?  
    SYStem:HELP 21-31  
SYStem Subsystem 21-1 - 21-40

**T**

:TABLE  
    MEMory 13-10  
    MEMory:CLEar 13-6

MEMORY:COPY 13-7  
MEMORY:EXChange 13-8  
.TABLE?  
    MEMORY:CATalog 13-6  
    MEMORY:FREE 13-9  
.TCONstant  
    SENSe:FILTER:DEMPhasis 18-34  
.TCONrol EXPonential  
    CALCulate:AVERage 4-7  
    SENSe:AVERage 18-7  
.TCONrol MOVing  
    CALCulate:AVERage 4-7  
    SENSe:AVERage 18-7  
.TCONrol NORMal  
    CALCulate:AVERage 4-7  
    SENSe:AVERage 18-7  
.TCONrol REPeat  
    CALCulate:AVERage 4-7  
    SENSe:AVERage 18-7  
.TDStamp  
    HCOPy:ITEM 10-11  
.TEMPerature  
    SOURce 19-88 - 19-92  
    SOURce:LIST 19-52  
    UNIT 25-2  
.TERMinals  
    ROUTe 17-6  
TEST Subsystem 22-1 - 22-2  
.TEXT  
    DISPlay:WINDOW 8-9  
    HCOPy:ITEM:LABEL 10-10  
    HCOPy:ITEM:WINDOW 10-11  
.THD  
    SENSe:FUNCTION:POWER 18-51 - 18-53, 18-55  
.THRehold  
    SOURce:DM 19-29  
    SYStem:COMM:SERial:RECeive 21-8  
.TIME  
    CALCulate:FILTER:GATE 4-11  
    CALCulate:TRANsform 4-25  
    MEASure:FALL 3-13  
    MEASure:RISE 3-12  
    SENSe:CORRection:EDELay 18-19  
    SENSe:SWEep 18-82  
    SOURce:CORRection:EDELay 19-14  
    SOURce:SWEep 19-85  
    SYStem 21-37  
    SYStem:BEEPer 21-4  
    UNIT 25-2  
Time Domain Waveform Measurements 3-10  
.TIMER  
    TRIGger Subsystem:ARM

## 1999 SCPI Command Reference

:SEQuence:LAYer 24-19  
TRIGger Subsystem:TRIGger:SEQuence 24-30  
TINTerval  
    SENSe:FUNCtion 18-56  
:TMAXimum  
    MEASure 3-14  
:TMINimum  
    MEASure 3-14  
:TOP  
    DISPlay:WINDOW:TRACe:Y:SCALE 8-15  
TOTalize  
    SENSe:FUNCtion 18-56  
TPLoss  
    SENSe:FUNCtion 18-56  
:TRACe 23-1 - 23-6  
    DISPlay:WINDOW 8-10  
    HCOPy:ITEM:WINDOW 10-12  
    MMEMory:LOAD and :STORE 14-5  
    MMEMory:LOAD and  
:STORE:DINTerchange 14-4  
:TRACK  
    SENSe:BANDwidth|BWIDth:RESolution 18-10  
:TRAiling  
    SOURce:PULSe:TRANSition 19-73  
:TRANSform  
    CALCulate 4-23 - 4-30  
:TRANSition  
    SOURce:PULSe 19-72  
:TRANsmi<sup>t</sup>  
    SYSTem:COMM:SERial 21-8  
TRIGger 2-5  
    Instrument Model 2-5  
    TRIGger Subsystem 24-23 - 24-32  
    VXI:WSPRotocol:COMMAND 26-13  
TRIGger Subsystem 24-1 - 24-32  
:TRIGgered  
    SOURce:CURREnt:LEVel 19-21  
    SOURce:POWER:LEVel 19-63  
    SOURce:RESistance:LEVel 19-75  
    SOURce:VOLTage:LEVel 19-97  
:TRIPped?  
    OUTPUT:PROTection 15-12  
    SENSe:CURREnt:AC|DC:PROTection 18-26  
    SENSe:POWER:AC|DC:PROTection 18-65  
    SENSe:VOLTage:AC|DC:PROTection 18-85  
    SOURce:CURREnt:PROTection 19-23  
    SOURce:POWER:PROTection 19-66  
    SOURce:RESistance:PROTection 19-77  
    SOURce:VOLTage:PROTection 19-99  
:TTL  
    TRIGger Subsystem:ARM  
    :SEQuence:LAYer 24-19

TRIGger Subsystem:TRIGger:SEQuence 24-30  
:TTLTrg  
    OUTPut Subsystem 15-12  
:TYPE  
    CALCulate 4-18  
    INPUT 11-14  
    OUTPut Subsystem 15-14  
    SENSe:AM 18-4  
    SOURce:AM 19-9  
:TYPE BESSel  
    OUTPut:FILTter:HPASs 15-4  
    OUTPut:FILTter:LPASs 15-4  
:TYPE BPASs  
    CALCulate:FILTter:GATE:FREQuency 4-13  
    CALCulate:FILTter:GATE:TIME 4-11  
    CALCulate:TRANSform:DISTance 4-27  
    CALCulate:TRANSform:FREQuency 4-29  
    CALCulate:TRANSform:TIME 4-25  
:TYPE CHEByshev  
    OUTPut:FILTter:HPASs 15-4  
    OUTPut:FILTter:LPASs 15-4  
:TYPE COMPlEx  
    CALCulate:AVERage 4-8  
    SENSe:AVERage 18-8  
:TYPE CURRent  
    INPUT:BIAS 11-4  
:TYPE EDGE  
    TRIGger Subsystem:ARM  
    :SEQuence:LAYer 24-19  
    TRIGger Subsystem:TRIGger:SEQuence 24-31  
:TYPE ENVelope  
    CALCulate:AVERage 4-8  
    SENSe:AVERage 18-8  
:TYPE EVEN  
    SYSTem:COMM:SERial:RECeive 21-8  
    SYSTem:COMM:SERial:TRANsmi<sup>t</sup> 21-9  
:TYPE FLATtop  
    SENSe:WINDOW 18-89  
:TYPE HAMMING  
    SENSe:WINDOW 18-89  
:TYPE HANNing  
    SENSe:WINDOW 18-89  
:TYPE IGNore  
    SYSTem:COMM:SERial:RECeive 21-8  
:TYPE LPASs  
    CALCulate:TRANSform:DISTance 4-27  
    CALCulate:TRANSform:FREQuency 4-29  
    CALCulate:TRANSform:TIME 4-25  
:TYPE MAXimum  
    CALCulate:AVERage 4-8  
    SENSe:AVERage 18-8  
:TYPE MINimum

## 1999 SCPI Command Reference

CALCulate:AVERage 4-8  
SENSe:AVERage 18-8  
:TYPE NONE  
  SYSTem:COMM:SERial:RECeive 21-8  
  SYSTem:COMM:SERial:TRANsmit 21-9  
:TYPE NOTCh  
  CALCulate:FILTer:GATE:FREQuency 4-13  
  CALCulate:FILTer:GATE:TIME 4-11  
:TYPE ODD  
  SYSTem:COMM:SERial:RECeive 21-8  
  SYSTem:COMM:SERial:TRANsmit 21-9  
:TYPE ONE  
  SYSTem:COMM:SERial:RECeive 21-8  
  SYSTem:COMM:SERial:TRANsmit 21-9  
:TYPE RECTangular  
  SENSe:WINDOW 18-89  
:TYPE RMS  
  CALCulate:AVERage 4-8  
  SENSe:AVERage 18-8  
:TYPE SCALAR  
  CALCulate:AVERage 4-8  
  SENSe:AVERage 18-8  
:TYPE UNIFORM  
  SENSe:WINDOW 18-89  
:TYPE VIDEO  
  TRIGger Subsystem:ARM  
  :SEQUence:LAYer 24-19  
  TRIGger Subsystem:TRIGger:SEQUence 24-31  
:TYPE VOLTage  
  INPut:BIAS 11-4  
:TYPE ZERO  
  SYSTem:COMM:SERial:RECeive 21-8  
  SYSTem:COMM:SERial:TRANsmit 21-9  
:TYPE?  
  MEMory 13-9  
:TZONe  
  SYSTem 21-38

### U

UINTeger 9-2  
:UNIT  
  HCOPy:DEvice:RESolution 10-7  
  HCOPy:DEvice:SPEed 10-7  
  HCOPy:PAGE 10-15  
UNIT Subsystem 25-1 - 25-2  
:UPHase  
  CALCulate:FORMAT 4-16  
:UPPer  
  CALCulate:LIMit 4-19  
  SENSe:AM:DEPTH:RANGE 18-4  
  SENSe:CURREnt:AC:DC:RANGE 18-27

SENSe:FM:DEViation:RANGE 18-36  
SENSe:FREQuency:RANGE 18-39  
SENSe:PM:DEViation:RANGE 18-62  
SENSe:POWER:AC|DC:RANGE 18-65  
SENSe:POWER:ACHannel:SPACing 18-64  
SENSe:RESistance:RANGE 18-69  
SENSe:VOLTage:AC|DC:RANGE 18-86  
:URIGHT  
  DISPlay:WINDOW:GEOMetry 8-7  
  HCOPy:PAGE:DIMensions 10-14  
User Request Event  
  SYSTem:ERRor? 21-27

**V**

:VALUE  
  CALibration 5-6  
  TRACe | DATA:DATA 23-3  
:VCDevice  
  CONTrol 6-4 - 6-6  
:VELOCITY  
  INPut:POSITION:X:ANGLE 11-9  
  INPut:POSITION:X:DISTance 11-10  
  INPut:POSITION:Y:ANGLE 11-11  
  INPut:POSITION:Y:DISTance 11-12  
  INPut:POSITION:Z:ANGLE 11-13  
  INPut:POSITION:Z:DISTance 11-14  
  OUTPut:POSITION:X:ANGLE 15-6  
  OUTPut:POSITION:X:DISTance 15-7  
  OUTPut:POSITION:Y:ANGLE 15-8  
  OUTPut:POSITION:Y:DISTance 15-9  
  OUTPut:POSITION:Z:ANGLE 15-10  
  OUTPut:POSITION:Z:DISTance 15-11  
:VERBOSE?  
  VXI:CONFigure:HIERarchy? 26-4  
  VXI:CONFigure:INFormation? 26-6  
  VXI:REGister:READ? 26-8  
  VXI:RESET? 26-9  
:VERSion?  
  SYSTem 21-38 - 21-40  
:VERTical  
  INPut:Polarization 11-7  
  OUTPut:Polarization 15-5  
:VIDEO  
  SENSe:BANDwidth|BWIDth 18-10  
  TRIGger Subsystem:ARM  
  :SEQUence:LAYer 24-19  
  TRIGger Subsystem:TRIGger:SEQUence 24-31  
:VOLTage  
  :MEASure 3-8  
  INPut:BIAS 11-4  
  MEMory:TABLE 13-19

# 1999 SCPI Command Reference

SENSe:FUNCtion 18-56  
SOURce:LIST 19-53  
UNIT 25-1  
VOLTage Subsystem  
    SENSe 18-84 - 18-88  
    SOURce 19-93 - 19-102  
:VOLUME  
    SYSTem:BEEPer 21-4  
:VXI ASYNchronous  
    TRIGger Subsystem:ARM  
        :SEQuence:LAYER:PROTocol 24-17  
    TRIGger Subsystem:TRIGger  
        :SEQuence:PROTocol 24-29  
:VXI SSYNchronous  
    TRIGger Subsystem:ARM  
        :SEQuence:LAYER:PROTocol 24-17  
    TRIGger Subsystem:TRIGger  
        :SEQuence:PROTocol 24-29  
VXI Subsystem 26-1 - 26-18  
:VXI SYNChronous  
    TRIGger Subsystem:ARM  
        :SEQuence:LAYER:PROTocol 24-17  
    TRIGger Subsystem:TRIGger  
        :SEQuence:PROTocol 24-29

## W

:WAIT  
    PROGram:EXPLicit 16-6  
    PROGram:SElected 16-4  
:WARMup  
    CALibration 5-6  
:WAVeguide  
    SENSe:CORRection:RVELOCITY 18-22  
    SOURce:CORRection:RVELOCITY 19-15  
:WIDTH  
    HCOPY:PAGE 10-15  
    OUTPut:TTLTrg|ECLTrg 15-13  
    SOURce:PULSe 19-71  
:WINDOW  
    DISPLAY 8-6 - 8-16  
    HCOPY:ITEM 10-11  
:WINDOW FLATtop  
    CALCulate:FILTter:GATE:FREQUENCY 4-14  
    CALCulate:FILTter:GATE:TIME 4-12  
    CALCulate:TRANSform:DISTANCE 4-29  
    CALCulate:TRANSform:FREQUENCY 4-31  
    CALCulate:TRANSform:TIME 4-26  
:WINDOW HAMMING  
    CALCulate:FILTter:GATE:FREQUENCY 4-14  
    CALCulate:FILTter:GATE:TIME 4-12  
    CALCulate:TRANSform:DISTANCE 4-29

CALCulate:TRANSform:FREQUENCY 4-31  
CALCulate:TRANSform:TIME 4-26  
:WINDOW RECTangular  
    CALCulate:FILTter:GATE:FREQUENCY 4-14  
    CALCulate:FILTter:GATE:TIME 4-12  
    CALCulate:TRANSform:DISTANCE 4-29  
    CALCulate:TRANSform:FREQUENCY 4-31  
    CALCulate:TRANSform:TIME 4-26  
WINDOW Subsystem  
    SENSe 18-89 - 18-90  
:WINDOW UNIFORM  
    CALCulate:FILTter:GATE:FREQUENCY 4-14  
    CALCulate:FILTter:GATE:TIME 4-12  
    CALCulate:TRANSform:DISTANCE 4-29  
    CALCulate:TRANSform:FREQUENCY 4-31  
    CALCulate:TRANSform:TIME 4-26  
:WRITE  
    VXI:REGister 26-8  
:WSPRotocol  
    VXI 26-9

## X

:X  
    DISPlay:WINDOW:TRACe 8-12  
    INPut:POSition 11-8  
    OUTPut:POSition 15-5  
:XCURRENT  
    SENSe:FUNCtion:<presentation\_layer> 18-47  
:XFREquency  
    SENSe:FUNCtion:<presentation\_layer> 18-47  
:XNONE  
    SENSe:FUNCtion:<presentation\_layer> 18-47  
:XPPOWER  
    SENSe:FUNCtion:<presentation\_layer> 18-47  
:XTIME  
    SENSe:FUNCtion:<presentation\_layer> 18-47  
:XVOLtage  
    SENSe:FUNCtion:<presentation\_layer> 18-47

## Y

:Y  
    DISPlay:WINDOW:TRACe 8-13  
    INPut:POSition 11-10  
    OUTPut:POSition 15-8

## Z

:Z  
    INPut:POSition 11-12  
    OUTPut:POSition 15-10

## **1999 SCPI Command Reference**

:ZERO  
CALibration 5-7 - 5-10

**Standard  
Commands for  
Programmable  
Instruments  
(SCPI)**

**Volume 3: Data Interchange Format**

---

VERSION 1999.0  
May, 1999  
Printed in U.S.A.

# Table of Contents

---

## Chapter 1 Introduction

1.1	Overview .....	1-1
1.2	Defined Blocks .....	1-2
1.3	Implicit and Explicit Dimensions .....	1-2

## Chapter 2 Style

## Chapter 3 Syntax

3.1	Character Set .....	3-1
3.2	White Space .....	3-1
3.3	Blocks, Block Modifiers, and Keywords .....	3-2
3.4	Value Types .....	3-2
3.4.1	<Numeric> .....	3-2
3.4.2	<+Numeric> .....	3-2
3.4.3	<0+Numeric> .....	3-3
3.4.4	<NR1> .....	3-3
3.4.5	<+NR1> .....	3-3
3.4.6	<0+NR1> .....	3-3
3.4.7	<Block> .....	3-3
3.4.8	<String> .....	3-3
3.4.9	<Label> .....	3-3
3.4.10	Enumerated Set .....	3-4

## Chapter 4 Grammar

4.1	Grammar Notation .....	4-1
4.1.1	Non-terminals .....	4-1
4.1.2	Pseudo-terminals .....	4-1
4.1.3	Meta-symbols .....	4-1
4.1.4	Terminal Symbols .....	4-1
4.2	Grammar Description .....	4-1
4.3	Required Blocks .....	4-6
4.4	Required Order .....	4-6
4.5	Unrecognized Keywords and Blocks .....	4-6

## Chapter 5 Data Format Extensions

5.1	Extensions .....	5-1
5.2	Enhancements .....	5-1
5.3	Extension Example .....	5-1

# 1999 SCPI Data Interchange Format

## Chapter 6 Block Descriptions

6.1	Top Level Block Organization .....	6-1
6.2	DATA .....	6-1
6.2.1	NOTE .....	6-2
6.2.2	DELTa .....	6-2
6.2.2.1	NOTE .....	6-2
6.2.2.2	DIMension .....	6-3
6.2.2.2.1	SCALe .....	6-3
6.2.2.2.2	OFFSet .....	6-3
6.2.2.2.3	SIZE .....	6-3
6.2.2.3	DATE .....	6-3
6.2.2.4	TIME .....	6-3
6.2.3	CURVe .....	6-4
6.2.3.1	NOTE .....	6-4
6.2.3.2	NAME .....	6-4
6.2.3.3	CTYPe .....	6-4
6.2.3.4	VALues .....	6-5
6.2.3.5	CSUM .....	6-5
6.2.4	WAveform .....	6-5
6.2.4.1	NOTE .....	6-8
6.2.4.2	NAME .....	6-8
6.2.4.3	TRACe .....	6-8
6.2.4.4	HLMETHOD .....	6-8
6.2.4.5	HIGH .....	6-8
6.2.4.6	LOW .....	6-9
6.2.4.7	REFerence .....	6-9
6.2.4.7.1	HIGH .....	6-9
6.2.4.7.2	LOW .....	6-9
6.2.4.7.3	MID .....	6-9
6.2.4.7.4	METHod .....	6-9
6.2.4.8	AMPLitude .....	6-10
6.2.4.9	PWIDth .....	6-10
6.2.4.10	NWIDth .....	6-10
6.2.4.11	PERiod .....	6-10
6.2.4.12	FREQuency .....	6-10
6.2.4.13	PDUTcycle   DCYCLE .....	6-11
6.2.4.14	NDUTcycle .....	6-11
6.2.4.15	RISE .....	6-11
6.2.4.15.1	TIME .....	6-11
6.2.4.15.2	OVERshoot .....	6-11
6.2.4.15.3	PRESHoot .....	6-11
6.2.4.16	FALL .....	6-11
6.2.4.16.1	TIME .....	6-12
6.2.4.16.2	OVERshoot .....	6-12

## 1999 SCPI Data Interchange Format

6.2.4.16.3	PREShoot .....	6-12
6.2.4.17	MAXimum .....	6-12
6.2.4.18	MINimum .....	6-12
6.2.4.19	TMAXimum .....	6-12
6.2.4.20	TMINimum .....	6-12
6.2.4.21	MEAN .....	6-13
6.2.4.22	RMS .....	6-13
6.2.4.23	SDEViation .....	6-13
6.2.4.24	PTPeak .....	6-13
6.2.4.25	CYCLE .....	6-13
6.2.4.25.1	COUNt .....	6-13
6.2.4.25.2	MEAN .....	6-14
6.2.4.25.3	RMS .....	6-14
6.2.4.25.4	SDEViation .....	6-14
6.2.5	MEASurement .....	6-14
6.2.5.1	NOTE .....	6-15
6.2.5.2	NAME .....	6-15
6.2.5.3	UNITs .....	6-15
6.2.5.4	TYPE .....	6-15
6.2.5.5	TRACe .....	6-16
6.2.5.6	LOCation .....	6-16
6.2.5.6.1	LABel .....	6-16
6.2.5.6.2	INdex .....	6-16
6.2.5.7	VALues .....	6-16
6.2.6	DATA Block Example .....	6-17
6.3	DIMension .....	6-17
6.3.1	NOTE .....	6-18
6.3.2	NAME .....	6-18
6.3.3	TYPE .....	6-18
6.3.4	SCALe .....	6-19
6.3.5	OFFSet .....	6-19
6.3.6	SIZE .....	6-19
6.3.7	UNITs .....	6-20
6.3.8	ENCode .....	6-20
6.3.9	DIMension Block Example .....	6-20
6.4	ENCode .....	6-21
6.4.1	NOTE .....	6-21
6.4.2	FORMAT .....	6-22
6.4.3	NVALue .....	6-23
6.4.4	ORANGE .....	6-23
6.4.5	URANGE .....	6-23
6.4.6	HRANGE .....	6-23
6.4.7	LRANGE .....	6-24
6.4.8	RESolution .....	6-24
6.4.9	ENCode Block Example .....	6-24

## 1999 SCPI Data Interchange Format

6.5	IDE <sup>N</sup> tify .....	6-24
6.5.1	NOTE .....	6-25
6.5.2	NAME .....	6-25
6.5.3	TECHnician .....	6-25
6.5.4	PROject .....	6-26
6.5.5	DATE .....	6-26
6.5.6	TIME .....	6-26
6.5.7	UUT .....	6-26
6.5.7.1	NAME .....	6-26
6.5.7.2	ID .....	6-27
6.5.7.3	DESign .....	6-27
6.5.8	TEST .....	6-27
6.5.8.1	NAME .....	6-27
6.5.8.2	SERies .....	6-27
6.5.8.3	NUMBer .....	6-27
6.5.9	HISTory .....	6-27
6.5.10	IDE <sup>N</sup> tify Block Example .....	6-28
6.6	ORDer .....	6-28
6.6.1	NOTE .....	6-28
6.6.2	BY .....	6-28
6.6.3	Examples .....	6-29
6.7	REMark .....	6-31
6.7.1	NOTE .....	6-32
6.7.2	REMark Block Example .....	6-32
6.8	DIF .....	6-32
6.8.1	NOTE .....	6-32
6.8.2	VERSion .....	6-32
6.8.3	SCOPe .....	6-33
6.8.4	DIF Block Example .....	6-33
6.9	TRACe .....	6-33
6.9.1	NOTE .....	6-34
6.9.2	NAME .....	6-34
6.9.3	SYMMetry .....	6-34
6.9.4	INDependent .....	6-34
6.9.4.1	LABel .....	6-35
6.9.4.2	STARt .....	6-35
6.9.4.3	STOP .....	6-35
6.9.5	DEPendent .....	6-35
6.9.5.1	LABel .....	6-35
6.9.6	TRACe Block Example .....	6-36
6.10	VIEW .....	6-36
6.10.1	NOTE .....	6-37
6.10.2	NAME .....	6-37
6.10.3	ENVelope .....	6-37
6.10.3.1	UPPer .....	6-38

## **1999 SCPI Data Interchange Format**

6.10.3.2	LOWER .....	6-38
6.10.3.3	FUNCTION .....	6-38
6.10.4	RCOMplex .....	6-38
6.10.4.1	REAL .....	6-38
6.10.4.2	IMAGinary .....	6-38
6.10.5	PCOMplex .....	6-38
6.10.5.1	MAGNitude .....	6-38
6.10.5.2	PHASe .....	6-39
6.10.6	VIEW BLOCK EXAMPLE .....	6-39

## **Chapter 7 Data Format Example**

**1999 SCPI Data Interchange Format**

## 1 Introduction

The data interchange format described in this section lets software packages and instruments share waveform and other data. It is designed to be flexible and extensible, and can accommodate a wide range of data structures and formats. In addition to describing the data itself, the data interchange format provides information regarding the data environment, structure, and other related specifications. Only a minimum of information about the data is actually required but a wide range of information can be associated with the data. Complex data dimensioned sets and simpler data are equally representable using the block structure of this data format.

Because this data format is hierarchically structured, it can be subordinate to other structures. It is suitable as a file format and its encoding formats allow it to be easily transmitted across *IEEE 488.1/488.2, RS-232, SCSI, IEEE 802*, and various other transmission media and protocols.

The syntax of the data interchange format is compatible with *IEEE 488.2* syntax. With minor modifications, an *IEEE 488.2* parser, or at least its lexical analyzer, can also analyze the syntax of this format. While the format imposes its own rules within *IEEE 488.2* expressions, these rules conform to the *IEEE 488.2* restrictions for expressions.

### 1.1 Overview

The data interchange format is block-structured. A data set generally consists of several description blocks and a data block. A block contains keywords and subordinate blocks. Within a block or sub-block are keywords with values that define characteristics of the data or the environment in which it was acquired. Many blocks accept a modifier label value and it is required for some blocks. A block modifier allows multiples of the same block type to be distinguished from one another.

Each block addresses a different aspect of data description. The DATA block contains the data. The IDENTify block describes the manner and environment in which the data was obtained. The DIMension and ENCode blocks describe how the data is physically represented and logically organized. The TRACe and VIEW blocks provide semantic information about the data. A REMark block contains textual comments regarding the data. Finally, the DIF block identifies the block as a <dif\_expression> and describes the version of SCPI used.

This data interchange format focuses on data as an abstraction of instrument- or algorithm-generated data. The method of data acquisition or creation is handled as descriptive information that is not required for proper interpretation of the data. Information about the encoding, structure, scaling, range, etc., of the data is independent of the source of the data.

For example, instrument-dependent settings and characteristics such as sample rate are translated into acquisition independent parameters such as RESolution.

# 1999 SCPI Data Interchange Format

## 1.2 Defined Blocks

The Data Interchange Format consists of the following major blocks:

- **DIF** identifies the expression as a <dif\_expression> and contains the version of the standard used to create the data set.
- **REMark** contains general comments in human-readable text regarding the data.
- **IDENtify** names the data set and describes the environment in which it was generated. Environment description includes project name, test number and series, date and time, and source.
- **ENCode** specifies the encoding format for data in the DATA and ENCode blocks. Also covered are signal values for underflow, overflow and no value, as well as the range and resolution of the data.
- **DIMension** specifies the structure and format of data in the DATA block. Provision is made for data scaling, offset, naming, and units specification.
- **ORDer** specifies ordering of the data elements in the DATA(CURVe) block for each dimension.
- **TRACe** logically groups dimensions as functions, surfaces, sets, etc. TRACe blocks provide semantic information about the data and are used to build VIEW blocks. They also provide a convenient and powerful mechanism for subsetting data.
- **VIEW** provides a second level of semantic information about the data. VIEW describes logical relationships among traces defined in the TRACe block such as envelope traces.
- **DATA** contains the actual data. Different subordinate blocks describe dimensioned data, waveform parameter measurements, and point values.

## 1.3 Dimensioned Data

There are many ways to describe the structure and order of dimensioned data, and not all terms are well defined or understood. The data interchange format treats all CURVe data as coordinates in n-dimensional space. If a coordinate is actually present in the data, its dimension is said to be explicit. If the coordinate value of a dimension is not present as a data value, but is instead derived from the location of other coordinate values, the dimension is said to be implicit.

Suppose you mark off an evenly spaced grid on the floor of a room, and then make measurements of temperature and humidity at some (varying) height above the floor. As you make these measurements, you write down the temperature and humidity measurements and the location (X, Y, and Z positions relative to one corner of the room). The result is 5-tuple. You must write each set of measurements in a consistent order (such as X, Y, Z, temperature, and humidity), but you may write the sets of measurements in any order.

## 1999 SCPI Data Interchange Format

A 5-tuple may be viewed, as this format does, as a point in a 5-dimensional space. All five dimensions are explicit, meaning that for each dimension, you have explicitly written down its coordinate.

Now suppose you make the measurements according to some consistent pattern. You start with the first grid line and make the measurements sequentially at each Y point along the X grid line. You then make the measurements along the second X grid line, and so on, until the last grid line is completed.

If you keep track of the order in which the measurements were made, you only need to write down two measurements: temperature and humidity. The other three measurements (X location, Y location, and height) are derivable from the order of the data. The data is still 5-dimensional, but three of the dimensions are implicit and two are explicit.

Simple waveforms acquired from instruments like oscilloscopes (when clocked internally) commonly consist of a time series of voltage (vertical axis) measurements. For the time-voltage space, only the voltage value or coordinate is provided by the oscilloscope; the time (horizontal axis) is derived from the order of the data, an initial starting time, and a scale factor. The voltage dimension is explicit and the time dimension is implicit.

This format classifies all dimensions as implicit or explicit. The order of the data, which is essential to the correct derivation of implicit dimension coordinates or values, is strictly defined by the ORDer block or its default values.

## **1999 SCPI Data Interchange Format**

## 2 Style

The data interchange format uses, for the most part, standard *IEEE 488.2* syntactic elements along with the nested parentheses which together comprise an Expression type parameter (“Volume 1: Syntax and Style,” Section 8).

The format also uses the concept of “friendly” listening and “precise” talking followed by *IEEE 488.2*. There are some differences, which are clarified in 3 of this section.

Keywords may take single or multiple values. If a keyword takes multiple values, all values shall be of the same type.

A few keywords have default values. If the keyword is omitted, the default value prevails. These values are specified in the block descriptions in 6 of this section.

An invalid set reports either a Command Error or an Execution Error as described in “Volume 2: Command Reference,” 21.8.9 and 21.8.10. While uppercase characters are specified, lowercase characters may be accepted without error if interpretation of the data set is unambiguous.

It is not required that a data format parser accept all values of an enumerated set (3.4.10 of this volume). An error may be generated on receipt of an unimplemented value.

## **1999 SCPI Data Interchange Format**

## 3 Syntax

The data format is block-structured. Each hierarchical level is introduced by a block name, with its subordinate elements enclosed in parentheses. A block may have a modifier and may contain subordinate blocks and keyword units, or both. Keyword units consist of a keyword followed by one or more values. If a keyword has more than one value, the values are separated by commas.

The following example shows a simple data set. It is formatted so that all blocks and keywords are indented to show their hierarchical relationship.

3

```
(DIF (VERSion 1993.0)
IDENtify (
    NAME "Data Format Example"
    TEST (
        NUMBER "7D4", "2.4"))
ENCode (
    HRANge 75
    LRANge 25)
DIMension=X (
    TYPE IMPLicit
    SCALe 0.01
    SIZE 7
    UNITS "S")
DIMension=Y (
    TYPE EXPLicit
    SCALe 0.02
    OFFSet 0.1
    UNITS "V")
DATA (
    CURVe (
        VALues 49.0, 48.0, 50.2, 61.3, 68.5, 38.6, 48.0)))
```

The data interchange format overall is formatted as an *IEEE 488.2 <EXPRESSION PROGRAM DATA>* element. Within this element, the various blocks, modifiers, keywords, and value types are composed of syntactic elements that, for the most part, are identical to the corresponding types specified in *IEEE 488.2* or a subset of them.

### 3.1 Character Set

All syntactic elements except <Block> contain only 7-bit ASCII-formatted (*ANSI X3.4-1977*) data with the high-order eighth bit always set to zero.

### 3.2 White Space

White space consists of the ASCII space character, 32 decimal. A space is used along with the parentheses, comma, and equal sign to separate syntactic elements.

## 1999 SCPI Data Interchange Format

When an instrument accepts data, redundant white space may appear without semantic effect anywhere except within: block names, block modifiers, keywords, and all value types.

Note that the characters constituting white space may be a part of the semantic content of <String> and <Block> value types.

When an instrument sends data, redundant white space is not allowed.

### 3.3

#### Blocks, Block Modifiers, and Keywords

When an instrument accepts data, block names, block modifiers, and keywords conform to the *IEEE 488.2 <CHARACTER PROGRAM DATA>* syntax. Abbreviated short forms and any specified aliases shall be accepted.

When an instrument sends data, the *IEEE 488.2 <CHARACTER RESPONSE DATA>* syntax is used. If a long and short form element is specified, the short form shall be sent.

As the data format is hierarchically organized, mnemonic generation rules for compound headers apply. See “Syntax and Style” 2.2.1. Specific syntactic restrictions on the use of blocks, block modifiers, and keywords are covered in 4 of this section.

### 3.4

#### Value Types

The grammar employs the following value types.

##### 3.4.1

#### <Numeric>

The <Numeric> value type is used to specify zero, positive and negative numeric values.

When an instrument accepts data, <Numeric> follows one of the following two *IEEE 488.2* program data formats:

<DECIMAL NUMERIC PROGRAM DATA>, or  
<NON-DECIMAL NUMERIC PROGRAM DATA>

Contrary to *IEEE 488.2*, <Numeric> values with embedded white spaces do not have to be accepted.

When an instrument sends data, <Numeric> follows one of the following six response data formats:

<NR1 NUMERIC RESPONSE DATA>,  
<NR2 NUMERIC RESPONSE DATA>,  
<NR3 NUMERIC RESPONSE DATA>,  
<OCTAL NUMERIC RESPONSE DATA>,  
<BINARY NUMERIC RESPONSE DATA>, or  
<HEXADECIMAL NUMERIC RESPONSE DATA>

##### 3.4.2

#### <+Numeric>

The <+Numeric> value type is used to specify positive numeric values greater than zero. Other than this restriction, the value type is identical to <Numeric>.

**3.4.3 <0+Numeric>**

The <0+Numeric> value type is used to specify positive numeric values equal to or greater than zero. Other than this restriction, the value type is identical to <Numeric>.

**3.4.4 <NR1>**

The <NR1> value type is used to specify zero, positive and negative integer numeric values.

An instrument shall accept any <Numeric> value, rounding as necessary to obtain an integer value. When an instrument sends data, <NR1> follows the IEEE 488.2 <NR1 DECIMAL NUMERIC RESPONSE DATA> format.

Note that rounding may result in an invalid integer value.

**3.4.5 <+NR1>**

The <+NR1> value type is used to specify positive integer numeric values greater than zero. Other than this restriction, the value type is identical to <NR1>.

**3.4.6 <0+NR1>**

The <0+NR1> value type is used to specify positive numeric values equal to or greater than zero. Other than this restriction, the value type is identical to <NR1>.

**3.4.7 <Block>**

The <Block> value type is used to specify 8-bit, 16-bit, 32-bit, or 64-bit significant data. <Block> is only allowed for use with the DATA(CURVe(VALues)) keyword.

The FORMat keyword from the applicable ENCode block determines the interpretation of the data values. Bytes are grouped according to the number of bits required. For example, bytes are grouped by four's and interpreted as a 32-bit integer for INT32.

When an instrument accepts data, <Block> values shall follow the format of *IEEE 488.2 <ARBITRARY BLOCK PROGRAM DATA>* format with the restriction that only definite length blocks are accepted.

When an instrument sends data, <Block> values shall follow the format of *IEEE 488.2 <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>* format.

**3.4.8 <String>**

The <String> is used to represent quoted strings.

When an instrument accepts data, <String> data conforms to *IEEE 488.2 <STRING PROGRAM DATA>* syntax.

When an instrument sends data, <String> data conforms to *IEEE 488.2 <STRING RESPONSE DATA>* syntax.

**3.4.9 <Label>**

The <Label> value type is used to represent short, unquoted string labels for block modifier and keyword values.

When an instrument accepts data, <Label> follows the *IEEE 488.2 <CHARACTER PROGRAM DATA>* syntax.

## 1999 SCPI Data Interchange Format

When an instrument sends data, <Label> follows the *IEEE 488.2 <CHARACTER RESPONSE DATA>* syntax. A <Label> has no short form.

### 3.4.10 **Enumerated Set**

Some keywords specify a value from an completely enumerated set of values.

When an instrument accepts data, an enumerated set member follows the *IEEE 488.2 <CHARACTER PROGRAM DATA>* syntax. If specified, acceptance of long and short form as well as aliases is required.

When an instrument sends data, an enumerated set member follows the *IEEE 488.2 <CHARACTER RESPONSE DATA>* syntax. If both a long and short form element are specified, only the short form shall be sent.

It is not required that a data format parser accept all values of an enumerated set. An error may be generated on receipt of an unimplemented value. See 2 of this section.

Mnemonic generation rules for character data apply. See “Syntax and Style,” 2.2.1.

## 4 Grammar

This chapter specifies the grammar of the data format. This grammar describes the contents of a <dif\_expression> that represents a Data Interchange Format.

### 4.1 Grammar Notation

The grammar notation consists of the non-terminal symbols, pseudo-terminal symbols, meta-symbols, and terminal symbols defined below.

#### 4.1.1 Non-terminals

Non-terminals are represented as all lowercase words.

#### 4.1.2 Pseudo-terminals

Pseudo-terminals stand for any from a set of terminal symbols (value types) as described in 3.4 of this section and are represented as words bracketed between < and >.

#### 4.1.3 Meta-symbols

In the following table, “.” represents an arbitrary string of terminals and non-terminals. The meta-symbols for the grammar are ->, {, }, \*, +, [ , ], and | .

They have the following meanings:

->	Separates the left and right hand sides of a production
{ . }*	The enclosed item occurs zero or more times.
{ . }+	The enclosed item occurs one or more times.
[ . ]	The enclosed item occurs zero or one times.
{ .   .   . }	One and only one of the two or more enclosed items separated by   occurs

#### 4.1.4 Terminal Symbols

Terminal symbols are represented in mixed uppercase and lowercase to indicate long and short form (“Syntax and Style,” 1).

### 4.2 Grammar Description

```

dif ->      difid
              [remark]
              [identify]
              [encode]
              {dimension}+
              [order]
              {trace}* 
              {view}*
              {data}+
  
```

## 1999 SCPI Data Interchange Format

```
difid -> DIF(
    [NOTE] <String>
    [VERSION] <+Numeric>
    [SCOPe] PREamble | FULL
)

remark -> REMark(
    [NOTE] <String> { ,<String>}*
)

4 identify -> IDENTify (
    [NOTE] <String>
    [NAME] <String>
    [TECHnician] <String> { ,<String>}*
    [PROject] <String>
    [DATE] <NR1>, <+NR1>, <+NR1>
           [ ,<NR1>, <+NR1>, <+NR1> ]
    [TIME] <0+NR1>, <0+NR1>, <0+Numeric>
           [ ,<0+NR1>, <0+NR1>, <0+Numeric> ]
    [UUT(
        [NAME] <String>
        [ID] <String> { ,<String>}*
        [DESign] <String> { ,<String>}*
    )]
    [TEST(
        [NAME] <String>
        [SERies] <String>
        [NUMBER] <String> { ,<String>}*
    )]
    [HISTORY] <String> { ,<String>}*
)
)

encode -> ENCode (
    [NOTE] <String>
    [FORMAT] { IFP32 | IFP64
               | INT8 | INT16 | INT32 | INT64
               | SFP32 | SFP64
               | SINT16 | SINT32 | SINT64
               | SUINT16 | SUINT32 | SUINT64
               | UINT8 | UINT16 | UINT32 | UINT64 }
    [NVALue] <Numeric>
    [ORANGE] <Numeric>
    [URANGE] <Numeric>
    [HRANGE] <Numeric>
    [LRANGE] <Numeric>
)
```

## 1999 SCPI Data Interchange Format

```
[RESolution    <+Numeric>]
)

dimension -> DIMension = <Label>(
    [NOTE          <String>]
    [NAME          <String>]
    TYPE          {IMPLicit | EXPLicit}
    [SCALe         <Numeric>]
    [OFFSet        <Numeric>]
    [SIZE          <+NR1>]
    UNITS         <String>
    [encode]
)
order -> ORDer (
    [NOTE          <String>]
    [BY            {TUPLe | DIMension}]
)
trace -> TRACe = <Label>(
    [NOTE          <String>]
    [NAME          <String>]
    [SYMMetry      <String>]
    {INDependent(
        LABel        <Label>
        [START        <+NR1>]
        [STOP         <+NR1>]
    )}**
    {DEPendent(
        LABel        <Label>
    )}**
)
view -> VIEW = <Label> (
    [NOTE          <String>]
    [NAME          <String>]
    {ENVelope(
        UPPer        <Label>
        LOWER        <Label>
        [FUNCTION    <Label>]
    )}**
)
```

## 1999 SCPI Data Interchange Format

```
4  
data ->      DATA [= <Label>] (  
           [NOTE          <String>]  
           [delta]  
           [curve]  
           {waveform}*  
           {measurement}*  
           )  
  
delta ->     DELTa (   
           [NOTE          <String>]  
           {DIMension =  <Label> (  
               [SCALE        <Numeric>]  
               [OFFSet       <Numeric>]  
               [SIZE         <+NR1>]  
           ) } +  
           [DATE          <NR1> , <+NR1> , <+NR1>  
            [ , <NR1> , <+NR1> , <+NR1> ] ]  
           [TIME          <0+NR1> , <0+NR1> , <0+Numeric>  
            [ , <0+NR1> , <0+NR1> , <0+Numeric> ] ]  
           )  
  
curve ->    CURVe (   
           [NOTE          <String>]  
           [NAME          <String>]  
           [CTYPe         {SUM8| CRC16| CCITT | SUM16}]  
           [VALues        {<Block> | <Numeric>}  
            [ , {<Block> | <Numeric>} ] * ]  
           [CSUM          <+NR1>]  
           )
```

NOTE: The presence of VALues is conditional on the DIF(SCOPe) keyword; refer to the Required Blocks, section 4.3.

## 1999 SCPI Data Interchange Format

```
waveform -> WAVEform[ = <Label>] (
    [NOTE <String>]
    [NAME <String>]
    [TRACe <Label>]
    [HLMETHOD <String>]
    [HIGH <Numeric>]
    [LOW <Numeric>]
    [REFERENCE(
        [HIGH <Numeric>]
        [LOW <Numeric>]
        [MID <Numeric>]
        [METHOD <String>]
    )]
    [AMPLitude <Numeric>]
    [PWIDth <Numeric>]
    [NWIDth <Numeric>]
    [PERiod <Numeric>]
    [FREQuency <Numeric>]
    [PDUTcycle
    | DCYCLE <Numeric>]
    [NDUTcycle <Numeric>]
    [RISE(
        [TIME <Numeric>]
        [OVERshoot <Numeric>]
        [PRESHoot <Numeric>]
    )]
    [FALL(
        [TIME <Numeric>]
        [OVERshoot <Numeric>]
        [PRESHoot <Numeric>]
    )]
    [MAXimum <Numeric>]
    [MINimum <Numeric>]
    [TMAXimum <Numeric>]
    [TMINimum <Numeric>]
    [MEAN <Numeric>]
    [RMS <Numeric>]
    [SDEViation <0+Numeric>]
    [PTPeak <0+Numeric>]
    [CYCLE(
        [COUNT <0+Numeric>]
        [MEAN <Numeric>]
        [RMS <Numeric>]
        [SDEViation <0+Numeric>]
    )]
)
```

## 1999 SCPI Data Interchange Format

```
measurement -> MEASurement [= <Label>] (
    [NOTE      <String>]
    [NAME      <String>]
    [UNITS     <String>]
    [TYPE      <String>]
    [TRACe     <Label>]
    {LOCATION (
        LABEL   <Label>
        INDEX   <+NR1>
    )}**
    [VALues    <Numeric> { , <Numeric>}* ]
)
```

4

### 4.3 Required Blocks

Note that a data set shall contain a DIF block and at least one DIMension and one DATA block.

All TRACe blocks referenced by keywords in the VIEW and DATA blocks shall be present.  
All DIMension blocks referenced by LABel keywords in the TRACe block shall be present.

If the DIF(SCOPe) keyword is not present or is set to FULL, any CURVe block shall contain a VALues keyword and parameters. If DIF(SCOPe) is present and set to PREamble, VALues keywords and parameters shall be omitted from all CURVe blocks.

### 4.4 Required Order

When an instrument accepts data, certain blocks shall be sent in the following order:

```
DIF
ENCode
DIMension
TRACe
VIEW
DATA
```

In addition, the ORDer must appear before the DATA block. The REMark and IDENTify blocks may appear anywhere after the DIF block.

Within the DATA block, the DELTa sub-block shall precede all other sub-blocks.

Within a block, individual keywords may also be accepted in any order with the exception that the CTYPe keyword must come before the VALues keyword.

When an instrument sends data, the block and keyword order specified in the grammar is required.

### 4.5 Unrecognized Keywords and Blocks

A listener is not required to interpret all keywords and blocks present in a data set, but only those required for the type of data of interest. For example, all listeners must interpret most keywords and sub-blocks in the DIF, DIMension, ENCode, and ORDer blocks since

## 1999 SCPI Data Interchange Format

information contained in these blocks is required for the correct interpretation of data in the DATA block. Exceptions are keywords such as NOTE, RESolution, HRANge, etc. Other keywords and blocks may or may not be of interest to a listener. For example, a listener that seeks only WAVEform data will ignore CURVe data, and vice versa.

This is consistent with the philosophy that different listeners may accept the same data set but use it for different purposes. In addition, the data format provides an extension mechanism that is designed to allow older listeners to accept newer data sets that may contain new keywords and sub-blocks. See 5 of this section for further details on extension mechanisms.

Therefore, listeners must be able to ignore unrecognized (but syntactically correct) keywords and blocks. The extent, if any, to which a listener notifies a user that an unrecognized keyword or block was encountered; and, the extent, if any, of required user action is implementation dependent.

Note that a block name is always followed by an open parenthesis; or, an equal sign, modifier, and open parenthesis sequence. This following structure may be ignored by searching forward to the matching closing parenthesis. A keyword is always followed by one or more comma separated values.

## **1999 SCPI Data Interchange Format**

## 5 Data Format Extensions

This format may be extended and enhanced with one of the following two methods.

### 5.1 Extensions

Keywords, blocks (sub-blocks) may be created to accomodate information that is not otherwise provided for by existing keywords or blocks. This method of extension can be done without conflict because parsers ignore unknown keywords and blocks. Name aliases may be specified for the new keywords or blocks. However, the use of aliases is strongly discouraged and should only be provided if there is widespread industry usage of more than one name.

### 5.2 Enhancements

A second method of extension involves replacing an existing keyword with a sub-block of the same name. This method is used when the semantic information represented by an existing keyword is insufficient. The replacement sub-block shall include a keyword that represents the exact same semantic information (including the same value type and range) of the original keyword. This keyword is distinguished from all other keywords in the new sub-block by appending an underscore to it. No alias names may be added for the new sub-block.

A parser which was implemented before the extension was made recognizes this special keyword by the presence of the underscore and then takes the value associated with this flagged keyword and assigns it to the old keyword. The remainder of the sub-block is discarded.

### 5.3 Extension Example

Suppose sometime in the future a need for a sub-block under RANGe is recognized. In this version the syntax for RANGe is:

RANGe      <+Numeric>

The extended syntax might appear as:

```
RANGe (
    HIGH_      <+Numeric>
    [LOW       <+Numeric>]
    [OVER      <+Numeric>]
)
```

A newer implementation would understand the meaning for all three values under RANGe. An older parser takes the values associated with the flagged keyword in the sub-block, in this case, HIGH\_, and uses its value as the value for RANGe. Since the older parser has no need for the LOW and OVER values, they are discarded.

## 1999 SCPI Data Interchange Format

This extension can be continued. At some even later date, more sub-blocks are needed. The syntax might now become:

```
RANGe (
    HIGH_(
        EXPected_ <+Numeric>
        [ALLOWed <+Numeric>]
    )
    LOW (
        EXPected_ <+Numeric>
        [ALLOWed <+Numeric>]
    )
    OVER (
        EXPected_ <+Numeric>
        [ALLOWed <+Numeric>]
    )
)
```

The original parser could still successfully determine which value to associate with the old keyword RANGe and safely discard the rest. Both the intermediate and newest parsers can handle this syntax without conflict.

## 6 Block Descriptions

This chapter details all of the block descriptions for the data interface format.

### 6.1 Top Level Block Organization

The entire data set is composed of the following sequence of mandatory and optional blocks.

```
dif ->      difid
              [remark]
              [identify]
              [encode]
              {dimension}+
              [order]
              {trace}*
              {view}*
              {data}+
```

6

A data set is built from the blocks described in 6.2 through 6.10 of this section.

### 6.2 DATA

A DATA block contains the data of primary interest. It may contain dimensioned data such as DATA(CURVe), or specific sets of data (WAVEform, etc.). At least one DATA block is required, and multiple DATA blocks are allowed. A DATA block is defined as:

```
data ->      DATA [= <Label>] (
              [NOTE <String>]
              [delta]
              [curve]
              {waveform}*
              {measurement}*
            )
```

<Label> is optional and allows for the unique identification of the DATA block; no two DATA blocks may have the same <Label> value within a data set.

The subordinate blocks within a particular DATA block are assumed to be related. That is, if CURVe and WAVEform measurements are present in the same DATA block, the WAVEform measurements are assumed to have been derived from the CURVe data. If that is not the case, different DATA blocks should be used for the CURVe and WAVEform measurements.

## 1999 SCPI Data Interchange Format

### 6.2.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “Humidity was 45%”

### 6.2.2 DELTa

A DELta sub-block provides for sequences of closely related data. When a data set contains multiple occurrences of DATA blocks, the DELta block differentiates the characteristics of each of the DATA blocks. Note that the DELta block is limited to redefining descriptive information, encoding information, and dimension size, such as SCALe, DATE, and OFFSet. A DELta block cannot change the organization of the data, such as the number of dimensions or the data ordering.

The values specified in a DELta block override those specified at a higher hierarchical level. For example, the value for SCALe in the DATA(DELta(DIMension)) block overrides the value for SCALe in the higher level DIMension block, but only for data within the DATA block that contains the DELta block. If there are multiple DATA blocks with subordinate DELta blocks, the DELta blocks have no effect on each other; that is, they each individually specify a modification of the same (higher level) IDENTify, DIMension, or ENCode block.

The DELta sub-block may occur only once and, if present, shall be the first block in the DATA block. The DELta sub-block is defined as:

```
delta ->      DELTa(
                [ NOTE          <String>]
                { DIMension =   <Label> (
                    [ SCALe        <Numeric>]
                    [ OFFSet       <Numeric>]
                    [ SIZE         <+NR1>]
                ) } +
                [ DATE          <NR1>,<+NR1>,<+NR1>
                  [ ,<NR1>,<+NR1>,<+NR1> ] ]
                [ TIME          <0+NR1>,<0+NR1>,<0+Numeric>
                  [ ,<0+NR1>,<0+NR1>,<0+Numeric> ] ]
            )
)
```

### 6.2.2.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “Sequence 43”

### 6.2.2.2 DIMension

The DIMension sub-block is a limited subset of the higher level DIMension block (see 6.3 in this section). The <Label> modifier is required and is restricted to a value defined by the <Label> of a DIMension block at the same hierarchical level as the DATA block.

#### 6.2.2.2.1 SCALe

This is the scale value for the dimension specified by the DIMension sub-block <Label>.

Value type is <Numeric>. SCALe may occur only once. See SCALe in 6.3 in this section for additional definition and restrictions.

Example: SCALe 1E-2

#### 6.2.2.2.2 OFFSet

This is the offset value for the dimension specified by the DIMension sub-block <Label>.

Value type is <Numeric>. OFFSet may occur only once. See OFFSet in 6.3 in this section for additional definition and restrictions.

Example: OFFSet 6.0E-1

#### 6.2.2.2.3 SIZE

The size value for the dimension specified by the DIMension sub-block <Label>.

Value type is <+NR1>. SIZE may occur only once. See SIZE in 6.3 in this section for additional definition and restrictions.

Example: SIZE 2048

#### 6.2.2.3 DATE

This is the date of data creation or acquisition.

This keyword takes a set of three comma separated value types to specify the date corresponding to year, month, and day.

The years parameter is type <NR1>. It always includes century and millennium information.

The months parameter is type <+NR1>. Its range is 0 to 12.

The days parameter is type <+NR1>. Its range is 0 to 31 (depending on the month).

If two sets of values are specified, they define an inclusive range of dates.

Example: DATE 1990,01,01,1992,10,11 indicates a range of dates from January 1, 1990 through October 11, 1992.

#### 6.2.2.4 TIME

This is the time of data creation or acquisition.

This keyword takes a set of three comma separated value types to specify the time corresponding to hour, minute, and second.

The hours parameter is type <0+NR1>. Its range is 0 to 23. This parameter is always expressed in 24 hour format.

## 1999 SCPI Data Interchange Format

The minutes parameter is type <0+NR1>. Its range is 0 to 59.

The seconds parameter is type <0+Numeric>. Its range is 0 to 60 for instruments receiving data and 0 to 59.999... for instruments sending data.

If two sets of values are specified, they define an inclusive range of time.

Example: TIME 9,0,59.095,13,1,0 indicates a range of time from 59.095 seconds after 9 AM to one minute after 1 PM.

### 6.2.3 CURVe

Curve data is dimensioned. Values may be organized as n-tuples or by dimension. This is determined by the ORDER block or, in its absence, the defaults for the BY keyword in the ORDER block.

Only one occurrence of CURVe is allowed in a DATA block. The CURVe block is defined as:

```
curve ->      CURVe(  
    [ NOTE    <String> ]  
    [ NAME    <String> ]  
    [ CTYPe   {SUM8| CRC16| CCITT | SUM16} ]  
    [ VALues  {<Block> | <Numeric>}  
        [, {<Block> | <Numeric>}]* ]  
    [ CSUM    <+NR1> ]  
)
```

The presence of VALUES is conditional on the DIF(SCOPe) keyword; refer to the Required Blocks, section 4.3.

#### 6.2.3.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “Data taken with low line voltage”

#### 6.2.3.2 NAME

This is an arbitrary name or description of the curve.

Value type is <String>. It is recommended that the string length be 32 characters or less. NAME may occur only once and only one value is allowed.

Example: NAME “DS1C”

#### 6.2.3.3 CTYPe

This indicates the checksum type or the absence of a checksum.

CTYPe takes a specified enumerated set of values. CTYPe may occur only once and requires one and only one of the following enumerated set values:

## 1999 SCPI Data Interchange Format

CRC16	16-bit cyclic redundancy check code. $x^{16} + x^{15} + x^2 + 1$ . (default)
CCITT	16-bit cyclic redundancy check code. $x^{16} + x^{12} + x^5 + 1$ .
SUM8	8-bit longitudinal redundancy check code (8 bit sum). $x^8 + 1$ .
SUM16	(SUM8 is provided for compatibility and its use is discouraged).
NONE	16-bit longitudinal redundancy check code (16 bit sum). $x^{16} + 1$ . indicates there is no checksum.

See “IEEE Micro”, Vol 8, No 4, August 1988, pp 62-76.

Note that CTYPe must precede the VALUes keyword.

Example: CTYPe SUM8

### 6.2.3.4 **VALUes**

This specifies the data values.

Value type is <Block> or <Numeric>. Multiple values are allowed.

6

VALUes shall occur exactly once if the DIF(SCOPe) keyword is not present or is set to FULL. If DIF(SCOPe) is present and set to PREamble, VALUes shall be omitted.

The physical format of each value is determined by the FORMat keyword of the ENCode block that applies. If the dimension has a subordinate ENCode block, the subordinate ENCode block prevails; otherwise the ENCode block at the same hierarchical level as the DIMension blocks prevails.

Example: VALUes 1.2, 3.3, 4.5, 2.7, 0.4

### 6.2.3.5 **CSUM**

This indicates the checksum for DATA(CURVe(VALUes)) data

Value type is <+NR1>.

The checksum is computed according to the rules specified by CTYPe. If data is formatted in <Numeric> value type, the checksum is computed using the 8-bit ASCII decimal equivalent of each character of the number. This includes all signs, decimal points, and exponent characters but not white space.

CSUM may occur only once and takes only one value.

Example: CSUM 27

### 6.2.4 **WAveform**

This block describes waveform parameter measurements of a two dimensional waveform. The descriptions assume a single-valued function of time in which the independent dimension is the time dimension. However, the actual units are specified in the DIMension block and it is possible to represent other quantities which are not functions of time.

Multiple occurrences of WAveform are allowed. The WAveform block is defined as:

## 1999 SCPI Data Interchange Format

```
waveform -> WAVEform[= <Label>] (
    [NOTE] <String>
    [NAME] <String>
    [TRACe] <Label>
    [HLMETHOD] <String>
    [HIGH] <Numeric>
    [LOW] <Numeric>
    [REFERENCE(
        [HIGH] <Numeric>
        [LOW] <Numeric>
        [MID] <Numeric>
        [METHOD] <String>
    )]
    [AMPLITUDE] <Numeric>
    [PWIDTh] <Numeric>
    [NWIDTh] <Numeric>
    [PERIOD] <Numeric>
    [FREQuency] <Numeric>
    [PDUTYcycle]
    | DCYCLE <Numeric>
    [NDUTYcycle] <Numeric>
    [RISE(
        [TIME] <Numeric>
        [OVERshoot] <Numeric>
        [PRESHoot] <Numeric>
    )]
    [FALL(
        [TIME] <Numeric>
        [OVERshoot] <Numeric>
        [PRESHoot] <Numeric>
    )]
    [MAXimum] <Numeric>
    [MINimum] <Numeric>
    [TMAXimum] <Numeric>
    [TMINimum] <Numeric>
    [MEAN] <Numeric>
    [RMS] <Numeric>
    [SDEViation] <0+Numeric>
    [PTPeak] <0+Numeric>
    [CYCLE(
        [COUNT] <0+Numeric>
        [MEAN] <Numeric>
        [RMS] <Numeric>
        [SDEViation] <0+Numeric>
    )]
)
```

## 1999 SCPI Data Interchange Format

In the absence of a TRACe block, the following rules are used to determine the independent and dependent dimensions:

1. The independent dimension is the first implicit dimension or, if no implicit dimensions are present, then the first explicit dimension
2. The dependent dimension is the first explicit dimension if there is at least one implicit dimension. Otherwise it is the second explicit dimension.

The block modifier <Label> value is optional and allows for the unique identification of the WAVEform block. No two WAVEform blocks may have the same <Label> within the same DATA block.

If WAVEform measurements are found together with a CURVe block in a DATA block, they are considered to be derived from or related to the CURVe data (or a subset of it). However, the exact relationship of waveform measurements to the data can be determined only if the TRACe keyword is present.

It is permissible for a data set to contain waveform measurements without a DATA(CURVe) block, but DIMension blocks for the independent and dependent dimensions shall still be specified.

Waveform measurements derived from DATA block data are considered to have been derived from the data after SCALe and OFFSet (specified in the DIMension block) have been applied. WAVEform measurements are derived from DIMension block units.

Note that this standard requires the syntax for WAVEform data to be correct, but makes no claim for the correctness or consistency of the data values. Keywords in the WAVEform block are representative of terms generally used in the industry. Keyword descriptions are general and, except where specifically provided (such as for HLMETHOD), no requirement for specific calculation methods are claimed.

The definitions for functions and settings within the waveform block are illustrated with the following figure:

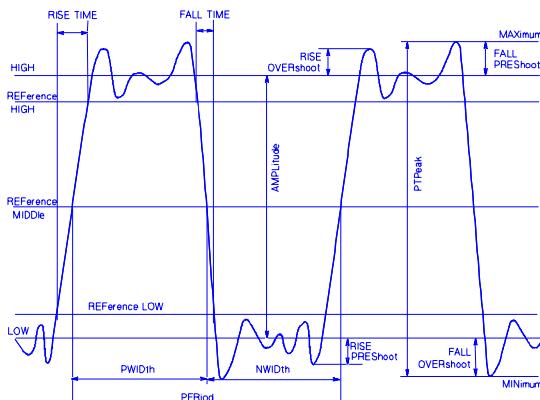


Figure 6-1 Waveform Terminology

## 1999 SCPI Data Interchange Format

### 6.2.4.1 NOTE

This is arbitrary text. Text should be human-readable.

Value type is <String>. Only one occurrence of the NOTE keyword is allowed and one value is allowed.

Example: NOTE “500 ps Reference Waveform”

### 6.2.4.2 NAME

This is an arbitrary name of the waveform measurement. Typically, NAME is more descriptive than the WAVEform block modifier <Label> value and it does not have to be unique.

Value type is <String>. It is recommended that the string length be 32 characters or less. NAME may occur only once and only one value is allowed.

Example: NAME “Frame bit”

### 6.2.4.3 TRACe

This is the TRACe from which the waveform parameters were extracted. TRACe allows a sub-set of the data to be referenced.

Value type is <Label>. Only values defined by a TRACe block modifier <Label> value are allowed.

Only one occurrence of TRACe is allowed and only one value is allowed. The TRACe referenced shall be two-dimensional.

Example: TRACe Y2

### 6.2.4.4 HLMethod

This notes the method used to determine the HIGH and LOW magnitudes.

For *IEEE 181* methods, HIGH becomes the more positive (less negative) and LOW becomes the more negative (less positive) of the Top Magnitude and Base Magnitude values as defined by *IEEE 194*, sections 3.2.1 and 3.2.2.

Value type is <String>. It is recommended that the string length be 32 characters or less. The following values are predefined:

UNKNOWN	Unknown or Unspecified. Default.
MEAN	Mean of Density Algorithm ( <i>IEEE 181</i> , section 4.3.1.1)
MODE	Mode of Density Algorithm ( <i>IEEE 181</i> , section 4.3.1.2)
PEAK	Peak Magnitude ( <i>IEEE 181</i> , section 4.3.1.4)
ABSOLUTE	Absolute, user-specified levels

Example: HLMethod “MEAN”

### 6.2.4.5 HIGH

Indicates the high magnitude.

Value type is <Numeric>. Value is required if keyword is present. HIGH may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: HIGH 4.2

#### **6.2.4.6 LOW**

This is the low magnitude.

Value type is <Numeric>. Value is required if keyword is present. LOW may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: LOW -0.21

#### **6.2.4.7 REFERENCE**

The REReference sub-block contains reference level information used to calculate other parameters in the WAveform block.

##### **6.2.4.7.1 HIGH**

Indicates the high or upper reference level for measurement of rise and fall time.

Value type is <Numeric>. Value is required if keyword is present. HIGH may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: HIGH 4.8

##### **6.2.4.7.2 LOW**

This is the low or lower reference level for measurement of rise and fall time.

Value type is <Numeric>. Value is required if keyword is present. LOW may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: LOW 0.3

##### **6.2.4.7.3 MID**

This is the mid reference level for measurement of width, duty cycle, and period.

Value type is <Numeric>. Value is required if keyword is present. MID may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: MID 3.3

##### **6.2.4.7.4 METHod**

This is the reference method. Notes the method used to determine HIGH, LOW, and MID REReference level values.

Value type is <String>. It is recommended that the string length be 32 characters or less. The following values are predefined:

UNKNOWN

Unknown or unspecified (default).

RELATIVE

Reference levels are set according to percentages of amplitude given by:

$$\text{high\_percentage} = \frac{(\text{WAveform}(REFerence(HIGH)) - \text{WAveform}(LOW)) * 100}{\text{WAveform}(AMPLitude)}$$

## 1999 SCPI Data Interchange Format

$$low\_percentage = \frac{(WAVEform(REFerence(LOW))-WAVEform(LOW))*100}{WAVEform(AMPLitude)}$$

$$mid\_percentage = \frac{(WAVEform(REFerence(MID))-WAVEform(LOW))*100}{WAVEform(AMPLitude)}$$

ABSOLUTE

Reference levels set in absolute signal value units.

Example: METHod “RELATIVE”

### 6.2.4.8 AMPLitude

Waveform amplitude. HIGH minus LOW.

Value type is <Numeric>. Value is required if keyword is present. AMPLitude may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: AMPLitude 4.03

### 6.2.4.9 PWIDth

Indicates positive width. Duration from a positive mid-reference crossing followed by at least one high-reference crossing to the following negative mid-reference crossing.

Value type is <Numeric>. Value is required if keyword is present. PWIDth may occur only once and only one value is allowed. Units are of the independent dimension (time).

Example: PWIDth 2.6E-7

### 6.2.4.10 NWIDth

Indicates negative width. Duration from a negative mid-reference crossing followed by at least one low-reference crossing to the following positive mid-reference crossing.

Value type is <Numeric>. Value is required if keyword is present. NWIDth may occur only once and only one value is allowed. Units are of the independent dimension (time).

Example: NWIDth 2.5E-7

### 6.2.4.11 PERiod

This is the repetition period. Duration from a mid-reference crossing (whether positive or negative) followed by at least one high-reference crossing and at least one low-reference crossing to the following mid-reference crossing in the same direction as the first mid-reference crossing.

Value type is <Numeric>. Value is required if keyword is present. PERiod may occur only once and only one value is allowed. Units are of the independent dimension (time).

Example: PERiod 1.602E-6

### 6.2.4.12 FREQuency

This is the frequency. FREQuency is the reciprocal of PERiod.

Value type is <Numeric>. Value is required if keyword is present. FREQuency may occur only once and only one value is allowed. Units are the reciprocal of the PERiod units.

Example: FREQuency 1.87222E+9

#### **6.2.4.13 PDUTycycle | DCYCle**

Indicates the positive dutycycle. Ratio of PWIDth to PERiod.

Value type is <Numeric>. Value is required if keyword is present. PDUTycycle may occur only once and only one value is allowed. Units are dimensionless.

Example: PDUTycycle .48

#### **6.2.4.14 NDUTycycle**

Indicates the negative dutycycle. Ratio of NWIDth to PERiod.

Value type is <Numeric>. Value is required if keyword is present. NDUTycycle may occur only once and only one value is allowed. Units are dimensionless.

Example: NDUTycycle .52

#### **6.2.4.15 RISE**

This sub-block deals with the first rising transition of the waveform where rising is defined as increasing data values.

##### **6.2.4.15.1 TIME**

This is the rise time (positive going transition). The first time interval during which the instantaneous amplitude of a waveform increases from the low reference point, REF (LOW), to the high reference point, REF (HIGH).

Value type is <Numeric>. Value is required if keyword is present. TIME may occur only once and only one value is allowed. Units are of the independent dimension (time).

##### **6.2.4.15.2 OVERshoot**

This is the rising edge overshoot. The difference between the HIGH level and the positive peak signal value to which the waveform initially rises expressed as a percentage of the waveform amplitude.

Value type is <Numeric>. OVERshoot may occur only once and only one value is allowed.

Example: OVERshoot 20.5

##### **6.2.4.15.3 PREshoot**

This is the rising edge preshoot. The difference between the LOW level and the negative peak signal value to which the waveform initially falls expressed as a percentage of the waveform amplitude.

Value type is <Numeric>. PREShoot may occur only once and only one value is allowed.

Example: PREShoot 1.2

#### **6.2.4.16 FALL**

This sub-block deals with the first falling transition of the waveform where falling is defined as decreasing data values.

## 1999 SCPI Data Interchange Format

### 6.2.4.16.1 TIME

Indicates the fall time (negative going transition). The first time interval during which the instantaneous amplitude of a waveform decreases from the high reference point, REF (HIGH), to the low reference point, REF (LOW).

Value type is <Numeric>. Value is required if keyword is present. TIME may occur only once and only one value is allowed. Units are of the independent dimension (time).

### 6.2.4.16.2 OVERshoot

This is the falling edge overshoot. The difference between the LOW level and the negative peak signal value to which the waveform initially falls expressed as a percentage of the waveform amplitude.

Value type is <Numeric>. OVERshoot may occur only once and only one value is allowed.

Example: OVERshoot 12.62

### 6.2.4.16.3 PREShoot

This is the falling edge preshoot. The difference between the HIGH level and the positive peak signal value to which the waveform initially rises expressed as a percentage of the waveform amplitude.

Value type is <Numeric>. PREShoot may occur only once and only one value is allowed.

Example: PREShoot 4.3

### 6.2.4.17 MAXimum

This is the maximum value of the entire waveform as specified by TRACe.

Value type is <Numeric>. MAXimum may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: MAXimum 4.5903

### 6.2.4.18 MINimum

This is the minimum value of the entire waveform as specified by TRACe.

Value type is <Numeric>. MINimum may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: MINimum -0.12

### 6.2.4.19 TMAXimum

This indicates the time of the first MAXimum value in the waveform as specified by TRACe.

Value type is <Numeric>. TMAXimum may occur only once and only one value is allowed. Units are of the independent dimension (time).

Example: TMAXimum 9.51

### 6.2.4.20 TMINimum

This indicates the time of the first MINimum value in the waveform as specified by TRACe.

Value type is <Numeric>. TMINimum may occur only once and only one value is allowed. Units are of the independent dimension (time).

Example: TMINimum 3.876

#### 6.2.4.21 MEAN

This is the arithmetic mean. Mean value of the entire waveform as specified by TRACe.

Value type is <Numeric>. MEAN may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: MEAN 2.025

#### 6.2.4.22 RMS

This is the root mean square. Root mean square value of the entire waveform as specified by TRACe.

Value type is <Numeric>. RMS may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: RMS 12.3

#### 6.2.4.23 SDEViation

This is the standard deviation. The standard deviation of the entire waveform as specified by TRACe.

Value type is <0+Numeric>. SDEV may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: SDEViation 45.68

#### 6.2.4.24 PTPeak

This is the waveform Peak-to-peak value. Peak-to-peak value of the entire waveform as specified by TRACe. PTPeak is equivalent to the MAXimum minus the MINimum values.

Value type is <0+Numeric>. PTPeak may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: PTPeak 4.5

#### 6.2.4.25 CYCLe

The CYCLe sub-block specifies parameters dealing with a specified number of complete cycles in the waveform as specified by the COUNT field.

##### 6.2.4.25.1 COUNT

This is the cycle count representing the number of cycles used for MEAN, RMS, and SDEViation in the CYCLe sub-block. Cycle count evaluates to an integer representing the number of cycles over which the CYCLe parameters apply. Default value for COUNT is 1.

If one complete cycle is not present, and COUNT is specified then the value should be zero. In this case, any related CYCLe parameters are meaningless.

Value type is <0+Numeric>. COUNT may occur only once and only one value is allowed.

## 1999 SCPI Data Interchange Format

Example: COUNT 10

### 6.2.4.25.2 MEAN

This is the cycle arithmetic mean. This indicates the mean value of the waveform as specified by COUNT.

Value type is <Numeric>. MEAN may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: MEAN 2.025

### 6.2.4.25.3 RMS

This is the cycle root mean square. It indicates the root mean square value of the waveform as specified by COUNT.

Value type is <Numeric>. RMS may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: RMS 12.3

### 6.2.4.25.4 SDEViation

This is the cycle standard deviation. This indicates the standard deviation of the entire waveform as specified by COUNT.

Value type is <0+Numeric>. SDEV may occur only once and only one value is allowed. Units are of the dependent dimension (signal value).

Example: SDEViation 45.68

### 6.2.5 MEASurement

This is the measurement of one or more specified data points. This block contains data that does not fit in other sub-blocks of the DATA block. A MEASurement block may occur multiple times, but is not required.

The MEASurement sub-block is defined as:

```
measurement -> MEASurement [= <Label>] (
    [ NOTE      <String> ]
    [ NAME      <String> ]
    [ UNITS     <String> ]
    [ TYPE      <String> ]
    [ TRACe     <Label> ]
    { LOCation (
        LABel    <Label>
        INDex    <+NR1>
    ) } *
    [ VALues    <Numeric> { , <Numeric> } * ]
)
```

## 1999 SCPI Data Interchange Format

The block modifier <Label> value is optional and allows for the unique identification of the MEASurement sub-block. As all <Label> values are unique, no two MEASurement sub-blocks may have the same <Label> within the same DATA block.

The TRACe keyword allows association of the measurement or point with an arbitrary subset of DATA(CURVe) data.

Measurements derived from DATA block data are considered to have been derived from the data after SCALe and OFFSet (specified in the DIMension block) have been applied.

### 6.2.5.1    **NOTE**

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “This data is retest of test 14”

6

### 6.2.5.2    **NAME**

This is an arbitrary name of the measurement or point. NAME is typically more descriptive than the MEASurement block modifier <Label> and it does not have to be unique.

Value type is <String>. It is recommended that the string length be 32 characters or less. NAME may occur only once and only one value is allowed.

Example: NAME “Trigger pattern 4”

### 6.2.5.3    **UNITS**

This indicates units, such as Volts, Amps, Hz, etc without multipliers.

Value type is <String>. It is recommended that the string length be 32 characters or less. These units may differ from those specified in other DIMension blocks since the values represented herein may be derived or ancillary values.

Recommended units are those specified by IEEE 488.2 <SUFFIX PROGRAM DATA>, the null string, and UNKNOWN. The null string implies that the data has no dimensional units and UNKNOWN indicates that the units are unknown.

If the TRACe keyword is present, the UNITS value may differ from that of the trace. Only one occurrence of UNITS is allowed and only one value is allowed. If omitted, the units are as specified in the DIMension block.

Example: UNITS A

### 6.2.5.4    **TYPE**

This indicates the type of measurement or point.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of TYPE is allowed and only one value is allowed.

The one defined type is:

UNKNOWN   Unknown (default)

## 1999 SCPI Data Interchange Format

Example: TYPE "UNKNOWN"

### 6.2.5.5 **TRACe**

This is the trace associated with the point(s). TRACe allows a sub-set of the data to be referenced.

Value type is <Label>. Only values defined by the TRACe block modifier <Label> value are allowed. Only one occurrence of TRACe is allowed and only one value is allowed. If this keyword is present, it determines which dimensions shall be included in the LOCation sub-block for MEASurement.

Example: TRACe ADDR

### 6.2.5.6 **LOCation**

This specifies the label and index for one independent dimension. This sub-block is optional and may occur multiple times. There shall be a LOCation sub-block with a label value for each independent dimension present, or if a TRACe keyword is specified, then for each independent dimension in the specified trace.

#### 6.2.5.6.1 **LABel**

This is the label of the independent dimension.

Value type is <Label>. Only values previously defined by a DIMension block modifier <Label> value are allowed and the dimension shall be implicit. LABel is required and may occur only once. Only one value is allowed.

Example: LABel A1

#### 6.2.5.6.2 **INDex**

This specifies the index for the point.

Value type is <+NR1>. INDex is required and only one occurrence of INDex is allowed.

Example: INDex 274

### 6.2.5.7 **VALues**

This indicates the data value(s).

Value type is <Numeric>. VALues is optional and may occur only once. Multiple values are allowed.

Example: VALues 25.3, 93.7

## 6.2.6 DATA Block Example

```

DATA(
    NOTE "File 1A-23"
    DELTa(
        DIMension=Y(
            SCALe 0.1
            OFFSet 12.3
            SIZE 4096)
        DATE 1988,9,2
        TIME 23,3,0.25)
    CURVe(
        NAME ""
        CTYPe CRC16
        VALue 23.2,14.5,16.02,12.0
        CSUM 45)
    WAveform(
        NAME "19th Pulse"
        TRACe Z
        HLMETHOD "MODE"
        AMPLitude 3.902
        PWIDth 2.6E-7
        PDUTycycle 32.4
        RISE(
            TIME 2.8E-8)
        FALL(
            TIME 2.703E-8)
        REFERENCE(
            HIGH 5.85E-5
            LOW 5.14E-5
            MID 5.32E-5)
        RMS 6.4
        PTPeak 12.5)))

```

### 6.3 Dimension

Each DIMension block describes one dimension of DATA(CURVe) data. A dimension may be either explicit, in which case values are present in the DATA(CURVe) block, or implicit, in which case the values are not present, but are instead generated by a function. See 1.3 in this section. The order in which dimensions appear determines the data ordering in the DATA block.

The DIMension and ORDer blocks describe the logical structure and organization of data in the DATA(CURVe) block. The ENCode block describes the physical representation or encoding of the data. The ORDer block describes the ordering of the dimensional data elements in the DATA(CURVe) block. Together they provide the sufficient and necessary information for correct extraction of the data.

## 1999 SCPI Data Interchange Format

Whether a dimension is to be considered an independent or dependent dimension is determined by the TRACe block. In the absence of a TRACe block, the following simple rule is applied: if one or more implicit dimensions are present, they are considered to be independent, and all explicit dimensions are considered to be dependent.

A DIMension block is required for each dimension of the data (implicit and explicit). The DIMension block is defined as:

```
dimension -> DIMension = <Label>(
    [NOTE    <String>]
    [NAME    <String>]
    TYPE    {IMPLicit | EXPLicit}
    [SCALe   <Numeric>]
    [OFFSet  <Numeric>]
    [SIZE    <+NRL>]
    UNITS   <String>
    [encode]
)
```

Each dimension is uniquely distinguished by its required block modifier <Label> value. No two DIMension blocks may have the same <Label> value. Dimension labels should be kept short, yet meaningful.

### 6.3.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “DS1C at test point 12”

### 6.3.2 NAME

This is an arbitrary name or description of the dimension. NAME is typically more descriptive than the dimension block label.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of NAME is allowed and only one value is allowed.

Example: NAME “DS1C Voltage/Amplitude”

### 6.3.3 TYPE

This indicates if the dimension is implicit or explicit.

TYPE takes a specified enumerated set of values. TYPE is required, may occur only once, and takes one of the two following enumerated set values:

IMPLicit	Values for this dimension are not in DATA(CURVe), but are derived from a linear function, $y = mx + b$ .
----------	---

EXPLicit	Values for this dimension are present in DATA(CURVe).
----------	---

Example: TYPE EXPL

**6.3.4 SCALE**

Scaling factor to be applied to the dimension's values in DATA(CURVe(VALues)).

Value type is <Numeric>. Default value is 1.0. Only one occurrence of SCALE is allowed and only one value is allowed. The SCALE value may be overridden by DATA(DELTa(DIMension(SCALE))).

SCALE is always applied to DATA(CURVe) data, but not to any other data, such as DATA(WAVEform).

For implicit dimensions:

$$X' = (\text{SCALE} * i) + \text{OFFSet} \quad i = 1 \text{ to SIZE}$$

For explicit dimensions:

$$X' = (\text{SCALE} * X) + \text{OFFSet}$$

where X is a value from DATA(CURVe(VALues)). Also see Section 5.6, ORDER.

Example: SCALE 0.5E-2

**6.3.5 OFFSet**

Offset factor to be applied to the dimension's values in DATA(CURVe(VALues)).

Value type is <Numeric>. Default value is zero. Only one occurrence of OFFSet is allowed and only one value is allowed. The OFFSet value may be overridden by DATA(DELTa(DIMension(OFFSet))).

OFFSet is always applied to DATA(CURVe) data, but not to any other data, such as DATA(WAVEform).

For implicit dimensions:

$$X' = (\text{SCALE} * i) + \text{OFFSet} \quad i = 1 \text{ to SIZE}$$

For explicit dimensions:

$$X' = (\text{SCALE} * X) + \text{OFFSet}$$

X is a value from DATA(CURVe(VALues)).

Example: OFFSet -1.0E-2

**6.3.6 SIZE**

This is the number of points or values in the dimension.

Value is type <+NR1>. Only one occurrence of SIZE is allowed and only one value is allowed. The SIZE value may be overridden by DATA(DELTa(DIMension(SIZE))).

The following two invariants hold:

1. All explicit dimensions must have the same SIZE value.
2. The product of the SIZE values of the implicit dimensions must equal the SIZE value of any explicit dimension.

If SIZE is not specified for a dimension, it is assumed to be consistent with the above invariants. It is an error if there is insufficient information to determine a dimension's SIZE. For example, SIZE cannot be determined if two or more implicit dimensions do not specify a

## 1999 SCPI Data Interchange Format

SIZE, nor can it be determined if two or more explicit dimensions specify different SIZE values.

It is strongly recommended that SIZE be specified for all dimensions whenever possible.

Example: SIZE 512

### 6.3.7 UNITS

This indicates units, such as Volts, Amps, Hz, etc. Multipliers are not allowed.

Value type is <String>. It is recommended that the string length be 32 characters or less. Recommended units are those specified by IEEE 488.2 <SUFFIX PROGRAM DATA>, the null string, and UNKNOWN. The null string implies that the data has no dimensional units and UNKNOWN indicates that the units are unknown or unspecified. UNITS is required and only one occurrence of UNITS is allowed.

Example: UNITS "V"

### 6.3.8 ENCode

An ENCode sub-block may appear subordinate to a DIMension block. The DIMension(ENCode) block keywords and sub-blocks are the same as the major ENCode block described in 6.4 in this section. They define data resolution, special range values, and encoding format for <Block> value type data.

The scope of a DIMension(ENCode) block includes only the DATA(CURVe) block and the DIMension(ENCode) block itself. The scope of the major ENCode block includes all other values in the DATA block.

Each keyword specified in a DIMension(ENCode) block overrides the corresponding keyword in the major ENCode block, but only for the values in the DATA(CURVe) block for the specific dimension.

Only one ENCode block can be specified subordinate to a DIMension block. It is not required.

### 6.3.9 DIMension Block Example

```
DIM=X (
    NOTE "Procedure 12.8"
    NAME "DS3"
    TYPE EXPLicit
    UNITS "S"
    SCALe 1.0
    OFFSet 0.0
    SIZE 1024
    ENCode (
        NOTE "This ENCode block affects only X dimension values in the
              DATA(CURVe) block"
        FORMat INT16
        NVALue -32768
```

```
ORANge 32767
URANge -32767
HRANge 32766
LRANge -32766
RESolution 1))
```

## 6.4

**ENCode**

The ENCode block addresses data resolution, special range values, and encoding format <Block> value type data.

The scope of an ENCode block is determined by its hierarchical level in the data format structure. If the ENCode block is at the same level as DIMension blocks, it applies globally to all data in the DATA blocks. A global encoding specification can, however, be overridden on a dimension-by-dimension basis with an ENCode block that is subordinate to the particular dimension. A keyword specification at the higher level rules unless explicitly superceded by the same keyword at the subordinate level.

Only one ENCode block can be specified at the same hierarchical level as DIMension blocks, and only one ENCode block can be specified subordinate to a DIMension block.

The ENCode block is not required at any hierarchical level. However, the specification of HRANge and LRANge for all explicit dimensions is strongly recommended. The ENCode block is defined as:

```
encode -> ENCode (
    [ NOTE      <String> ]
    [ FORMAT    { IFP32 | IFP64
                  | INT8  | INT16 | INT32 | INT64
                  | SFP32 | SFP64
                  | SINT16 | SINT32 | SINT64
                  | SUINT16 | SUINT32 | SUINT64
                  | UINT8 | UINT16 | UINT32 | UINT64 } ]
    [ NValue     <Numeric> ]
    [ ORANge    <Numeric> ]
    [ URANge    <Numeric> ]
    [ HRANge    <Numeric> ]
    [ LRANge    <Numeric> ]
    [ RESolution <+Numeric> ]
)
```

If no ENCode block is specified, the default values for each keyword prevail. The ENCode block sub-blocks and keywords are:

## 6.4.1

**NOTE**

This is arbitrary text. Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “Not-a-number and plus and minus infinity do not follow the IEEE 488.2 recommendation”

### 6.4.2 FORMat

This is the data encoding format for CURVe(VALUE) data. <Block> value type is introduced by a pound sign followed by one or more decimal digit characters. See section 3.4.5. ASCII representation uses comma separated <Numeric> data.

Specifying a block FORMat does not require that the values in DATA block be formatted as <Block> value type. Values may always be specified as <Numeric>.

If the ENCode block is subordinate to a DIMension block, then FORMat applies only to values in DATA(CURVe(VALUes)) for the particular dimension.

FORMat takes a specified enumerated set of values. Only one occurrence of FORMat is allowed and only one of the following enumerated set values is allowed:

ASCII	Comma-separated <Numeric> data
IFP32	IEEE floating point, 32 bits.
IFP64	IEEE floating point, 64 bits.
INT8	IEEE 8-bit signed integer (default).
INT16	IEEE 16-bit signed integer.
INT32	IEEE 32-bit signed integer.
INT64	IEEE 64-bit signed integer.
SFP32	IEEE swapped floating point, 32 bits.
SFP64	IEEE swapped floating point, 64 bits.
SINT16	Swapped IEEE 16-bit signed integer.
SINT32	Swapped IEEE 32-bit signed integer.
SINT64	Swapped IEEE 64-bit signed integer.
SUINT16	Swapped IEEE 16-bit unsigned integer.
SUINT32	Swapped IEEE 32-bit unsigned integer.
SUINT64	Swapped IEEE 64-bit unsigned integer.
UINT8	IEEE 8-bit unsigned integer.
UINT16	IEEE 16-bit unsigned integer
UINT32	IEEE 32-bit unsigned integer
UINT64	IEEE 64-bit unsigned integer

All signed integers are two's complement. IEEE formats are defined by *IEEE 488.2* Binary Integer and Binary Unsigned Integer codes. "Swapped" means that the byte order (from low to high address) is from least significant to most significant byte.

For example, to transfer #H1234 as an INT16 (16-bit integer) across a byte oriented transmission medium, transfer the first byte as 12 (hex) and the second byte as 34 (hex). To transfer #H1234 as an SINT16 (swapped 16-bit integer), transfer the first byte as 34 (hex) and the second byte as 12 (hex).

Similarly, to transfer #H12345678 as an INT32 (32-bit integer), transfer the bytes in the order: 12 (hex), 34 (hex), 56 (hex), and 78 (hex). To transfer #H12345678 as an SINT32

(32-bit swapped integer), transfer the bytes in the order: 78 (hex), 56 (hex), 34 (hex), and 12 (hex).

Example: FORMat INT16

#### 6.4.3 **NVALue**

This is the numeric value (before application of SCALe and OFFSet) that is to be treated as implying “not a number” or “no value recorded” for values in the DATA block.

Value type is <Numeric>. Only one occurrence of NVALue is allowed and only one value is allowed. If NVALue is not present, no not-a-number value is defined for integer formats. For IEEE floating point formats, no NVALue is ever specified. In this case, NVALue is as defined by IEEE 754 for not-a-number. For ASCII, the default value is 9.91E+37.

Example: NVALue 9999

#### 6.4.4 **ORANge**

This indicates the numeric value (before application of SCALe and OFFSet) that is to be treated as over range for values in the DATA block.

Value type is <Numeric>. Only one occurrence of ORANge is allowed and only one value is allowed. If ORANge is not present, no over range value is defined for integer formats. For IEEE floating point formats, no ORANge is ever specified. In this case, ORANGe is as defined by IEEE 754 for plus infinity. For ASCII, the default value is 9.9E+37.

Example: ORANge #H7FFF

#### 6.4.5 **URANge**

This is the numeric value (before application of SCALe and OFFSet) that is to be treated as under range for values in the DATA block.

Value type is <Numeric>. Only one occurrence of URANge is allowed and only one value is allowed. If URANge is not present, no under range value is defined for integer formats. For IEEE floating point formats, no URANge is ever specified. In this case, URANGe is as defined by IEEE 754 for negative infinity. For ASCII, the default value is -9.9E+37.

Example: URANge -32767

#### 6.4.6 **HRANge**

This is the greatest signed value (before application of SCALe and OFFSet) that may be present in the DATA(CURVe(VALUE)).

Value type is <Numeric>. Only one occurrence of HRANge is allowed and only one value is allowed. The value specified shall be at least as great as the greatest value present in the data but may be greater.

NVALue, ORANge, and URANge values are not considered when determining range. Units are the measurement curve units after SCALe and OFFSet are applied to the data.

Example: HRANge 32766

## 1999 SCPI Data Interchange Format

### 6.4.7 LRAnge

This is the least value (before application of SCALe and OFFSet) that may be present in DATA(CURVe(VALUE)).

Value type is <Numeric>. Only one occurrence of LRAnge is allowed and only one value is allowed. The value specified shall be at least as small as the least value present in the data but may be less.

NVALue, ORAnge, and URAnge values are not considered when determining range. Units are the measurement curve units after SCALe and OFFSet are applied to the data.

Example: LRAnge -32766

### 6.4.8 RESolution

This is the minimum resolution of data values present in the DATA block.

Value type is <+Numeric>. Only one occurrence of REsolution is allowed and only one value is allowed.

Units are the measurement curve units after SCALe and OFFSet are applied to the data.

Example: RESolution 1

### 6.4.9 ENCode Block Example

```
ENCode (
    NOTE "Acquisition resolution was 10 bits"
    FORMat INT16
    NVALue -32768
    ORANge 32767
    URANge -32767
    HRANge 32766
    LRANge -32766
    RESolution 1)
```

### 6.5 IDENtify

This is the IDENtify block contains fields uniquely identifying the data and describing the environment in which it was created or acquired. The precise definitions for the contents of all text string values in the IDENtify block is left to the creator of the data set. However, the general category shall be followed.

The IDENtify block may occur only once, but is not required. The IDENtify block is defined as:

```

identify -> IDENTify (
    [NOTE           <String>]
    [NAME           <String>]
    [TECHnician    <String> {,<String>}*]
    [PROJect        <String>]
    [DATE           <NR1>,<+NR1>,<+NR1>
                   [, <NR1>,<+NR1>,<+NR1>]]
    [TIME           <0+NR1>,<0+NR1>,<0+Numeric>
                   [, <0+NR1>,<0+NR1>,<0+Numeric>]]
    [UUT(
        [NAME           <String>]
        [ID             <String> {,<String>}*]
        [DESIGN         <String> {,<String>}*]
    )]
    [TEST(
        [NAME           <String>]
        [SERIES         <String>]
        [NUMBER         <String> {,<String>}*]
    )]
    [HISTORY        <String> {,<String>}*]
)

```

### 6.5.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “24 AWG may have been out of calibration”

### 6.5.2 NAME

This is the user-defined name identifying the data set. NAME is typically more descriptive than the block modifier.

Value type is <String>. It is recommended that the string length be 32 characters or less. This field is optional. Only one occurrence of the NAME keyword is allowed. Only one value is allowed.

Example: NAME “Spectral Power Test”

### 6.5.3 TECHnician

This identifies the technician(s) involved in the generation or modification of the data.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of the TECHnician keyword is allowed. Multiple values, up to 7, are allowed.

Example: TECHnician “John H. Doe”

## 1999 SCPI Data Interchange Format

### 6.5.4 PROject

This is the name of project, job, task, etc.

Value type is <String>. It is recommended that the string length be 32 characters or less.  
Only one occurrence of PROject is allowed and only one value is allowed.

Example: PROject “S27A408”

### 6.5.5 DATE

This is the date of data creation or acquisition.

This keyword takes a set of three comma separated value types to specify the date corresponding to year, month, and day.

The years parameter is type <NR1>. It always includes century and millennium information.

The months parameter is type <+NR1>. Its range is 0 to 12.

The days parameter is type <+NR1>. Its range is 0 to 31 (depending on the month).

If two sets of values are specified, they define an inclusive range of dates.

Example: DATE 1776,07,04 indicates the 4th of July, 1776

### 6.5.6 TIME

This is the time of data creation or acquisition.

This keyword takes a set of three comma separated value types to specify the time corresponding to hour, minute, and second.

The hours parameter is type <0+NR1>. Its range is 0 to 23. This parameter is always expressed in 24 hour format.

The minutes parameter is type <0+NR1>. Its range is 0 to 59.

The seconds parameter is type <0+Numeric>. Its range is 0 to 60 for instruments receiving data and 0 to 59.999... for instruments sending data.

If two sets of values are specified, they define an inclusive range of time.

Example: TIME 23,59,59 indicates one second before midnight.

### 6.5.7 UUT

This is a description of the “unit under test”. This block may occur only once.

#### 6.5.7.1 NAME

This is the name or type of unit under test. NAME is similar in use to ID, except that it is typically more descriptive and only one value is allowed.

Value type is <String>. It is recommended that the string length be 32 characters or less.  
Only one occurrence of the NAME keyword is allowed and only one value is allowed.

Example: NAME “Model 29AP402”

## 6.5.7.2 ID

This is the identification of the unit under test.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of the ID keyword is allowed. Multiple values, up to 7, per keyword are allowed.

Example: ID “AA32303”,“B7”

## 6.5.7.3 DESign

This identifies the design specification.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of the DESign keyword is allowed. Multiple values, up to 7, are allowed.

Example: DESign “2W32 Rev 3C”

## 6.5.8 TEST

This is the test description. This block may occur only once.

### 6.5.8.1 NAME

This is the test series name.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of the NAME keyword is allowed and only one value is allowed.

Example: NAME “DC Parametrics”

### 6.5.8.2 SERies

This is the number or id of test series.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of SERies is allowed and only one value is allowed.

Example: SERies “24B208”

### 6.5.8.3 NUMBER

This is the number or id of particular test.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of the NUMBER keyword is allowed. Multiple values, up to 7, are allowed.

Example: NUMBER “A3”,“B6”,“B7”

## 6.5.9 HISTory

This is the derivation history of the data set, such as names of tests, instruments, instrument systems, and software packages.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of the HISTory keyword is allowed. Multiple values, up to 7, are allowed and imply an ordering from oldest to most recent.

## 1999 SCPI Data Interchange Format

Example: HISTory “T24.WFM”, “CCITT DS3 Template”

### 6.5.10 IDENTify Block Example

```
IDENTify (
    NOTE "This IDENTify block example contains examples of all keywords"
    NAME ""
    TECHnician "Matt Wilson"
    PROject "Acorn"
    DATE 1988,9,2
    TIME 23,3,0.25
    UUT (
        NAME "Ironman"
        ID "007"
        DESign "Rev D")
    TEST (
        NAME "Cal Menu"
        SERies "3A"
        NUMBER "5")
    HISTory "Tinman", "Strawman")
```

6

### 6.6 ORDER

The ORDER block defines the physical layout of data within the DATA(CURVe) block. Data may be ordered as n-tuples, in which the data for each explicit dimension is “inter-mixed”. This is also called row order. Data may also be ordered by dimension. When ordered by dimension, all data for each dimension is grouped consecutively, with all data for one dimension followed by all data for the next dimension. This is also called column order.

When more than one implicit dimension exists, the first implicit dimension index always increments the slowest and the last implicit dimension index always increments the fastest.

The ORDER block may occur only once, but is not required. If omitted, the default values for its keywords prevail. The ORDER block is defined as:

```
order -> ORDER (
    [NOTE <String>]
    [BY {TUPLE | DIMension}]
)
```

#### 6.6.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “Most important first”

#### 6.6.2 BY

This indicates if the ordering is by tuple or dimension.

## 1999 SCPI Data Interchange Format

BY takes a specified enumerated set of values. BY may occur only once, and takes one of the two following enumerated set values:

TUPLe	Data values are organized by n-tuple (default).
DIMension	Data values are organized by dimension.

### 6.6.3 Examples

Suppose the sample measurements in the following example were made according to the scenario of 1.3 of this section. The data has five dimensions. Assume also that the five DIMension blocks have label values corresponding to the headings in as shown below.

Sample Measurements

X	Y	Z	Temp	Hum
5	1	8.1	18.1	61
5	2	9.2	20.2	62
7	1	6.3	16.3	63
7	2	3.4	16.4	64
9	1	8.5	18.5	65
9	2	3.6	16.6	66

6

Example 1:

If all values are present in the data, the five DIMension blocks will have TYPE EXPLicit (see 6.3 in this section). If no ORDER block is present, the data will be ordered:

HUM, TEMP, X, Y, Z, HUM, TEMP, X, Y, Z ...

That is, the data is ordered by 5-tuple and by column order of occurrence within a 5-tuple. Note that the rows may appear in any order since there are no implicit dimensions.

The DIMension and DATA blocks for this example are:

```
DIMension=HUM (
    TYPE EXPLicit
    SIZE 6
    UNITs "PCT"
    ENCode (
        HRANge 60
        LRANge 66))
DIMension=TEMP (
    TYPE EXPLicit
    SIZE 6
    UNITs "CEL"
    ENCode (
        HRANge 25
        LRANge 15))
DIMension=X (
    TYPE EXPLicit
    SIZE 6)
```

## 1999 SCPI Data Interchange Format

```
6
    UNITS "M"
    ENCode (
        HRAnge 9
        LRAnge 5))
    DIMension=Y(
        TYPE EXPLicit
        SIZE 6
        UNITS "M"
        ENCode (
            HRAnge 2
            LRAnge 1))
    DIMension=Z(
        TYPE EXPLicit
        SIZE 6
        UNITS "M"
        ENCode (
            HRAnge 10
            LRAnge 0))
    DATA (
        CURVe (
            VALues 61, 18.1, 5, 1, 8.1,
            64, 16.4, 7, 2, 3.4,
            65, 18.5, 9, 1, 8.5,
            66, 16.6, 9, 2, 3.6,
            62, 20.2, 5, 2, 9.2,
            63, 16.3, 7, 1, 6.3))
```

Example 2:

If the coordinate values for the X and Y dimensions are omitted from the data, these dimensions are implicit (TYPE is IMPLicit).

The X and Y dimensions are easily derived by linear functions:

$$x = 2m + 3$$

$$y = n$$

where m and n are indices related to the ordering of the values for Z, TEMP, and HUM.

The DIMension and DATA blocks for this example are:

```
DIMension=TEMP(
    TYPE EXPLicit
    SIZE 6
    UNITS "CEL"
    ENCode (
        HRAnge 25
        LRAnge 15))
DIMension=X(
    TYPE IMPLicit
    SIZE 3)
```

```

    UNITS "M"
    SCALe 2
    OFFSet 3)
DIMension=Y(
    TYPE IMPLicit
    SIZE 2
    UNITS "M")
DIMension=Z (
    TYPE EXPLicit
    SIZE 6
    UNITS "M"
    ENCode (
        HRANge 10
        LRANge 0))
DIMension=HUM (
    TYPE EXPLicit
    SIZE 6
    UNITS "PCT"
    ENCode (
        HRANge 60
        LRANge 66))
DATA (
    CURVe (
        VALUes 18.1, 8.1, 61,
        20.2, 9.2, 62,
        16.3, 6.3, 63,
        16.4, 3.4, 64,
        18.5, 8.5, 65,
        16.6, 3.6, 66))

```

Note that the humidity DIMension now appears last and that DATA values in the DATA block are therefore in a different order. The reordering of the X and Y DIMension blocks affects only the default determination of the independent and dependent dimensions (see chapter 6.3). In this second example, the presence of implicit dimensions forces a required row order on the VALUES values.

## 6.7

### REMark

This block is for information not fitting elsewhere.

The REMark block may occur only once, but is not required. The REMark block is defined as:

```

remark ->   REMark(
                [NOTE   <String> { ,<String>}*]
            )

```

## 1999 SCPI Data Interchange Format

### 6.7.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed. Multiple values are allowed and no ordering is implied.

Example: NOTE “The REMark block is the appropriate place for observations and annotations about the data, its acquisition, creation, or modification. In general, it's a place to stuff technical information.”

### 6.7.2 REMark Block Example

REMark (NOTE “A remarkable data structure”)

### 6.8 DIF

The DIF block is the first block in a dif data set. It uniquely defines the expression data as being a dif data set with the beginning DIF character sequence. The DIF block provides version information to assist the parser in determining compatibility. It further defines the scope of the data set.

The DIF block may occur only once and is required. The DIF block is defined as:

```
difid ->      DIF(
                  [ NOTE           <String> ]
                  VERSION         <+Numeric>
                  [ SCOPE          PREamble | FULL ]
)

```

#### 6.8.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “First Version”

#### 6.8.2 VERSion

The DIF(VERSion) number exactly tracks the SCPI SYSTem:VERSion number (Command Reference, 21.20) and correlates changes and extensions in the DIF volume with the rest of SCPI.

Note: The DIF(VERSion) keyword was new in 1994.0 and was created as a part of a one-time major restructuring and simplification of DIF. DIF parsers designed prior to 1994 will not be compatible with 1994.0 and following versions.

Value type is <+Numeric>. Only one value is allowed. Only one occurrence of VERSion is allowed.

Note that while version numbers increase in value in time sequence, lesser version numbers will not necessarily be guaranteed to be fully compatible.

Example: VERSion 1993.0

### 6.8.3 SCOPe

This indicates the scope of the DIF block.

SCOPe takes a specified enumerated set of values. SCOPe may occur only once and takes one of the two following enumerated set values:

When SCOPe is set to PREamble, the DIF block omits the following data:

- dif blocks of waveform, measurement, and delta.
- dif curve block keywords VALues and CSUM and their parameters.

The various notes and identification parameters may be sent if absolutely required.

When SCOPe is set to FULL, a complete dif data set as described in Section 4.3 is present (default).

Example: SCOPe PREamble

### 6.8.4 DIF Block Example

```
DIF(
    NOTE "Third revision of standard"
    VERSion 1993.0)
```

## 6.9 TRACe

TRACe blocks describe logical relationships among all or some of the dimensions. They may define functions, curves, surfaces, or arbitrary sets or data groupings. They also provide a convenient means of describing subsets of the data. The windowing effect of the TRACe blocks may, for example, be used to make two-dimensional “cuts” through three-dimensional data. Other blocks, such as the VIEW block, build on TRACe blocks. TRACe block information is built from DIMension blocks; VIEW blocks are built from TRACe block information.

If no TRACe block is present, the independence or dependence of a dimension is determined by rules described in 6.3 in this section.

The TRACe block is optional and multiple TRACe blocks are allowed. The TRACe block is defined as:

```
trace ->      TRACe = <Label>(
                [ NOTE           <String> ]
                [ NAME           <String> ]
                [ SYMMetry       <String> ]
                { INDependent(
                    LAbel        <Label>
                    [ START        <+NR1> ]
                    [ STOP         <+NR1> ]
                ) } *
                { DEPendent(
```

## 1999 SCPI Data Interchange Format

```
    LABEL      <Label>
) } *
)
```

The TRACe block modifier <Label> value is required. It provides for the unique identification of the TRACe block. As <Label> for the entire data set are unique, no two TRACe blocks may have the same <Label>.

Explicit and implicit, as applied to dimensions, are distinguished from independent and dependent as follows. Implicit and explicit are concerned with how the data is physically represented and structured. Independent and dependent relate to how a dimension is interpreted.

### 6.9.1 NOTE

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “Values generated pursuant to 12A34D”

### 6.9.2 NAME

This is an arbitrary name or description of the trace.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of NAME is allowed and only one value is allowed.

Example: NAME “Gain”

### 6.9.3 SYMMetry

This specifies the type of symmetry on a per dimension basis. Only one occurrence of SYMMetry is allowed.

Value type is <String>. It is recommended that the string length be 32 characters or less. The following values are predefined:

UNKNOWN	Symmetry is unknown. (default)
NONE	No symmetry.
PEVEN	Positive and even symmetry.
NEVEN	Negative and even symmetry.
PODD	Positive and odd symmetry.
NODD	Negative and odd symmetry.

Example: SYMMetry “UNKNOWN”

### 6.9.4 INDependent

This identifies one independent dimension that constitutes the trace. If independence and dependence are not meaningful, the trace should be defined using the INDependent block. This sub-block may occur multiple times.

STARt and STOP parameters represent integer values referring to indexes on implicit dimensions and therefore apply only to implicit dimensions

#### 6.9.4.1 **LABel**

This is the label of the independent dimension.

Value is of type <Label>. Only values defined by the DIMension block label are allowed. LABel is required and may occur only once. Only one value is allowed.

Example: LABel X

#### 6.9.4.2 **STARt**

This specifies the starting index (inclusive).

Value is of type <+NR1>. Only one occurrence of STARt is allowed and only one value is allowed. This keyword may be present only if the dimension is implicit.

If STARt is omitted, then the default value is 1. If the index value is greater than the dimension's SIZE value (from the applicable DIMension or DATA(DELTa(DIMension)) block), then an error is reported.

Example: STARt 12

#### 6.9.4.3 **STOP**

This specifies the ending index (inclusive).

Value is of type <+NR1>. Only one occurrence of STOP is allowed and only one value is allowed. This keyword may be present only if the dimension is implicit.

If STOP is omitted then SIZE is used. If the index value is greater than the dimension's SIZE value (from the applicable DIMension or DATA(DELTa(DIMension)) block), then an error is reported.

If the stopping index value is less than the starting index value then an error is reported.

Example: STOP 982

### 6.9.5 **DEPendent**

This identifies a dependent dimension that constitutes the trace. This sub-block may occur multiple times.

#### 6.9.5.1 **LABel**

This is the label of the dependent dimension.

Value is of type <Label>. Only values defined by the DIMension block label are allowed. LABel is required and may occur only once. Only one value is allowed.

Example: LABel Y

### 6.9.6 TRACe Block Example

```
TRACe=TEMP (
    NOTE "Missing values were interpolated"
    NAME "Transient acceleration"
    SYMMetry "UNKNOWN"
    INDependent (
        LABel X
        STARt 1
        STOP 10)
    INDependent (
        LAB Y
        STARt 100
        STOP 200)
    INDependent (
        LABel Z
        STARt 2
        STOP 4)
    DEPendent (
        LABel A))
```

6

### 6.10 VIEW

VIEW blocks provide semantic information about the data in DATA(CURVe). With a VIEW block, one can show relationships between and among subsets of the data as defined by TRACe blocks. For example, a complex waveform or surface can be easily defined, as can a waveform and its upper and lower envelopes.

The VIEW block builds on traces described in a TRACe block. A trace commonly defines a simple function, but the data format allows for the definition of sets, surfaces and other complex functions.

For example, a time varying envelope might be described by three DIMension blocks. The two explicit dimensions define the two Y-axis values for the upper and lower components of the envelope. The implicit dimension defines the common X-axis. While the organizational structure of the data is defined by the DIMension and ORDer blocks, its meaning is determined by a VIEW block. It is the view block that identifies which dimension is the upper envelope and which is the lower.

Alternatively, the same data could be viewed as a complex waveform. The DIMension blocks would be the same, but a VIEW block would identify which dimension is the real component and which is the complex component.

The VIEW block is optional and multiple VIEW blocks are allowed. The VIEW block is defined as:

## 1999 SCPI Data Interchange Format

```
view ->      VIEW = <Label> (
    [ NOTE           <String> ]
    [ NAME           <String> ]
    { ENVelope(
        UPPer         <Label>
        LOwEr         <Label>
        [ FUNCTIon   <Label> ]
    ) } *
    { RCOMplex(
        REAL          <Label>
        IMAGinary     <Label>
    ) }
    { PCOMplex(
        MAGNitude    <Label>
        PHASE         <Label>
    ) }
)
```

6

The modifier label is required. It provides for the unique identification of the VIEW block. As all <Label> elements are unique, no two VIEW blocks may have the same label.

A VIEW block takes the following sub-blocks. Except for NAME and NOTE, only one type of the other sub-blocks may be present:

### 6.10.1 **NOTE**

This is arbitrary text.

Value type is <String>. Text should be human-readable. Only one occurrence of the NOTE keyword is allowed and only one value is allowed.

Example: NOTE “Template defines the max and min values for calibrating mil spec 97654-D”

### 6.10.2 **NAME**

This is the name of the view. NAME is typically more descriptive than the VIEW block label and does not need to be unique.

Value type is <String>. It is recommended that the string length be 32 characters or less. Only one occurrence of NAME is allowed and only one value is allowed.

Example: NAME “97654-D”

### 6.10.3 **ENVelope**

This identifies which two traces constitute the envelope for a third in DATA(CURVe) data. Multiple occurrences of ENVelope are allowed.

## 1999 SCPI Data Interchange Format

### 6.10.3.1 **UPPer**

This specifies the name of the upper envelope trace.

Value type is <Label> and is restricted to a value defined by a TRACe block modifier <Label> value. Only one occurrence of UPPer is allowed and only one value is allowed.

Example: UPPer YH

### 6.10.3.2 **LOWer**

This specifies the name of the lower envelope trace.

Value type is <Label> and is restricted to a value defined by a TRACe block modifier <Label> value. Only one occurrence of LOWer is allowed and only one value is allowed.

Example: LOWER YL

### 6.10.3.3 **FUNCTION**

This specifies the set of data being enveloped.

Value type is <Label> and is restricted to a value defined by a TRACe block <Label>. Only one occurrence of FUNCtion is allowed and only one value is allowed.

Example: FUNCtion Y

### 6.10.4 **RCOMPlEx**

This identifies which two TRACes constitute a rectangular complex waveform. RCOMPlEx may only occur once.

#### 6.10.4.1 **REAL**

This specifies the name of the real trace.

Value is of type <Label>. Only values defined by the TRACe block are allowed. REAL is required and may occur only once. Only one value is allowed.

Example: REAL S11REAL

#### 6.10.4.2 **IMAGinary**

This specifies the name of the imaginary trace.

Value is of type <Label>. Only values defined by the TRACe block are allowed. IMAGinary is required and may occur only once. Only one value is allowed.

Example: IMAGinary S11IMAG

### 6.10.5 **PCOMplex**

This identifies the which two TRACes constitute a polar complex waveform. PCOMplex may only occur once.

#### 6.10.5.1 **MAGNitude**

This specifies the name of the magnitude trace.

Value is of type <Label>. Only values defined by the TRACe block are allowed. MAGNitude is required and may occur only once. Only one value is allowed.

Example: REAL S11MAG

### 6.10.5.2 PHASE

This specifies the name of the phase trace.

Value is of type <Label>. Only values defined by the TRACe block are allowed. PHASE is required and may occur only once. Only one value is allowed.

Example: PHASE S11PHASE

### 6.10.6 VIEW BLOCK EXAMPLE

```
VIEW=T3_J4 (
    NOTE "Mil spec 97654-D replaces 54679-D"
    NAME "Mil spec 97654-D"
    ENV (
        UPPer YH
        LOWER YL))
```

## **1999 SCPI Data Interchange Format**

---

## 7 Data Format Example

The following data set exemplifies the use of several related blocks in the data interchange format. The example below is shown encapsulated in parentheses; that is, as a <dif\_expression>.

```
( DIF (
    VERsion 1993.0)
IDENtify (
    DATE 1993,4,23
    TIME 16,4,14.23)
ENCode (
    FORMat INT8
    HRANge 127
    LRANge -128)
DIMension=YH (
    TYPE EXPLicit
    SCALe 2.000000E-002
    OFFSet -3.500000E-001
    SIZE 512
    UNITS "V")
DIMension=YL (
    TYPE EXPLicit
    SCALe 2.000000E-002
    OFFSet -3.500000E-001
    SIZE 512
    UNITS "V")
DIMension=X (
    TYPE IMPLICit
    SCALe 2.000000E-005
    OFFSet -1.024000E-002
    SIZE 512
    UNITS "s")
TRACe=H (
    INDependent(
        LABel X)
    DEPendent(
        LABel YH))
TRACe=L (
    INDependent(
        LABel X)
    DEPendent (
        LABel YL))
VIEW=ENV1 (
    ENVelope (
        UPPer H
        LOWER L))
```

## 1999 SCPI Data Interchange Format

```
DATA (
    CURVe (
        VALue #41024 {1024 8-bit significant bytes of data goes here}
        CSUM 139))
    WAveform (
        TRACe H
        RISE (
            TIME 1.04E-003)
        FALL (
            TIME 0.86E-003)))
```

In this example, we have a data set that follows the 1993.0 version of SCPI and is dated at about 4 PM on April 23rd, 1993. The data is encoded as 8-bit signed integers ranging from -128 to 127 and is presented as a sequence of 512 byte pairs representing two voltage signal levels (YH and YL). The raw bit values must be multiplied by .02 with .35 subtracted to yield the measured value in Volts. The first data pair occurred at a measured time (X) of (20 - 10,240 microseconds) or -10.22 milliseconds. Subsequent data pairs were recorded at 20 microsecond intervals.

Two traces (H and L) are specified with YH and YL voltage levels as the dependent variables and time as the independent variable. The traces can be viewed as envelope-type data and referred to as ENV1. The data itself is formatted in IEEE 488.2 block format (1024 bytes long) and its integrity can be verified by computing a 16-bit cyclic redundancy check and comparing the results to the stated (CSUM) value of 139.

Finally, the rise and fall times of the H trace have been computed and recorded in the data set as 1.04 milliseconds and 0.86 milliseconds.

# Index

---

## A

AMPLitude 6-10

## B

block 3-3  
    definition 3-2  
    unrecognized 4-6  
block modifier 3-2  
BY 6-28

## C

CCITT 6-5  
character set 3-1  
COUNT 6-13  
CRC16 6-5  
CSUM 6-5  
CTYPe 6-4  
CURVe 6-4  
CYCle block 6-13

## D

DATA block 6-1  
data format  
    example 7-1  
    extension 5-1  
    overview 1-1  
Data Interchange Format 1-2  
DATE  
    DELTa block 6-3, 6-26  
DELTa block 6-2  
DEPendent 6-35  
DESign 6-27  
DIF block 6-32  
`<dif_expression>` 1-1  
dimension 6-29  
    explicit 1-2  
    implicit 1-2  
DIMension block 6-3, 6-17

## E

ENCode block 6-21  
DIMension block 6-20

enumerated set  
    definition 3-4  
    errors 2-1  
ENVelope 6-37  
EXPLicit 6-18

## F

FALL block 6-11  
FREQuency 6-10  
FUNCTION 6-38

## G

grammar  
    notation 4-1  
    required blocks 4-6  
    required order 4-6  
    unrecognized blocks 4-6  
    unrecognized keywords 4-6

## H

HIGH  
    REFerence block 6-9  
    WAVeform block 6-8  
HISTory 6-27  
HLMETHOD 6-8  
HRANGE 6-23

## I

ID 6-27  
IDE NTify block 6-24  
IFP32 6-22  
IFP64 6-22  
IMPLICIT 6-18  
INDependent 6-34  
INDEX 6-16  
INT16 6-22  
INT32 6-22  
INT64 6-22  
INT8 6-22

## 1999 SCPI Data Interchange Format

### K

keyword  
  default values 2-1  
  definition 3-2  
  errors 2-1  
  multiple values 2-1  
  unrecognized 4-6

### L

<Label> 3-3  
  DEPendent block 6-35  
  INDependent block 6-35  
  LOCation block 6-16  
  LOCATION block 6-16  
LOW  
  REFerence block 6-9  
  WAveform block 6-9  
LOWER 6-38  
LRANge 6-24

### M

MAXimum 6-12  
MEAN  
  CYCLE block 6-14  
  WAveform block 6-13  
MEAsurement block 6-14  
METHOD 6-9  
MID 6-9  
MINimum 6-12

### N

NAME  
  CURVe block 6-4  
  DIMension block 6-18  
  IDENTify block 6-25  
  MEASurement block 6-15  
  TEST block 6-27  
  TRACe block 6-34  
  UUT block 6-26  
  VIEW block 6-37  
  WAveform block 6-8  
NDUTcycle 6-11  
NOTE  
  CURVe block 6-4  
  DATA block 6-2  
  DELTa block 6-2  
  DIF block 6-32  
  DIMension block 6-18

ENCode block 6-21  
IDENTify block 6-25  
MEASurement block 6-15  
ORDer block 6-28  
REMark block 6-32  
TRACe block 6-34  
VIEW block 6-37  
WAveform block 6-8

NUMber 6-27  
<Numeric> 3-2—3-3  
<+Numeric> 3-2  
<+Numeric> 3-3  
NVALue 6-23  
NWIDth 6-10

### O

OFFSet  
  DIMension block 6-3, 6-19  
ORAnge 6-23  
ORDer block 6-28  
OVERshoot  
  FALL block 6-12  
  RISE block 6-11

### P

PDUTcycle 6-11  
PERiod 6-10  
PREShoot  
  FALL block 6-12  
  RISE block 6-11  
PROject 6-26  
PTPeak 6-13  
PWIDth 6-10

### R

REFerence block 6-9  
REMark block 6-31  
RESolution 6-24  
RISE block 6-11  
RMS  
  CYCLE block 6-14  
  WAveform block 6-13

### S

SCALE  
  DIMension block 6-3, 6-19  
Scope 6-33  
SDEViation

## 1999 SCPI Data Interchange Format

CYCLE block 6-14  
WAVeform block 6-13  
SERies 6-27  
SFP32 6-22  
SFP64 6-22  
SINT16 6-22  
SINT32 6-22  
SINT64 6-22  
SIZE  
DIMension block 6-3, 6-19  
STARt 6-35  
STOP 6-35  
<String> 3-3  
SUINT16 6-22  
SUINT32 6-22  
SUINT64 6-22  
SUM16 6-5  
SUM8 6-5  
SYMMetry 6-34  
syntax  
block 3-2  
block modifier 3-2  
character set 3-1  
keyword 3-2  
overview 3-1  
value types 3-2  
white space 3-1

### T

TECHnician 6-25  
TEST block 6-27  
TIME  
DELTa block 6-3, 6-26  
FALL block 6-12  
RISE block 6-11  
TMAXimum 6-12  
TMINimum 6-12  
TRACe  
MEASurement block 6-16  
WAVeform block 6-8  
TRACe block 6-33  
TUPle 6-29  
TYPE  
DIMension block 6-18  
MEASurement block 6-15

### U

UINT8 6-22  
UNITS  
DIMension block 6-20

MEASurement block 6-15  
UPPer 6-38  
URAnge 6-23  
UUT block 6-26

### V

value type  
<+NR1> 3-3  
<+Numeric> 3-2  
<0+NR1> 3-3  
<0+Numeric> 3-3  
<Block> 3-3  
<Label> 3-3  
<NR1> 3-3  
<Numeric> 3-2  
<String> 3-3  
enumerated set 3-4

### VALues

CURVe block 6-5  
MEASurement block 6-16  
VERSion 6-32  
VIEW block 6-36

### W

WAVeform block 6-5  
white space 3-1

**Standard  
Commands for  
Programmable  
Instruments  
(SCPI)**

## **Volume 4: Instrument Classes**

---

**VERSION 1999.0**

**May, 1999**

**Printed in U.S.A.**

# Table of Contents

---

## Chapter 1 Introduction

1.1	Scope .....	1-1
1.2	Definition of Terms .....	1-1
1.3	Purpose .....	1-1
1.3.1	Guiding Designers .....	1-1
1.3.2	Achieving Consistency .....	1-2
1.4	Instrument Classification .....	1-2
1.4.1	Syntax .....	1-2
1.4.2	Examples .....	1-4
1.5	Compliance .....	1-4
1.6	Chapter Organization .....	1-5

## Chapter 2 Chassis Dynamometers

2.1	Base Functionality .....	2-2
2.1.1	Base Measurement Instructions .....	2-3
2.1.2	Base Device-oriented Functions .....	2-3
2.1.2.1	CALibration subsystem .....	2-4
2.1.2.2	CONTrol subsystem .....	2-5
2.1.2.3	MEMORY subsystem .....	2-5
2.1.2.4	SENSe subsystem .....	2-8
2.1.2.5	SOURce subsystem .....	2-8
2.1.3	STATUs Subsystem .....	2-10
2.1.3.1	OPERation .....	2-10
2.1.3.1.1	CDYNo:CONDition?<16 bit integer word> .....	2-10
2.1.3.2	QUEstionable .....	2-12
2.1.3.2.1	CDYNo:CONDition?<16 bit integer word> .....	2-12
2.2	Additional Functionality .....	2-14
2.3	Programming Examples .....	2-14
2.3.1	Coastdown .....	2-14
2.3.2	Road Load Simulation .....	2-16
2.4	New Commands for this Instrument Class .....	2-18
2.4.1	Dynamometer CALibration Subsystem .....	2-18
2.4.1.1	:BINertia .....	2-18
2.4.1.1.1	:AVERage? .....	2-19
2.4.1.1.2	:HSPEED <numeric_value> .....	2-19
2.4.1.1.3	:INITiate .....	2-19
2.4.1.1.4	:LSPEED <numeric_value> .....	2-20
2.4.1.1.5	:NRUNS <numeric_value> .....	2-20
2.4.1.1.6	:SDEViation? .....	2-20
2.4.1.1.7	:UPDAtE .....	2-20

## 1999 SCPI Instrument Classes

2.4.1.2	:PLOSS .....	2-20
2.4.1.2.1	:APCOff <numeric_value>,numeric_value>,<numeric_value>... .....	2-21
2.4.1.2.2	:INITiate .....	2-21
2.4.1.2.3	:LATime .....	2-21
2.4.1.2.4	:STIMe <numeric_value> .....	2-22
2.4.1.2.5	:UPDate .....	2-22
2.4.1.3	:WARMup .....	2-22
2.4.1.3.1	:INITiate .....	2-22
2.4.1.3.2	:SPEEd <numeric_value> .....	2-23
2.4.1.3.3	:TIMEout <numeric_value> .....	2-23
2.4.1.4	:ZERO .....	2-23
2.4.1.4.1	:FSENsor .....	2-23
2.4.1.4.1.1	:INITiate .....	2-23
2.4.1.4.1.2	:LATime .....	2-24
2.4.1.4.1.3	:LEVel? .....	2-24
2.4.1.4.1.4	:SPEEd <numeric_value> .....	2-24
2.4.1.4.1.5	:STIMe <numeric_value> .....	2-24
2.4.1.4.1.6	:UPDate .....	2-24
2.4.2	DIAGnostic Subsystem .....	2-24
2.4.3	MEMory Subsystem .....	2-24
2.4.3.1	:ARATe .....	2-26
2.4.3.1.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-26
2.4.3.1.1.1	:POINts? .....	2-26
2.4.3.2	:CINertia .....	2-26
2.4.3.2.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-26
2.4.3.2.1.1	:POINts? .....	2-26
2.4.3.3	:DRATE .....	2-26
2.4.3.3.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-26
2.4.3.3.1.1	:POINts? .....	2-27
2.4.3.4	:FORCe .....	2-27
2.4.3.4.1	:AACCELERation .....	2-27
2.4.3.4.1.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-27
2.4.3.4.2	:ADECELERation .....	2-27
2.4.3.4.2.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-27
2.4.3.4.3	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-27
2.4.3.4.3.1	:POINts? .....	2-28
2.4.3.4.4	>ZOFFset .....	2-28
2.4.3.4.4.1	[:MAGNitude] .....	2-28
2.4.3.5	:DLOSS .....	2-28
2.4.3.5.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-28
2.4.3.5.1.1	:POINts? .....	2-28
2.4.3.6	:PCoefficients<n> .....	2-28
2.4.3.6.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-28
2.4.3.6.1.1	:POINts? .....	2-28
2.4.3.7	:SPEEd .....	2-29

## 1999 SCPI Instrument Classes

2.4.3.7.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-29
2.4.3.7.1.1	:POINts? .....	2-29
2.4.3.7.2	:START .....	2-29
2.4.3.7.2.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-29
2.4.3.7.3	:STOP .....	2-29
2.4.3.7.3.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-29
2.4.3.8	:TIME .....	2-29
2.4.3.8.1	:ACCEleration .....	2-30
2.4.3.8.1.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-30
2.4.3.8.2	:DECeleration .....	2-30
2.4.3.8.2.1	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-30
2.4.3.8.3	[:MAGNitude] <numeric_value>{,<numeric_value>} .....	2-30
2.4.3.8.3.1	:POINts? .....	2-30

## Chapter 3 Digital Meters

3.1	Base Functionality .....	3-3
3.1.1	Base Measurement Instructions .....	3-3
3.1.1.1	Effects of MEASure Query and CONFigure .....	3-3
3.1.1.2	The <expected_value> parameter .....	3-4
3.1.1.3	The <resolution> parameter .....	3-4
3.1.2	Base Device-oriented Functions .....	3-4
3.1.2.1	FORMat Subsystem .....	3-4
3.1.2.2	SENSe subsystem .....	3-4
3.1.2.3	TRIGger subsystem .....	3-5
3.1.3	Base Status Reporting .....	3-5
3.1.3.1	QUEstionable Status .....	3-5
3.1.3.2	OPERation Status .....	3-5
3.2	Additional functionality .....	3-6
3.2.1	Extended Trigger .....	3-6
3.2.1.1	ETRIGGER Measurement Instructions .....	3-6
3.2.1.2	ETRIGGER Device-oriented Functions .....	3-6
3.2.1.2.1	TRIGger Subsystem .....	3-6
3.2.1.3	ETRIGGER Status Reporting .....	3-6
3.2.2	Multiple Terminals .....	3-6
3.2.2.1	TERMINALS Measurement Instructions .....	3-6
3.2.2.2	TERMINALS Device-oriented Functions .....	3-6
3.2.2.2.1	ROUTE Subsystem .....	3-6
3.2.2.3	TERMINALS Status Reporting .....	3-6
3.2.3	Offset Compensation .....	3-6
3.2.3.1	OCOMPENSATED Measurement Instructions .....	3-7
3.2.3.2	OCOMPENSATED Device-oriented Functions .....	3-7
3.2.3.2.1	SENSe Subsystem .....	3-7
3.2.3.3	OCOMPENSATED Status Reporting .....	3-7
3.3	Various Meter Classes .....	3-8

## 1999 SCPI Instrument Classes

3.3.1	DC Voltmeter .....	3-8
3.3.1.1	Range .....	3-8
3.3.1.2	SENSe Commands .....	3-8
3.3.2	AC RMS Voltmeter .....	3-8
3.3.2.1	Range .....	3-8
3.3.2.2	SENSe Commands .....	3-8
3.3.3	Ohmmeter .....	3-8
3.3.3.1	SENSe Commands .....	3-8
3.3.3.2	4-wire Ohmmeter .....	3-8
3.3.3.3	SENSe Commands .....	3-8
3.3.4	DC Ammeter .....	3-8
3.3.4.1	Range .....	3-9
3.3.4.2	SENSe Commands .....	3-9
3.3.5	AC RMS Ammeter .....	3-9
3.3.5.1	Range .....	3-9
3.3.5.2	SENSe Commands .....	3-9
3.4	Programming Examples .....	3-10
3.4.1	Simple Measurement .....	3-10
3.4.2	Time Critical Measurement .....	3-10
3.4.3	Multiple Measurements .....	3-10

## Chapter 4 Digitizers

4.1	Base Functionality .....	4-2
4.1.1	Base Measurement Instruction .....	4-2
4.1.2	Base Device-oriented Functions .....	4-2
4.1.2.1	Input functions .....	4-2
4.1.2.2	SENSe amplitude functions .....	4-3
4.1.2.3	SENSE timebase functions .....	4-3
4.1.2.4	SENSe One-of-N Function Control .....	4-3
4.1.2.5	Formatting .....	4-4
4.1.2.6	Trigger functions .....	4-4
4.1.3	Base Status Reporting .....	4-4
4.1.3.1	QUEStionable .....	4-5
4.1.3.2	OPERation .....	4-5
4.2	Programming Example .....	4-5

## Chapter 5 Emissions Benches

5.1	Base Functionality .....	5-1
5.1.1	Base Measurement Instructions .....	5-1
5.1.2	Base Device-Oriented Functions .....	5-1
5.1.2.1	CALibration subsystem .....	5-2
5.1.2.2	CONTrol subsystem .....	5-2
5.1.2.3	DIAGnostic subsystem .....	5-2
5.1.2.4	INSTrument subsystem .....	5-3

## 1999 SCPI Instrument Classes

5.1.2.5	MEMory subsystem .....	5-3
5.1.2.6	ROUTe subsystem .....	5-4
5.1.2.7	SENSe subsystem .....	5-4
5.1.2.8	SYSTem subsystem .....	5-6
5.1.2.9	TRIGger subsystem .....	5-7
5.1.3	Base Status Reporting .....	5-7
5.1.3.1	OPERation Status .....	5-8
5.1.3.2	QUEstionable Status .....	5-9
5.2	Additional functionality .....	5-10
5.3	Examples .....	5-10
5.3.1	Zero/Span/Zero Procedure .....	5-10
5.3.2	Bag Read Procedure .....	5-11
5.3.3	Start of Diesel Test Procedure .....	5-12
5.4	CALibration Subsystem (Bench Commands) .....	5-15
5.4.1	:LINearize .....	5-15
5.4.1.1	:ACCept .....	5-15
5.4.2	:ACQuire .....	5-15
5.4.2.1	:AUTO ONCE .....	5-15
5.4.2.2	:CALCulate AUTO POLYnomial<n> SRATional<n> .....	5-16
5.4.2.3	:CURVe .....	5-16
5.4.2.3.1	[TYPE] POLYnomial<n>   SRATional<n>, <numeric_value>{,<numeric_value>} .....	5-16
5.4.2.3.2	:ZFORce <Boolean> .....	5-17
5.4.2.4	:VERify .....	5-17
5.4.2.4.1	:ACQuire .....	5-17
5.4.2.4.2	:TOlerance <numeric_value> .....	5-17
5.4.2.4.3	:TYPE .....	5-18
5.5	DIAgnostic Subsystem (Bench Commands) .....	5-18
5.5.1	:HUP .....	5-18
5.5.1.1	:ACQuire .....	5-18
5.5.1.2	:CALCulate .....	5-18
5.5.2	:LEAK .....	5-19
5.5.2.1	:ACQuire .....	5-19
5.5.2.2	:CALCulate .....	5-19
5.5.3	:NEFFiciency .....	5-19
5.5.3.1	:ACQuire .....	5-19
5.5.3.2	:CALCulate .....	5-19

## Chapter 6 Emission Test Cell

6.1	Base Functionality .....	6-1
6.1.1	Base Measurement Instructions .....	6-1
6.1.2	Base Device-oriented Functions .....	6-1
6.1.2.1	SYSTem subsystem .....	6-1
6.1.2.2	TRIGger subsystem .....	6-2

## **1999 SCPI Instrument Classes**

6.2	Additional Functionality .....	6-2
6.3	Programming Examples .....	6-2
6.3.1	Instrument Capability and Version Example .....	6-2
6.3.2	Command Channel Example .....	6-3
6.3.3	Two Channel Example .....	6-3
6.3.4	Returning Data using a Table Example .....	6-4
6.3.5	Time Example .....	6-4

## **Chapter 7 Power Supplies**

7.1	Base Functionality .....	7-2
7.1.1	Base Measurement Instructions .....	7-2
7.1.2	Base Device-oriented Functions .....	7-2
7.1.2.1	Outputs .....	7-2
7.1.2.2	SOURce subsystem .....	7-2
7.1.3	Base Status Reporting .....	7-3
7.1.3.1	QUEStionable .....	7-3
7.1.3.2	OPERation .....	7-3
7.2	Additional Functionality .....	7-4
7.2.1	Measurement capability .....	7-4
7.2.1.1	MEASURE Measurement Instructions .....	7-4
7.2.1.2	MEASURE Device-oriented Functions .....	7-4
7.2.1.3	MEASURE Status Reporting .....	7-4
7.2.2	Multiple supplies .....	7-4
7.2.2.1	MULTIPLE Measurement Instructions .....	7-4
7.2.2.2	MULTIPLE Device-oriented Functions .....	7-4
7.2.2.3	MULTIPLE Status Reporting .....	7-5
7.2.3	Triggering Capability .....	7-5
7.2.3.1	TRIGGER Measurement Instructions .....	7-5
7.2.3.2	TRIGGER Device-oriented Functions .....	7-5
7.2.3.3	TRIGGER Status Reporting .....	7-6
7.3	Programming Examples .....	7-7
7.3.1	Simple .....	7-7
7.3.2	Time Critical .....	7-7
7.3.3	Level Verification .....	7-7
7.3.4	Multiple Supplies .....	7-8

## **Chapter 8 RF & Microwave Sources**

8.1	Base Functionality .....	8-2
8.1.1	Base Measurement Instructions .....	8-2
8.1.2	Base Device-oriented Functions .....	8-2
8.1.2.1	SOURce subsystem .....	8-2
8.1.2.2	OUTPut Subsystem .....	8-3
8.1.2.3	UNIT Subsystem .....	8-3
8.1.3	Base Status Reporting .....	8-3

## 1999 SCPI Instrument Classes

8.1.3.1	QUEStionable Status .....	8-3
8.1.3.2	OPERation Status .....	8-3
8.2	Additional functionality .....	8-4
8.2.1	Amplitude Modulation .....	8-4
8.2.1.1	AM Measurement Instructions .....	8-4
8.2.1.2	AM Device Oriented Commands .....	8-4
8.2.1.3	AM Status Reporting .....	8-4
8.2.2	Frequency Modulation .....	8-4
8.2.2.1	FM Measurement Instructions .....	8-4
8.2.2.2	FM Device Oriented Commands .....	8-4
8.2.2.3	Status Reporting .....	8-5
8.2.3	Pulse Modulation .....	8-5
8.2.3.1	PULM Measurement Instructions .....	8-5
8.2.3.2	PULM Device Oriented Commands .....	8-5
8.2.3.3	PULM Status Reporting .....	8-5
8.2.4	Analog Frequency Sweeping .....	8-5
8.2.4.1	FASWEEP Measurement Instructions .....	8-5
8.2.4.2	FASWEEP Device Oriented Commands .....	8-5
8.2.4.3	FASWEEP Status Reporting .....	8-6
8.2.5	Stepped Frequency Sweeping .....	8-6
8.2.5.1	FSSWEEP Measurement Instructions .....	8-6
8.2.5.2	FSSWEEP Device Oriented Commands .....	8-6
8.2.5.3	FSSWEEP Status Reporting .....	8-7
8.2.6	Analog Power Sweeping .....	8-7
8.2.6.1	PASWEEP Measurement Instructions .....	8-7
8.2.6.2	PASWEEP Device Oriented Commands .....	8-7
8.2.6.3	PASWEEP Status Reporting .....	8-8
8.2.7	Stepped Power Sweeping .....	8-8
8.2.7.1	PSSWEEP Measurement Instructions .....	8-8
8.2.7.2	PSSWEEP Device Oriented Commands .....	8-8
8.2.7.3	PSSWEEP Status Reporting .....	8-9
8.2.8	Frequency List .....	8-9
8.2.8.1	FLIST Measurement Instructions .....	8-9
8.2.8.2	FLIST Device Oriented Commands .....	8-9
8.2.8.3	FLIST Status Reporting .....	8-9
8.2.9	Marker Function .....	8-9
8.2.9.1	MARKER Measurement Instructions .....	8-9
8.2.9.2	MARKER Device Oriented Commands .....	8-9
8.2.9.3	MARKER Status Reporting .....	8-10
8.2.10	Trigger Function .....	8-10
8.2.10.1	TRIGGER Measurement Instructions .....	8-10
8.2.10.2	TRIGGER Device Oriented Commands .....	8-10
8.2.10.3	TRIGGER Status Reporting .....	8-11
8.2.11	Reference Oscillator .....	8-11
8.2.11.1	REFERENCE Measurement Instructions .....	8-11

## 1999 SCPI Instrument Classes

8.2.11.2	REFERENCE Device Oriented Commands .....	8-11
8.2.11.3	REFERENCE Status Reporting .....	8-11
8.3	Programming Examples .....	8-12
8.3.1	Simple .....	8-12
8.3.2	Modulation .....	8-12
8.3.3	Analog Sweep .....	8-12
8.3.4	Triggered Analog Sweep .....	8-13
8.3.5	Sweep with Marker .....	8-13
8.3.6	Reference oscillator .....	8-13

## Chapter 9 Signal Switchers

9.1	Base Functionality .....	9-2
9.1.1	Base Measurement Instructions .....	9-2
9.1.2	Base Device-oriented Functions .....	9-2
9.1.2.1	ROUTE Subsystem .....	9-2
9.1.3	Base Status Reporting .....	9-2
9.2	Additional Functionality .....	9-3
9.2.1	Scanning .....	9-3
9.2.1.1	SCAN Measurement Instructions .....	9-3
9.2.1.2	SCAN Device-oriented Functions .....	9-3
9.2.1.2.1	ROUTE Subsystem .....	9-3
9.2.1.2.2	TRIGger Subsystem .....	9-3
9.2.1.3	SCAN Status Reporting .....	9-4
9.2.2	Extended Trigger .....	9-4
9.2.2.1	ETRIGGER Measurement Instructions .....	9-4
9.2.2.2	ETRIGGER Device-oriented Functions .....	9-4
9.2.2.2.1	TRIGger Subsystem .....	9-4
9.2.2.3	ETRIGGER Status Reporting .....	9-4
9.3	Programming Examples .....	9-5
9.3.1	Making and Breaking Connections .....	9-5
9.3.2	Programmed Connection Sequences .....	9-5

# 1 Introduction

This volume defines the SCPI commands and behavior needed to implement functionality sets associated with common classes of instruments. Each chapter covers a particular instrument class. A class could be, for example, a power supply, voltmeter, or switcher. Each chapter stands by itself so a designer may focus on only the chapters that apply to a particular design.

## 1.1 Scope

This volume covers the major classes of electronic instruments supported by the commands and behaviors as defined in SCPI Volumes 1 through 3. This volume does not extend SCPI with unsupported or contradictory commands or behaviors.

As new applications for SCPI are explored, additional chapters will be added.

## 1.2 Definition of Terms

Functionality - an aspect of instrument operation which is described by SCPI commands and their associated behavior.

- **Functionality set**  
A grouping of functionality which shall be implemented as a whole.
- **Instrument class**  
A generally recognized type of instrument, for example switcher, power supply, or voltmeter.
- **Base functionality**  
That functionality set required in an instrument of a given class.
- **Additional functionality**  
One or more functionality sets defined for an instrument class beyond the base functionality.

## 1.3 Purpose

This volume is designed to further the overall SCPI goal of reducing ATE program development time by:

- Reducing the ATE product development time by guiding designers in the usage of SCPI from the more familiar, instrument class point of view.
- Achieving a higher degree of consistency among implementations of the same instrument class.

### 1.3.1 Guiding Designers

The Command Reference Volume describes commands that cover a wide variety of instrument classes and technologies. Picking commands from this alphabetically sorted list appropriate for a particular instrument can be daunting.

## 1999 SCPI Instrument Classes

This volume uses terms familiar to the designer to help relate SCPI to previous design experiences. Volume 4 provides a translation from the SCPI Command Reference vocabulary to the vocabulary common to an instrument class.

1

The Command Reference purposely presents certain subsystems in broad terms. For example, triggering is a function used in almost all instruments, but in slightly different ways. This volume provides further guidance to designers on how to apply certain subsystems, like TRIGger, to a particular instrument class.

### 1.3.2 Achieving Consistency

Because SCPI covers many instrument classes, it cannot require all instruments to implement all commands. It can, however, define an instrument base functionality by specifying a relatively small set of commands and behaviors. A programmer can depend on a level of consistency among SCPI instruments of the same class.

By staying within the base functionality, a programmer can avoid all Command Errors. Execution errors are still possible. For example, this volume cannot specify <numeric\_value> ranges. The programmer, however, can use the required MIN/MAX queries to establish range limits. Specific required values are shown where the parameter type is <CHARACTER PROGRAM DATA>.

Additional functionality may be defined for some instrument classes. None of these functionality sets are required, but if any set is implemented it shall be implemented in its entirety.

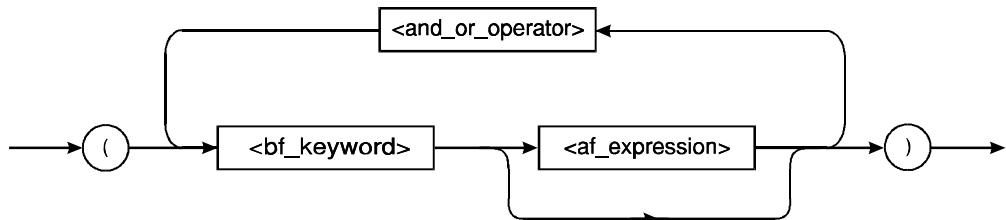
### 1.4 Instrument Classification

Each chapter defines base functionality for an instrument class. This base functionality is designated with a keyword using SCPI program mnemonic rules. Keywords are also defined for additional functionality. An ongoing goal of this volume is to further classify additional functionality for each instrument class.

An instrument is classified using one or more base functionality keywords along with optional additional functionality keywords. This combination of keywords is an <instrumentSpecifier>. Some instruments implement the SYSTem:CAPability? query, whose response is its <instrumentSpecifier>.

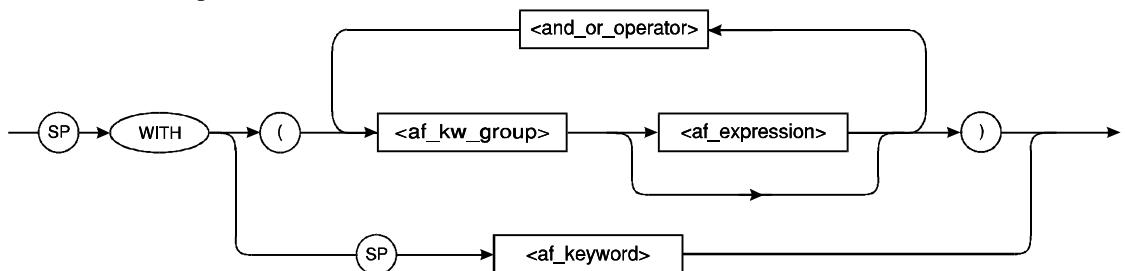
### 1.4.1 Syntax

The syntax of <instrumentSpecifier> is defined as:



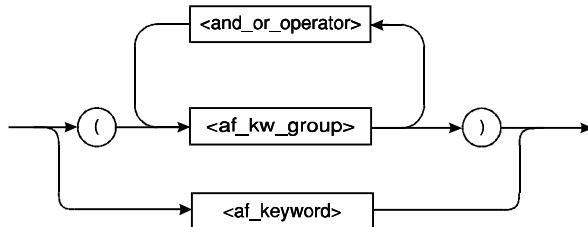
## 1999 SCPI Instrument Classes

<bf\_keyword> is a base functionality keyword as defined in one of the instrument chapters.  
<af\_expression> is defined as:



1

<af\_kw\_group> is defined as:



<af\_keyword> is an additional functionality keyword defined in the same instrument chapter which defines the associated <bf\_keyword>.

Both <bf\_keyword> and <af\_keyword> use the mnemonic generation rules described in Syntax & Style, 6.2.1 when constructing the keywords. The keywords have no short form, but may have a numeric suffix. Only <bf\_keyword> and <af\_keyword> elements defined in this volume may appear in the response to SYSTem:CAPability?.

The <and\_or\_oper> is either '&' (ASCII hexadecimal 26) or '|' (ASCII hexadecimal 7C). The '&' operator connects functionalities which may be active concurrently. The '|' operator indicates that a choice of functionality must be made.

The operators used in the preceding syntax definitions follow the rules of the following the precedence rules in Table 1-1.

## 1999 SCPI Instrument Classes

Operator	Associativity
()	left to right
WITH	right to left
&	left to right
	left to right

**Table 1-1  
Precedence and Order of Evaluation  
of <instrumentSpecifier> Operators**

1

### 1.4.2 Examples

The simplest <instrumentSpecifier> describes an instrument with just one base functionality. A power supply with no additional functionality might have an <instrumentSpecifier> of:

( PSUPPLY )

A simple digital multi-meter might contain several base functionalites, so its <instrumentSpecifier> could be:

( DCVOLTMETER | ACVOLTMETER | OHMMETER )

The operator, |, indicates that only one of the base functionalities is available at a time.

A more sophisticated DMM could have an <instrumentSpecifier> like:

( DCVOLTMETER WITH( Etrigger&TERMINALS ) | ACVOLTMETER  
WITH( Etrigger&TERMINALS ) | DCAMMETER  
WITH( Etrigger&TERMINALS ) | ACAMMETER WITH( Etrigger&TERMINALS )  
| OHMMETER WITH( Etrigger&TERMINALS&OCOMPENSATED ) | FOHMMETER  
WITH( Etrigger&TERMINALS&OCOMPENSATED ) )

Within the <af\_expression>, the operator, &, indicates all the additional functionalities are available simultaneously.

An instrument may contain multiple base functionalities that are used at the same time. The base functionalities may even be described in different chapters of this volume. A scanning voltmeter could be described with:

( DCVOLTMETER&SWITCHER WITH SCAN )

### 1.5 Compliance

An instrument that complies with SCPI even though it has less than base functionality for its instrument class is still a SCPI instrument, and may be claimed by its manufacturer as such. However, the manufacturer may not use <instrumentSpecifier> to describe its instrument unless it meets the requirements of Volume 4.

## 1999 SCPI Instrument Classes

1

Any product which uses an <instrumentSpecifier> to describes its capabilities and whose response to SYSTem:VERSion? is greater than or equal to 1994.0 shall also implement SYSTem:CAPability?

Note that every instrument will likely have functionality not covered by this volume. The designer must still follow the requirements in other volumes when implementing SCPI for those functionalities.

### 1.6 Chapter Organization

Chapters are nominally organized into sections as follows, where x is the chapter number:

x.1	Base Functionality
x.1.1	Base Measurement Instructions (as applies)
x.1.2	Base Device-oriented Functions
x.1.3	Base Status Reporting
x.2	Additional Functionality (as applies)
x.3	Programming Examples
x.4	Instrument Class specific commands for CALibration
x.5	Instrument Class specific commands for DIAGnostics
x.6	MEMory subsystem additions

Some instrument classes naturally divide into major sub-classes. For example, meters fall into sub-classes of voltmeter, ammeter, ohmmeter, and others. In this situation, separate chapters or other variations on chapter organization may be used.

In some instances, it is desired to standardize the way an Instrument or Instrument Class uses commands in the CALibration and/or DIAGnostic subsystem beyond the commands that are included in Volume II of SCPI for all instruments. In these cases, Instrument Class specific commands can be defined in the Instrument Classes volume for SCPI, but only for CALibration and DIAGnostic commands. When this is the case, the commands will be listed as required commands in the Base Functionality or Extended functionality sections of the Instrument Class chapter, and then the commands will be defined in x.4 Instrument Class specific commands for CALibration and/or x.5 Instrument Class specific commands for DIAGnostics sections of Instrument Class chapter to which they apply.

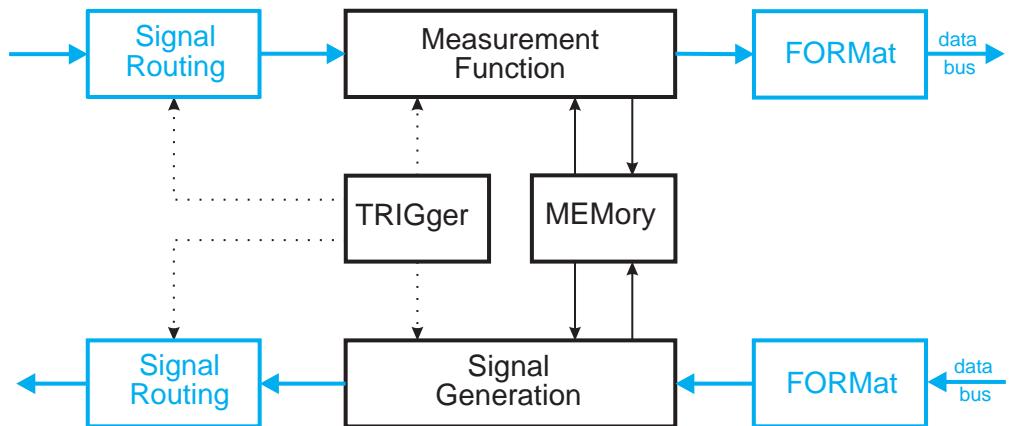
## **1999 SCPI Instrument Classes**

## 2 Chassis Dynamometers

The Chassis Dynamometer is a basic sourcing instrument. It supplies a force to a vehicle to simulate actual road load driving conditions. The SOURce root node is optional.

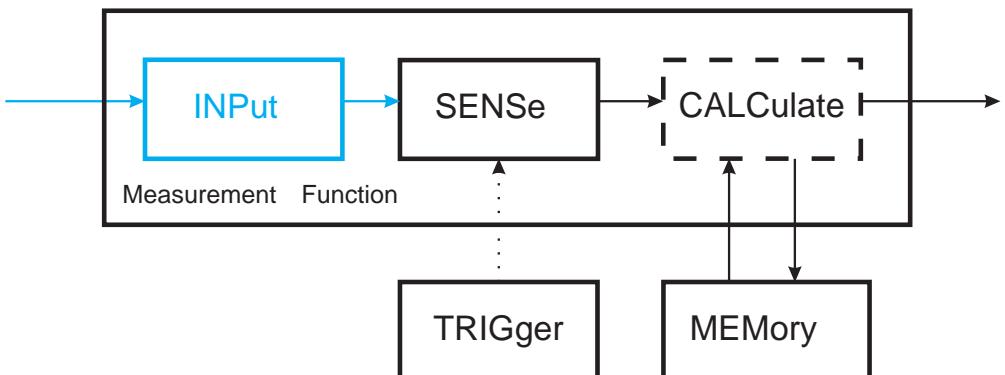
All SCPI compliant products implement the commands listed in Syntax & Style section 4.2.

Figure 2-1 shows a simplified model for a chassis dynamometer. It is derived from Command Reference, Chapter 2. The shaded parts are not described here.



**Figure 2-1 Simplified Model for a Chassis Dynamometer**

Figure 2-2 shows how the “Measurement Function” block in Figure 2-1 is expanded in a chassis dynamometer to measure Force, Speed and Distance. The CALCulate block is dashed and its behavior is not described here.

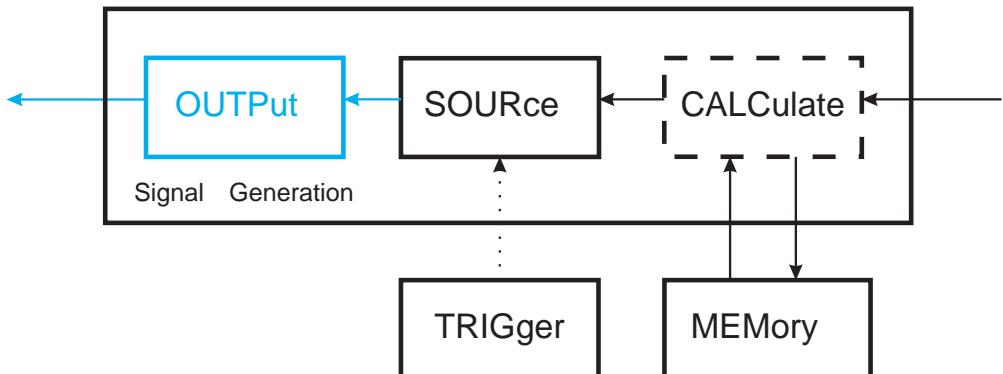


**Figure 2-2 Expanded Measurement Function Model of a Chassis Dynamometer**

## 1999 SCPI Instrument Class Applications

2

Figure 2-3 shows how the “Signal Generation” block in Figure 2-1 is expanded in a chassis dynamometer to source Force, Speed and other parameters to the chassis dynamometer.



**Figure 2-3 Expanded Signal Generation Model of a Chassis Dynamometer**

Figure 2-4 shows an instrument model for a Chassis Dynamometer. It indicates how the “Measurement Function” and “Signal Generation” blocks of the standard SCPI instrument model, Figure 2-1 of Command Reference, are modified for the dynamometer. Additionally, it shows how this measurement system uses system, status, calibration and control functions to set-up for performance of functional tasks. Calculations are part of the functional tasks of the dynamometer and thus are shown as part of the internal function block as are dynamometer specific tasks, e.g., coast downs, road load simulations/derivations and warm-ups. SCPI commands or responses to these chassis dynamometer internal functions are communicated through the I/O block shown.

The dynamometer has eleven possible states.

1. Idle
2. Warmup
3. Force Sensor Zero
4. Drive Line Loss
5. Coast Down
6. Parasitic Losses
7. Base Inertia
8. Go to Speed
9. Go to Force
10. Road Load Determination
11. Road Load Simulation.

### 2.1

### Base Functionality

This section describes the base functionality for Chassis Dynamometer.

The <bf\_keyword> for a Chassis Dynamometer is CDYNO. In addition to the functions shown below, all Chassis Dynamometers for use in emissions test cells shall implement the

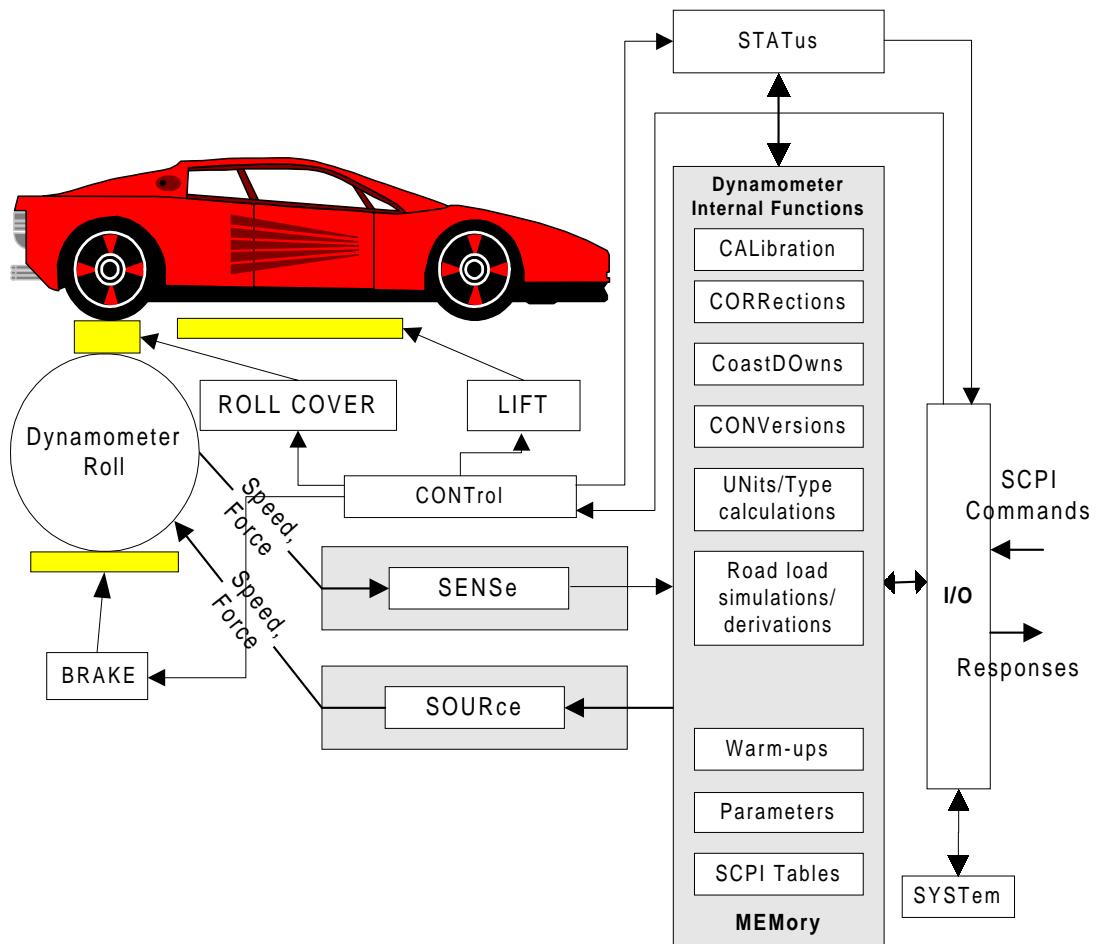


Figure 2-4 SCPI Chassis Dynamometer Model

functions shown in the base functionality for Emission Test Cells, <bf\_keyword> ETCELL, shown in Chapter 6 of SCPI Instrument Classes.

### 2.1.1 Base Measurement Instructions

The base functionality for a chassis dynamometer contains no MEASurement instructions, (commands from the MEASurement subsystem), but does include SENSe commands, which set up and cause measurements such as speed and force to be made.

### 2.1.2 Base Device-oriented Functions

The base functionality for a chassis dynamometer include :SOURce, :SENSe, :MEMory, CONTrol, and :CALibration subsystems. It also defines device dependent STATus bits and

## 1999 SCPI Instrument Class Applications

2

instructions for chassis dynamometers. A SCPI-compliant chassis dynamometer shall implement all of the commands listed in this section.

### 2.1.2.1 CALibration subsystem

The Chassis Dynamometer CALibration subsystem shall provide for calibrating or verifying various components that affect a chassis dynamometer measurement capability by implementing the following commands. The INITiate command causes the calibration function to be run. The CALibration subsystem commands for a chassis dynamometer are as follows.

KEYWORD	PARAMETER FORM	SCPI Reference
:CALibration		Vol 2-5
:BINertia		Vol 4-2.4.1.1
:AVERage?	<numeric_value>	Vol 4-2.4.1.1.1
:HSPEED	<numeric_value>	Vol 4-2.4.1.1.2
:INITiate		Vol 4-2.4.1.1.3
:LSPeed	<numeric_value>	Vol 4-2.4.1.1.4
:NRUNs	<numeric_value>	Vol 4-2.4.1.1.5
:SDEViation?	<numeric_value>	Vol 4-2.4.1.1.6
:UPDate		Vol 4-2.4.1.1.7
:PLOSS		Vol 4-2.4.1.2
:APCoefficients	<numeric_value>,<numeric_value>,...	Vol 4-2.4.1.2.1
:INITiate		Vol 4-2.4.1.2.2
:LATime	<numeric_value>	Vol 4-2.4.1.2.3
:STIMe	<numeric_value>	Vol 4-2.4.1.2.4
:UPDate		Vol 4-2.4.1.2.5
:WARMup		Vol 4-2.4.1.3
:INITiate		Vol 4-2.4.1.3.1
:SPEed	<numeric_value>	Vol 4-2.4.1.3.2
:TIMEout	<numeric_value>	Vol 4-2.4.1.3.3
:ZERO		Vol 4-2.4.1.4
:FSENsor		Vol 4-2.4.1.4.1
:INITiate		Vol 4-2.4.1.4.1.1
:LEVel?		Vol 4-2.4.1.4.1.3
:LATime	<numeric_value>	Vol 4-2.4.1.4.1.2
:SPEed	<numeric_value>	Vol 4-2.4.1.4.1.4
:STIMe	<numeric_value>	Vol 4-2.4.1.4.1.5
:UPDate		Vol 4-2.4.1.4.1.6

### 2.1.2.2 CONTrol subsystem

The CONTrol subsystem commands for a chassis dynamometer are as follows.

KEYWORD	PARAMETER FORM	SCPI Reference
:CONTrol		Vol 2-6
:BRAKE		Vol 2-6.3
[:STATE]	<Boolean>	Vol 2-6.3.1
:COVer		Vol 2-6.5
[:ADJust]	<OPEN CLOSE SCLOSE SOPEN>	Vol 2-6.5.1
[:POSIon?]		Vol 2-6.5.2
:IDLE		Vol 2-6.7
:INITiate		Vol 2-6.7.1
:LIFT		Vol 2-6.8
[:ADJust]	<UP DOWN>	Vol 2-6.8.1
[:POSIon?]		Vol 2-6.8.2
:MCONTrol		Vol 2-6.9
[:STATE]	<Boolean>	Vol 2-6.9.1
:ROTation		Vol 2-6.10
[:DIRECTION]	<FORWARD REVERSE>	Vol 2-6.10.1
:VCDevice		Vol 2-6.11
[:STATE]	<Boolean>	Vol 2-6.11.1
:TDIameter	<numeric_value>	Vol 2-6.11.2

### 2.1.2.3 MEMory subsystem

The MEMory subsystem of the chassis dynamometer shall include the following commands:

KEYWORD	PARAMETER FORM	SCPI Reference
:MEMory		Vol 2-13
:CLEar		Vol 2-13.2
[:NAME]	<name>	Vol 2-13.2.1
:TABLE	<name>	Vol 2-13.2.2
:DATA?	<data>	Vol 2-13.4
:DELETE		Vol 2-13.5
:ALL	<name>	Vol 2-13.5.1
[:NAME]	<name>	Vol 2-13.5.2
:TABLE		Vol 2-13.11
:ARATE		Vol 4-2.4.3.1
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.1.1
:POINTs?		Vol 4-2.4.3.1.1.1
:CINertia		Vol 4-2.4.3.2
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.2.1
:POINTs?		Vol 4-2.4.3.2.1.1
:DEFine	<structure_string>[,<numeric_value>]	Vol 2-13.11.11
:DRate		Vol 4-2.4.3.3
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.3.1
:POINTs?		Vol 4-2.4.3.3.1.1

## 1999 SCPI Instrument Class Applications

2

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
:FORCe		Vol 2-13.11.12
:ACCEleration		Vol 4-2.4.3.4.1
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.4.1.1
:POINts?		Vol 4-2.4.3.4.1.1.1
:ADECeleration		Vol 4-2.4.3.4.2
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.4.2.1
:POINts?		Vol 4-2.4.3.4.2.1.1
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 2-13.11.12.1
:POINts?		Vol 2-13.11.12.1.1
:ZOFFset		Vol 4-2.4.3.4.4
[:MAGNitude]	<numeric_value>	Vol 4-2.4.3.4.4.1
:DLOSSs		Vol 4-2.4.3.5
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.5.1
:POINts?		Vol 4-2.4.3.5.1.1
:PCOEfficients<n>		Vol 4-2.4.3.6
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.6.1
:POINts?		Vol 4-2.4.3.6.1.1
:SElect	<table_name>	Vol 2-13.11.22
:SPEed		Vol 2-13.11.23
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 2-13.11.23.1
:POINts?		Vol 2-13.11.23.1.1
:STARt		Vol 4-2.4.3.7.2
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.7.2.1
:POINts?		Vol 4-2.4.3.7.2.1.1
:STOP		Vol 4-2.4.3.7.3
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.7.3.1
:POINts?		Vol 4-2.4.3.7.3.1.1
:TIME		Vol 2-13.11.24
:ACCEleration		Vol 4-2.4.3.8.1
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.8.1.1
:POINts?		Vol 4-2.4.3.8.1.1.1
:DECeleration		Vol 4-2.4.3.8.2
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 4-2.4.3.8.2.1
:POINts?		Vol 4-2.4.3.8.2.1.1
[:MAGNitude]	<numeric_value> {,<numeric_value>}	Vol 2-13.11.24.1
:POINts?		Vol 2-13.11.24.1.1

## 1999 SCPI Instrument Class Applications

The Chassis Dynamometer shall have the following tables.

<b>Table Name</b>	<b>Elements</b>	<b>Description</b>
CDownSpeed	SPEed:STARt, SPEed:STOP	Coastdown speed interval definitions
CDownResult	TIME, FORCe, POWER	Results of the coastdown runs. Each coastdown has ‘n’ intervals as defined in the CDownSpeed table
CDownMCoefficients	PCO0, PCO1, PCO2, PCO3	Coastdown measured Polynomial Coefficients for a coastdown or road load derivation procedure
CDownDCoefficients	PCO0, PCO1, PCO2, PCO3	Coastdown calculated dynamometer setting coefficients during coastdown or road load derivation
ParasiticLOSSs	SPEed, DLOSSs	parasitic loss computation results
PCoefficients	PCO0, PCO1, PCO2, PCO3	Parasitic data curve fit results from the Parasitic Loss procedure
DriveLineLoss	SPEed, DLOSSs	Steady state drive line loss measurement
DriveLineCoefficients	PCO0, PCO1, PCO2, PCO3	Results of a curve fit of the DriveLineLoss table data, after the Drive Line Loss procedure
BaseINertia	CINertia, TIME:ACCEL, TIME:DECCEL, ARATE, DRATE, FORCe:AACCEL, FORCe:ADECEL	base inertia procedure data
FsensorZero	ZOFFset	Single value table whose value represents the zero offset used to compensate for shift. CALibration:ZERO:FSENsor:LEVel value will replace this when CAL:ZERO:FSENsor:UPDate is sent.

For definition of the new :MEMORY:TABLE Commands defined above, see Section 2.4.3 in this chapter of Volume 4, Instrument Classes.

### 2.1.2.4 SENSe subsystem

The SENSe subsystem of a chassis dynamometer shall provide access to function values by implementing the following commands. The <sensor\_function> options shall include:  
 FORCe|TPLoss|SPEEd[:REAR]|SPEEd:FRONT|FERRor|ACCeleration|DISTance||  
 DISTance:RESET|TIME

KEYWORD	PARAMETER FORM	SCPI Reference
:SENSe		Vol 2-18
:DATA?	<data_handle>	Vol 2-18.13.1
:DISTance		Vol 2-18.9
:RESET		Vol 2-18.9.1
:FUNCTION		Vol 2-18.13.2
:CONCurrent	<Boolean>	Vol 2-18.13.2.1
:OFF	<sensor_function>{,<sensor_function>}	Vol 2-18.13.2.2
	where the sensor functions to be supported shall include: FORCe TPLoss SPEEd[:REAR] SPEEd:FRONT	
	FERRor ACCeleration DISTance	
	DISTance:RESET TIME	
:ALL		Vol 2-18.13.2.2.1
:COUNt?		Vol 2-18.13.2.2.2
[:ON]	<sensor_function>{,<sensor_function>}	Vol 2-18.13.2.3
	where the sensor functions supported	
	are same as :OFF	
:ALL		Vol 2-18.13.2.3.1
:COUNt?		Vol 2-18.13.2.3.2
:STATE?	<sensor_function>	Vol 2-18.13.2.4

### 2.1.2.5 SOURce subsystem

The SOURce subsystem of the chassis dynamometer shall provide the following commands

KEYWORD	PARAMETER FORM	SCPI Reference
:SOURce		Vol 2-19
:ACCeleration		Vol 2-19.1
[:LEVel]	<numeric_value>	Vol 2-19.1.1
:FORCe		Vol 2-19.8
:CDOWn		Vol 2-19.8.1
:INITiate		Vol 2-19.8.1.1
:NRUNs	<numeric_value>	Vol 2-19.8.1.3
:RLDerivation		Vol 2-19.8.1.4
:FACCeptance	<numeric_value>	Vol 2-19.8.1.4.1
:INITiate		Vol 2-19.8.1.4.2
:RMAXimum	<numeric_value>	Vol 2-19.8.1.4.3
:RVERify	<numeric_value>	Vol 2-19.8.1.4.4
:SOFFset	<numeric_value>	Vol 2-19.8.1.2
:CONFIGure		Vol 2-19.8.2

## 1999 SCPI Instrument Class Applications

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
:ABRake		Vol 2-19.8.2.1
:GAIN	<numeric_value>	Vol 2-19.8.2.1.1
[:STATe]	<Boolean>	Vol 2-19.8.2.1.2
:THreshold	<numeric_value>	Vol 2-19.8.2.1.3
:GRADE		Vol 2-19.8.2.2
:LEVel	<numeric_value>	Vol 2-19.8.2.2.1
[:STATe]	<Boolean>	Vol 2-19.8.2.2.3
:SOURce	INTernal   EXTernal	Vol 2-19.8.2.2.2
[:VEHicle]		Vol 2-19.8.2.3
:DCOefficient	<numeric_value_list>,<numeric_value>	Vol 2-19.8.2.3.1
:DINertia	<numeric_value>	Vol 2-19.8.2.3.2
[:STATe]	<Boolean>	Vol 2-19.8.2.3.3
:TCOefficient	<numeric_value_list>,<numeric_value>	Vol 2-19.8.2.3.4
:TINertia	<numeric_value>	Vol 2-19.8.2.3.5
:WEIGHT	<numeric_value>	Vol 2-19.8.2.3.6
:INITiate		Vol 2-19.8.3
[:LEVel]	<numeric_value>	Vol 2-19.8.4
:RLSimulation		Vol 2-19.8.5
:INITiate		Vol 2-19.8.5.1
:SPEed		Vol 2-19.20
:INITiate		Vol 2-19.20.1
[:LEVel]	<numeric_value>	Vol 2-19.20.2
:SSDLoss		Vol 2-19.20.3
:INITiate		Vol 2-19.20.3.1
:LATime	<numeric_value>	Vol 2-19.20.3.2
:STIMe	<numeric_value>	Vol 2-19.20.3.3

## 2.1.3 STATus Subsystem

Chassis Dynamometer shall implement the status reporting structure described in SCPI Syntax & Style Volume 1, Chapter 9, Status Reporting. The SCPI Command Reference Volume 2, Chapter 20, STATus Subsystem defines the commands that shall be used to control the status reporting structure.

KEYWORD	PARAMETER FORM	SCPI Reference
:STATUs		Vol 2-20
:OPERation		Vol 2-20.1
:BIT<n>	<n=8 -12>	Vol 2-20.1.1
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:NTRansition	<NRf> <non-decimal numeric>	Vol 2-20.1.6
:PTRansition	<NRf> <non-decimal numeric>	Vol 2-20.1.7
:CONDITION?		Vol 2-20.1.2
:QUEstionable?		Vol 2-20.3
:BIT<n>	<n=9 -12>	Vol 2-20.3.1
:CONDITION?		Vol 2-20.3.2
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.3.3
[:EVENT]?		Vol 2-20.3.4
:NTRansition		Vol 2-20.3.6
:PTRansition		Vol 2-20.3.7
:CONDITION?		Vol 2-20.3.2
[:EVENT]		Vol 2-20.3.4

### 2.1.3.1 OPERation

:STATUs:OPERation

For a chassis dynamometer, the bit of interest in the OPERation status structure is the CDYNO summary bit. While the chassis dynamometer is performing its function, bit 9 (CDYNO summary) shall be used for the chassis dynamometer status. The fanned out register is shown in Figure 2-5.

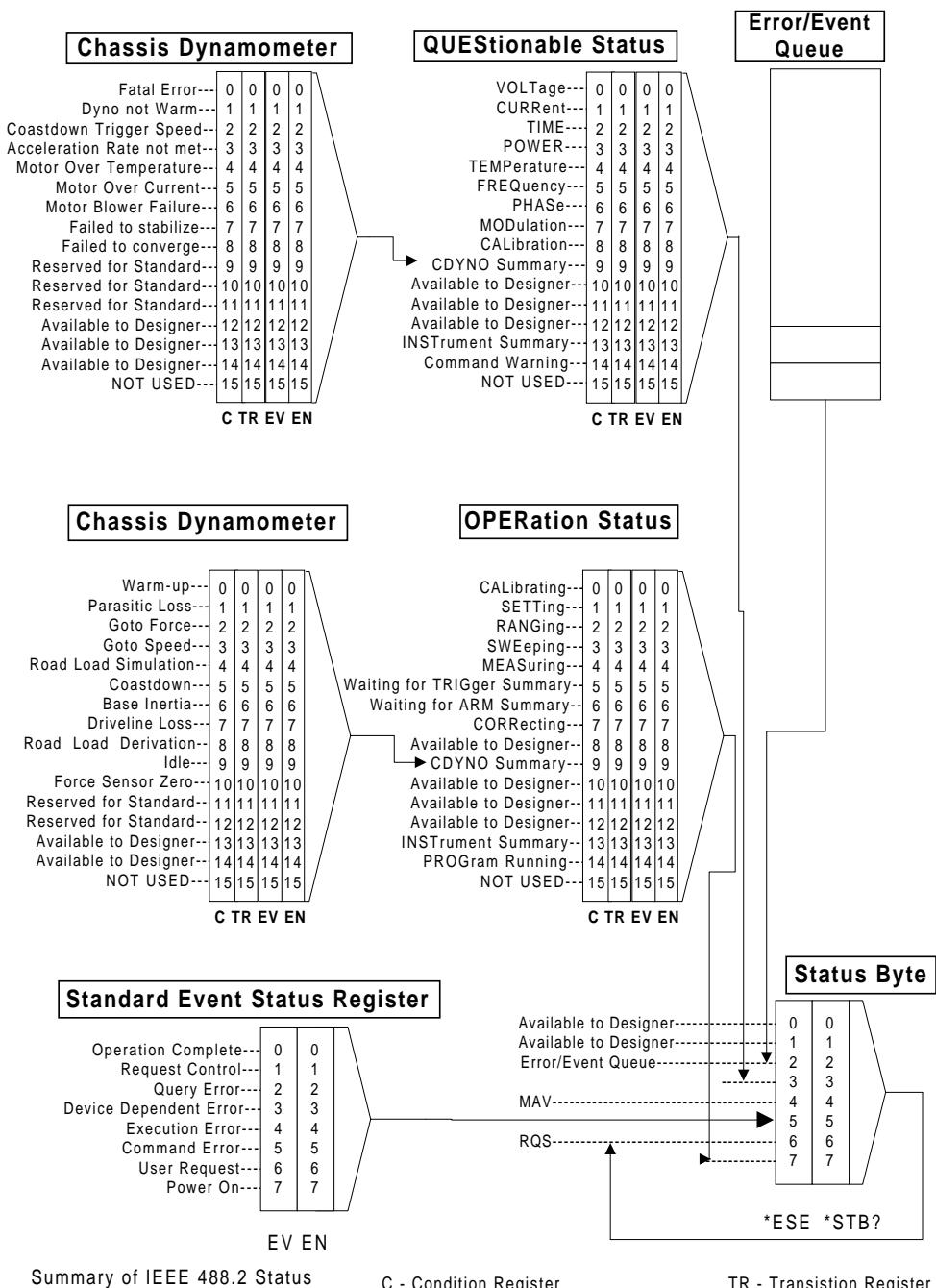
#### 2.1.3.1.1 CDYNo:CONDITION?<16 bit integer word>

:STATUs:OPERation:CDYNo:CONDITION?

The Dynamometer Program Status Register contains 16 bits that indicate what operation is being performed on the dynamometer. The most significant bit (bit 15) is always 0. The dynamometer will be in one of the following processes at all times. All processes are mutually exclusive and are overlapped, (i.e. with overlapped processes, a command may be sent while a process is executing) but will not be executed until that procedure is completed.

## 1999 SCPI Instrument Class Applications

2



**Figure 2-5 SCPI Status Registers**

## 1999 SCPI Instrument Class Applications

### CDYNO OPERation CONDITION Status Register

2

Bit #	Event	
0	Warm Up	Set if Dynamometer is in Warm Up Process
1	Parasitic Loss	Set if Dynamometer is in Parasitic Loss Process.
2	Go To Force	Set if Dynamometer is in Go To Force Process.
3	Go To Speed	Set if Dynamometer is in Go To Speed Process.
4	Road Load Simulation	Set if Dynamometer is in Road Load Simulation Process.
5	Coast Down	Set if Dynamometer is in Coast Down Process.
6	Base Inertia	Set if Dynamometer is in Base Inertia Process.
7	Driveline Loss	Set if Dynamometer is in Driveline Loss Process.
8	Road Load Derivation	Set if Dynamometer is in Road Load Derivation Process.
9	Idle	Set if Dynamometer is in Idle state.
10	Force Sensor Zero	Set if Dynamometer is in Force Sensor Zero state
11	Reserved for standard	
12	Reserved for standard	
13	Available to Designer	
14	Available to Designer	
15	Not Used	Will always read as 0

#### 2.1.3.2 **QUESTIONable**

:STATus:QUESTIONable

For the chassis dynamometer, the bit of interest in the QUESTIONable status structure is the CDYNO summary bit. When the chassis dynamometer suspects the dynamometer specification is violated, bit 9 (CDYNO summary) shall be set. The fanned out register is as follows:

##### 2.1.3.2.1 **CDYNo:CONDition?<16 bit integer word>**

:STATus:QUESTIONable:CDYNo:CONDition?

The Dynamometer Questionable Status Register contains 16 bits that indicate whether the data being acquired is of questionable quality. The most significant bit (bit 15) is always 0.

The dynamometer updates this register immediately when the dynamometer encounters an error that may generate questionable data. Bits 0-9 are the minimum subset of errors that may generate faulty data. Bits 10-14 are available for a dynamometer manufacturer for other questionable data or for other summary bits.

### CDYNO QUESTionable CONDition Status Register

Bit #	Questionable Event	
0	Fatal Error	Set when the dynamometer experiences a fatal error that causes the motor contactor to drop out, such as Torque Safety Limit, Speed Safety Limit or other dynamometer specific fatal error. An appropriate message must be placed on the Error/Event message queue when set.
1	Dynamometer not Warm	This bit is set if the dynamometer is not warm or has timed out after being warm but not used within the TIMEOUT period.
2	Coast down Trigger Speed	For a coast down, the dynamometer determines an acceptable trigger speed based on coast down inputs. If the dynamometer does not agree with the set trigger speed, this bit shall be set.
3	Acceleration Rate not met	The dynamometer will set this bit if the requested acceleration rate for a Process is out of tolerance with the actual acceleration. The tolerance is based on the physical characteristic of the dynamometer.
4	Motor Over Temperature	For dynamometers that monitor motor temperature
5	Motor Over Current	For dynamometers that monitor motor current
6	Motor Blower Failure	For dynamometers that monitor the blower
7	Failed To Stabilize	The dynamometer failed to stabilize
8	Failed to Converge	The dynamometer failed to converge.
9	Reserved for Standard	
10	Reserved for Standard	
11	Reserved for Standard	
12	Available to Designer	
13	Available to Designer	

## 1999 SCPI Instrument Class Applications

2

14	Available to Designer	
15	Not Used	Will always read as 0

### 2.2 Additional Functionality

Not applicable.

### 2.3 Programming Examples

#### 2.3.1 Coastdown

Note that this example does not start with \*RST or \*CLS. These commands would not be appropriate when carryover settings or status are important.

Brake off	:CONTrol:BRAKE OFF	
Contactor On -	:CONTrol:MCONtrol ON	
See if Dyno is warm	STATus:QUEStionable:CDYNo :CONDition?	Controller reads response, looks at bit 1
Let Dyno know if vehicle is on or off	:SOURce:FORCe:CONFigure :VEHicle ON	OFF or ON for vehicle
Define table of speed points for input into coastdown	:MEMory:TABLE:SElect CdownSpeed	This table represents the coastdown speed interval that data will be measured and collected.
	:MEMory:TABLE:DEFine “SPEed:STARt,SPEed:STOP”,5	For this example, 5 elements have been allocated (30 maximum).
	:MEMory:TABLE:SPEed :STARt 24.6,20.2,15.7,11.2,6.7	Meters/sec values corresponding to 55, 45, 35, 25, 15 MPH start, and 45, 35, 25, 15, 5 MPH stop speeds
	:MEMory:TABLE:SPEed :STOP 20.2,15.7,11.2,6.7,2.2	
Define the number of coastdown runs	:SOURce:FORce:CDOWn :NRUNs 5	Repeat coastdown 5 times
Define table for coastdown outputs	:MEMory:TABLE:SElect CDownResult	

## 1999 SCPI Instrument Class Applications

	:MEMORY:TABLE:DEFine “TIME, FORCe, POWER”, 6	Theoretical and measured values from coastdown. The table is NRUNs + 1 big.
Set Inertia	:SOURce:FORCe:CONFigure :VEHicle:DINERTia 1350	Dyno Inertia setting in kilograms
Set Road load coefficients	:SOURce:FORCe:CONFigure :VEHicle:DCoefficient 26.7, -0.98, 0.65	Numeric list of polynomial coefficients. These corresponds to 6, -0.1 and 0.3 in english units (pounds force (lbf) and miles per hour).
	:SOURce:FORCe:CONFigure :VEHicle:TCoefficient 26.7, -0.98, 0.65	Target settings
Set coastdown input parameters	:SOURce:ACCELERate 1	How fast to accelerate the dyno to top coastdown point (meters/sec^2)
	:SOURce:FORCe:CONFigure :VEHicle:TINertia 1350	Target inertia.
	:SOURce:FORCe:CDOWN :SOFFset 2.2	Speed above top set point (meters/sec)
Setup to sense data	:SENSe:FUNCTION:CONCurrent ON :SENSe:FUNCTION:ON “FORCe”, “SPEed”, “ACCELERation”, “DISTance”	Allow multiple sensor functions to be sensed. Turn on desired sensor functions.
Setup trigger system for sensing data	:TRIGger:SOURce TIMER :TRIGger:TIMER 0.100	Select internal timer on dynamometer Take samples every 100 ms, samples are averaged over 100 ms
Initiate Coastdown	:SOURce:FORCe:CDOWN :INITiate	Start coastdown (this command clears output)
Coastdown Complete?	:STATus:OPERation:CDYNo :CONDITION?	Wait until bit 5 is false indicating coastdown complete.
Checkout completion	:STATus:QUESTIONable :CONDITION?	If bit 9 is true then :STATus:QUESTIONable :CDYNo:CONDITION? can be used to check for specific errors.

## 1999 SCPI Instrument Class Applications

2

Iteratively review realtime data	:INITiate:CONTinuous ON:SENSe:DATA?	Start acquiring data Returns FORCe, SPEEd, ACCeleration, and DISTance.
Retrieve Coastdown results	:MEMory:DATa? CDownResult	Retrieve Theoretical and Measured data
	:MEMory:DATa? CDownMCoefficients	For definition of CDownMCoefficient see SAE J2264. This is a curve fit of the data
Idle the dynamometer	:CONTrol:IDLE:STATe ON	

### 2.3.2 Road Load Simulation

Note that this example does not start with \*RST or \*CLS. These commands would not be appropriate when carryover settings or status are important.

Brake off	:CONTrol:BRAKe OFF	
Contactor On	:CONTrol:MCONtrol ON	
See if dyno is warm	STATus:QUEStionable:DYNO :CONDition?	Controller reads response, looks at bit 1
Set Inertia	:SOURce:FORCe:CONFigure :VEHicle:DINERTia 1350	
Set Road load coefficients	:SOURce:FORCe:CONFigure :VEHicle:RLOad:DCoefficient 26.7, -0.98, 0.65	Numeric list of polynomial coefficients. These corresponds to 6, -0.1 and 0.3 in english units (pounds force (lbf) and miles per hour).
Set Vehicle Weight	:SOURce:FORCe:CONFigure :VEHicle:WEIGHT 1300	Vehicle weight in kilograms
Set the augmented braking	:SOURce:FORCe:CONFigure :ABRake ON	Augmented braking
Set the gain for augmented braking	:SOURce:FORCe:CONFigure :ABRake:GAIN 0.00	
Set the threshold for augmented braking	:SOURce:FORCe:CONFigure :ABRake:THRehold 500	Sets the threshold to 500 Newtons

## 1999 SCPI Instrument Class Applications

Set up the grade simulation	:SOURce:FORCe:CONFigure :GRADE ON		2
	:SOURce:FORCe:CONFigure :GRADE:LEVel 5	%in terms in grade - 100% is equal to sin(45deg)	
Setup to sense data	:SENSe:FUNCTION:CONCurrent ON	Allow multiple sensor functions to be sensed.	
	:SENSe:FUNCTION:ON “FORCe”, “SPEed”, “ACCeeration”, “DISTance”, “FERRor”	Turn on desired sensor functions.	
Setup trigger system for sensing data	:TRIGger:SOURce TIMER :TRIGger:TIMER 0.100	Select internal timer on dynamometer. Take samples every 100 ms, samples are averaged over 100 ms	
Initiate Road Load Simulation	:SOURce:FORCe:RLSimulate :INITiate	Reset the distance to zero.	
RLS Complete?	:STATus:OPERation:CDYNo :CONDition?	Make sure bit 4 is on indicating dyno is in simulation mode.	
Check for errors	:STATus:QUEstionable :CONDition?	If bit 9 is true then :STATus:QUEstionable:CDYNo:CONDition? can be used to check for specific errors.	
Iteratively review realtime data	:INITiate:CONTinuous ON :SENSe:DATA?	Start acquiring data. Returns FORCe, SPEed, ACCeleration, DISTance and FERRor.	
Brake on Brake off	:CONTrol:BRAKe ON :CONTrol:BRAKe OFF	Optional dynamic control of the mechanical brake. Used in conjunction with an automatic driver to hold zero speed during idle periods.	
Optionally reset the distance	:SENSe:DISTance:RESet		
Idle the Dynamometer	:CONTrol:IDLE:STATE ON	If idle is complete i.e., dyno at zero speed the Operation complete flag is true (same as above wrt. Checkout completion)	

## 2.4 New Commands for this Instrument Class

### 2.4.1 Dynamometer CALibration Subsystem

The Dynamometer CALibration subsystem contains the commands needed for calibrating or verifying various components that affect a dynamometers' measurement capabilities. This includes load sensor zero, parasitic loss determination, base inertia determination, and dynamometer warm-up.

KEYWORD	PARAMETER FORM	NOTES
:CALibration		
:BINertia		
:AVERage?	<numeric_value>	[query only]
:HSPEED	<numeric_value>	
:INITiate		[event; no query]
:LSPED	<numeric_value>	
:NRUNS	<numeric_value>	
:SDEVIATION?	<numeric_value>	[query only]
:UPDate		[event; no query]
:PLOSS		
:APCoefficients	<numeric_value>,<numeric_value>,...	
:INITiate		[event; no query]
:LATime	<numeric_value>	
:STIME	<numeric_value>	
:UPDate		[event; no query]
:WARMup		
:INITiate		[event; no query]
:SPEED	<numeric_value>	
:TIMEout	<numeric_value>	
:ZERO		
:FSENsor		
:INITiate		[event; no query]
:LATime	<numeric_value>	
:LEVEL?		[query only]
:SPEED	<numeric_value>	
:STIME	<numeric_value>	
:UPDate		[event; no query]

#### 2.4.1.1 :BINertia

CALibration:BINertia

Base Inertia

For chassis dynamometers, this node describes the procedure to determine the base mechanical inertia of the dynamometer roll system (all rotating mechanical components outside of the control loop). This procedure consists of an acceleration and deceleration of

the dynamometer rolls, from which a mechanical inertia calculation is made. The dynamometer is motored from an initial speed to a final speed at a given constant acceleration, and this interval is timed. Immediately, the dynamometer is motored back from the final speed to the initial speed at the same constant acceleration (inverted), and this interval is timed. For this pair of intervals, the force transducer output is continuously measured, and the average interval force calculated. The average acceleration is calculated using the interval initial and final speed values, and the interval time. This data is used to calculate the mechanical inertia (Inertia=Force/acceleration). The results are stored in the BaseINertia predefined table.

#### 2.4.1.1.1 :AVERage?

CALibration:BINertia:AVERage?

This query returns the running average of all base inertia values acquired since the last CALibration:BINertia:INITiate was invoked.

#### 2.4.1.1.2 :HSPEED <numeric\_value>

CALibration:BINertia:HSPEED

High Speed

The higher magnitude speed (m/s) of the speed interval.

At \*RST, this value is set to 16.

#### 2.4.1.1.3 :INITiate

CALibration:BINertia:INITiate

Begin motoring the dynamometer through the ACCeleration/deceleration pairs, using :SOURce:ACCEleration as the acceleration/deceleration rate, from LSPeed to HSPEED and back, NRUNs times. All query commands must be accepted and responded to while this command is executing. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

1. Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
2. Set the DYNO operation condition register Base Inertia Procedure bit indicating the Base Inertia is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
3. Accelerate/decelerate the dynamometer through the speed intervals, recording time, average acceleration, average force for the specified number of runs. The data is placed in the BaseInertia table after each acceleration/deceleration is run. Update the :CALibration:BINertia:SDEViation value based on this run. End the procedure when the specified runs are complete.
4. Clear the DYNO operation condition register Base Inertia Procedure bit indicating the Base Inertia is not executing.

## 1999 SCPI Instrument Class Applications

2

5. Issue the process complete message.
6. Execute the Idle Procedure.
7. Update the on-line mechanical inertia value, if desired.

### 2.4.1.1.4 :LSPeed <numeric\_value>

CALibration:BINertia:LSPeed

Low Speed

The lower magnitude speed (m/s) of the speed interval.

At \*RST, this value is set to 6.

### 2.4.1.1.5 :NRUNs <numeric\_value>

CALibration:BINertia:NRUNs

Number of Runs - NRUNs

The number of ACCeleration/deceleration pairs to perform. NRUNs will equal the number of base inertia values acquired.

At \*RST, this value is set to 10.

### 2.4.1.1.6 :SDEViation?

CALibration:BINertia:SDEViation?

Standard Deviation

This query returns the running standard deviation of all base inertia values acquired since the last CALibration:BINertia:INITiate was invoked.

### 2.4.1.1.7 :UPDate

CALibration:BINertia:UPDate

Place on-line the last determined average base mechanical inertia value. This command describes an event and therefore has no associated \*RST condition.

### 2.4.1.2 :PLOSS

:CALibration:PLOSS

Parasitic Loss

For chassis dynamometers this node describes the commands associated with the parasitic loss procedure. This procedure measures and, if requested, updates the parasitic loss corrections for the dynamometer. The parasitic losses are measured by operating the dynamometer at various selected speeds and measuring the force required to maintain the constant speed. This procedure uses the :MEMORY table 'ParasiticLOSS' that defines the speed points at which parasitic loss measurements are taken. The parasitic loss coefficients are calculated at the end of the procedure and reside in the memory table PCoefficient.

**2.4.1.2.1 :APCoeff <numeric\_value>,<numeric\_value>,<numeric\_value>...**

:CALibration:PLOSSs:APCoeff

Active Parasitic Coefficients - APCoefficients

This is the presently active parasitic loss curve used by the dynamometer. The dynamometer will maintain coefficients for forward and reverse operations. The appropriate values must be placed in the APCoeff memory, based on roll rotation. The Active Parasitic Coefficients set of retrieved will be based on the currently selected direction.

**2.4.1.2.2 :INITiate**

:CALibration:PLOSSs:INITiate

This command starts the parasitic loss procedure. This is an overlapped command. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

1. Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
2. Set the DYNO operation condition register Parasitic Loss Procedure bit indicating the Parasitic Loss procedure is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
3. Accelerate dynamometer to the first speed point defined in the ParasiticLOSs memory table. Stabilize at speed point for defined stabilization time.
4. Average the force at the roll surface required to maintain the speed set point over the latency time. The average force value is placed in the ParasiticLOSs table for the specified speed point. Continue until all speed points have been executed in the ParasiticLOSs table. The first zero speed in the ParasiticLOSs table ends the procedure.
5. Perform a curve fit on the ParasiticLOSs table data and place the coefficients in the memory table PCoeff.
6. Clear the DYNO operation condition register Parasitic Loss Procedure bit indicating the parasitic loss procedure is not executing.
7. Issue the process complete message.
8. Execute the Idle Procedure.

**2.4.1.2.3 :LATime**

:CALibration:PLOSSs:LATime

Loss Averaging Time - sec

The length of time in seconds to average the losses at a speed point.

At \*RST, this value is set to 5.

## 1999 SCPI Instrument Class Applications

### 2.4.1.2.4 :STIMe <numeric\_value>

CALibration:PLOSS:STIMe

Stabilization Time - sec

Time for stabilization prior to LATime.

At \*RST, this value is set to 5.

### 2.4.1.2.5 :UPDate

:CALibration:PLOSS:UPDate

Update the dynamometer actual current parasitic loss coefficients (APCoefficients node) with the coefficients contained in memory table PCoefficients.

This command is an event and has no associated \*RST condition.

### 2.4.1.3 :WARMup

CALibration:WARMup

This node controls parameters relating to the warmup of the device.

For chassis dynamometers, this node describes the procedure to set up the conditions required to ready the dynamometer for testing. The warm-up consists of running the dynamometer at a fixed speed over a period of time. Warm-up completion is defined by either running the dynamometer for a fixed period of time, or by monitoring parasitic losses and comparing them to a target value tolerance.

### 2.4.1.3.1 :INITiate

CALibration:WARMup:INITiate

Start the warmup procedure.

For chassis dynamometers, motor the dynamometer roll to SPEEd, and hold this speed for a TIMEout period of time. All query commands must be accepted and responded to while this command is executing. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

1. Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
2. Set the DYNO operation condition register Warm up Procedure bit indicating the Warm-up is executing. Note: The DYNO operation condition register will be further defined in Volume 4.
3. Accelerate dynamometer to the speed point defined by :CALibration:WARMup:SPEEd.
4. Average the force at the roll surface required to maintain the speed set point over :CALIbrate:WARMup:LATime. The average force can be monitored by :SENSe:FORCe?
5. The procedure will end when the elapsed time is greater than :CALibration:WARMup:TIMEout.

6. Clear the DYNO operation condition register Warm up Procedure bit indicating the Warm-up is not executing.
7. Issue the process complete message.
8. Execute the Idle Procedure.

### 2.4.1.3.2 :SPEed <numeric\_value>

CALibration:WARMup:SPEed

For chassis dynamometers this is the fixed roll speed (m/s) at which the warm-up will be run.

At \*RST, this value is set to 22.

### 2.4.1.3.3 :TIMEout <numeric\_value>

CALibration:WARMup:TIMEout

For chassis dynamometers this is the length of time in seconds the warm-up will run before automatically returning to zero speed.

At \*RST, this value is device dependent.

### 2.4.1.4 :ZERO

:CALibration:ZERO

For chassis dynamometers, this node describes the procedure to set up the conditions required and perform a mathematical zero correction of the roll force measurement system calibration. The correction is determined from a measurement of the roll force sensor at a given roll speed. The results are stored in the FSensorZero pre-defined table.

### 2.4.1.4.1 :FSENsor

CALibration:ZERO:FSENsor

Force Sensor

For chassis dynamometers this node sets up the conditions required and performs a mathematical zero correction of the roll force measurement system calibration. The correction is determined from a measurement of the roll force sensor at a given roll speed.

#### 2.4.1.4.1.1 :INITiate

CALibration:ZERO:FSENsor:INITiate

This is an overlapped command. Perform the measurement of the roll force, acquiring for the length of time specified by :LATime. This command describes an event and therefore has no associated \*RST condition.

When this command is initiated for dynamometers, the dynamometer must:

1. Check critical ancillaries, if any, for proper conditions: Brakes Off, Motor Contactor On, parameters within dynamometer safety limits. For conditions that prohibit operation of this procedure, issue appropriate warning messages, and execute the Idle Procedure.
2. Set the DYNO operation condition register Force Sensor Zero Procedure bit indicating the Zero is executing. Note: The DYNO operation condition register will be further defined in Volume 4.

## 1999 SCPI Instrument Class Applications

2

3. Accelerate dynamometer to the speed specified in :CALibration:ZERO:FSENsor:SPEEd.  
Stabilize for :STIMe.
4. Average the force at the roll surface required to maintain the speed set point over the :LATime. The average force value is placed in :LEVel.
5. Clear the DYNO operation condition register Force Sensor Zero Procedure bit indicating the Zero is not executing.
6. Issue the process complete message.
7. Execute the Idle Procedure.

### 2.4.1.4.1.2 :LATime

CALibration:ZERO:FSENsor:LATime

Loss Averaging Time - sec

The length of time to average the measured zero reading.

At \*RST, this value is set to 0.5.

### 2.4.1.4.1.3 :LEVel?

CALibration:ZERO:FSENsor:LEVel?

Returns the measured force value used to determine the mathematical zero correction. This value is in units of force at the roll surface. This is query only.

### 2.4.1.4.1.4 :SPEed <numeric\_value>

CALibration:ZERO:FSENsor:SPEed

The roll speed set point for the force measurement period in m/s.

At \*RST, this value is set to 0.

### 2.4.1.4.1.5 :STIMe <numeric\_value>

CALibration:ZERO:FSENsor:STIMe

Stabilization Time - sec

Time for stabilization prior to LATime.

At \*RST, this value is set to 5.

### 2.4.1.4.1.6 :UPDate

CALibration:ZERO:FSENsor:UPDate

Once the mathematical zero correction has been determined, apply the results to the on-line calibration. This is not a queryable command and therefore has no \*RST associated with it.

## 2.4.2 DIAGnostic Subsystem

There are no Instrument Class specific commands for DIAGnostics.

## 2.4.3 MEMory Subsystem

The MEMory subsystem of the chassis dynamometer shall include the following new commands:

## 1999 SCPI Instrument Class Applications

KEYWORD	PARAMETER FORM	NOTES
:MEMORY		
:TABLE		
:ARATE		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:CINERTIA		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:DEFINE	<structure_string>[,<numeric_value>]	
:DRATE		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:FORCE		
:ACCeleration		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:DECeleration		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:MAGNITUDE	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:ZOFFSET		
[:MAGNITUDE]	<numeric_value>	
:DLOSs		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:PCoefficients<n>		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:SELECT	<table_name>	
:SPEED		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:STARt	<numeric_value>{,<numeric_value>...}	
:POINTs?		[query only]
:STOP	<numeric_value>{,<numeric_value>...}	
:POINTs?		[query only]
:TIME		
:ACCeleration		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]
:DECeleration		
[:MAGNITUDE]	<numeric_value> {,<numeric_value>}	
:POINTs?		[query only]

## 1999 SCPI Instrument Class Applications

2

KEYWORD	PARAMETER FORM	NOTES
[:MAGNitude]       :POINts?	<numeric_value> {,<numeric_value>}	[query only]

New commands for the MEMory:TABLE subsystem are defined below.

### 2.4.3.1 :ARATe

MEMory:TABLE:ARATE

Specifies the AccelerationRATE points of the TABLE.

#### 2.4.3.1.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMory:TABLE:ARATE:MAGNitude

Specifies the AccelerationRATE[:MAGNitude] points of the TABLE. The default units of this command are meters per second per second.

At \*RST, the contents of this list item is device dependent.

#### 2.4.3.1.1.1 :POINts?

MEMory:TABLE:ARATE:MAGNitude:POINts?

Returns the number of points in the ARATE[:MAGNitude] list of the selected TABLE. If no ARATE[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

### 2.4.3.2 :CINertia

MEMory:TABLE:CINertia

Specifies the CalculatedINertia points of the TABLE.

#### 2.4.3.2.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMory:TABLE:CINertia:MAGNitude

Specifies the CalculatedINertia[:MAGNitude] points of the TABLE. The default units of this command are kilograms.

At \*RST, the contents of this list item is device dependent.

#### 2.4.3.2.1.1 :POINts?

MEMory:TABLE:CINertia:MAGNitude:POINts?

Returns the number of points in the CINertia[:MAGNitude] list of the selected TABLE. If no CINertia[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

### 2.4.3.3 :DRATe

MEMory:TABLE:DRATE

Specifies the DecelerationRATE points of the TABLE.

#### 2.4.3.3.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMory:TABLE:DRATE:MAGNitude

Specifies the DecelerationRATE[:MAGNitude] points of the TABLE. The default units of this command are meters per second per second.

At \*RST, the contents of this list item is device dependent.

**2.4.3.3.1.1 :POINts?**

MEMORY:TABLE:DRATe:MAGNitude:POINts?

Returns the number of points in the DRATe[:MAGNitude] list of the selected TABLE. If no DRATe[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**2.4.3.4 :FORCe**

MEMORY:TABLE:FORCe

See Volume 2, Section 13.11.12

**2.4.3.4.1 :AACCeleration**

MEMORY:TABLE:FORCe:AACCeleration

Specifies the AverageACCeleration points of the TABLE.

**2.4.3.4.1.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:FORCe:AACCeleration:MAGNitude

Specifies the AverageACCeleration[:MAGNitude] points of the TABLE. The default units of this command are meters per second per second.

At \*RST, the contents of this list item is device dependent.

**2.4.3.4.1.1.1 :POINts?**

MEMORY:TABLE:FORCe:AACCeleration:MAGNitude:POINts?

Returns the number of points in the AACCELERATION[:MAGNitude] list of the selected TABLE. If no AACCELERATION[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**2.4.3.4.2 :ADECeleration**

MEMORY:TABLE:FORCe:ADECeleration

Specifies the AverageDECeleration points of the TABLE.

**2.4.3.4.2.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:FORCe:ADECeleration:MAGNitude

Specifies the AverageDECeleration[:MAGNitude] points of the TABLE. The default units of this command are meters per second per second.

At \*RST, the contents of this list item is device dependent.

**2.4.3.4.2.1.1 :POINts?**

MEMORY:TABLE:FORCe:ADECeleration:MAGNitude:POINts?

Returns the number of points in the ADECeleration[:MAGNitude] list of the selected TABLE. If no ADECeleration[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**2.4.3.4.3 [:MAGNitude]<numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:FORCe:MAGNitude

See Volume 2, Section 13.11.12.1

### 2.4.3.4.3.1 :POINts?

MEMory:TABLE:FORCe:MAGNitude:POINts?

See Volume 2, Section 13.11.12.1.1

### 2.4.3.4.4 ZOFFset

MEMory:TABLE:FORCe:ZOFFset

Defines the ZeroOFFset value.

### 2.4.3.4.4.1 [:MAGNitude]

MEMory:TABLE:FORCe:ZOFFset:MAGNitude

Defines the ZeroOFFset value. For chassis dynamometers, this is a single value. This is a force, and the default units are newtons.

At \*RST, the contents of this list item is device dependent.

### 2.4.3.5 :DLOSS

MEMory:TABLE:DLOSSs

Specifies the DLOSS points of the TABLE.

### 2.4.3.5.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMory:TABLE:DLOSSs:MAGNitude

Specifies the DLOSSs[:MAGNitude] points of the TABLE. The default units of this command are newtons.

At \*RST, the contents of this list item is device dependent.

### 2.4.3.5.1.1 :POINts?

MEMory:TABLE:DLOSSs:MAGNitude:POINts?

Returns the number of points in the DLOSSs[:MAGNitude] list of the selected TABLE. If no DLOSSs[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

### 2.4.3.6 :PCoefficients<n>

MEMory:TABLE:PCoefficients

Specifies the PolynomialCOefficients points of the TABLE.

### 2.4.3.6.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMory:TABLE:PCoefficients:MAGNitude

Specifies the PolynomialCOefficients[:MAGNitude] points of the TABLE. The default units of this command are as follows: Units of the polynomial: N, N/(m/s), N/m/s)^2, N/(m/s)^3, ...

At \*RST, the contents of this list item is device dependent.

### 2.4.3.6.1.1 :POINts?

MEMory:TABLE:PCoefficients:MAGNitude:POINts?

Returns the number of points in the PCoefficients[:MAGNitude] list of the selected TABLE. If no PCoefficients[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**2.4.3.7 :SPEed**

MEMORY:TABLE:SPEed

See Volume 2, Section 13.11.23

**2.4.3.7.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:SPEed:MAGNitude

See Volume 2, Section 13.11.23.1

**2.4.3.7.1.1 :POINts?**

MEMORY:TABLE:SPEed:MAGNitude:POINts?

See Volume 2, Section 13.11.23.1.1

**2.4.3.7.2 :STARt**

MEMORY:TABLE:SPEed:STARt

For chassis dynamometers, this specifies the start speed used in Coastdown

**2.4.3.7.2.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:SPEed:STARt:MAGNitude

Specifies the STARt[:MAGNitude] points of the TABLE. The default units of this start speed command are meters per second.

At \*RST, the contents of this list item is device dependent.

**2.4.3.7.2.1.1 :POINts?**

MEMORY:TABLE:SPEed:STARt:MAGNitude:POINts?

Returns the number of points in the STARt[:MAGNitude] list of the selected TABLE. If no STARt[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**2.4.3.7.3 :STOP**

MEMORY:TABLE:SPEed:STOP

For chassis dynamometers, this specifies the stop speed used in Coastdown

**2.4.3.7.3.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}**

MEMORY:TABLE:SPEed:STOP:MAGNitude

Specifies the STOP[:MAGNitude] points of the TABLE. The default units of this start speed command are meters per second.

At \*RST, the contents of this list item is device dependent.

**2.4.3.7.3.1.1 :POINts?**

MEMORY:TABLE:SPEed:STOP:MAGNitude:POINts?

Returns the number of points in the STOP[:MAGNitude] list of the selected TABLE. If no STOP[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

**2.4.3.8 :TIME**

MEMORY:TABLE:TIME

See Volume 2, Section 13.11.24

### 2.4.3.8.1 :ACAcceleration

MEMORY:TABLE:TIME:ACAcceleration

2

For chassis dynamometers, this command defines the elapsed time over an acceleration interval.

#### 2.4.3.8.1.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMORY:TABLE:TIME:ACAcceleration:MAGNitude

Specifies the TIME:ACAcceleration[:MAGNitude] points of the TABLE. The default units of this command are seconds.

At \*RST, the contents of this list item is device dependent.

### 2.4.3.8.1.1.1 :POINts?

MEMORY:TABLE:TIME:ACAcceleration:MAGNitude:POINts?

Returns the number of points in the ACAcceleration[:MAGNitude] list of the selected TABLE. If no ACAcceleration[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

### 2.4.3.8.2 :DECeleration

MEMORY:TABLE:TIME:DECeleration

For chassis dynamometers, this command defines the elapsed time over a deceleration interval.

#### 2.4.3.8.2.1 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMORY:TABLE:TIME:DECeleration:MAGNitude

Specifies the TIME:DECeleration[:MAGNitude] points of the TABLE. The default units of this command are seconds.

At \*RST, the contents of this list item is device dependent.

### 2.4.3.8.2.1.1 :POINts?

MEMORY:TABLE:TIME:DECeleration:MAGNitude:POINts?

Returns the number of points in the DECeleration[:MAGNitude] list of the selected TABLE. If no DECeleration[:MAGNitude] values have been sent POINts? returns 0. If no TABLE has been selected, POINts? returns NAN.

### 2.4.3.8.3 [:MAGNitude]<numeric\_value>{,<numeric\_value>}

MEMORY:TABLE:TIME:MAGNitude

See Volume 2, Section 13.11.24.1

### 2.4.3.8.3.1 :POINts?

MEMORY:TABLE:TIME:MAGNitude:POINts?

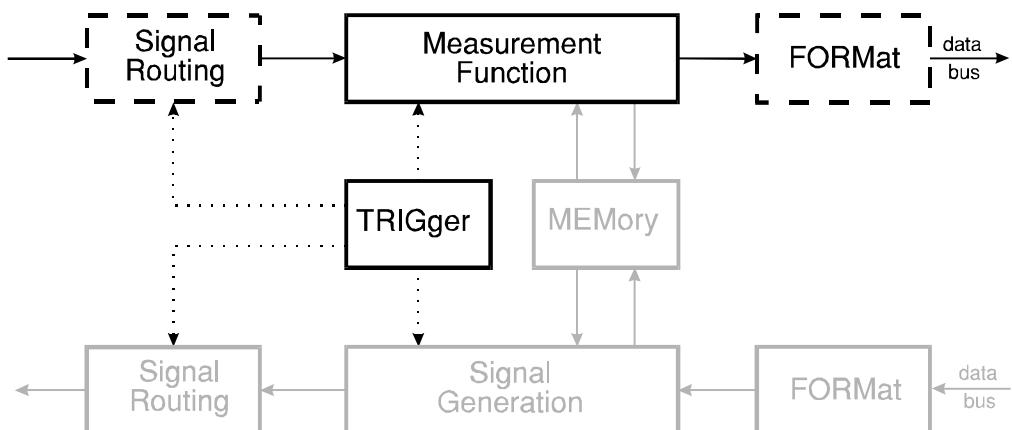
See Volume 2, Section 13.11.24.1.1

## 3 Digital Meters

A digital meter is a basic measuring instrument. It typically measures a single value at one point in time. This type of instrument is so general that it is first described generically. Attributes specific to a particular type of meter are described in their own subsections.

All SCPI compliant products implement the commands listed in Syntax & Style, 4.2.

Figure 3-1 shows an instrument model for a meter. It is derived from Command Reference, chapter 2. It is essentially the same as Figure 2-1 in Command Reference with various parts gray or dashed. The bold boxes and lines, solid and dashed, are described in this chapter. The FORMat and ROUTe blocks are shown with dashed lines because many meters use their default settings. \*RST sets FORMat[:READings][:DATA] to ASCII and ROUTe:TERMinals to FRONt. The gray boxes are not described here. While a meter may have MEMORY, its functionality is optional and thus not described.



**Figure 3-1 Simplified Model for a Programmable Meter**

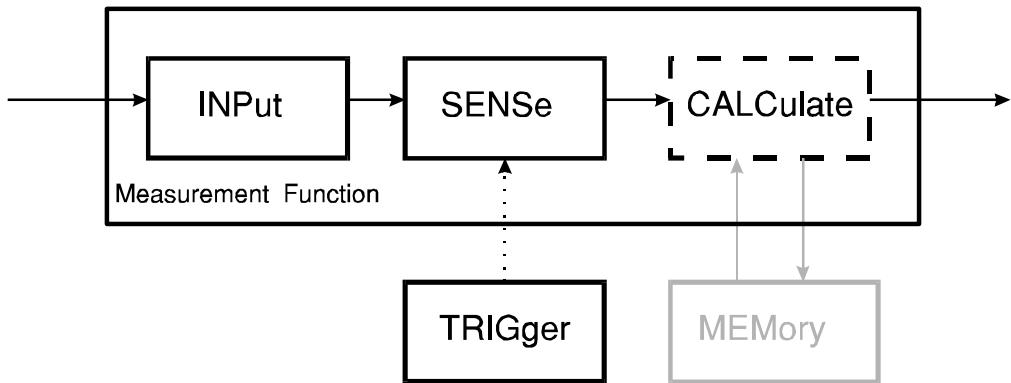
Figure 3-2 shows how the “Measurement Function” block in Figure 3-1 is expanded in a meter. It is the same as Figure 2-4 in Command Reference with some dashed and gray boxes. The CALCulate block is dashed because after \*RST this block has no effect on the data. This chapter does not describe any CALCulate functions. The MEMORY block is gray because while a meter may contain this function, this chapter does not address it.

A designer would never implement a generic meter, but would rather build one of the specific types. Table 1 lists the various classes of meters considered in this chapter along with the quantity they measure. A <meter\_fn> is associated with each class of meter. The contents of the <meter\_fn> shown in the table are used in the actual commands.

Some meters contain several instrument classes. A Digital Multi-Meter might contain a DC voltmeter, an AC voltmeter, a DC ammeter, an AC ammeter, an ohmmeter, and a 4-wire

## 1999 SCPI Instrument Classes

3



**Figure 3-2 Expanded Measurement Function Model**

ohmmeter. The SENSe:FUNCTION commands switch the instrument among its implemented classes. The TRIGger subsystem commands only appear once in the instrument and are shared by all the meters. Such a meter would not have TRIG1 for the DC voltmeter, TRIG2 for the AC voltmeter, etc.

Base Functionality Keyword <bf_keyword>	Common Usage Name	Measured Quantity	<meter_fn>
DCVOLTMETER	DC Voltmeter	DC Volts	VOLTage[:DC]
ACVOLTMETER	AC RMS Voltmeter	RMS Volts	VOLTage:AC
DCAMMETER	DC Ammeter	DC Amperes	CURREnt[:DC]
ACAMMETER	AC RMS Ammeter	RMS Amperes	CURREnt:AC
OHMMETER	Ohmmeter	Ohms	RESistance
FOHMMETER	4-wire Ohmmeter	Ohms	FRESistance

**TABLE 1. Instrument Classes for Meters**

## 3.1 Base Functionality

This section describes the functionality all meters implement.

### 3.1.1 Base Measurement Instructions

Since the basic function of a meter is to measure some quantity, it understands the measurement instructions in Command Reference, 3. A meter shall implement these commands with the appropriate substitution for <meter\_fn>.

KEYWORD	PARAMETERS
CONFigure	
[:SCALar]	
:<meter_fn>	[<expected_value>[,<resolution>]]
CONFigure?	
FETCh	
[:SCALar]	
[:<meter_fn>?]	[<expected_value>[,<resolution>]]
READ	
[:SCALar]	
[:<meter_fn>?]	[<expected_value>[,<resolution>]]
MEASure	
[:SCALar]	
:<meter_fn>?	[<expected_value>[,<resolution>]]

The syntax of these commands is found in chapter 3 of the Command Reference. Meters may accept parentheses and <source\_list> as described by the full syntax.

The response from \*RST;MEAS:<meter\_fn>? is a single <NR1>, <NR2>, or <NR3> because \*RST sets FORMat to ASCII, TRIGger:COUNT to 1, and ARM:COUNT to 1.

READ? and FETCh? shall return a <RESPONSE MESSAGE UNIT> containing TRIGger:COUNT times ARM:COUNT numbers corresponding to that many measurements. If ARM:COUNT is not implemented, its value is one. The timing of measurements is controlled by the other settings in the TRIGger subsystem.

The command sequence MEAS:<meter\_fn>?;FETCH? shall return two identical <RESPONSE MESSAGE UNIT> elements separated by a semicolon.

#### 3.1.1.1 Effects of MEASure Query and CONFigure

MEASure query and CONFigure shall set:

SENSe:FUNCTION[:ON]	to	"[XNONE:]<meter_fn>"
INITiate:CONTinuous	to	OFF
TRIGger[:SEQUence]:SOURce	to	IMMEDIATE
TRIGger[:SEQUence]:COUNt	to	1
TRIGger[:SEQUence]:DELay	to	MINimum

For meters that implement additional functionality ETRIGGER (see 3.2.1), MEASure query and CONFigure shall set:

## 1999 SCPI Instrument Classes

ARM[:SEQUence][:LAYer]:SOURce to	IMMEDIATE
ARM[:SEQUence][:LAYer]:COUNt to	1

For meters that implement additional functionality TERMINALS (see 3.2.2), MEASure query and CONFigure shall set:

ROUTE:TERMinals	unaffected
-----------------	------------

3

Note: Additional instrument settings may need to be changed to support the execution of the high-level measurement instruction. These settings are under the control of commands not described in this chapter and thus are not defined here.

### 3.1.1.2 The <expected\_value> parameter

If <expected\_value> is present and its value is not DEFault, SENSe:<meter\_fn>:RANGE[:UPPer] shall be set to the legal value equal to or just larger than <expected\_value>. SENSe:<meter\_fn>:RANGE:AUTO shall be set OFF.

If <expected\_value> is not present or its value is DEFault, SENSe:<meter\_fn>:RANGE:AUTO shall be set ON and the instrument shall autorange to the best possible range for the measurement.

### 3.1.1.3 The <resolution> parameter

If <resolution> is present and its value is not DEFault, SENSe:<meter\_fn>:RESolution shall be set to the legal value equal to or just less than <resolution>.

If <resolution> is not present or its value is DEFault, SENSe:<meter\_fn>:RESolution shall be set to a legal value that gives a reasonably accurate result without taking excessively long.

## 3.1.2 Base Device-oriented Functions

### 3.1.2.1 FORMat Subsystem

This subsystem is not required. A meter shall implement the FORMat[:READings][:DATA] ASCII and FORMat[:READings]:DINTerchange OFF settings. Note: These are the \*RST values, so the commands are not required.

### 3.1.2.2 SENSe subsystem

Any meter shall sense a selected function and control the range and resolution of the sensor. The ability to autorange is required. The ability to turn on multiple <sensor\_functions> simultaneously is not required.

A meter shall implement these SENSe commands with the appropriate substitution for <meter\_fn>.

KEYWORD	PARAMETER FORM	SCPI Reference
SENSe		Vol 2-18
:FUNCTION[:ON]	"[XNONE:]<meter_fn>"	Vol 2-18.13.2
:<meter_fn>		
:RANGE		
[:UPPer]	<numeric_value>	
:AUTO	<Boolean>	
:RESolution	<numeric_value>	

### 3.1.2.3 TRIGger subsystem

The primary purpose of the TRIGger subsystem is to control when measurements are made. The default timing is generally to take a measurement as soon as possible.

A meter shall implement these TRIGger commands.

KEYWORD	PARAMETER FORM	SCPI Reference
INITiate		Vol 2-24.7
	[:IMMEDIATE]	Vol 2-24.7.2
	[::ALL]	Vol 2-24.7.2.1
ABORT		Vol 2-24.5
TRIGger		Vol 2-24.8
	[::SEQUENCE]	Vol 2-24.8.1
	:COUNT <numeric_value>	Vol 2-24.8.1.2
	:DELAY <numeric_value>	Vol 2-24.8.1.5
	:SOURCE {BUS   IMMEDIATE   EXTERNAL   ...}	Vol 2-24.8.1.17

Requiring BUS as a legal parameter to TRIGger:SOURce has the side effect of requiring DT1 capability in an IEEE 488.2 device. It also means the meter implements the \*TRG common command.

A VXIbus device with a trigger function is required to implement the *Trigger* command. When TRIGger:SOURce is set BUS, a VXIbus meter shall use the *Trigger* command as the TRIGger source.

### 3.1.3 Base Status Reporting

Syntax & Style, chapter 9 describes the status reporting structure used in all SCPI instruments. Command Reference, chapter 20 defines the commands which control the status reporting structure.

#### 3.1.3.1 QUESTIONable Status

When a meter is measuring DC volts or AC volts, the meter shall set bit 0, VOLTage, in the QUESTIONable status structure when the meter suspects the voltage is of questionable accuracy. When a meter is measuring DC current or AC current, the meter shall set bit 1, CURRrent, in the QUESTIONable status structure when the meter suspects the current is of questionable accuracy. When a meter is measuring ohms, the meter shall set either bit 0, VOLTage, or bit 1, CURRrent when the meter suspects either is of questionable accuracy.

#### 3.1.3.2 OPERATION Status

While the meter is performing one of the operations listed in the OPERATION status structure, that bit shall be set in the condition register. Every meter shall report at least RANGing, bit 2, MEASuring, bit 4, and Waiting for TRIG, bit 5.

## 3.2 Additional functionality

### 3.2.1 Extended Trigger

The <af\_keyword> for extended trigger is ETRIGGER.

Meters with extended trigger functionality provide two layer triggering where a separate event must occur before the TRIGger layer is entered. The commands under ARM are used to provide this functionality rather than another SEQuence.

3

#### 3.2.1.1 ETRIGGER Measurement Instructions

ETRIGGER adds no Measurement Instructions.

#### 3.2.1.2 ETRIGGER Device-oriented Functions

##### 3.2.1.2.1 TRIGger Subsystem

KEYWORD	PARAMETER FORM	SCPI Reference
ARM		Vol 2-24.6
[:SEQUence]		Vol 2-24.6.1
[:LAYer]		Vol 2-24.6.1.2
:SOURce	{BUS   IMMEDIATE   ... }	Vol 2-24.6.1.2.14

#### 3.2.1.3 ETRIGGER Status Reporting

Bit 6 (Waiting for ARM) in the OPERATION Status Register is used to indicate the status of the trigger model.

### 3.2.2 Multiple Terminals

The <af\_keyword> for multiple terminals is TERMINALS.

A meter generally has only a single input, though some meters may be able switch the input between front and rear terminals. These meters have the TERMINALS additional functionality.

#### 3.2.2.1 TERMINALS Measurement Instructions

TERMINALS adds no Measurement Instructions.

#### 3.2.2.2 TERMINALS Device-oriented Functions

##### 3.2.2.2.1 ROUTe Subsystem

A meter which can select between front and rear terminals shall implement

KEYWORD	PARAMETER FORM	SCPI Reference
ROUTe		Vol 2-17
:TERMinals	FRONt   REAR	Vol 2-17.7

#### 3.2.2.3 TERMINALS Status Reporting

TERMINALS adds no Status Reporting requirements.

### 3.2.3 Offset Compensation

The <af\_keyword> for offset compensation is OCOMPENSATED.

An ohmmeter or 4-wire ohmmeter may be unable to measure resistance when residual voltages are present. An ohmmeter or 4-wire ohmmeter which can compensate for residual voltages when measuring resistance has the OCOMPENSATED additional functionality.

### 3.2.3.1 OCOMPENSATED Measurement Instructions

OCOMPENSATED adds no measurement instructions.

MEASure:RESistance? and CONFigure:RESistance shall set SENSe:RESistance:OCOMpensated to OFF. MEASure:FRESistance? and CONFigure:FRESistance shall set SENSe:FRESistance:OCOMpensated to OFF.

### 3.2.3.2 OCOMPENSATED Device-oriented Functions

#### 3.2.3.2.1 SENSe Subsystem

An ohmmeter or 4-wire ohmmeter which can compensate for residual voltages shall implement:

KEYWORD	PARAMETER FORM	SCPI Reference
SENSe		Vol 2-18
:RESistance		Vol 2-18.18
:FRESistance		
:OCOMpensated	<Boolean>	Vol 2-18.18.3

#### 3.2.3.3 OCOMPENSATED Status Reporting

OCOMPENSATED adds no Status Reporting requirements.

### 3.3 Various Meter Classes

This section provides additional information and rules of behavior for specific meter classes.

#### 3.3.1 DC Voltmeter

DC voltmeters measure the average voltage level of the signal across their inputs.

##### 3.3.1.1 Range

DC voltmeters shall measure negative as well as positive voltages symmetrically about zero. SENSe:RANGE[:UPPer] determines how far from zero measurements can be made. The MINimum and MAXimum values for SENSe:RANGE[:UPPer] shall be positive. Since the range is symmetric about zero, SENSe:RANGE:LOWER is not needed, but if implemented its value shall be coupled to UPPer by the equation:

$$\text{LOWER} = -\text{UPPer}.$$

##### 3.3.1.2 SENSe Commands

SENSe:VOLTage[:DC] commands are found in Command Reference, 18.20.

#### 3.3.2 AC RMS Voltmeter

AC RMS voltmeters measure the root mean square of the voltage level across their inputs.

##### 3.3.2.1 Range

Since the RMS of any signal is non-negative, the MINimum and MAXimum values for SENSe:RANGE[:UPPer] shall be positive. SENSe:RANGE:LOWER is not needed.

##### 3.3.2.2 SENSe Commands

SENSe:VOLTage:AC commands are found in Command Reference 1994, 18.21.

#### 3.3.3 Ohmmeter

Ohmmeters measure the resistance across their input terminals.

Typically, a DC current is sourced and the DC voltage is measured. Resistance is calculated by dividing the current into the voltage.

##### 3.3.3.1 SENSe Commands

SENSe:RESistance commands are found in Command Reference 1994, 18.16.

##### 3.3.3.2 4-wire Ohmmeter

4-wire ohmmeters measure the resistance across two of their input terminals with the assistance of two additional wires.

Typically, a DC current is sourced through one set of wires and the DC voltage is measured across the other set. As with a 2-wire ohms measurement, resistance is calculated by dividing the current into the voltage.

##### 3.3.3.3 SENSe Commands

SENSe:FRESistance commands are found in Command Reference 1994, 18.16.

#### 3.3.4 DC Ammeter

DC ammeters measure the average value of the current through their inputs.

## 1999 SCPI Instrument Classes

### 3.3.4.1 Range

DC ammeters shall follow the same constraints on RANGe as DC voltmeters.

### 3.3.4.2 SENSe Commands

SENSe:CURRent[:DC] commands are found in Command Reference 1994, 18.6.

### 3.3.5 AC RMS Ammeter

AC RMS ammeters measure the root mean square of the current through their inputs.

#### 3.3.5.1 Range

AC RMS ammeters shall follow the same constraints on RANGe as AC voltmeters.

#### 3.3.5.2 SENSe Commands

SENSe:CURRent:AC commands are found in Command Reference 1994, 18.6.

## 3.4 Programming Examples

This section discusses how a user might program a meter for certain applications. The examples assume the meter is a DC voltmeter, but a meter of any type could have been used.

### 3.4.1 Simple Measurement

In an application where the user wants to make a simple measurement with no special consideration for signal conditioning or timing, the MEASure query is an excellent choice. An example might be checking if a power supply voltage is at a proper level. The instrument might receive the following query to make a voltage measurement:

```
MEASure:VOLTage:DC?
```

The instrument autoranges to the best range and makes the measurement with a reasonable resolution. A single value is returned as soon as possible. The instrument does not wait for any other triggers.

Another user might send:

```
MEASure:VOLTage:DC? 5,.05
```

A voltmeter that has 1, 10, and 100 volt ranges would be in the 10 volt range. Another voltmeter with 3, 30, and 300 volt ranges would use the 30 volt range. Resolution would probably get set to 10 mv corresponding to 3 1/2 digits. Again the measurement is made as soon as possible.

This same measurement could be made using low level commands.

```
*RST
SENSe:FUNCTION "VOLTage:DC";VOLTage:RANGE 5V;RESolution .05V
INITiate;FETCH?
```

Note that \*RST sets all TRIGger subsystem SOURce to IMMEDIATE and COUNT to 1.

### 3.4.2 Time Critical Measurement

To use an external triggering signal, the user cannot use MEASure query because MEASure query sets TRIGger:SOURce to IMMEDIATE. Instead, CONFigure and READ? with TRIGger commands between them is the proper choice. The sequence:

```
CONFigure:VOLTage:DC 5V,.05V
TRIGger:SOURce EXTernal
READ?
```

sets up the SENSe functions easily. The single TRIGger command modifies the triggering subsystem just enough to do the job. The READ query initiates a measurement which waits for the external trigger before returning the result.

### 3.4.3 Multiple Measurements

The overhead of sending commands may prevent taking measurements made close together in time using a separate MEASure query or READ? for each measurement. The instrument could, however, buffer the readings. The sequence:

## 1999 SCPI Instrument Classes

```
CONFigure:VOLTage:DC 5V,.05V  
TRIGger:SOURce EXTERNAL;COUNT 10  
READ?
```

returns ten separate measurements taken when ten external triggers occur. A defensive programmer would then send SYSTem:ERRor? and check for error -210, “Trigger error” or -211, “Trigger ignored” to see if there were any problems with the triggering.

## **1999 SCPI Instrument Classes**

## 4 Digitizers

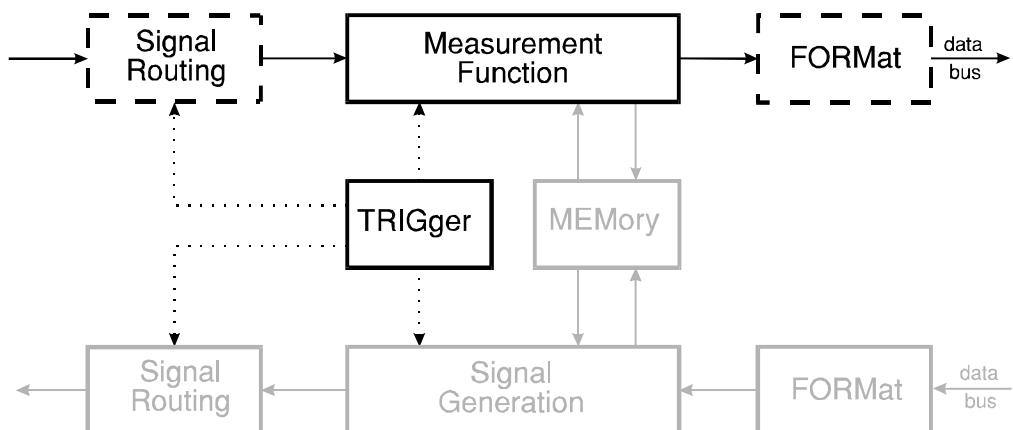
A digitizer is an instrument designed primarily to measure time-swept voltage waveforms. Because a digitizer is primarily a sensing device, the SENSe root node is optional.

All SCPI compliant products implement the commands listed in Syntax & Style, 4.2.

Figure 4-1 shows an instrument model for a digitizer. It is derived from Command Reference, chapter 2. It is essentially the same as Figure 2-1 in Command Reference with various parts gray or dashed. The bold boxes and lines, solid and dashed, are described in this chapter. The ROUTe block are shown with dashed lines because many digitizers use their default settings.

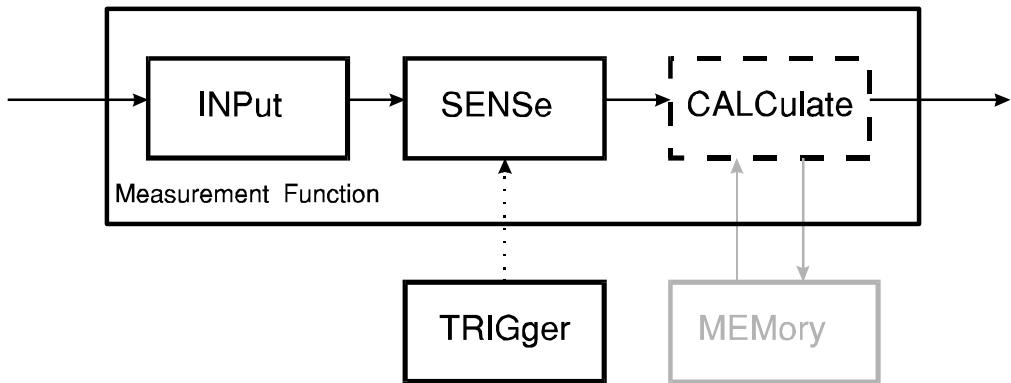
4

The gray boxes are not described here. While a digitizer may have MEMory, its functionality is optional and thus not described.



**Figure 4-1 Simplified Model of a Digitizer**

Figure 4-2 shows how the “Measurement Function” block in Figure 4-1 is expanded in a digitizer. It is the same as Figure 2-4 in Command Reference with some dashed and gray boxes. The CALCulate block is dashed because after \*RST this block has no effect on the data. This chapter does not describe any CALCulate functions. The MEMory block is gray because while a digitizer may contain this function, this chapter does not address it.



**Figure 4-2 Expanded Measurement Function Model**

## 4.1 Base Functionality

This section describes the functionality all digitizers implement.

The <bf\_keyword> for a digitizer is DIGITIZER.

Digitizer inputs are typically called channels. These are generally numbered 1, 2, 3, ... or lettered A, B, C, .... Several SCPI commands in this section use numeric suffixes on header keywords or parameters. The intention is that these suffixes unify operations that apply to the associated channel.

Two-channel capability is shown in the examples to illustrate this concept. A single channel digitizer may be implemented by omitting the command and or parameter options with the numeric suffix 2. More than two channels may be implemented by extending the numeric suffix.

### 4.1.1 Base Measurement Instruction

The base functionality for a digitizer requires no measurement instructions (MEASure, CONFigure, READ, FETCh).

### 4.1.2 Base Device-oriented Functions

The base functionality for digitizers shall contain the commands described in this section.

#### 4.1.2.1 Input functions

KEYWORD	PARAMETER FORM	SCPI Reference
INPut[1] INPut2		Vol 2-11
:COUpling	AC   DC   GND	Vol 2-11.3

This command controls the coupling characteristics of the input to the amplifier.

At \*RST, the coupling for digitizers shall be DC.

As previously mentioned, INPut1 is associated with the first input channel, and so on.

#### 4.1.2.2 SENSe amplitude functions

KEYWORD	PARAMETER FORM	SCPI Reference
[SENSe]		Vol 2-18
:VOLTage[1]:VOLTage2		Vol 2-18.24
[:DC]		Vol 2-18.24.1
:RANGE		Vol 2-18.24.1.5
:LOWER <numeric_value>		Vol 2-18.24.1.5.2
:OFFSet <numeric_value>		Vol 2-18.24.1.5.4
:PTPeak <numeric_value>		Vol 2-18.24.1.5.5
[:UPPer] <numeric_value>		Vol 2-18.24.1.5.1

VOLTage1 is associated with INPut1, VOLTage2 is associated with INPut2, and so on.

The PTPeak command sets the amplifier gain of the specified channel.

The OFFSet command sets the voltage offset of the amplifier, the midpoint of the PTPeak range. Measurement results remain constant when offset is added.

UPPer and LOWER, also expressed in volts, are coupled to OFFSet and PTPeak as described in the introduction to Command Reference, chapter 18.

#### 4.1.2.3 SENSE timebase functions

KEYWORD	PARAMETER FORM	SCPI Reference
[SENSe]		Vol 2-18
:SWEep		Vol 2-18.23
:POINts <numeric_value>		Vol 2-18.23.8
:TIME <numeric_value>		Vol 2-18.23.12
:TINTerval <numeric_value>		Vol 2-18.23.13

POINts controls the record length of the acquisition.

TIME is the full time span of the acquisition.

TINTerval controls the time interval between points.

NOTE that if the number of sweep points and the time interval are changed then the sweep time will change as well.

At \*RST, a digitizer shall set its SWEep parameters to a known, device-specified value.

#### 4.1.2.4 SENSe One-of-N Function Control

Digitizers shall implement the following commands that allow the selection of the active channel using [SENSe:]FUNCtion.

## 1999 SCPI Instrument Classes

KEYWORD	PARAMETER FORM	SCPI Reference
[SENSe]		Vol 2-18
:DATA?	[<data_handle>]	Vol 2-18.13.1
:FUNCtion		
:CONCurrent	<Boolean>	Vol 2-18.13.2.1
:OFF	<sensor_function>	Vol 2-18.13.2.2
[:ON]	<sensor_function>	Vol 2-18.13.2.3
:STATe?	<sensor_function>	Vol 2-18.13.2.4

4

Where <sensor\_function> and <data\_handle> for channel 1 shall be specified as "XTIME:VOLTage[:DC] 1" and so on.

[SENSe]:DATA? shall return data for the channel specified in the query parameter or shall return data for the active channel(s).

### 4.1.2.5 Formatting

KEYWORD	PARAMETER FORM	SCPI Reference
:FORMat		
[:DATA]	<type> [, <numeric_value>]	Vol 2-

The parameter ASCii and the <numeric\_value> zero shall be supported.

At \*RST, the :FORMAT type shall be ASCii,0.

### 4.1.2.6 Trigger functions

KEYWORD	PARAMETER FORM	SCPI Reference
INITiate		
[:IMMEDIATE]		Vol 2-24.7.2
[:ALL]		Vol 2-24.7.2.1
ABORT		Vol 2-24.5
TRIGger		Vol 2-24.8
[:SEQUENCE[1]]		
:COUPling	AC   DC	Vol 2-24.8.1.3
:LEVel	<numeric_value>	Vol 2-24.8.1.12
:SLOPe	POSitive   NEGative   EITher	Vol 2-24.8.1.16
:SOURce	INTERNAL[1]   ...	Vol 2-24.8.1.17

Triggering sourced from channel 1 shall be specified as INTERNAL1 and so on.

At \*RST, SOURce shall be set to INTERNAL1, COUPLING shall be set to DC, SLOPe shall be set to POSitive, and LEVel shall be set to a known, device-specified value.

### 4.1.3 Base Status Reporting

All SCPI digitizers shall implement the status reporting structure described in Syntax & Style, Status Reporting. Command Reference, STATus Subsystem defines the commands which shall be used to control the status reporting structure.

**4.1.3.1 QUESTIONable**

The base functionality of a digitizer adds no requirements for this register.

**4.1.3.2 OPERATION**

At a minimum, the digitizer shall report Waiting for TRIG, bit 5.

**4.2 Programming Example**

The following set of commands sets up the digitizer to acquire a waveform on channel 1 for AC coupling, 10 peak to peak signal with a sample recorded every microsecond. The trigger source is channel 1 and is DC coupled, looking for a positive slope transition of 100 millivolts. After setting up the digitizer the trigger system is initiated and the resultant captured waveform is returned to the controller in ASCII format.

Longform version:

```
*RST  
INPUT:COUpling AC  
SENSe:VOLTage:DC:RANGE:PTPeak 10 V  
SENSe:SWEep:TINTerval 1E-6 S  
SENSe:FUNCTION:CONCurrent OFF  
SENSe:FUNCTION:ON "XTIMe:VOLTage:DC 1"  
TRIGger:SEQuence1:LEVEL 0.1V  
INITiate:IMMEDIATE:ALL  
SENSe:DATA?
```

Shortform version:

```
*RST  
INP:COUP AC  
VOLT:RANG:PTP 10  
SWE:TINT 1E-6 S  
FUNC:CONC 0;ON "XTIM:VOLT 1"  
TRIG:LEV 0.1  
INIT  
DATA?
```

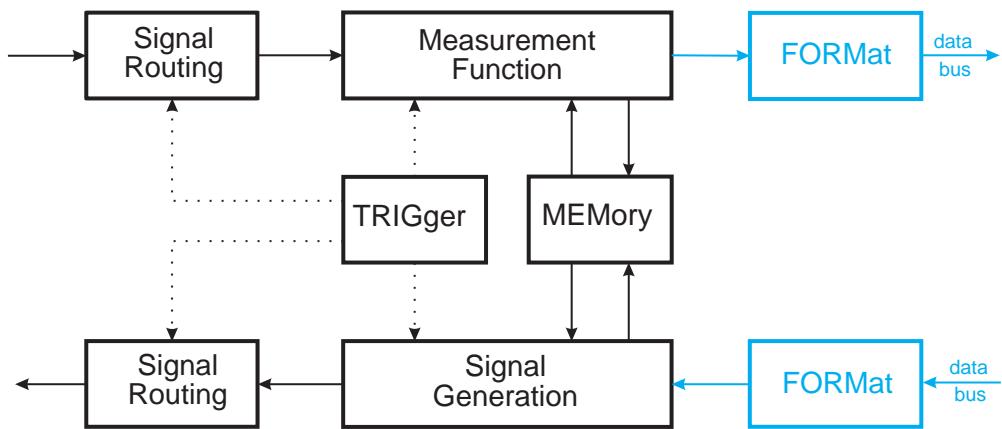
## **1999 SCPI Instrument Classes**

## 5 Emissions Benches

An emissions bench is a complex measuring device. It measures the constituent concentration in a sample of gas introduced into the bench. It typically consists of several instruments, at least one of which is a gas analyzer, but also can include a mini-diluter, a gas divider, as well as temperature, pressure, and flow measurement instruments.

All SCPI compliant products implement the commands listed in Syntax and Style, 4.2.

5



**Figure 5-1 Simplified Model for an Emissions Bench**

As indicated in Figure 5-1 above, emissions benches make use of signal (gas) routing commands, the TRIGger subsystem, extensive use of measurement function commands (using the :SENSe block), and extensive use of the :MEMory subsystem. In addition, the :CALibrate and :SYSTem subsystems are utilized in both standard and emissions bench-specific ways.

### 5.1 Base Functionality

This section describes the base functionality for emissions benches.

The <bf\_keyword> for an emissions bench is EBENCH.

#### 5.1.1 Base Measurement Instructions

The base functionality for an emissions bench contains no commands from the :MEASurement subsystem.

#### 5.1.2 Base Device-Oriented Functions

The base functionality for an emissions bench include :SENSe, :MEMory, :TRIGger, :CALibration, :CONTrol, :SYSTem, :DIAGnostic, :INSTRument, and :ROUTE subsystems. It also defines device dependent :STATus bits and instructions for emissions benches. A SCPI-compliant emissions bench shall implement all of the commands listed in this section.

### 5.1.2.1 CALibration subsystem

CALibration commands for emissions benches, detailed in section 5.4 of this volume, are as follows:

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
:CALibration		Vol 4-5.4
:LINearize		Vol 4-5.4.1
:ACCept		Vol 4-5.4.1.1
:ACQuire		Vol 4-5.4.2
:AUTO	ONCE	Vol 4-5.4.2.1
:CALCulate	AUTO   POLYnomial<n>   SRATional<n>	Vol 4-5.4.2.2
:CURVe		Vol 4-5.4.2.3
[:TYPE]	POLYnomial<n>   SRATional<n>,<numeric_value>{,<numeric_value>}	Vol 4-5.4.2.3.1
	<Boolean>	Vol 4-5.4.2.3.2
:ZFORce		Vol 4-5.4.2.4
:VERify		Vol 4-5.4.2.4.1
:ACQuire		Vol 4-5.4.2.4.2
:TOLERance	<numeric_value>	Vol 4-5.4.2.4.3
:TYPE	CCURve   NCURve	Vol 4-5.4.2.4.3

### 5.1.2.2 CONTrol subsystem

CONTrol commands for emissions benches are as follows:

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
:CONTrol		Vol 2-6
:EBENch		Vol 2-6.6
:CLEan		Vol 2-6.6.1
:DURATION	<numeric_value>	Vol 2-6.6.1.2
[:INIT]		Vol 2-6.6.1.1

### 5.1.2.3 DIAGnostic subsystem

DIAGnostic commands for emissions benches, detailed in section 5.5 of this volume, are as follows:

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
:DIAGnostic		Vol 4-5.5
:HUP		Vol 4-5.5.1
:ACQuire		Vol 4-5.5.1.1
:CALCulate		Vol 4-5.5.1.2
:LEAK		Vol 4-5.5.2
:ACQuire		Vol 4-5.5.2.1
:CALCulate		Vol 4-5.5.2.2
:NEFFiciency		Vol 4-5.5.3
:ACQuire		Vol 4-5.5.3.1
:CALCulate		Vol 4-5.5.3.2

### 5.1.2.4 INSTRument subsystem

INSTRument commands for emissions benches are as follows:

KEYWORD	PARAMETER FORM	SCPI Reference
:INSTrument		Vol 2-12
:CATalog?		Vol 2-12.1
:FULL?		Vol 2-12.1.1
:DEFIne		Vol 2-12.3
:GROup	<identifier>, <identifier_list>	Vol 2-12.3.1
[:NAME]	<identifier>, <numeric_value>	Vol 2-12.3.2
:DELete		Vol 2-12.4
:ALL		Vol 2-12.4.1
[:NAME]	<identifier>	Vol 2-12.4.2
[:SElect]	<identifier>	Vol 2-12.6

### 5.1.2.5 MEMory subsystem

MEMory commands for emissions benches are as follows:

KEYWORD	PARAMETER FORM	SCPI Reference
:MEMory		Vol 2-13
:TABLE		Vol 2-13.11
:BNUMber<n>	<numeric_value> {,<numeric_value>}	Vol 2-13.11.1
:POINTs?		Vol 2-13.11.1.1
:CCURve	<numeric_value> {,<numeric_value>}	Vol 2-13.11.2
:POINTs?		Vol 2-13.11.2.1
:CONCetration	<numeric_value> {,<numeric_value>}	Vol 2-13.11.3
:POINTs?		Vol 2-13.11.3.1
:CPOint	<numeric_value> {,<numeric_value>}	Vol 2-13.11.5
:POINTs?		Vol 2-13.11.5.1
:DEFIne	<structure_string> [,<numeric_value>]	Vol 2-13.11.11
:DFACtory	<numeric_value> {,<numeric_value>}	Vol 2-13.11.7
:POINTs?		Vol 2-13.11.7.1
:DLASt	numeric_value> {,<numeric_value>}	Vol 2-13.11.8
:POINTs?		Vol 2-13.11.8.1
:DLINearize	<numeric_value> {,<numeric_value>}	Vol 2-13.11.9
:POINTs?		Vol 2-13.11.9.1
:EXPected	<numeric_value> {,<numeric_value>}	Vol 2-13.11.10
:POINTs?		Vol 2-13.11.10.1
:LABEL	<string> {,<string>}	Vol 2-13.11.14
:POINTs?		Vol 2-13.11.14.1
:LOG	<string> {,<string>}	Vol 2-13.11.16
:POINTs?		Vol 2-13.11.16.1
:NCURve	<numeric_value> {,<numeric_value>}	Vol 2-13.11.18
:POINTs?		Vol 2-13.11.18.1

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>NOTES</b>
:RAW	<numeric_value> {,<numeric_value>}	Vol 2-13.11.20
:POINts?		Vol 2-13.11.20.1
:SElect	<identifier>	Vol 2-13.11.22
:TOLerance	<numeric_value> {,<numeric_value>}	Vol 2-13.11.25
:POINts?		Vol 2-13.11.25.1
:WFACtor	<numeric_value> {,<numeric_value>}	Vol 2-13.11.28
:POINts?		Vol 2-13.11.28.1

### 5.1.2.6 ROUTe subsystem

ROUTe commands for emissions benches are as follows:

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
:ROUTE		Vol 2-17
:SAMPlE		Vol 2-17.4.3.1
:CATalog?		Vol 2-17.5.1
[:OPEN]	BAG   DILute   PRE   POST   MID   CEFFiciency   NONE   ZERO   SPAN   VERify   MANifold<n>	Vol 2-17.5.2

Note that some emissions benches may not implement all of the above sample paths.

### 5.1.2.7 SENSe subsystem

SENSe commands for emissions benches are as follows:

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
:SENSe		Vol 2-18
:AVERage		Vol 2-18.2
:COUNT	<numeric_value>	Vol 2-18.2.2
:TCOnrol	EXPonential   MOVing	Vol 2-18.2.4
	NORMal   REPeat	
:TYPE	COMPLex   ENVelope   MAXimum   MINimum   RMS   SCALar	Vol 2-18.2.5
:CONCetration		Vol 2-18.4
:CSET	<numeric_value>,<numeric_value>	Vol 2-18.4.1
	{,<numeric_value>,<numeric_value>,...}	
:LOWER	<numeric_value>{,<numeric_value>}	Vol 2-18.4.2
:LSET	POLYnomial<n>	Vol 2-18.4.3
	SRATional<n>,<numeric_value>	
	{,<numeric_value>},{POLYnomial<n>}	
	SRATional<n>,<numeric_value>	
	{,<numeric_value>})	
:RANGE		Vol 2-18.4.4
:AUTO		Vol 2-18.4.4.1
:LOWER	<numeric_value>{,<numeric_value>}	Vol 2-18.4.4.1.1
[:STATE]	<Boolean>	Vol 2-18.4.4.1.2
:UPPer	<numeric_value>{,<numeric_value>}	Vol 2-18.4.4.1.3
[:FIXed]	<numeric_value>{,<numeric_value>}	Vol 2-18.4.4.2

## 1999 SCPI Instrument Class Applications

:TALign	<numeric_value> {,<numeric_value>}	Vol 2-18.4.5
:UPPer	<numeric_value>{,<numeric_value>}	Vol 2-18.4.6
:CORRection		Vol 2-18.6
:AUTO		Vol 2-18.6.1
:CALCulate		Vol 2-18.6.2
:SPOint		Vol 2-18.6.10
:ACQuire		Vol 2-18.6.10.1
:DTOLerance	<numeric_value>{,<numeric_value>}	Vol 2-18.6.10.2
:STATe	<Boolean>	Vol 2-18.6.11
:ZERO		Vol 2-18.6.12
:ACQuire		Vol 2-18.6.12.1
:DTOLerance	<numeric_value>{,<numeric_value>}	Vol 2-18.6.12.2
:DATA?		Vol 2-18.13.1
:FUNCTion	<function_name> {,<function_name>}	Vol 2-18.13.2
	where function_name can be:	
	CONCentration<n>,TIMer:COUNt,	
	CONCentration<n>;SDEViation,	
	CONCentration<n>;TALign,	
	CONCentration<n>;RAW	
:CONCurrent	<Boolean>	Vol 2-18.13.2.1
:STABilize		Vol 2-18.22
:NTOLerance	<numeric_value>{,<numeric_value>}	Vol 2-18.22.1
[:STATe]	<Boolean>	Vol 2-18.22.2
:TIME<n>	<numeric_value>{,<numeric_value>}	Vol 2-18.22.3

5

### NOTE:

:SENSe:CORRection:CALCulate calculates the zero/span adjustment of the form

$$y=mx+b$$

where  $x = \%FSRAW$  Data Value

$y = \%FSADJ$  Data Value

$m$  = Zero/Span Adjustment Slope Coefficient

$b$  = Zero/Span Adjustment Intercept Coefficient

and

$$m = \frac{S_0 - Z_0}{S_m - Z_m}$$

$$b = \frac{Z_0 S_m - Z_m S_0}{S_m - Z_m}$$

where  $Z_0$  = Zero Gas Expected

$Z_m$  = Zero Gas Measured

$S_0$  = Span Gas Expected

$S_m$  = Span Gas Measured

## 1999 SCPI Instrument Class Applications

### 5.1.2.8 SYSTem subsystem

SYSTem commands for emissions benches are as follows:

KEYWORD	PARAMETER FORM	SCPI Reference
:SYSTem		Vol 2-21
:COMMunicate		Vol 2-21.4
:SOCKet<n>		Vol 2-21.4.4
:ADDRess <string> {<IPaddress> <Host Name>}		Vol 2-21.4.4.1
:CONNect		Vol 2-21.4.4.2
:DISConnect		Vol 2-21.4.4.3
:FEED <data_handle>{,<data_handle>}		Vol 2-21.4.4.4
:OCONDition <event_handle>		Vol 2-21.4.4.4.1
:SCONDition <event_handle>		Vol 2-21.4.4.4.2
:LISTen		Vol 2-21.4.4.5
:PORT <NRF> <non-decimal numeric>		Vol 2-21.4.4.6
:TYPE TCP UDP		Vol 2-21.4.4.7
:DATE <year>,<month>,<day>		Vol 2-21.7
:ERRor		Vol 2-21.8
:ALL?		Vol 2-21.8.4
:CODE		Vol 2-21.8.5
:ALL?		Vol 2-21.8.5.1
[:NEXT]?		Vol 2-21.8.5.2
:ENABLE		Vol 2-21.8.7
:ADD <numeric list>		Vol 2-21.8.7.1
:DELETE <numeric list>		Vol 2-21.8.7.2
[:LIST]		Vol 2-21.8.7.3
[:NEXT]?		Vol 2-21.8.8
:LOCK		Vol 2-21.14
:RELEASE		Vol 2-21.14.2
:REQuest?		Vol 2-21.14.3
:OWNer?		Vol 2-21.14.1
:TIME		Vol 2-21.19
:TIMer		Vol 2-21.19.1
:COUNt <numeric_value>		Vol 2-21.19.1.1
[:STATe] <Boolean>		Vol 2-21.19.1.2

### 5.1.2.9 TRIGger subsystem

TRIGger commands for emissions benches are as follows:

KEYWORD	PARAMETER FORM	SCPI Reference
:TRIGger		Vol 2-24.8
[:SEQUence]		Vol 2-24.8.1
:ECOunt	<numeric_value>	Vol 2-24.8.1.7
:LINK	<event_handle>	Vol 2-24.8.1.13
:SOURCE	TIMer	Vol 2-24.8.1.17
:TIMER	<numeric_value>	Vol 2-24.8.1.18

### 5.1.3 Base Status Reporting

Volume 1, Syntax and Style, Status Reporting, describes the status reporting structure used in all SCPI instruments. Volume 2, Command Reference, STATus Subsystem, defines the commands that control the status reporting structure.

### 5.1.3.1 OPERATION Status

:STATus:OPERation commands for benches utilize bit 9 of the OPERation status register (see SCPI Volume 1 figure 9-1) for bench-wide status and assume a parallel structure to the :STATus:QUEstionable commands referred to in section 5.1.3.2 below. Therefore, bit 10 of an instrument summary operations register is used for instrument-specific status.

KEYWORD	PARAMETER FORM	SCPI Reference
:STATus		Vol 2-20
:OPERation		Vol 2-20.1
:BIT9		Vol 2-20.1.1
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRF> <non-decimal numeric>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:NTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.6
:PTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.7
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRF> <non-decimal numeric>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:INSTRument		
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRF> <non-decimal numeric>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:ISUMmary<n>		
:BIT10		Vol 2-20.1.1
:CALibrating		
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRF> <non-decimal numeric>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:NTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.6
:PTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.7
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRF> <non-decimal numeric>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:NTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.6
:PTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.7
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRF> <non-decimal numeric>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:NTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.6
:PTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.7
:NTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.6
:PTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.7
:NTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.6
:PTRansition	<NRF> <non-decimal numeric>	Vol 2-20.1.7

### 5.1.3.2 **QUESTIONable Status**

:STATus:QUESTIONable commands for benches utilize bit 9 of the QUESTIONable status register (see SCPI Volume 1 figure 9-1) for bench-wide status and bit 10 of the instrument summary status register (see SCPI volume 1 figure 9-3) for instrument-specific status as follows:

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>	<b>5</b>
:STATus		Vol 2-20	
:QUESTIONable		Vol 2-20.3	
:BIT9		Vol 2-20.3.1	
:CONDITION?		Vol 2-20.3.2	
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.3.3	
[:EVENT]?		Vol 2-20.3.4	
:NTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.6	
:PTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.7	
:CONDITION?		Vol 2-20.3.2	
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.3.3	
[:EVENT]?		Vol 2-20.3.4	
:INSTrument			
:CONDITION?		Vol 2-20.3.2	
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.3.3	
[:EVENT]?		Vol 2-20.3.4	
:ISUMmary<n>			
:BIT10		Vol 2-20.3.1	
:CALibration			
:CONDITION?		Vol 2-20.3.2	
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.3.3	
[:EVENT]?		Vol 2-20.3.4	
:NTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.6	
:PTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.7	
:CONDITION?		Vol 2-20.3.2	
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.3.3	
[:EVENT]?		Vol 2-20.3.4	
:NTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.6	
:PTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.7	
:CONDITION?		Vol 2-20.3.2	
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.3.3	
[:EVENT]?		Vol 2-20.3.4	
:NTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.6	
:PTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.7	
:CONDITION?		Vol 2-20.3.6	
:ENABLE	<NRf> <non-decimal numeric>	Vol 2-20.3.7	
:NTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.6	
:PTRansition	<NRf> <non-decimal numeric>	Vol 2-20.3.7	

### 5.2 Additional functionality

All functionality described for emissions benches is considered base functionality at this time.

### 5.3 Examples

#### 5.3.1 Zero/Span/Zero Procedure

The following illustrates a possible command sequence for performing a zero/span/zero procedure:

SCPI Command	Notes
:INST:DEFIne:GROup "BENCH", "CONC1", "CONC2"	
:INST:SElect "BENCH"	<i>Select desired analyzers</i>
:SENSe:FUNCTION:CONCurrent ON	
:SENSe:FUNCTION "CONC1", "CONC1:SDEV", "CONC2", "CONC2:SDEV"	<i>Applies to selected analyzers; used in :SENS:DATA</i>
:SENSe:CORRection:ZERO:DTOLerance 5	<i>Allow 5% tolerance for zero drift</i>
:SENSe:CORRection:SPOint:DTOLerance 5	<i>Allow 5% tolerance for set point (span) drift</i>
:SENSe:CORRection:STATE OFF	<i>New Z/S/Z needs uncorrected values; drift check would require :SENS:CORR:STATE ON</i>
:SENSe:CORRection:ZERO:ACQuire	<i>request and execute zero procedure</i>
:STATus:OPERation:INSTRument:ISUMmary1:BIT10:CALibrating?	<i>Keep polling while bit 1 (zero) is set (CONC1)</i>
:STATus:OPERation:INSTRument:ISUMmary2:BIT10:CALibrating?	<i>Keep polling while bit 1 (zero) is set (CONC2)</i>
:STATus:QUESTIONable:INSTRument:ISUMmary1:BIT10:CALibrating?	<i>0 = successful zero; Bit 1 set = failure</i>
:STATus:QUESTIONable:INSTRument:ISUMmary2:BIT10:CALibrating?	<i>0 = successful zero; Bit 1 set = failure</i>
:SENSe:DATA?	<i>Retrieve Zero Concentration Readings and SDEVs</i>
:MEMORY:DATA?ZDRift	<i>Retrieve Zero Drifts</i>
:SENSe:CORRection:SPOint:ACQuire	<i>execute SPOint (span) procedure</i>
*WAI	<i>Alternative to polling; wait for SPOint (span) operation to complete</i>
:STATus:QUESTIONable:INSTRument:ISUMmary1:BIT10:CALibrating?	<i>0 = successful zero; Bit 2 set = failure</i>
:STATus:QUESTIONable:INSTRument:ISUMmary2:BIT10:CALibrating?	<i>0 = successful zero; Bit 2 set = failure</i>
:SENSe:DATA?	<i>Retrieve Span Concentration Readings and SDEV's</i>
:MEMORY:DATA?SDRift	<i>Retrieve Span Drifts</i>
:SENSe:CORRection:CALCulate	<i>Calculate new m, b corrections (skip this if drift check)</i>
:SENSe:CORRection:ZERO:ACQuire	<i>do re-zero</i>

## 1999 SCPI Instrument Class Applications

```

:STATus:OPERation:INSTRument:ISUMmary1:BIT10:CALibrating?
    Keep polling while bit 1 (zeroing) is set
    (CONC1)
:STATus:OPERation:INSTRument:ISUMmary2:BIT10:CALibrating?
    Keep polling while bit 1 (zeroing) is set
    (CONC2)
:STATus:QUESTIONable:INSTRument:ISUMmary1:BIT10:CALibrating?
    0 = successful zero; Bit 1 set = failure
:STATus:QUESTIONable:INSTRument:ISUMmary2:BIT10:CALibrating?
    0 = successful zero; Bit 1 set = failure
:SENSe:DATA?          Retrieve Zero Readings and SDEVs
:MEMORY:DATA?ZDRift Retrieve Zero Drifts
:SENSe:CORRection:ZERO:DTOLerance 2
    Allow 2% tolerance for zero drift
:SENSe:CORRection:STATE[:ON]
    start applying new m, b
:SENSe:CONC:CSET?    Retrieve m,b for this range of this analyzer

```

### 5.3.2 Bag Read Procedure

The following command sequence illustrates a possible command sequence for bag reads:

SCPI Command	Notes
:SYSTem:LOCK:REQuest?	<i>Assert host control of bench if 1 is returned</i>
:INSTRument:DEFIne:GROUP "Bag_Analyzers", "CONC1", "CONC2", "CONC3", "CONC4"	<i>Sets up the group "Bag_Analyzers" to use four concentrations</i>
:INSTRument:SELect "Bag_Analyzers"	
:CONTrol:EBENch:CLEan:DURation 60	<i>Set a gas line "cleanup" for 60 seconds</i>
:CONTrol:EBENch:CLEan[:INIT]	<i>Start CLEAN procedure</i>
:SENSe:FUNCTion:CONCurrent ON	
:SENSe:FUNCTion "CONC1", "CONC1:SDEV", "CONC2", "CONC2:SDEV", "CONC3", "CONC3:SDEV", "CONC4", "CONC4:SDEV"	
:TRIGger:SEQ1:SOURce TIMER	
:TRIGger:SEQ1:TIMER 0.1	<i>".1" means .1 seconds</i>
:SENSe:STABilized:STATE ON	<i>Use Stabilized mode</i>
<i>Start "Sniff"</i>	
:SENSe:CONC:RANGE:AUTO ON	<i>Auto range to find correct range</i>
<i>Host command to sampling system that selects bag</i>	
:ROUTe:SAMPle BAG Cold_Transient_Sample	
:INITiate:IMMEDIATE:SEQUence1	
:STATus:OPERation:BIT9?	<i>Keep polling while bit 5 (sampling) is set</i>
:STATus:QUESTIONable:BIT9?	

## 1999 SCPI Instrument Class Applications

5

```
:SENSe:DATA?          0 = successful bag sample
:SENSe:CONC:RANGE:AUTO OFF
:SENSe:CONC:RANGE?      Retrieve four Sniff concentrations
                        Leave in selected range
:SENSe:CONC:RANGE:FIXed?
:ROUTe:SAMPle NONE    Retrieve range numbers (n,n,n,n)
                        Turn off gas flow
```

Perform ZERO/SPAN/ZERO PROCEDURE - See Section 5.3.1

Host command to sampling system that selects bag

```
:ROUTe:SAMPle BAG   Cold_Transient_Sample
:INITiate:IMMEDIATE:SEQUence1
:STATus:OPERation:BIT9?
                        Keep polling while bit 5 (sampling) is set
:STATus:QUESTIONable:BIT9?
                        0 = successful bag sample
:SENSe:DATA?          Retrieve the four Sample Bag concentrations and
                        SDEVs
```

Host command to sampling system that selects bag

```
:INITiate:IMMEDIATE:SEQUence1
:STATus:OPERation:BIT9?
                        Keep polling while bit 5 (sampling) is set
:STATus:QUESTIONable:BIT9?
                        0 = successful bag sample
:SENSe:DATA?          Retrieve the four Ambient Bag concentrations and
                        SDEVs
:ROUTe:SAMPle NONE   Turn off gas flow
```

Perform Z/S Drift check procedure - similar to section 5.3.1

```
:SYSTem:LOCK:RELEASE
                        Release control of bench
```

### 5.3.3 Start of Diesel Test Procedure

The following command sequence illustrates a possible command sequence for starting a diesel test:

SCPI Command	Notes
<i>Setup Phase</i>	
:SYSTem:LOCK:REQuest?	Assert host control of bench
:INSTRument:DEFIne:GROup "All_Bag_Analyzers", "CONC1", "CONC2", "CONC3", CONC4"	Sets up the groups to use
:INSTRument:SElect "All_Bag_Analyzers"	
:CONTrol:EBENch:CLEAR:DURATION 60	Do a gas line "cleanup" for 60 seconds
:CONTrol:EBENch:CLEAR[:INIT]	Do a gas line "cleanup" for 60 seconds
:SENSe:FUNCTION:CONCurrent ON	Applies to selected analyzers
:SENSe:FUNCTION "CONC1", "CONC1:SDEV", "CONC2", "CONC2:SDEV",	

## 1999 SCPI Instrument Class Applications

```
"CONC3", "CONC3:SDEV", "CONC4", "CONC4:SDEV"
                           Applies to selected analyzers
:SENSe:STABilization:STATE ON
                           Applies to selected analyzers
:SENSe:STABilization:TIME1 60,50,45,55
                           Stabilization rise time
:SENSe:STABilization:TIME2 15,15,15,15
                           Stabilization evaluation period
:SENSe:STABilization:TIME3 5,5,5,5
                           Stabilization averaging period
:SENSe:STABilization:TIME4 120,120,120,120
                           Stabilization maximum duration
:SENSe:STABilization:TOLerance 4,4,4,4
                           Stabilization tolerance in %
:INSTRument:DEFIne:GROUP
"Bag_With_Out_Fid", "CONC2", "CONC3", "CONC4"
:INSTRument:SElect "Bag_With_Out_Fid"
:SENSe:FUNCtion:CONCurrent ON
                           Applies to selected analyzers
:SENSe:FUNCtion"CONC2", "CONC2:SDEV",
"CONC3", "CONC3:SDEV", "CONC4", "CONC4:SDEV"
                           Applies to selected analyzers
:SENSe:STABilization:STATE ON
                           Applies to selected analyzers
:SENSe:STABilization:TIME1 50,45,55
                           Stabilization rise time
:SENSe:STABilization:TIME2 15,15,15
                           Stabilization evaluation period
:SENSe:STABilization:TIME3 5,5,5
                           Stabilization averaging period
:SENSe:STABilization:TIME4 120,120,120
                           Stabilization maximum duration
:SENSe:STABilization:TOLerance 4,4,4
                           Stabilization tolerance in %
:INSTRument:DEFIne:GROUP "Just_Heated Fid", "CONC1"
:INSTRument:SElect "Just_Heated Fid"
:SENSe:AVERage1:COUNT 10
:SENSe:AVERage1:TCONtrol MOVing
:SENSe:AVERage1:TYPE SCALar
:SENSe:AVERage1:STATE ON
:SENSe:FUNCtion:CONCurrent ON
:SENSe:FUNCtion "CONC1 ON AVERagel", "CONC1:SDEV ON AVERagel"
:SENSe:STABilization:STATE ON
:SENSe:STABilization:TIME1 60
                           Stabilization rise time
:SENSe:STABilization:TIME2 15
                           Stabilization evaluation period
:SENSe:STABilization:TIME3 5
                           Stabilization averaging period
:SENSe:STABilization:TIME4 120
```

## 1999 SCPI Instrument Class Applications

*Stabilization maximum duration*

```
:SENSe:STABilization:TOlerance 4
    Stabilization tolerance in %
:TRIGger:SEQ1:SOURce TIMer
    Set up 10 Hz Sampling
:TRIGger:SEQ1:TIMer 0.1
    Set up integration rate
:TRIGger:SEQ2:LINK "TRIGger:SEQence1"
    Seq 2 slaved to Seq1
:TRIGger:SEQ2:ECOut 10
    Sequence 2 once per second
:SYSTem:COMMUnicatE:SOCKEt1:ADDReSS "123.456.789.1"
    Separate Data feed
:SYSTem:COMMUnicatE:SOCKEt1:PORT 701
    to the host
:SYSTem:COMMUnicatE:SOCKEt1:TYPE TCP
    A TCP Socket
:SYSTem:COMMUnicatE:SOCKEt1:FEED:OCON "TRIG:SEQ1"
    Collect 10/sec
:SYSTem:COMMUnicatE:SOCKEt1:FEED:SCON "TRIG:SEQ2"
    Send once/sec
:SYSTem:COMMUnicatE:SOCKEt1:CONNect
```

### *Pretest function*

```
:INSTRument:SElect "Just_Heated_Fid"
:SENSe:CONCentratiOn:RANGE:FIXed n
    Alternate form for 1 analyzer
```

### *Perform ZERO/SPAN/ZERO PROCEDURE - See Section 5.3.1*

```
:SENSe:STABilization:STATe OFF
    For command channel "reads"
:SENSe:FUNCTION "TIMER:COUNT", "CONC1:TALign", "TEMP1"
:SYSTem:COMMUnicatE:SOCKEt1:FEED "TIMER:COUNT", "CONC1:TAL",
"TEMP1"           For socket feed
:ROUTe:SAMPle DILute
```

### *Phase 1*

```
:INITiate:CONTinuous ON
    Start socket feed
```

*drive phase 1*

## 5.4

**CALibration Subsystem (Bench Commands)**

KEYWORD	PARAMETER FORM	NOTES
:CALibration		
:LINearize		
:ACCept		[event]
:ACQuire		[event]
:AUTO	ONCE	[event]
:CALCulate	AUTO   POLYnomial<n>	
	SRATional<n>	[event]
:CURve		
[:TYPE]	POLYnomial<n>	
	SRATional<n>, <numeric_value>{,<numeric_value>}	
:ZFORce	<Boolean>	
:VERify		
:ACQuire		[event]
:TOLERance	<numeric_value>	
:TYPE	CCURve   NCURve	

5

## 5.4.1

**:LINearize**

CALibration:LINearize

The LINearize subsystem is used to produce a calibration curve used to linearize a non-linear instrument. An instance of this subsystem exists for each range in an instrument. The data defined and queried and the events will only be performed on the currently selected range and instrument.

## 5.4.1.1

**:ACCept**

CALibration:LINearize:ACCept

ACCept causes an already-calculated or supplied set of coefficients to be placed into regular use by an instrument. This consists of replacing the “current” values with the “new” values. The current values can be obtained by querying :SENSe:CONCetration:RANGE:LSET?.

## 5.4.2

**:ACQuire**

CALibration:LINearize:ACQuire

ACQuire causes the automatic collection of linearization data POINTs for the currently selected range of the currently selected analyzer to begin. Results are placed in a predefined table as they are collected. This procedure stores the expected gas concentration, causes gas flow for the defined cutPOINTs, waits for a stabilized reading of each one, and records the measured values. :SENSe:STABilize:STATE must be ON when this command is issued. Gases are selected for the duration of the procedure.

## 5.4.2.1

**:AUTO ONCE**

CALibration:LINearize:AUTO

This command shall perform linearization based on a vendor defined procedure, including :CALibrate:LINearize:ACQuire and possibly other procedures, for example :SENSe:CORRection:ZERO and :SPOint. :SENSe:STABilize:STATE must be ON when this command is issued. Gases are selected for the duration of the procedure. This command is

## 1999 SCPI Instrument Class Applications

provided to allow the instrument to perform an automated linearization procedure according to the manufacturer's recommended practice.

### 5.4.2.2 :CALCulate AUTO|POLYnomial<n>|SRATional<n>

CALibration:LINearize:CALCulate

Using the results data stored in a predefined linearization table, calculate a linearization curve. The results are placed in that table. A curve fit of type AUTO allows the device to calculate the curve fit according to the manufacturer's recommended practice. A POLYnomial<n> parameter shall limit the calculation to the specified polynomial curve type. A SRATional<n> parameter shall limit the linearization to the specified simple rational curve fit. The <n> designates the order of the curve. A curve fit can automatically be forced through the origin, if :CALibration:LINearize:CURVe:ZFORce has been turned on. The results can be stored in the instrument at a later time (see ACCept). An example of the predefined table follows:

**LSI01R01**

CPOint (%FS)	EXPected	RAW	CCURve	NCURve
6.7	10000	464646	10015	10001
20.0	30000	484848	30109	30000
33.3	50000	515151	49905	49999
46.7	70000	525252	71004	70001
60	90000	555555	89950	90005
73.3	110000	565656	111070	110100
86.7	130000	575859	129051	129999
100	150000	590101	150031	149000

### 5.4.2.3 :CURVe

CALibration:LINearize:CURVe

This node controls analyzer curve fitting.

#### 5.4.2.3.1 [:TYPE] POLYnomial<n> | SRATional<n>, <numeric\_value>{,<numeric\_value>}

CALibration:LINearize:CURVe[:TYPE]

[:TYPE] sets or retrieves the information used to determine a calibration curve.

The first parameter is the curve type, which consists of either POLYnomial for a polynomial curve fit or SRATional for a simple rational polynomial curve fit, followed by a positive integer <n>, which is the curve order.

The additional parameters are the curve coefficients. These are listed starting with the 0th order term. Note that the number of coefficients depends on the order of the curve - there are exactly n+1 coefficients for a curve of order n. For example:

```
:CALibration:LINearize:CURVe:TYPE  
POLY3,-0.117,-7.43,1.02,-0.034
```

defines a calibration curve of type 3rd order polynomial and the four coefficients used.

As a query, TYPE returns latest new pending curve fit type and calculated curve coefficients.

If, as in the above example,

$$f(x) = -0.117 - 7.43*x + 1.02*x^2 - 0.034*x^3$$

results from the curve fit, the returned curve values shall be:

POLY3,-0.117,-7.43,1.02,-0.034

A third order simple rational polynomial

$$f(x) = \frac{x}{0.81 - 2.0*x + 1.17*x^2 + 0.02*x^3}$$

shall be represented by the parameters

SRAT3,0.81,-2.0,1.17,0.02

#### 5.4.2.3.2 :ZFORce <Boolean>

CALibration:LINEarize:CURVe:ZFORce

The setting of ZFORce (Zero FORce) determines whether or not a newly calculated curve will be fit such that the scalar polynomial term is 0. This affects the :CALibration:LINEarize:CALCulate command.

At \*RST, ZFORce is not changed.

#### 5.4.2.4 :VERify

CALibration:LINEarize:VERify

VERify causes the instrument to check the validity of a set of linearization coefficients by checking the difference from the concentration measured with the curve to the expected concentration of the verification gas.

This command is an event, and therefore has no \*RST value associated with it.

#### 5.4.2.4.1 :ACQuire

CALibration:LINEarize:VERify:ACQuire

ACQuire causes the flow and stabilization of a verification gas to occur. First, verification gas flow is enabled, next a stabilized reading is taken, and finally the verification gas flow is disabled. The concentration determined from this procedure is compared against a known concentration stored in a designated table. If the difference between the expected concentration of the verification gas and the concentration measured using the curve coefficients is greater than the tolerance value set by :CALibration:LINEarize:VERify:TOL, then the appropriate questionable status bit is set.

This command is an event, and therefore has no \*RST value associated with it.

#### 5.4.2.4.2 :TOLerance <numeric\_value>

CALibration:LINEarize:VERify:TOLerance

TOLerance sets the allowable tolerance between the expected concentration of the verification gas and the concentration measured using the curve coefficients in % of full scale of the active range of the selected analyzer.

## 1999 SCPI Instrument Class Applications

At \*RST, TOLerance is not changed.

### 5.4.2.4.3 :TYPE

CALibration:LINearize:VERify:TYPE

TYPE selects the linearization coefficients to be verified. Possible types are:

- CCURve - The curve coefficients currently in effect
- NCURve - A newly calculated or downloaded set of coefficients that are not yet in use.

At \*RST, TYPE is set to CCURve.

## 5.5 DIAGnostic Subsystem (Bench Commands)

KEYWORD	PARAMETER FORM	NOTES
:DIAGnostic		
:HUP		
:ACQuire		[no query]
:CALCulate		[no query]
:LEAK		
:ACQuire		[no query]
:CALCulate		[no query]
:NEFFiciency		
:ACQuire		[no query]
:CALCulate		[no query]

### 5.5.1 :HUP

DIAGnostic:HUP

Commands to perform a diesel HC HangUP check.

#### 5.5.1.1 :ACQuire

DIAGnostic:HUP:ACQuire

This procedure performs the procedure found the 1996 Code of Federal Regulations, 86.1340-90 Exhaust Sample Analysis, Paragraph (e) (3); it first takes a zero direct reading;, then a zero overflow reading. This command has no query form.

This command is an event, and therefore has no \*RST value associated with it.

#### 5.5.1.2 :CALCulate

DIAGnostic:HUP:CALCulate

Compare the direct zero value with the overflow zero value. These values and their difference are stored in memory table HUPResults. The difference is compared with the value in memory table HUPTolerance. Error bit 4 of :STATus:QUESTIONable:INSTrument:ISUMmary5:BIT10 is set if the value of the difference is outside of the tolerance values stored in HUPTolerance. These tolerance values are upper and lower tolerance in ppm.

This command is an event, and therefore has no \*RST value associated with it.

**5.5.2 :LEAK**

DIAGnostic:LEAK

Commands to help perform a bench leak check.

**5.5.2.1 :ACQuire**

DIAGnostic:LEAK:ACQuire

Initiate a bench leak check. This causes the bench perform a device-dependent leak check procedure. This command has no query form.

This command is an event, and therefore has no \*RST value associated with it.

**5.5.2.2 :CALCulate**

DIAGnostic:LEAK:CALCulate

This command performs a device-dependent leak check calculation to indicate success or failure of the leak check.

Error bit 4 of :STATus:QUEStionable:INSTRument:ISUMmary5:BIT10 is set if any of the calculated values are outside of these tolerance values. The lower and upper tolerance values are stored in table LTOLerance; the value which is compared against these tolerances is stored in table LREsults. This command has no query form.

This command is an event, and therefore has no \*RST value associated with it.

**5.5.3 :NEFFiciency**

DIAGnostic:NEFFiciency

Commands to initiate the NOx efficiency diagnostic procedure and its calculations.

**5.5.3.1 :ACQuire**

DIAGnostic:NEFFiciency:ACQuire

Initiate the NOx efficiency procedure. This is a six-step test and is performed according to the EPA's 1996 code of Federal Regulations, section 86.123-78.

This command is an event, and therefore has no \*RST value associated with it.

**5.5.3.2 :CALCulate**

:DIAGnostic:NEFFiciency:CALCulate

Calculate the NOx efficiency results, per the EPA's 1996 code of Federal Regulations, section 86.123-78. This command has no query form. The results of the calculations are placed in the memory table NEResults. Error bit 4 of :STATus:QUEStionable:INSTRument:ISUMmary5:BIT10 is set if any of the calculated values are outside of the tolerance values resident in the memory table NETolerance.

This command is an event, and therefore has no \*RST value associated with it.

## **1999 SCPI Instrument Class Applications**

## 6 Emission Test Cell

An Emission Test Cell consists of one or more computers and a set of instruments that are used to perform emissions testing. The instruments which are included in an ETCELL may include Emission Benches, Dynamometers, Sampling Systems, and other emissions measurement or other devices. In order to standardize the interfaces within the cell it is desirable that all instrument implement a core set of commands. These commands are described in this chapter.

All SCPI compliant Emission Test Cells shall implement the commands listed in Syntax & Style sections 4.2.

### 6.1 Base Functionality

This section describes the base functionality for an Emission Test Cell.

The <bf\_keyword> for an Emissions Test Cell is ETCELL.

#### 6.1.1 Base Measurement Instructions

The base functionality for an ETCELL contains no commands from the MEASure subsystem.

#### 6.1.2 Base Device-oriented Functions

ETCELL compliant instruments shall include SYSTem commands, which set up and cause communication between instruments and the site computer and the set up and transfer of data from instruments.

##### 6.1.2.1 SYSTem subsystem

The SYSTem subsystem of an Emission Test Cell shall include the following commands.

KEYWORD	PARAMETER FORM	SCPI Reference
SYSTem		V2-21
:CAPability?		V2-21.3
:COMMunicate		V2-21.4
:SOCKet<n>		V2-21.4.4
:ADDRess <string>(Ipaddress   Host Name)		V2-21.4.4.1
:CONNect		V2-21.4.4.2
:DISConnect		V2-21.4.4.3
:FEED <data_handle>{,<data_handle>}		V2-21.4.4.4
:OCONDition <event_handle>		V2-21.4.4.4.1
:SCONDition <event_handle>		V2-21.4.4.4.2
:LISTen		V2-21.4.4.5
:PORT <NRf> <non-decimal numeric>		V2-21.4.4.6
:TYPE TCP   UDP		V2-21.4.4.7
:DATE <year>,<month>,<day>		V2-21.7
:ERRor		V2-21.8
:ALL?		V2-21.8.4
:CODE?		V2-21.8.5
:ALL?		V2-21.8.5.1
:[NEXT]?		V2-21.8.5.2

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
:ENABle		V2-21.8.7
:ADD	<numeric_list>	V2-21.8.7.1
:DELetE	<numeric_list>	V2-21.8.7.2
[:LIST]	<numeric_list>	V2-21.8.7.3
[:NEXT]?		V2-21.8.8
:LOCK		V2-21.14
:OWNer?		V2-21.14.1
:RELEASE		V2-21.14.2
:REQuest?		V2-21.14.3
:TIME		V2-21.19
:TImEr		V2-21.19.1
:COUNt	<numeric_value>	V2-21.19.1.1
[:STATe]	<Boolean>	V2-21.19.1.2
:VERSion?		V2-21.21

### 6.1.2.2 TRIGger subsystem

TRIGger commands for ETCELLs are as follows:

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	
:TRIGger		V2-24.8
[:SEQUence]		V2-24.8.1
:ECOunt	<numeric_value>	V2-24.8.1.7
:LINK	<event_handle>	V2-24.8.1.13
:SOURce	TIMer	V2-24.8.1.17
:TImEr	<numeric_value>	V2-24.8.1.18

## 6.2 Additional Functionality

Not applicable.

## 6.3 Programming Examples

This section discusses how a user might use commands in an Emission Test Cell

### 6.3.1 Instrument Capability and Version Example

:SYSTem:CAPability?	A chassis dynamometer for an emissions test cell returns: (ETCELL & CDYNO)<NL>
:SYSTem:VERSion?	A chassis dynamometer might return: 1999.0<NL>

### 6.3.2 Command Channel Example

Assume the Site computer wants to configure an Emission Test Cell instrument to perform a reading and return the data. The following commands might be issued to the instrument to perform this reading.

#### Commands

```

SENSe:FUNCtion:CONCurrent ON      Configure to sense data from
                                    more than one analyzer.

SENSe:FUNCtion "CONC1:TAL", "CONC2:TAL", "CONC3:TAL", "CONC4:TAL"
                  Select the required data.

SENSe:CONCntration:RANGE:AUTO ON
                  Set instrument to automatic range mode.

INITiate          Start the measurement

SENSe:DATA?        Return data values for CONC1:TAL etc

```

By default, the result is returned in ASCII. The FORMat subsystem can be used to configure the instrument to send the data in various binary formats.

### 6.3.3 Two Channel Example

This example illustrates how to configure the instrument to send continuously acquired data on a separate channel from the SCPI command channel. This allows results to be continuously streamed back from the instrument to the controller.

Assume the controller wants the instrument to take data every 0.1 second, average the data over one second, buffer it then feed it out a second channel every 5 seconds.

```

Setup Sense functions to get modal data averaged over a second
SENSe:FUNCtion:CONCurrent ON
SENSe:FUNCtion "CONC1 ON AVERage", "CONC2 ON AVERage",
                "CONC3 ON AVERage", "CONC4 ON AVERage"
Setup averaging parameter for the modal data
SENSe:AVERage:COUNT 10
SENSe:AVERage:TCONTrol REPeat
SENSe:AVERage:TYPE SCALAR
Turn the Averaging ON inside of the Sense block
SENSe:AVERage:STATe ON
Trigger Sequence will setup rates to sense the modal data
TRIGger:SEQUence1:SOURce TIMER
TRIGger:SEQUence1:TIMER 0.1      set sampling rate of 0.1sec
TRIGger:SEQUence2:LINK "TRIGger:SEQUence1"
TRIGger:SEQUence2:ECount 10      set integration rate
                                10 times 0.1=1sec
TRIGger:SEQUence3:LINK "TRIGger:SEQUence1"
TRIGger:SEQUence3:ECount 50      set transmission rate
                                50 times 0.1=5sec
Setup and connect on the communication socket
SYStem:COMMUnicatE:SOCKET:ADDReSS "123.45.67.89"
SYStem:COMMUnicatE:SOCKET:PORT 701
SYStem:COMMUnicatE:SOCKET:TYPE TCP

```

## 1999 SCPI Instrument Class Applications

```
SYStem:COMMunicate:SOCKet:CONNect  
Setup the Socket feed to data channel  
SYStem:COMMunicate:SOCKet:FEED "SYSTem:TIME:TEST", "CONC1:TAL  
ON AVERage".....  
SYStem:COMMunicate:SOCKet:OCONDition "TRIGger:SEQuence2"  
collects data into the socket every 1 sec  
SYStem:COMMunicate:SOCKet:SCONDition "TRIGger:SEQuence3"  
sends data out the socket every 5 sec
```

6

Note: you can feed the data from two logical instruments into the same socket. Here for example, RT analyzer data and RE analyzer data can both be sent on SOCKet.

```
Initiate all sequences for continuous sampling  
:INITiate:CONTinuous ON  
The data is sent every 5 seconds on the data channel.
```

### 6.3.4 Returning Data using a Table Example

Assuming that the table MYTABLE has already been defined with items FORCe, SPEed, and TIME:

MEMORY:TABLE:SElect MYTABLE	Select the table containing the data.
MEMORY:DATA? SPEed	Retrieve the speed data from the table.

### 6.3.5 Time Example

Standard SCPI time commands shall be used to synchronize each of the instrument clocks to the Site computer clock. This synchronization should be done during a period in the test when the instruments are in a quiescent state.

```
SYSTem:TIME 07,45,00.010
```

This command is used to control internal timer/counters in Emission Test Cell instruments. The Timer is used to timestamp data returned by Emission Test Cell instruments and to track time during an emission test. Set or reset the instrument timer to 0 at the beginning of a test.

```
SYSTem:TIME:TIMer:COUNT 0  
Start the instrument timer.  
SYSTem:TIME:TIMer:STATE ON  
Query the value of the instrument timer.  
SYSTem:TIME:TIMer:COUNT?
```

## 7 Power Supplies

A power supply is a basic sourcing instrument. It typically supplies a constant voltage or current at significant power levels to energize an electrical circuit. Because a power supply is primarily a source, the SOURce root node is optional.

All SCPI compliant products implement the commands listed in Syntax & Style section 4.2.

Figure 7-1 shows an instrument model for a power supply. It is derived from Command Reference, Chapter 2. It is essentially the same as Figure 2-1 in Command Reference with various parts shaded or dashed. The shaded boxes are not described here. While a power supply may have a MEMory subsystem, its operation is not described. The TRIGger block is dashed because not all power supplies have that functionality. While a power supply contains FORMat and Signal Routing functionality, their behaviors generally follow the \*RST setting. These blocks are not described so they are dashed.

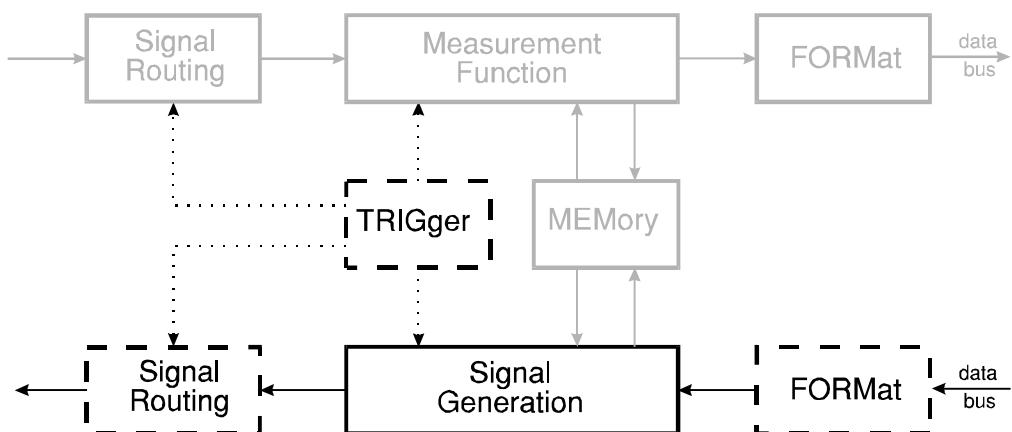


Figure 7-1 Simplified Model for a Power Supply

Figure 7-2 shows how the “Signal Generation” block in Figure 7-1 is expanded in a power supply. It is the same as Figure 2-5 in Command Reference with some shaded and dashed boxes. The **MEMory** block is shaded because while a power supply may contain this function, this chapter does not address it. The **CALCulate** block is dashed and its behavior not described. It behaves according to the \*RST value.

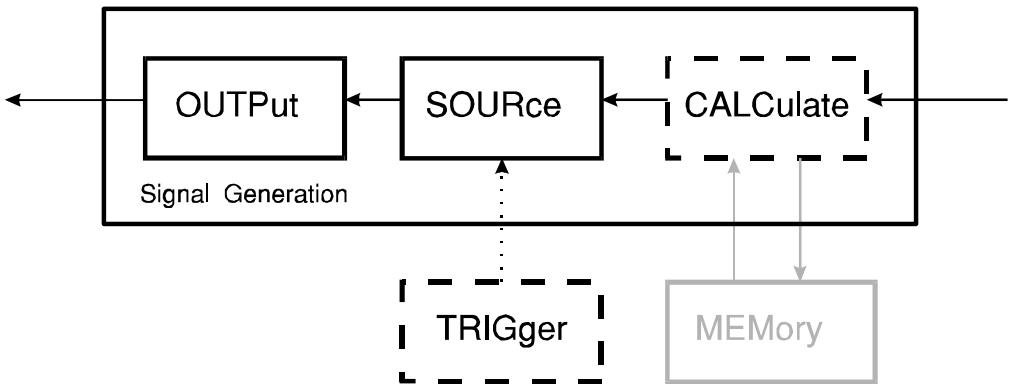


Figure 7-2 Expanded Signal Generation Model

## 7.1 Base Functionality

This section describes the base functionality for power supplies.

The <bf\_keyword> for a DC power supply is DCPSUPPLY.

### 7.1.1 Base Measurement Instructions

The base functionality for a power supply contains no measurement instructions.

### 7.1.2 Base Device-oriented Functions

#### 7.1.2.1 Outputs

A power supply shall have at least one output which can be turned on and off. This OUTPut subsystem command shall be implemented.

KEYWORD	PARAMETER FORM	SCPI Reference
OUTPut		Vol 2-15
[:STATe]	<Boolean>	Vol 2-15.12

As stated in Command Reference, \*RST sets the output state to off.

#### 7.1.2.2 SOURce subsystem

A power supply shall provide control of voltage and current by implementing these commands.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
[SOURce]		Vol 2-19
:CURRent		Vol 2-19.5
[:LEVel]		Vol 2-19.5.4
[:IMMediate]		Vol 2-19.5.4.1
[:AMPLitude] <numeric_value>		Vol 2-19.5.4.1.1
:VOLTage		Vol 2-19.23
[:LEVel]		Vol 2-19.23.4
[:IMMediate]		Vol 2-19.23.4.1
[:AMPLitude] <numeric_value>		Vol 2-19.23.4.1.1

A power supply can operate in one of two modes: voltage source or current source. The mode of operation depends on the load connected to it. When a power supply is operating as a voltage source, the :VOLTage command sets the output voltage and the :CURRent command sets the current limit. When a power supply is operating as a current source, the :VOLTage command sets the voltage limit and the :CURRent command sets the output current.

If the :CURRent and :VOLTage commands are implemented as overlapped commands, the associated Pending-Operation flags shall be reported in the No-Operation-Pending flag. The operation started by the :CURRent command completes when the output current has reached the programmed current value or when the voltage limit has been reached. Likewise, the operation started by the :VOLTage command completes when the output voltage has reached the programmed voltage value or the current limit has been reached.

At \*RST a power supply is required to set the voltage and current to “safe” conditions. This is generally achieved by setting them to their values closest to zero.

### 7.1.3 Base Status Reporting

All SCPI power supplies shall implement the status reporting structure described in Syntax & Style, Status Reporting. Command Reference, STATus Subsystem defines the commands which shall be used to control the status reporting structure.

#### 7.1.3.1 QUESTIONable

For a power supply, the bits of interest in the QUESTIONable status structure are VOLTage and CURRent. When a power supply is operating as a voltage source, bit 1 (CURRent) shall be set. When a power supply is operating as a current source, bit 0 (VOLTage) shall be set. When the output is unregulated, both bits shall be set (for example, while the output is changing to a new programmed value).

#### 7.1.3.2 OPERATION

The base functionality of a power supply adds no requirements for this register.

## 7.2 Additional Functionality

This section describes additional functionality for power supplies.

### 7.2.1 Measurement capability

While the basic function of a power supply is to source voltage and current, this additional functionality provides for the measurement of the actual voltage and current delivered to the load.

The <af\_keyword> for measurement capability is MEASURE.

#### 7.2.1.1 MEASURE Measurement Instructions

The following MEASure:<function>? queries shall be implemented.

KEYWORD	PARAMETER FORM
---------	----------------

MEASure

[:SCALar]

:VOLTage[:DC]? [<expected\_value>[,<resolution>]]

:CURRent[:DC]? [<expected\_value>[,<resolution>]]

7

These queries read the actual voltage and or current at the power supply's output. They control a sensing function that is completely separate from the power supply itself.

Executing these queries shall have no effect on the power supply's settings. This functionality further require no additional Measurement Instruction commands and has no interaction with the power supply's triggering system, if implemented.

The <expected\_value> and <resolution> parameters are included solely for compatibility with the Command Reference. While parameters shall be accepted for syntactic compatibility, they are not required by MEASURE and may be ignored by the device.

#### 7.2.1.2 MEASURE Device-oriented Functions

MEASURE adds no device-oriented functions.

#### 7.2.1.3 MEASURE Status Reporting

MEASURE adds no Status Reporting requirements.

### 7.2.2 Multiple supplies

A power supply may contain multiple independent power supplies.

The <af\_keyword> for multiple supplies is MULTIPLE.

#### 7.2.2.1 MULTIPLE Measurement Instructions

MULTIPLE adds no measurement instructions.

#### 7.2.2.2 MULTIPLE Device-oriented Functions

An instrument which contains multiple power supplies shall use the INSTRument:NSELect and INSTRument[:SELect] commands to switch among them.

## 1999 SCPI Instrument Classes

KEYWORD	PARAMETER FORM	SCPI Reference
:INSTrument		Vol 2-12
:NSElect	<numeric_value>	Vol 2-12.5
[:SElect]	<identifier>	Vol 2-12.6

Valid values of the <numeric\_value> shall range from 1 to the number of power supplies.

### 7.2.2.3 MULTIPLE Status Reporting

Section 9.5 of Syntax & Style describes the implications of multiple instruments on the status reporting model. The INSTRument Summary bits in the QUESTionable and OPERation registers shall be used to summarize the status from all logical instruments. The following STATus subsystem commands are required.

KEYWORD	PARAMETER FORM	SCPI Reference
STATUs		Vol 2-20
:OPERation		Vol 2-20.1
:INSTRument		
:CONDition?		Vol 2-20.1.2
:ENABLE	<NRf>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:ISUMmary<n>		
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRf>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4
:QUESTionable		Vol 2-20.3
:INSTRument		
:CONDition?		Vol 2-20.3.2
:ENABLE	<NRf>	Vol 2-20.3.3
[:EVENT]?		Vol 2-20.3.4
:ISUMmary<n>		
:CONDITION?		Vol 2-20.1.2
:ENABLE	<NRf>	Vol 2-20.1.3
[:EVENT]?		Vol 2-20.1.4

7

### 7.2.3 Triggering Capability

Some power supplies are able to change their voltage and current based on the arrival of a command or external signal. Trigger commands, used in conjunction with source commands, control when the output changes.

The <af\_keyword> for trigger capability is TRIGGER.

#### 7.2.3.1 TRIGGER Measurement Instructions

TRIGGER adds no measurement instructions.

#### 7.2.3.2 TRIGGER Device-oriented Functions

Power supplies which have TRIGGER additional functionality shall implement the following SOURce and TRIGger commands.

## 1999 SCPI Instrument Classes

KEYWORD	PARAMETER FORM	SCPI Reference
INITiate		Vol 2-24.7
:CONTinuous	<Boolean>	Vol 2-24.7.1
[:IMMEDIATE]		Vol 2-24.7.2
[:ALL]		Vol 2-24.7.2.1
[SOURce]		Vol 2-19
:CURREnt		Vol 2-19.5
[:LEVel]		Vol 2-19.5.4
:TRIGgered		Vol 2-19.5.4.2
[:AMPLitude] <numeric_value>		Vol 2-19.5.4.2.1
:VOLTage		Vol 2-19.23
[:LEVel]		Vol 2-19.23.4
:TRIGgered		Vol 2-19.23.4.2
[:AMPLitude] <numeric_value>		Vol 2-19.23.4.2.1
TRIGger		Vol 2-24.8
[:SEQunce]		Vol 2-24.8.1
[:IMMEDIATE]		Vol 2-24.8.1.11
:SOURce	{BUS   EXTERNAL   ...}	Vol 2-24.8.1.17

7

The trigger commands associated with the TRIGGER additional functionality shall have no interaction with the MEASURE additional functionality.

Requiring BUS as a parameter to TRIGger:SOURce has the side effect of requiring DT1 capability in an IEEE 488.1 device. It also means the power supply implements the \*TRG common command.

A VXIbus device with a trigger function is required to implement the *Trigger* command. When TRIGger:SOURce is set to BUS, a VXIbus power supply uses the *Trigger* command as the TRIGger source.

When TRIGger:SOURce is set to IMMEDIATE, an INITiate command immediately transfers the TRIGgered[:AMPLitude] value to [:IMMEDIATE][:AMPLitude].

### 7.2.3.3 TRIGGER Status Reporting

Bit 5 (Waiting for TRIG) in the OPERation Status Register shall be used to indicate the status of the trigger model.

## 7.3 Programming Examples

This section discusses how a user might program a power supply for certain applications.

### 7.3.1 Simple

To use a power supply as a constant voltage supply the programmer could send:

```
*RST
SOURce:VOLTage:LEVel:IMMEDIATE:AMPLitude 5V
SOURce:CURREnt:LEVel:IMMEDIATE:AMPLitude MAXimum
OUTPut:STATE ON
```

or

```
*RST;VOLT 5V;CURR MAX;OUTP ON
```

### 7.3.2 Time Critical

With a power supply containing the TRIGGER additional functionality, an application could change the voltage at a specific time by sending:

```
*RST
SOURce:CURREnt:LEVel:IMMEDIATE:AMPLitude MAXimum
OUTPut:STATE ON
TRIGger:SEQuence:SOURce EXTERNAL
SOURce:VOLTage:LEVel:IMMEDIATE:AMPLitude 7.2V
SOURce:VOLTage:LEVel:TRIGgered:AMPLitude 9.6V
INITiate:IMMEDIATE
```

or

```
*RST;CURR MAX;OUTP ON;TRIG:SOUR EXT
VOLT:IMM 7.2V;TRIG 9.6V
INIT
```

The power supply sources 7.2 volts immediately and waits for a signal on the external trigger. When the trigger event occurs, the output voltage changes to 9.6 volts.

### 7.3.3 Level Verification

With a power supply containing the MEASURE additional functionality, an application could verify the level by sending:

```
*RST
OUTPut:STATE ON
SOURce:CURREnt:LEVel:IMMEDIATE:AMPLitude MAXimum
SOURce:VOLTage:LEVel:IMMEDIATE:AMPLitude 7.2V
*OPC?
```

or

```
*RST
OUTP ON;CURR MAX;VOLT 7.2;*OPC?
```

The application reads the ‘1’ returned by the supply when it completes the operation of changing the level.

## 1999 SCPI Instrument Classes

Now the actual voltage and current supplied to the load are measured with:

```
:MEASure:VOLTage:DC? ; :MEASure:CURREnt:DC?
```

or

```
MEAS:VOLT? ; CURR?
```

The supply returns the measured voltage and current, which the application can then compare with expected results.

### 7.3.4 Multiple Supplies

A device containing two power supplies sets supply #1 to 5 Volts and supply #2 to -5 volts with the following commands:

```
*RST  
INSTRument:NSELect 1  
SOURce:VOLTage:LEVel:IMMediate:AMPLitude 5V  
OUTPut:STATE ON  
INSTRument:NSELect 2  
SOURce:VOLTage:LEVel:IMMediate:AMPLitude -5V  
OUTPut:STATE ON
```

or

```
*RST  
INST:NSEL 1;VOLT 5;OUTP ON  
INST:NSEL 2;VOLT -5;OUTP ON
```

To check that both supplies are acting as voltage sources and are not current limited, the QUESTIONable Condition Register is queried using :

```
STATus:QUESTIONable:CONDITION?
```

If bit 0 is 0 and bit 1 is 1, both supplies are in the voltage source mode.

To check each supplies separately, the application subsequently reads the regulation status with these commands sent in the same program message:

```
STATus:QUESTIONable:INSTRument:ISUMmary1:CONDITION?;  
:STATus:QUESTIONable:INSTRument:ISUMmary2:CONDITION?
```

or

```
STAT:QUES:INST:ISUM1:COND? ; :STAT:QUES:INST:ISUM2:COND?
```

A possible response would be 2;3 indicating the CURREnt bit but not the VOLTage bit has been set on supply #1 (current limited, voltage regulated) and both bits set on supply #2 (unregulated).

## 8 RF & Microwave Sources

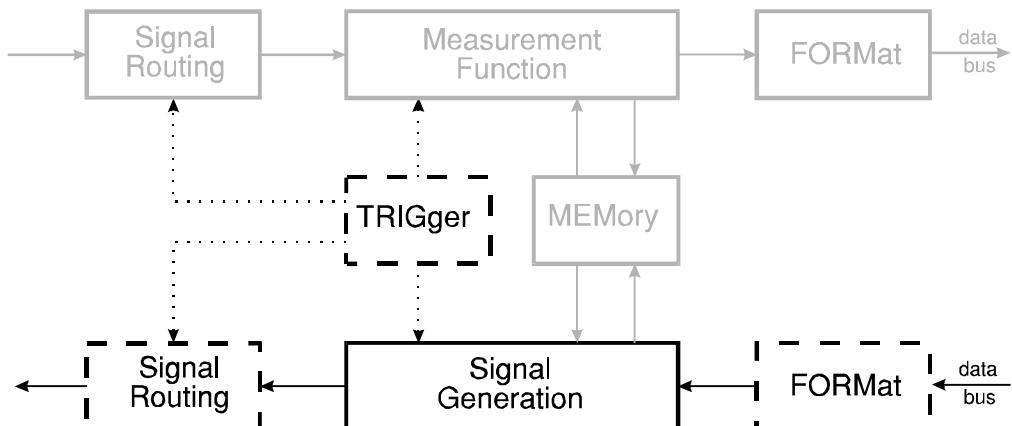
RF sources and microwave sources are basic sourcing instruments. They typically generate a single sinusoidal signal at one frequency and at constant output level. The frequency is typically in the range from above 10 MHz up to several GHz. Their output impedance is normally 50 Ohms and matched impedance is assumed. The output level is specified as power into a matched load rather than as voltage. Signal generators with modulation and sweepers are also covered by this chapter. Their special attributes are described as additional functionalities.

Excluded from this chapter are optical sources, function generators, arbitrary waveform generators, pulse generators and data pattern generators.

All SCPI compliant products implement the commands listed in Syntax & Style section 4.2.

Figure 8-1 shows an instrument model for RF & microwave sources. It is derived from Command Reference, chapter 2. It is essentially the same as Figure 2-1 in Command Reference with various parts shaded or dashed. The bold boxes and lines, solid and dashed are described in this chapter.

The FORMat and ROUTe blocks are shown with dashed lines because many RF & microwave sources use their default settings. \*RST sets FORMat[:DATA] to ASCII and ROUTe:TERMinals to FRONt. The shaded boxes are not described here.



**Figure 8-1 Simplified Model of an RF & Microwave Source**

Figure 8-2 shows how the “Signal Generation” block in Figure 8-1 is expanded in RF & microwave sources. It is the same as Figure 2-5 in Command Reference with some dashed and shaded boxes. The CALCulate block is dashed because after \*RST this block has no effect on the data. This chapter does not describe any CALCulate functions. The MEMORY block is shaded because while RF & microwave sources may contain this function, this chapter does not address it.

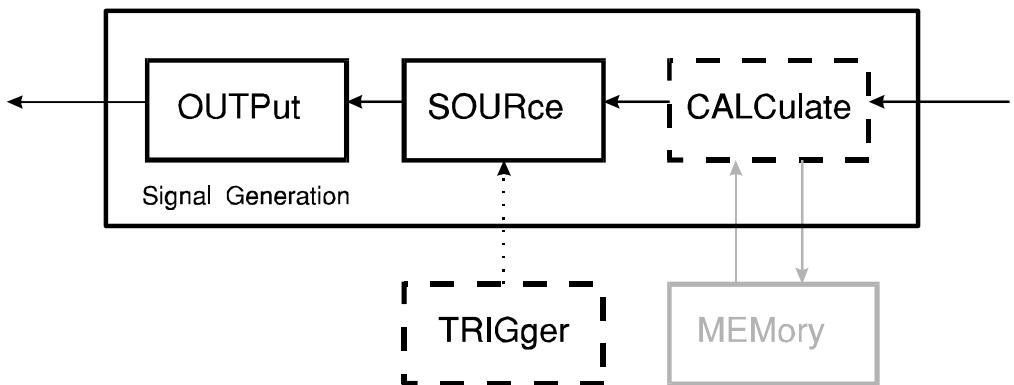


Figure 8-2 Expanded Signal Generation Model

## 8.1 Base Functionality

This section describes the base functionality for RF & microwave signal sources.

The <bf\_keyword> for RF & microwave sources is RFSOURCE.

### 8.1.1 Base Measurement Instructions

The base functionality for RF & microwave sources contains no measurement instructions.

### 8.1.2 Base Device-oriented Functions

#### 8.1.2.1 SOURce subsystem

RF & microwave sources shall provide control of its output frequency and level by implementing these commands.

KEYWORD	PARAMETER FORM	SCPI Reference
[SOURce]		Vol 2-19
:FREQuency		Vol 2-19.9
[:CW :FIXed]	<numeric_value>	Vol 2-19.9.2
:POWER		Vol 2-19.15
[:ALC]		Vol 2-19.15.2
[:STATE]	<Boolean>	Vol 2-19.15.2.1
[:LEVEL]		Vol 2-19.15.4
[:IMMediate]		Vol 2-19.15.4.1
[:AMPLitude]	<numeric_value>	Vol 2-19.15.4.1.1

If the :FREQuency and :POWER commands are implemented as overlapped commands, the associated Pending-Operation flags shall be reported in the No-Operation-Pending flag. The operation started by the :FREQuency command completes when the frequency has reached

## 1999 SCPI Instrument Classes

the programmed frequency value. Likewise, the operation started by the :POWER command completes when the output level has reached the programmed power value.

At \*RST a RF & microwave source should be set to a “safe” condition. This is generally achieved by setting the amplitude to its minimum value in conjunction with \*RST setting OUTPut:STATE to OFF.

### 8.1.2.2 OUTPut Subsystem

RF & microwave sources that have at least one output which can be turned on and off and shall implement the following command.

KEYWORD	PARAMETER FORM	SCPI Reference
OUTPut		Vol 2-15
[:STATE]	<Boolean>	Vol 2-15.12

As stated in Command Reference, \*RST sets the output state to off.

### 8.1.2.3 UNIT Subsystem

RF & microwave sources shall provide control of its UNITS for the output power by implementing this command.

KEYWORD	PARAMETER FORM	SCPI Reference
UNIT		Vol 2-25
:POWER	W V DBM DB<suffix mult.>W ...	Vol 2-25.2

At \*RST, this value shall be set to DBM.

### 8.1.3 Base Status Reporting

All SCPI RF & microwave sources shall implement the status reporting structure described in Syntax & Style, Status Reporting. Command Reference, STATus Subsystem defines the commands which shall be used to control the status reporting structure.

#### 8.1.3.1 QUESTIONable Status

RF & microwave sources shall set bit 3 (POWER) or bit 5 (FREQuency) in the QUESTIONable Status Register when the RF & microwave source suspects the power or the frequency is of questionable accuracy.

#### 8.1.3.2 OPERation Status

The base functionality of a RF & microwave source adds no requirements to the OPERation Status register.

## 8.2 Additional functionality

The basic function of a RF & microwave source is to generate a signal at fixed frequency and power. Typical signal generators and typical sweepers may include additional functionality as modulation, sweep of frequency or power, marker and trigger function and selection of the reference oscillator. This section describes this additional functionality for RF & microwave sources.

### 8.2.1 Amplitude Modulation

This additional functionality provides amplitude modulation of the RF output signal delivered to the load either by an internally generated modulation signal or by an external signal input.

The <af\_keyword> for amplitude modulation is AM.

#### 8.2.1.1 AM Measurement Instructions

AM adds no Measurement Instructions.

8

#### 8.2.1.2 AM Device Oriented Commands

RF & microwave sources which have the AM additional functionality shall provide control of amplitude modulation by implementing the following commands.

KEYWORD	PARAMETER FORM	SCPI Reference
[SOURce]		Vol 2-19
:AM		Vol 2-19.2
[:DEPTH]	<numeric_value>	Vol 2-19.2.2
:EXTernal		Vol 2-19.2.3
:COUpling	AC DC ...	Vol 2-19.2.3.1
:INTernal		Vol 2-19.2.4
:FREQuency	<numeric_value>	Vol 2-19.2.4.1
:SOURce	EXTernal INTernal	Vol 2-19.2.8
:STATE	<Boolean>	Vol 2-19.2.9

#### 8.2.1.3 AM Status Reporting

When the RF & microwave source suspects the additional function AM is of questionable accuracy, bit 7 (MODulation) shall be set in the QUESTionable Status Register.

### 8.2.2 Frequency Modulation

This additional functionality provides frequency modulation of the RF output signal delivered to the load either by an internally generated modulation signal or by an external signal input.

The <af\_keyword> for frequency modulation is FM.

#### 8.2.2.1 FM Measurement Instructions

FM adds no Measurement Instructions.

#### 8.2.2.2 FM Device Oriented Commands

RF & microwave sources which have the FM additional functionality shall provide control of frequency modulation by implementing the following commands.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
[SOURce]		Vol 2-19
:FM		Vol 2-19.7
[:DEViation]	<numeric_value>	Vol 2-19.7.2
[:EXTernal]		Vol 2-19.7.3
[:COUpling]	AC DC ...	Vol 2-19.7.3.1
[:INTernal]		Vol 2-19.7.4
[:FREQuency]	<numeric_value>	Vol 2-19.7.4.1
:SOURce	EXTernal INTernal	Vol 2-19.7.8
:STATe	<Boolean>	Vol 2-19.7.9

### 8.2.2.3 Status Reporting

When the RF & microwave source suspects the additional function FM is of questionable accuracy, bit 7 (MODulation) shall be set in the QUESTionable Status Register.

### 8.2.3 Pulse Modulation

This additional functionality provides pulse modulation of the RF output signal delivered to the load by an external modulation signal.

The <af\_keyword> for pulse modulation is PULM.

#### 8.2.3.1 PULM Measurement Instructions

PULM adds no Measurement Instructions.

#### 8.2.3.2 PULM Device Oriented Commands

RF & microwave sources which have the PULM additional functionality shall provide control of pulse modulation by implementing the following commands.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
[SOURce]		Vol 2-19
:PULM		Vol 2-19.16
[:POLarity]	NORMal INVerted	Vol 2-19.16.4
[:STATe]	<Boolean>	Vol 2-19.16.6

### 8.2.3.3 PULM Status Reporting

When the RF & microwave source suspects the additional functionality PULM is of questionable accuracy, bit 7 (MODulation) shall be set in the QUESTionable Status Register.

### 8.2.4 Analog Frequency Sweeping

RF & microwave sources may provide control of analog frequency sweep functions.

The <af\_keyword> for analog frequency sweeping is FASWEEP.

#### 8.2.4.1 FASWEEP Measurement Instructions

FASWEEP adds no Measurement Instructions.

#### 8.2.4.2 FASWEEP Device Oriented Commands

RF & microwave sources which have the FASWEEP additional functionality shall provide control of frequency sweep by implementing the following commands.

## 1999 SCPI Instrument Classes

KEYWORD	PARAMETER FORM	SCPI Reference
INITiate		Vol 2-24.7
:CONTinuous	<Boolean>	Vol 2-24.7.1
[:ALL]		Vol 2-24.7.1.1
[SOURce]		Vol 2-19
:FREQuency		Vol 2-19.9
:MODE	CW FIXed SWEep ...	Vol 2-19.9.4
:START	<numeric_value>	Vol 2-19.9.9
:STOP	<numeric_value>	Vol 2-19.9.10
:CENTer	<numeric_value>	Vol 2-19.9.1
:SPAN	<numeric_value>	Vol 2-19.9.8
:SWEep		Vol 2-19.21
:TIME	<numeric_value>	Vol 2-19.21.1
:AUTO	<Boolean> ONCE	Vol 2-19.21.1.1

8

See the beginning of the Command Reference, SOURce chapter for a description of the coupling among START, STOP, CENTER & SPAN.

Typically a RF & microwave source has just one single SWEEP source for frequency and power sweep. The setting of sweep parameters are then coupled and identical. If there is more than one sweep source they can be distinguished by use of a numeric suffix.

When the mode is changed from CW to SWEep, the instrument is not sweeping until INITiate:CONTinuous is set to ON.

### 8.2.4.3 FASWEEP Status Reporting

If the additional function Analog Frequency Sweep is implemented and the RF & microwave source is performing this operation, bit 3 (SWEEPing) shall be set in the OPERation Status Register.

### 8.2.5 Stepped Frequency Sweeping

RF & microwave sources may provide control of stepped frequency sweep functions.

The <af\_keyword> for stepped frequency sweeping is FSSWEEP.

#### 8.2.5.1 FSSWEEP Measurement Instructions

FSSWEEP adds no Measurement Instructions.

#### 8.2.5.2 FSSWEEP Device Oriented Commands

RF & microwave sources which have the FSSWEEP additional functionality shall provide control of frequency sweep by implementing the following commands.

KEYWORD	PARAMETER FORM	SCPI Reference
INITiate		Vol 2-24.7
:CONTinuous	<Boolean>	Vol 2-24.7.1
[:ALL]		Vol 2-24.7.1.1
[SOURce]		Vol 2-19
:FREQuency		Vol 2-19.9
:MODE	CW FIXed SWEep ...	Vol 2-19.9.4

## 1999 SCPI Instrument Classes

:STARt	<numeric_value>	Vol 2-19.9.9
:STOP	<numeric_value>	Vol 2-19.9.10
:CENTer	<numeric_value>	Vol 2-19.9.1
:SPAN	<numeric_value>	Vol 2-19.9.8
:SWEep		Vol 2-19.21
:DWELL	<numeric_value>	Vol 2-19.21.2
:STEP	<numeric_value>	Vol 2-19.21.7

See the beginning of the Command Reference, SOURce chapter for a description of the coupling among STARt, STOP, CENTer & SPAN.

Typically a RF & microwave source has just one single SWEEP source for frequency and power sweep. The setting of sweep parameters are then coupled and identical. If there is more than one sweep source they can be distinguished by use of a numeric suffix.

When the mode is changed from CW to SWEep, the instrument is not sweeping until INITiate:CONTinuous is set to ON.

### 8.2.5.3 FSSWEEP Status Reporting

If the additional function Stepped Frequency Sweep is implemented and the RF & microwave source is performing this operation, bit 3 (SWEeping) shall be set in the OPERation Status Register.

### 8.2.6 Analog Power Sweeping

RF & microwave sources may provide control of analog power sweep functions.

The <af\_keyword> for analog power sweeping is PASWEEP.

#### 8.2.6.1 PASWEEP Measurement Instructions

PASWEEP adds no Measurement Instructions.

#### 8.2.6.2 PASWEEP Device Oriented Commands

RF & microwave sources which have the PASWEEP additional functionality shall provide control of analog power sweep by implementing the following commands.

KEYWORD	PARAMETER FORM	SCPI Reference
INITiate		Vol 2-24.7
:CONTinuous	<Boolean>	Vol 2-24.7.1
[:ALL]		Vol 2-24.7.1.1
[SOURce]		Vol 2-19
:POWER		Vol 2-19.15
:MODE	FIXed SWEep	Vol 2-19.15.7
:STARt	<numeric value>	Vol 2-19.15.13
:STOP	<numeric value>	Vol 2-19.15.14
:CENTer	<numeric value>	Vol 2-19.15.3
:SPAN	<numeric value>	Vol 2-19.15.12
:SWEep		Vol 2-19.21
:TIME	<numeric value>	Vol 2-19.21.1
:AUTO	<Boolean> ONCE	Vol 2-19.21.1.1

See the beginning of the Command Reference, SOURce chapter for a description of the coupling among STARt, STOP, CENTER & SPAN.

Typically a RF & microwave source has just one single SWEEP source for frequency and power sweep. The setting of sweep parameters are then coupled and identical. If there is more than one sweep source they can be distinguished by use of a numeric suffix.

When the mode is changed from CW to SWEep, the instrument is not sweeping until INITiate:CONTinuous is set to ON.

### 8.2.6.3 PASWEEP Status Reporting

If the additional function Analog Power Sweep is implemented and the RF & microwave source is performing this operation, bit 3 (SWEEping) shall be set in the OPERation Status Register.

### 8.2.7 Stepped Power Sweeping

RF & microwave sources may provide control of stepped power sweep functions.

The <af\_keyword> for stepped power sweeping is PSSWEEP.

#### 8.2.7.1 PSSWEEP Measurement Instructions

PSSWEEP adds no Measurement Instructions.

#### 8.2.7.2 PSSWEEP Device Oriented Commands

RF & microwave sources which have the PSSWEEP additional functionality shall provide control of stepped power sweep by implementing the following commands.

KEYWORD	PARAMETER FORM	SCPI Reference
INITiate		Vol 2-24.7
:CONTinuous	<Boolean>	Vol 2-24.7.1
[:ALL]		Vol 2-24.7.1.1
[SOURce]		Vol 2-19
:POWER		Vol 2-19.15
:MODE	FIXed SWEEp	Vol 2-19.15.7
:STARt	<numeric value>	Vol 2-19.15.13
:STOP	<numeric value>	Vol 2-19.15.14
:CENTer	<numeric value>	Vol 2-19.15.3
:SPAN	<numeric value>	Vol 2-19.15.12
:SWEep		Vol 2-19.21
:DWELl	<numeric_value>	Vol 2-19.21.2
:STEP	<numeric_value>	Vol 2-19.21.7

See the beginning of the Command Reference, SOURce chapter for a description of the coupling among STARt, STOP, CENTER & SPAN.

Typically a RF & microwave source has just one single SWEEP source for frequency and power sweep. The setting of sweep parameters are then coupled and identical. If there is more than one sweep source they can be distinguished by use of a numeric suffix.

When the mode is changed from CW to SWEep, the instrument is not sweeping until INITiate:CONTinuous is set to ON.

### 8.2.7.3 PSSWEEP Status Reporting

If the additional function Stepped Power Sweep is implemented and the signal generator is performing this operation, bit 3 (SWEeping) shall be set in the OPERation Status Register.

### 8.2.8 Frequency List

RF & microwave sources may provide control of additional frequency sweep functions by a frequency list containing the sequence of frequencies to be stepped through.

The <af\_keyword> for frequency lists is FLIST.

#### 8.2.8.1 FLIST Measurement Instructions

FLIST adds no Measurement Instructions.

#### 8.2.8.2 FLIST Device Oriented Commands

RF & microwave sources which have the FLIST additional functionality shall provide control of frequency list by implementing the following commands.

KEYWORD	PARAMETER FORM	SCPI Reference
INITiate		Vol 2-24.7
:CONTinuous	<Boolean>	Vol 2-24.7.1
[:ALL]		Vol 2-24.7.1.1
[SOURce]		Vol 2-19
:FREQuency		Vol 2-19.9
:MODE	CW FIXed LIST ...	Vol 2-19.9.4
:LIST		Vol 2-19.11
:DWELL	<numeric value>	Vol 2-19.11.8
:POINTS?	query only	Vol 2-19.11.8.1
:FREQuency	<numeric_value>{,<numeric_value>}	Vol 2-19.11.9
:POINTS?	query only	Vol 2-19.11.9.1

This functionality only requires one parameter for DWELL, however the instrument may accept more than one.

#### 8.2.8.3 FLIST Status Reporting

FLIST adds no requirement for status reporting.

### 8.2.9 Marker Function

RF & microwave sources may provide control of marker functions.

The <af\_keyword> for marker function is MARKER.

#### 8.2.9.1 MARKER Measurement Instructions

MARKER adds no Measurement Instructions.

#### 8.2.9.2 MARKER Device Oriented Commands

RF & microwave sources which have the MARKER additional functionality shall provide control of markers by implementing the following commands.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
[SOURce]		Vol 2-19
:MARKer<n>		Vol 2-19.12
:AOFF		Vol 2-19.12.2
:FREQUency	<numeric_value>	Vol 2-19.12.3
[:STATE]	<Boolean>	Vol 2-19.12.7

RF & microwave sources may have more than one Marker. In this case a numeric suffix shall be used to distinguish between the different markers.

### 8.2.9.3 MARKER Status Reporting

MARKER adds no requirement for status reporting.

8

### 8.2.10 Trigger Function

RF & microwave sources uses the TRIGger Subsystem to control the start of the sweep or list function if the frequency and/or the amplitude is swept. When a trigger is received, the RF & microwave source starts and completes a sweep or list sequence and then waits for the next trigger event.

The <af\_keyword> for trigger function is TRIGGER.

#### 8.2.10.1 TRIGGER Measurement Instructions

TRIGGER adds no Measurement Instructions.

#### 8.2.10.2 TRIGGER Device Oriented Commands

RF & microwave sources which have the TRIGGER additional functionality shall implement the following trigger commands.

<b>KEYWORD</b>	<b>PARAMETER FORM</b>	<b>SCPI Reference</b>
INITiate		Vol 2-24.7
:CONTinuous	<Boolean>	Vol 2-24.7.1
[:ALL]		Vol 2-24.7.1.1
[:IMMEDIATE]		Vol 2-24.7.2
[:ALL]		Vol 2-24.7.2.1
ABORT		Vol 2-24.5
TRIGger		Vol 2-24.8
[:SEQUENCE]		Vol 2-24.8.1
[:IMMEDIATE]		Vol 2-24.8.1.11
:SOURce	BUS   IMMEDIATE   ... †	Vol 2-24.8.1.17

† In addition to the TRIGger:SOURce parameters listed, a RF & microwave source with FSSWEEP, PSSWEEP or FLIST shall implement an external trigger, such as EXTERNAL, ECLTrg<n> or TTLTrg<n>.

Requiring BUS as a legal parameter to TRIGger:SOURce has the side effect of requiring DT1 capability in a 488.1 device. It also means the RF & microwave source implements the \*TRG common command.

A VXIbus device with a trigger function is required to implement the *Trigger* command. When TRIGger:SOURce is set BUS, a VXIbus RF & microwave source shall use the *Trigger* command as the TRIGger source.

### 8.2.10.3 TRIGGER Status Reporting

If the additional function TRIGGER is implemented and the RF & microwave source is performing this operation, bit 5 (Waiting for TRIGger) shall be set in the OPERation Status Register.

### 8.2.11 Reference Oscillator

RF & microwave sources may have the functionality to select which reference oscillator is in use and to define the frequency of the external reference oscillator.

The <af\_keyword> for reference oscillator function is REFERENCE.

#### 8.2.11.1 REFERENCE Measurement Instructions

REFERENCE adds no Measurement Instructions.

#### 8.2.11.2 REFERENCE Device Oriented Commands

RF & microwave sources which have the additional functionality to select the reference oscillator shall implement the following commands.

KEYWORD	PARAMETER FORM	SCPI Reference
[:SOURce]		Vol 2-19
:ROSCillator		Vol 2-19.19
:EXTernal		Vol 2-19.19.2
:FREQuency	<numeric_value>	Vol 2-19.19.2.1
[:SOURce]	INTernal EXTernal	Vol 2-19.19.3

#### 8.2.11.3 REFERENCE Status Reporting

RF & microwave sources shall set bit 5 (FREQuency) in the QUESTionable Status Register when there is no appropriate reference signal available and the RF & microwave source suspects the output frequency is of questionable accuracy.

## **8.3 Programming Examples**

This section discusses how a user might program RF & microwave sources for certain applications.

### **8.3.1 Simple**

To use RF & microwave sources as a simple signal source for fixed frequency and output level the programmer could send:

```
*RST  
SOURce:CW:FREQuency 123.45 MHZ  
SOURce:POWer:LEVel:IMMEDIATE:AMPLitude -27 DBM  
OUTput:STATE ON
```

or

```
*RST  
FREQ 123.45MHZ;POW -27DBM;OUTP ON
```

**8**

### **8.3.2 Modulation**

To use a RF & microwave source having the functionality to generate a frequency modulated signal by use of the internal modulation source the programmer could send:

```
*RST  
SOURce:CW:FREQuency 89.5 MHZ  
SOURce:FM:DEVIation 200 KHZ  
SOURce:FM:SOURce INTERNAL  
SOURce:FM:INTERNAL:FREQuency 1 KHZ  
SOURce:POWer:LEVel:IMMEDIATE:AMPLitude 100 UW  
OUTPut:STATE ON
```

or

```
*RST  
FREQ 89.5MHZ  
FM 200KHZ;FM:SOUR INT;INT:FREQ 1KHZ  
POW 100UW;OUTP ON
```

### **8.3.3 Analog Sweep**

To use a RF & microwave source having the functionality to sweep the frequency with setting of start/stop frequency and sweep time the programmer could send:

```
*RST  
SOURce:FREQuency:MODE SWEep  
SOURce:FREQuency:START 100 MHZ  
SOURce:FREQuency:STOP 200 MHZ  
SOURce:SWEep:TIME 100 MS  
SOURce:POWer:LEVel:IMMEDIATE:AMPLitude -27 DBM  
OUTPut:STATE ON  
INITiate:CONTinuous ON
```

or

```
*RST
FREQ:MODE SWE
FREQ:STAR 100MHZ;STOP 200MHZ
SWE:TIME 100MS
POW -27 DBM;OUTP ON
INIT:CONT ON
```

### 8.3.4 Triggered Analog Sweep

To use a RF & microwave source as a analog sweeper in triggered single sweep mode the programmer could send:

```
*RST
SOURce:FREQuency:MODE SWEep
SOURce:FREQuency:START 100 MHZ
SOURce:FREQuency:STOP 200 MHZ
SOURce:SWEep:TIME 100 MS
SOURce:POWER:LEVel:IMMEDIATE:AMPLitude -27 DBM
OUTPut:STATE ON
TRIGger:SOURce EXTERNAL
INITiate:IMMEDIATE
```

or

```
*RST
FREQ:MODE SWE
FREQ:STAR 100MHZ;STOP 200MHZ
SWE:TIME 100MS
POW -27 DBM;OUTP ON
TRIG:SOUR EXT
INIT
```

Note: The SOURCE:SWEEP:COUNT and SOURCE:LIST:COUNT are set to 1 after \*RST i.e. the sweeper will perform single sweep function after occurrence of an external trigger signal and then waits for the next trigger event.

### 8.3.5 Sweep with Marker

To use a sweeper with setting of two markers the programmer could add the following commands to example 8.3.3:

```
SOURce:MARKer1:FREQuency 125 MHZ
SOURce:MARKer2:FREQuency 170 MHZ
```

or

```
MARK1:FREQ 125MHZ;MARK2:FREQ 170MHZ
```

### 8.3.6 Reference oscillator

To use a RF & microwave source with its output frequency derived from an external reference frequency the programmer could send:

## 1999 SCPI Instrument Classes

```
*RST  
SOURce:ROSCillator:EXTernal 10 MHZ  
SOURce:CW:FREQuency 2355.5 MHZ  
SOURce:POWer:LEVel:IMMediate:AMPLitude -6 DBM  
OUTPut:STATe ON
```

or

```
*RST  
ROSC:EXT 10MHZ  
FREQ 2355.5MHZ;POW -6DBM;OUTP ON
```

## 9 Signal Switchers

A signal switcher is a basic signal routing instrument. In its most basic form, it can make and break signal connections. These signal connections are usually referred to as channels.

Making a connection is closing a channel, and breaking a signal is opening a channel. More advanced signal switchers can be programmed with sequence lists of channels to close in response to trigger events. Because a signal switcher is primarily a signal router, the ROUTE root node is optional.

Figure 9-1 shows an instrument model for a signal switcher. It is derived from Command Reference, chapter 2. It is essentially the same as Figure 2-1 in Command Reference with various parts gray or dashed. The bold boxes and lines, solid and dashed, are described in this chapter.

All SCPI compliant products implement the commands listed in Syntax & Style section 4.2.

9

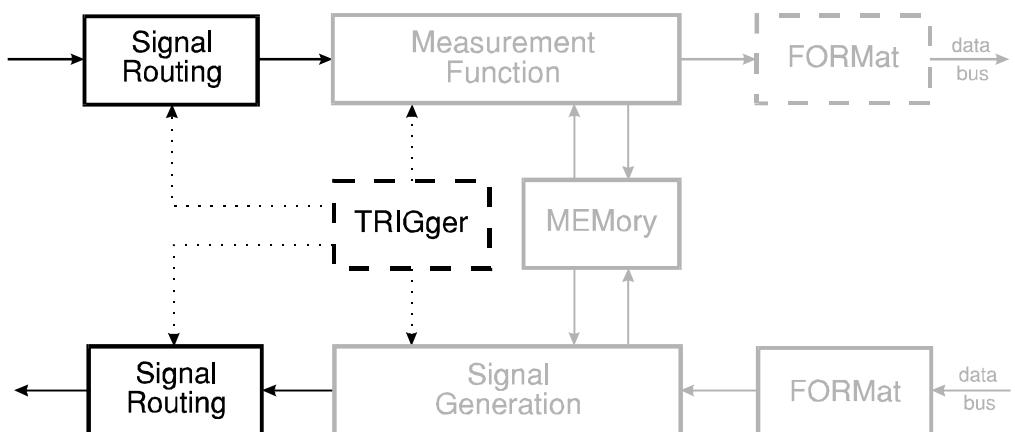


Figure 9-1 Simplified Model for a Programmable Signal Switcher

## 9.1 Base Functionality

The <bf\_keyword> for a signal switcher is SWITCHER.

### 9.1.1 Base Measurement Instructions

Since a signal switcher has no measuring capability, it has no measurement instructions.

### 9.1.2 Base Device-oriented Functions

#### 9.1.2.1 ROUTe Subsystem

A signal switcher can at least make and break signal connections by closing and opening channels.

The following commands shall be implemented.

KEYWORD	PARAMETER FORM	SCPI Reference
[ROUTe]		Vol 2-17
:CLOSe	<channel_list>	Vol 2-17.1
:STATe?		Vol 2-17.1.1
:OPEN	<channel_list>	Vol 2-17.3
:ALL		Vol 2-17.3.1

9

If the :CLOSe and :OPEN commands are implemented as overlapped commands, the associated Pending-Operation flags shall be reported in the No-Operation-Pending flag.

### 9.1.3 Base Status Reporting

All SCPI signal switchers shall implement the status reporting structure described in Syntax & Style, Status Reporting. Command Reference, STATus Subsystem defines the commands which shall be used to control the status reporting structure.

The base functionality of a switcher adds no additional status reporting requirements.

## 9.2 Additional Functionality

### 9.2.1 Scanning

The <af\_keyword> for scanning is SCAN.

Signal switchers which implement the SCAN additional functionality can be programmed with a list of channels to “scan”. Each time the signal switcher recognizes a trigger event, it opens the previously closed channel and closes the next channel in the list.

#### 9.2.1.1 SCAN Measurement Instructions

SCAN adds no measurement instructions.

#### 9.2.1.2 SCAN Device-oriented Functions

##### 9.2.1.2.1 ROUTe Subsystem

A signal switcher which has the SCAN additional functionality shall implement the following command.

KEYWORD	PARAMETER FORM	SCPI Reference
[ROUTe] :SCAN	<channel_list>	Vol 2-17 Vol 2-17.6

9

Note: The following paragraph is intended to provide guidance to designers of signal switchers. It is not intended to impose further design requirements.

The appropriate interaction between the SCAN command and the CLOSE and OPEN commands depends upon the intended application of the signal switcher. In some instances it is advantageous to be able to close a set of channels using the CLOSE command (e.g. to connect sourcing instruments to the device under test), then scan through a different set of channels (e.g. to make measurements on various nodes in the device under test) without opening those channels which were closed by the CLOSE command. In other instances it is desirable that the start of the scan operation first opens any channels which are already closed.

##### 9.2.1.2.2 TRIGger Subsystem

A signal switcher which has the SCAN additional functionality shall implement the following trigger commands.

KEYWORD	PARAMETER FORM	SCPI Reference
ABORT		Vol 2-24.5
INITiate		Vol 2-24.7
:CONTinuous	<Boolean>	Vol 2-24.7.1
[:IMMEDIATE]		Vol 2-24.7.2
[:ALL]		Vol 2-24.7.2.1
TRIGGER		Vol 2-24.8
[:SEQUENCE]		Vol 2-24.8.1
:COUNT	<numeric_value>	Vol 2-24.8.1.2
:SOURce	BUS   IMMEDIATE   ... †	Vol 2-24.8.1.17

## 1999 SCPI Instrument Classes

† In addition to the TRIGger:SOURce parameters listed, a signal switcher with SCAN shall implement an external trigger, such as EXTernal, ECLTrg<n> or TTLTrg<n>.

Requiring BUS as a legal parameter to TRIGger:SOURce has the side effect of requiring DT1 capability in a 488.1 device. It also means the signal switcher implements the \*TRG common command.

A VXIbus device with a trigger function is required to implement the *Trigger* command. When TRIGger:SOURce is set BUS, a VXIbus switcher shall use the *Trigger* command as the TRIGger source.

### 9.2.1.3 SCAN Status Reporting

Bit 5 (Waiting for TRIG) in the OPERation Status Register shall be used to indicate the status of the trigger model.

### 9.2.2 Extended Trigger

The <af\_keyword> for extended trigger is ETRIGGER. This <af\_keyword> is subservient to SCAN.

9

Some signal switchers which have SCAN functionality (and so have TRIGger commands) provide two layer triggering where a separate event must occur before the TRIGger layer is entered.

#### 9.2.2.1 ETRIGGER Measurement Instructions

ETRIGGER adds no Measurement Instructions.

#### 9.2.2.2 ETRIGGER Device-oriented Functions

##### 9.2.2.2.1 TRIGger Subsystem

The following ARM command shall be used to provide the ETRIGGER functionality.

KEYWORD	PARAMETER FORM	SCPI Reference
ARM		Vol 2-24.6
[:SEQUENCE]		Vol 2-24.6.1
[:LAYER]		Vol 2-24.6.1.2
:SOURCE	{BUS   IMMEDIATE   ... }	Vol 2-24.6.1.2.14

#### 9.2.2.3 ETRIGGER Status Reporting

Bit 6 (Waiting for ARM) in the OPERation Status Register shall be used to indicate the status of the trigger model.

## 9.3 Programming Examples

This section discusses how a user might program a signal switcher for certain applications. The examples assume the signal switcher contains a single switch module which is a 10-channel multiplexer which permits any combination of the 10 channels to be closed simultaneously.

### 9.3.1 Making and Breaking Connections

To close signal channel #3:

```
ROUTe:CLOSE (@3)
```

or

```
CLOS (@3)
```

To query whether channel #3 is closed:

```
ROUTe:CLOSE? (@3)
```

or

```
CLOS? (@3)
```

9

### 9.3.2 Programmed Connection Sequences

A signal switcher which can be programmed with a sequence of connections can improve the throughput of test systems by reducing the communication required between the system controller and the test instruments. For example, the output of the 10-channel multiplexer which has SCAN functionality, and includes EXTERNAL as a valid parameter for TRIGGER:SOURce, is connected to the inputs of a DMM. The “meter complete” signal from the DMM is connected to the external trigger input of the signal switcher. To configure this system to measure on all 10 channels, use this sequence of commands to the signal switcher:

```
*RST
TRIGger:SEQUence:COUNt 9;SOURce EXTERNAL
ROUTe:SCAN (@2,3,4,5,6,7,8,9,10);CLOSE (@1)
INITiate:IMMediate
```

or

```
*RST
TRIG:COUNt 9;SOURce EXT
SCAN (@2:10);CLOS (@1)
INIT
```

The \*RST command sets INITiate:CONTinuous to OFF and puts the trigger model in IDLE. After sending the INITiate:IMMediate command, the system controller triggers the DMM. When the meter is finished measuring, its “meter complete” signal activate the signal switcher’s external trigger input, causing it to close channel 2. Now the system controller triggers the DMM again, and so on. When the signal switcher has finished processing the 9th trigger (closing channel 10) it returns to the trigger model IDLE layer.

## **1999 SCPI Instrument Classes**