Contents

AEROFLEX

S-Series SGD Digital Signal Generator

Remote Command Reference Manual

Document part no. 47090/133 (PDF version)

S-Series Signal Sources

SGD DIGITAL SIGNAL GENERATOR

Remote Command Reference Manual

Longacres House
Six Hills Way
Stevenage SG1 2AN
UK

Document no. 47090/133 (PDF)
Issue 1

21 January 2013

Associated publications

If you want to…	Refer to…
View safety information and basic setup and operation instructions in **pdf** format.	**S-Series SGD Digital Signal Generator Getting Started** Part no. 47090/130 On the CD-ROM and at www.aeroflex.com/.
View safety information and basic setup and operation instructions in **printed** format.	**S-Series SGD Digital Signal Generator Getting Started** Part no. 47000/130 Supplied with the instrument.
View operating information for the instrument in **html** Help format	**S-Series SGD Digital Signal Generator Help** Part no. 47090/131 On the CD-ROM and at www.aeroflex.com/
View operating information for the instrument in **pdf** format.	**S-Series SGD Digital Signal Generator Operating Manual** Part no. 467090/132 On the CD-ROM and at www.aeroflex.com/
View operating information for the instrument in **printed** format.	**S-Series SGD Digital Signal Generator Operating Manual** Part no. 467000/132 Available as an optional extra

Contents

REMOTE OPERATION COMMANDS

Purpose

This document describes remote commands that are supported on the Aeroflex S-Series SGD Digital Signal Generator. This document describes various SCPI (Standard Commands for Programmable Instrument) mandatory and common commands required to support the instrument, together with IEEE optional commands

Introduction

This instrument may be operated remotely via an interface that conforms to:

IEEE Std 488.1-1987, which defines the electrical, mechanical and low-level protocol characteristics of the bus structure, the GPIB (General Purpose Interface Bus).

IEEE Std 488.2-1987, which defines standard codes, formats, protocols and common commands for use with IEEE Std 488.1.

The instrument is not fully compliant with SCPI (Standard Commands for Programmable Instruments) because many product features are not covered by that standard, and modern software trends favor the use of instrument drivers as a means of achieving interchangeability.

However, we recognize that SCPI is in common use by system developers and a number of SCPI features that make system integration easier have been implemented. These include the command mnemonic derivation rules (long and short form) and many of the most frequently used commands themselves. Refer to SCPI 1999 (standard available from the IVI Foundation) for details.

Where to find commands

Commands are grouped into particular subsystems on the following pages, as shown in the Contents. Under each heading is an overview of the commands within that subsystem, which will help you quickly locate commands by function. Commands are arranged alphabetically within subsystems.

CONVENTIONS USED IN THIS MANUAL

Abbreviations

Long and short forms

The elements of compound and query headers have a long and a short form, as defined by SCPI. Either the long or the short form may be entered as a command; other abbreviations are not permissible.

Example:

```
STATus:OPERation:EVENt?
```

is interpreted the same as

```
STAT:OPER:EVEN
```

The short form is marked by upper-case letters, the long form corresponds to the complete word. Upper-case and lower-case serve the above purpose only, as the instrument itself does not make any distinction between upper-case and lower-case letters.

Queries always return the short form, or a numeric response in those cases where the command provides a choice of numeric or character data.

Bracketed elements

Square brackets [ ]

Elements within the compound common program header structure that are enclosed within square brackets are optional and therefore may be omitted; the instrument processes the command in the same manner whether the bracketed element is included or not.

Example:

```
[SOURce:]POWer[:LEVel][:IMMediate][:AMPlitude]
```

is interpreted the same as

```
POWer
```

This applies to parameters also. The ability to recognize the full command length ensures that the instrument complies with the SCPI standard.

Angle brackets < >

Text within angle brackets represents an actual value that needs to be inserted: for example, <freq> shows that you need to insert a frequency value in the command at this point.

Case

The software is not case-sensitive. Upper- and lower-case characters are interchangeable. There is no conflict between milli (m) and mega (M) as both cannot be applied to the same data.

Choices

The vertical bar (|)

- separates a choice of parameters:

  for example, 0 | 1 means ‘0 or 1’

  or

- separates a choice of commands.

Compound program headers

Compound program headers allow a complex set of commands to be built up from a smaller set of basic elements in a tree structure. The elements of a compound program header are separated by a colon (:),each colon representing a change of level in the hierarchy. Each subsystem in this instrument is organized as a separate tree structure.

The compound program header may, optionally, be followed by one or more parameters encoded as program data functional elements.

Example:

```
CALibration:IQUSer:ADJust
```

***Note**:* *A leading colon is optional*

Program data

Program data functional elements contain the parameters related to the program header(s). The following program data functional elements are accepted by the instrument:

<CPD>	(also known as <CHARACTER PROGRAM DATA>)
<NRf>	(also known as <DECIMAL NUMERIC PROGRAM DATA>)
<ARBITRARY BLOCK PROGRAM DATA>	

These functional elements are defined in IEEE 488.2 and the SCPI Syntax and Style handbook.

A white space must separate the command header(s) and the program data.

<white space>, as defined in IEEE Std 488.2, can be any number of ASCII characters in the range 0–9, 11–32 decimal.

<white space> is also allowed at other points in a message.

<CPD

CPD (character program data) sets a parameter to one of a number of states that are best described by short alphanumeric strings.

Example:

ON

<NRf>

NRf (numeric representation, flexible) covers integer and floating-point representations.

Examples:

–466	Integer value
4.91	Explicitly-placed decimal point
59.5E+2	Mantissa and exponent representation

The format is known as ‘flexible’ because any of the three representations may be used for any type of numeric parameter.

Examples:

Where a parameter requires an integer value in the range 1 to 100, and you need to set its value to 42, the following values are accepted by the instrument:

42	Integer
42.0	Floating point
4.2E1, 4200E–2	Floating point – mantissa/exponent
41.5	Rounded up to 42
42.4	Rounded down to 42

<STRING PROGRAM DATA>

String program data consists of a number of ASCII characters enclosed in quotes. Use either pairs of single (ASCII 39) or double (ASCII 34) quotes, but do not mix single and double in a string. A quote within a string must be enclosed within an extra pair of quotes.

Example:

'This string contains the word ' 'Hello' ' '

is interpreted as

This string contains the word 'Hello'

and

"This string contains the word " "Hello" " "

is interpreted as

This string contains the word "Hello".

<Boolean>

<Boolean> is used as 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, and a nonzero result is interpreted as 1.

<CPD> elements ON and OFF are accepted as inputs, with ON corresponding to 1 and OFF corresponding to 0. Queries return 1 or 0, never ON or OFF.

Examples:

ON is interpreted as 1

0.4 is interpreted as 0

2.8 is interpreted as 1

Response data

The following response data functional elements are generated by the instrument:

<CRD>	(also known as <CHARACTER RESPONSE DATA>)
<NR1>	
<NR2>	
<STRING RESPONSE DATA>	

<CRD>

CRD (character response data) is returned when reading the value of a parameter that can take a number of discrete states. States are represented by short alphanumeric strings.

Example:

ON

<NR1

This type of NR (numeric response) returns the value of integer parameters, such as an averaging number or the number of measurement points.

Examples:

15
+3
–57

<NR2>

This type of NR (numeric response) includes an explicitly placed decimal point, but no exponent.

Examples:

17.91
–18.27
+18.83

Extended numeric parameters

Most subsystems use extended numeric parameters to specify physical quantities. Extended numeric parameters accept all numeric parameter values and other special values as well.

The following are examples of extended numeric parameters:

100	any simple numeric value
–100mV	negative 100 millivolts
10DEG	10 degrees

Extended numeric parameters also include the following special parameters:

MINimum | MAXimum

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

Remote command structure

This section describes the way remote commands are used in this document. IEEE optional commands for the SGD consist of the following structures:

SCPI subsystem	**System name**	**Instance**	**Setting/result**
(1 mnemonic)	(2 mnemonics)	(numeric suffix)	(1 or more mnemonics)
READ	:ANALyzer :MEAS	1 | 2	:POWer

The purpose of the four parts of the command are:

SCPI subsystem

The root mnemonic or command subsystem as specified in SCPI; for example, READ, FETCh, SOURce.

COMMON COMMANDS

Commands recognized by all IEEE 488.2 instruments

All SCPI instruments must implement the common commands declared mandatory by the IEEE 488.2 standard. These commands have the same effect on any instrument that conforms to the standard. The headers of these commands consist of an asterisk (*) followed by three letters. Many common commands refer to the status reporting system.

The most important of the common commands is *RST, which places the instrument in a defined state. It is good practice to send *RST at the start of any program.

*CLS

*ESE\?

*ESR?

*IDN?

*OPC\?

*OPT?

*RST

*SRE\?

*STB?

*WAI

*CLS

Description: Clear status clears the standard event register, the error/event queue, the operation status register and the questionable status register.

Parameters: None

*ESE

Description: The event status enable command sets the standard event status enable register to the value specified. This is an eight-bit register.

Parameters: <NRf>
Mask

Valid values: Mask: integer. Valid values are 0 to 255. Values outside the range are rejected and an error generated.

*ESE?

Description: Reads the event status enable register. This is an eight-bit register. The contents of the event status enable register are returned in decimal form.

Parameters: None

Response: <NR1>
Mask

Returned values: Mask: integer. Values are in the range 0 to 255.

*ESR?

Description: Reads the value of the standard event status register. This is an eight-bit register. The contents of the register are returned in decimal form. Subsequently the register is set to zero.

Parameters: None

Response: <NR1>
Register contents

Returned values: Register contents: integer. Values are in the range 0 to 255.

*IDN?

Description: The identification query command allows information about the instrument to be read.

Parameters: None

Response: <arbitrary ASCII response data>

Manufacturer, model, serial number, software part number and issue number

Returned values: Manufacturer: string

Always returns 'Aeroflex'.

Model: string

This is the instrument's model number in the form 'NSGD Radio Test Set'

Serial number: string

This is in the form ssssss/sss *where* s *is an* ASCII *digit in the range* 0 to 9.

Software part number and issue number: string

This is in the form ppppp/ppp/ii.ii *where* p *and* i *are* ASCII *digits in the range* 0 to 9.

*OPC

Description: The operation complete command sets the operation complete bit (bit 0) in the standard event status register when execution of the preceding operation is complete. This bit can be used to initiate a service request.

*OPC should be the final <program message unit> of the <program message>.

Parameters: None

*OPC?

Description: The operation complete query returns a '1' when the preceding operation has completed.

*OPC? should be the final <query message unit> of the <program message>.

Parameters: None

Response: <NR1>
Operation complete

Returned values: Operation complete: integer. Value is 1.

*OPT?

Description: Reads hardware options present. If no options are present a single '0' is returned, otherwise the response is up to six strings separated by commas.

Parameters: None

Response: <arbitrary ASCII response data>
Options

Returned values: Options: string

*RST

Description: Resets the instrument to a known configuration appropriate for remote operation.

Parameters: None

*SRE

Description: Sets the service request enable register. This is an eight-bit register.

Parameters: <NRf>
Mask

Valid values: Mask: integer. Valid values are 0 to 255. Values outside range are rejected and an error is generated.

*SRE?

Description: Reads the service request enable register. This is an eight-bit register.

Parameters: None

Response: <NR1>
Mask

Returned values: Mask: integer. Values are in the range 0 to 255.

*STB?

Description: Reads the status byte. This is an eight-bit register.

Parameters: None

Response: <NR1>
Status byte

Returned values: Status byte: integer. Values are in the range 0 to 255.

*WAI

Description Wait-to-Continue command. Prevents servicing of subsequent commands until all preceding commands have been executed and all signals have settled.

Parameters None

IEEE OPTIONAL COMMANDS

IEEE optional commands, also referred to as instrument-control commands, are based on a hierarchical structure and can be represented in a command tree. The command headers are built with one or several mnemonics (keywords). The first-level (root-level) mnemonic identifies a complete command system, for example:

> **SOUR**ce: this mnemonic identifies the SOURce command system, which provides generator settings.

The same mnemonics may be used on different command levels, not necessarily with the same meaning. The actual meaning of a mnemonic depends on its position in the command header.

The [SOURce] subsystem — an introduction

The SOURce subsystem contains commands that cover all aspects of modulation, frequency, power and pulse generation

The [SOURce] subsystem consists of:

- The [FREQuency] subsystem, which controls frequency parameters of the carrier and sweep signals
- The [LIST] subsystem, which controls list mode sweeping
- The [MODulation] subsystem, which controls all aspects of modulation
- The [POWer] subsystem, which sets all aspects of carrier and sweep levels
- The [PULSe] subsystem, which controls external and internal pulses and their profiles
- The [SWEep] subsystem, which controls the generation of frequency and power sweep signals.

Each of these subsystems is dealt with separately in the following sections.

The [SOURce] subsystem effectively controls the switching and configuration of internal and external signal sources and modulation paths within the instrument.

The menu structure of the [SOURce] subsystem is as follows:

```
[SOURce]
    [SIGGen]
        [:GENerator]
        (alias :SOURce)
            :FREQuency
            :LIST
            [:MODulation]
            :POWer
            :PULSe
            :SWEep
```

RF output frequency commands

([SOURce][:SIGGen][:GENerator]:FREQuency subsystem)

```
[SOURce]
    [:SIGGen]
        [:GENerator]
        (alias :SOURce)
            :FREQuency
                [:CW|:FIXed]\?
                :MODE\?
                :PHASe
                    [:ADJust]\?
                    :REFerence\?
                :SWEep
                    :DWELl\?
                    :POINts\?
                    :SPACing\?
                    :STARt\?
                    :STEP[1]
                    :STOP\?
```

[1] *Up to v1.2.0 software, :STEP included two enable parameters – SWEep:STEP:ENABLe and SWEep:STEP:INCRement. These are equivalent to, and are replaced by, SWEep:POINts (page 18) and SWEep:STEP (page 19) respectively in subsequent versions of software.*

[SOURce][:SIGGen][:GENerator]**:FREQ**uency[:CW|:FIXed]

Description: Sets the carrier frequency by value, to maximum or minimum, stepping up or down, returning to the last full setting, or transferring the current value to the new setting.

Parameters: <numeric_value>

Valid values: <NRf>(Hz) | MAXimum | MINimum | UP | DOWN | RETurn | XFER

Example: SOUR:SIGG:GEN:FREQ:CW:MAX

[SOURce][:SIGGen][:GENerator]**:FREQ**uency[:CW|:FIXed]?

Description: Queries the carrier frequency by value.

Parameters: None

Response: <NR2>

Returned values: Carrier frequency in Hz

Example: SOUR:SIGG:GEN:FREQ:CW?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:MODE**

Description: Sets the mode of operation of the carrier frequency.

Parameters: <CPD>

Valid values: CW | FIXed | SWEep | LIST

Example: SOUR:SIGG:GEN:FREQ:MOD SWE

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:MODE**?

Description: Queries the mode of operation of the carrier frequency.

Parameters: None

Response: <CRD>

Returned values: CW | FIX | SWE | LIST

Example: SOUR:SIGG:GEN:FREQ:MOD?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:PHAS**e[:ADJust]

Description: Sets the carrier frequency phase.

Parameters: <NRf>

Valid values: -360° to 0° to +360°

Example: SOUR:SIGG:GEN:FREQ:PHAS:ADJ 180

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:PHAS**e[:ADJust]?

Description: Queries the carrier frequency phase.

Parameters: None

Response: <NR2>

Returned values: Degrees

Example: SOUR:SIGG:GEN:FREQ:PHAS:ADJ?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:PHAS**e**:REF**erence

Description: Sets the current carrier frequency phase as a zero reference.

Parameters: None

Valid values: None

Example: SOUR:SIGG:GEN:FREQ:PHAS:REF

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:PHAS**e**:REF**erence?

Description: Queries the carrier frequency’s phase relative to the zero reference.

Parameters: None

Response: <NR2>

Returned values: Degrees

Example: SOUR:SIGG:GEN:FREQ:PHAS:REF?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:DWEL**l

Description: Sets the time per sweep step for the carrier frequency.

Parameters: <NRf>

Valid values: <NRf>(s) | MAXimum | MINimum | UP | DOWN | RETurn | XFER

Set by value, to maximum or minimum, stepping up or down, returning to the last full setting (RETurn), or transferring the current value to the new setting (XFER).

Example: SOUR:SIGG:GEN:FREQ:SWE:DWEL 5s

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:DWEL**l?

Description: Queries the time per sweep step for the carrier frequency.

Parameters: None

Response: <NR2>

Returned values: Time in s.

Example: SOUR:SIGG:GEN:FREQ:SWE:DWEL?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:POINts**

Description: Sets 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 the sweep step by the equation: STEP = SPAN/(POINts–1)

If POINts are changed, STEP is also changed, but not SPAN. In a logarithmic sweep, POINts determines the number of points/decade of sweep by the equation: POINts/DECADE = (POINts–1)/SPAN (in decades).

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.

Parameters: <numeric_value>

Valid values: <NRf> | MAXimum | MINimum

Example: SOUR:SIGG:GEN:FREQ:SWE:POINts 100

This replaces the SWEep:STEP:ENABle command after software v1.2.0.

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:POINts**?

Description: Queries the number of points in a stepped sweep.

Parameters: None

Response: <NR2>

Returned values: Number of points in a stepped sweep.

Example: SOUR:SIGG:GEN:FREQ:SWE:POINts?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:SPAC**ing

Description: Sets the carrier sweep step points to either linear or logarithmic spacing.

Parameters: <CPD>

Valid values: LINear | LOGarithmic

Example: SOUR:SIGG:GEN:FREQ:SWE:SPAC LIN

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:SPAC**ing?

Description: Queries whether carrier sweep step points have linear or logarithmic spacing.

Parameters: None

Response: <CRD>

Returned values: LIN | LOG

Example: SOUR:SIGG:GEN: FREQ:SWE:SPAC?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STAR**t

Description: Sets the start frequency for a carrier sweep.

Parameters: <numeric_value>

Valid values: <NRf>(Hz) | MAXimum | MINimum

Example: SOUR:SIGG:GEN:FREQ:SWE:STAR MAX

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STAR**t?

Description: Queries the start frequency for a carrier sweep.

Parameters: None

Response: <NR2>

Returned values: Start frequency in Hz

Example: SOUR:SIGG:GEN: FREQ:SWE:STAR?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STEP**

Description: Sets the step size for a carrier sweep.

Parameters: <numeric_value>

Valid values: <NRf> GHz | MHz | kHz | Hz

Example: SOUR:SIGG:GEN:FREQ:SWE:STEP MHz

This replaces the SWEep:STEP[:INCRement] command after software v1.2.0.

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STEP**?

Description: Queries the step size for a carrier sweep.

Parameters: None

Response: <NR2>

Returned values: Step size in Hz

Example: SOUR:SIGG:GEN: FREQ:SWE:STEP?

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STOP**

Description: Sets the stop frequency for the carrier sweep.

Parameters: <numeric_value>

Valid values: <NRf>(Hz) | MAXimum | MINimum

Example: SOUR:SIGG:GEN:FREQ:SWE:STOP MAX

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STOP**?

Description: Queries the carrier sweep's stop frequency.

Parameters: None

Response: <NR2>

Returned values: Sweep stop frequency in Hz

Example: SOUR:SIGG:GEN: FREQ:SWE:STOP?

Obsolescent commands

The following command is replaced by SWEep:POINts after software v1.2.0:

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STEP:ENABle**

Description: Enables stepped sweeping, either by size of step, or step count.

SIZe lets you query or set the step size, and query the number of points. You cannot set points.

COUNt lets you query or set the number of points, and query the step size. You cannot set the step size.

Parameters: <CPD>

Valid values: SIZe | COUNt

Example: SOUR:SIGG:GEN:FREQ:SWE:STEP:ENAB SIZ

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STEP:ENABle**?

Description: Queries how sweep steps are performed.

Parameters: None

Response: <CRD>

Returned values: SIZ | COUN

Example: SOUR:SIGG:GEN: FREQ:SWE:STEP:ENAB?

The following command is replaced by SWEep:STEP after software v1.2.0:

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STEP[:INCRement]**

Description: Sets the sweep step size.

Parameters: <numeric_value>

Valid values: <NRf> GHz | MHz |kHz | Hz

Example: SOUR:SIGG:GEN:FREQ:SWE:STEP:INCR 500 MHz

[SOURce][:SIGGen][:GENerator]**:FREQ**uency**:SWE**ep**:STEP[:INCRement]**?

Description: Queries the sweep step size.

Parameters: None

Response: <NR2>

Example: SOUR:SIGG:GEN: FREQ:SWE:STEP:INCR?

List commands

([SOURce]:LIST subsystem)

```
[SOURce]
    :LIST
        :CLEar
            :ALL
        :DWELl\?
        :FREQuency\?
        :INSert
        :POWer\?
        :RESet
        :STARt\?
        :STATe\?
        :STOP\?
        :VALue\?
```

[SOURce]**:LIST**

Description: Inserts a sequence of frequency and power values into the list in sequence, starting at the address given.

Parameters: <NRf>,<NRf>,<NRf>[,<NRf>,<NRf>...]

Valid values: <addr>,<freq>,<power>[,<freq>,<power>...] <addr> is an integer within the address range of the list

[SOURce]**:LIST:CLE**ar

Description: Clears the entry at this address.

Parameters: <NRf>

Valid values: <addr>, an integer within the address range of the list

[SOURce]**:LIST:CLE**ar**:ALL**

Description: Clears all entries in the list.

Parameters: None

Valid values: None

[SOURce]**:LIST:DWEL**l

Description: Sets the dwell time, the time spent at each address in the list.

Parameters: <NRf>

Valid values: <time (s)>

[SOURce]**:LIST:DWEL**l?

Description: Returns the dwell time.

Parameters: None

Response: <NR2>

Returned values: Dwell time in s

[SOURce]**:LIST:FREQ**uency

Description: Inserts a sequence of frequencies into the list, starting at the address given.

If there is already a list entry starting at this address, the command overwrites the frequency value(s) but does not modify the power value(s). If entries are not yet defined, the current power (specified by :SOURce:POWer?) is set as the power value.

Parameters: <NRf>,<NRf>[,<NRf>...]

Valid values: <addr>,<freq>[,<freq>...] <addr> is an integer within the address range of the list

[SOURce]**:LIST:FREQ**uency?

Description: Returns the frequency at a specified list address.

Parameters: <addr>

Response: <NR1>

Returned values: Frequency in Hz

[SOURce]**:LIST:INS**ert

Description: Inserts frequency and power values into the list at this address, shifting all following entries down.

Parameters: <NRf>,<NRf>,<NRf>

Valid values: <addr>,<frequency>,<power> <addr> is an integer within the address range of the list

[SOURce]**:LIST:POW**er

Description: Inserts a sequence of powers into the list, starting at the address given.

If there is already a list entry starting at this address, the command overwrites the power value(s) but does not modify the frequency value(s). If entries are not yet defined, the current frequency (specified by :SOURce:FREQuency?) is set as the frequency value.

Parameters: <NRf>,<NRf>[,<NRf>...]

Valid values: <addr>,<power>[,<power>...] <addr> is an integer within the address range of the list

[SOURce]**:LIST:POW**er?

Description: Returns the power at a specified list address.

Parameters: <addr>

Response: <NR1>

Returned values: Power in dBm

[SOURce]**:LIST:RES**et

Description: Returns the list sweep to its start address.

Parameters: None

[SOURce]**:LIST:STAR**t

Description: Defines the start address, from which the list sweep is executed.

Parameters: <NRf>

Valid values: <addr>, an integer within the address range of the list

*RST sets: 0

[SOURce]**:LIST:STAR**t?

Description: Returns the start address, from which the list sweep is executed.

Parameters: None

Response: <addr>

Returned values: Start address

[SOURce]**:LIST:STAT**e

Description: This command sets list mode to ON or OFF.

Parameters: <Boolean>

Valid values: 0 | 1 |ON | OFF

Example: SOUR:LIST:STAT ON

[SOURce]**:LIST:STATe**?

Description: Queries whether list mode is on or off.

Parameters: None

Response: <Boolean>

Returned values: 0 | 1

Example: SOUR:LIST:STAT?

[SOURce]**:LIST:STOP**

Description: Defines the stop address, at which the list sweep halts.

Parameters: <NRf>

Valid values: <addr>, an integer within the address range of the list

*RST sets: Maximum list address

[SOURce]**:LIST:STOP**?

Description: Returns the stop address, at which the list sweep halts.

Parameters: None

Response: <addr>

Returned values: Stop address

[SOURce]**:LIST:VAL**ue

Description: Modifies the frequency and power values at the specified address.

Parameters: <NRf>,<NRf>,<NRf>

Valid values: <addr>,<freq>,<power>

[SOURce]**:LIST:VAL**ue?

Description: Returns the frequency and power values at the specified address.

Parameters: None

Response: <NR1>,<NR2>,<NR2>

Returned values: Address and the associated frequency and power values

Modulation commands

([SOURce][:SIGGen][:GENerator][:MODulation] subsystem)

```
[SOURce]
    [:SIGGen]
        [:GENerator]
        (alias :SOURce)
            [:MODulation]
                :PULM
                    :STATe\?
```

[SOURce][:SIGGen][:GENerator][:MODulation]**:PULM:STAT**e

Description: Sets pulse modulation on or off.

Parameters: <Boolean>

Valid values: 0 | 1 |ON | OFF

Example: SOUR:SIGG:GEN:MOD:PULM:STAT ON

[SOURce][:SIGGen][:GENerator][:MODulation]:**PULM:STAT**e?

Description: Queries whether pulse modulation is on or off.

Parameters: None

Response: <Boolean>

Returned values: 0 | 1

Example: SOUR:SIGG:GEN:MOD:PULM:STAT?

Power commands

([SOURce][:SIGGen][:GENerator]:POWer subsystem)

```
[SOURce]
    [:SIGGen]
        [:GENerator]
        (alias :SOURce)
            :POWer
                :ALC
                    :STATe\?
                :ATTenuation
                [:LEVel]
                    [:IMMediate]
                        [:AMPLitude]\?
                :MODE\?
                :SWEEp
                    :DWELl\?
                    :POINts\?
                    :STARt\?
                    :STEP\?
                        :ENABle\?
                        [:INCRement]\?
                    :STOP\?
```

[SOURce]**:POWer:ALC**[:STATe]

Description: Sets the ALC state for optimum performance.

Parameters: <CPD>

Valid values: AUTO | NORMal | AM | FROZen | SCALed

*RST sets: NORMal

[SOURce]**:POWer:ALC**[:STATe]?

Description: Returns the ALC state.

Parameters: None

Response: <CRD>

Returned values: AUTO | NORM | AM | FROZ | SCAL

[SOURce][:SIGGen][:GENerator]:**POWer:ATT**enuation

Description: Sets the attenuation level. Note that when increasing the level by 10 dB, the magnitude of the outgoing signal, as well as the LEVel, is decreased by 10 dB. Default units are as determined in the unit subsystem.

Parameters: <numeric_value>

Valid values: <NRf>dB | UP | DOWN | RETurn | XFER

Set by value, to maximum or minimum, stepping up or down, returning to the last full setting (RETurn), or transferring the current value to the new setting (XFER).

Example: SOUR:SIGG:GEN:POW:ATT 2dB

[SOURce][:SIGGen][:GENerator]**:POWer:ATT**enuation?

Description: Queries the RF attenuation level.

Parameters: None

Response: <NR2>

Returned values: Attenuation level, in units set under the unit subsystem.

Example: SOUR:SIGG:GEN:POW:ATT?

[SOURce][:SIGGen][:GENerator]:**POW**er[:LEVel][:IMMediate] [:AMPLitude]

Description: Sets the carrier level.

Parameters: <numeric_value>

Valid values: <NRf> | MAXimum | MINimum | UP | DOWN | RETurn | XFER

Set by value, to maximum or minimum, stepping up or down, returning to the last full setting (RETurn), or transferring the current value to the new setting (XFER).

<NRf> is in units set by :UNIT:POW.

Example: SOUR:SIGG:GEN:POW:LEV:IMM:AMP MAX

[SOURce][:SIGGen][:GENerator]**:POW**er[:LEVel][:IMMediate] [:AMPLitude]?

Description: Queries the carrier level by value.

Parameters: None

Response: <NR2>

Returned values: Carrier level, in units set by :UNIT:POW

Example: SOUR:SIGG:GEN:POW:LEV:IMM:AMP?

[SOURce]**:POW**er**:MODE**

Description: Sets the mode of the carrier level operation.

Parameters: <CPD>

Valid values: FIXed | SWEep | LIST

*RST sets: FIX

[SOURce]**:POW**er**:MODE**?

Description: Returns the mode of carrier level operation.

Parameters: None

Response: <CRD>

Returned values: FIX | SWE | LIST

[SOURce]**:POW**er**:SWE**ep**:DWEL**l

Description: Sets the time per sweep step for carrier level.

Parameters: <numeric_value>

Valid values: <NRf> (s) | MAXimum | MINimum | UP | DOWN | RETurn | XFER

Set by value, to maximum or minimum, stepping up or down, returning to the last full setting (RETurn), or transferring the current value to the new setting (XFER).

*RST sets: 50 ms

[SOURce]**:POW**er**:SWE**ep**:DWEL**l?

Description: Queries the time per sweep step for carrier level.

Parameters: None

Response: <NR2>

Returned values: Time per sweep step in s

[SOURce]**:POW**er**:SWE**ep**:POINts**

Description: Sets 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 the sweep step by the equation: STEP = SPAN/(POINts−1)

If POINts are changed, STEP is also changed, but not SPAN. In a logarithmic sweep, POINts determines the number of points/decade of sweep by the equation: POINts/DECADE = (POINts–1)/SPAN (in decades).

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.

Parameters: <numeric_value>

Valid values: <NRf> | MAXimum | MINimum

Example: SOUR:POW:SWE:POINts 100

[SOURce]**:POW**er**:SWE**ep**:POINts**?

Description: Queries the number of points in a stepped sweep.

Parameters: None

Response: <NR2>

Returned values: Number of points in a stepped sweep.

Example: SOUR:POW:SWE:POINts?

[SOURce]**:POWer:SWEep:STARt**

Description: Sets the start level for a power sweep.

Parameters: <numeric_value>

Valid values: <NRf>(dB) | MAXimum | MINimum

*RST sets: MIN

[SOURce]**:POWer:SWEep:STARt**?

Description: Queries the start level for a power sweep.

Parameters: None

Response: <NR2>

Returned values: Start level

[SOURce]**:POWer:SWEep:STEP**

Description: Sets the step level for a power sweep.

Parameters: <numeric_value>

Valid values: <NRf>(dB) | MAXimum | MINimum | UP | DOWN | RETurn | XFER

Set by value, to maximum or minimum, stepping up or down, returning to the last full setting (RETurn), or transferring the current value to the new setting (XFER).

*RST sets: MAX

[SOURce]**:POWer:SWEep:STEP**?

Description: Queries the step level for a power sweep.

Parameters: None

Response: <NR2>

Returned values: Step level

[SOURce]**:POWer:SWEep:STEP:ENABle**

Description: Enables stepped sweeping, either by size of step, or incremental points mode.

SIZe lets you query or set the step size, and query the number of points. You cannot set points.

POINts lets you query or set the number of points, and query the step size. You cannot set the step size.

Parameters: <CPD>

Valid values: SIZe | POINts

Example: SOUR:POW:SWE:STEP ENAB SIZ

[SOURce]**:POWer:SWEep:STEP:ENABle**?

Description: Queries how sweep steps are performed.

Parameters: None

Response: SIZ | POIN

Returned values: Sweep step

Example: SOUR:POW:SWE:STEP:ENAB?

[SOURce]**:POWer:SWEep :STEP[:INCRement]**

Description: Sets the sweep step size.

Parameters: <numeric_value>

Valid values: <NRf> (dB)

Example: SOUR:POW:SWE:STEP:INCR 3

[SOURce]**:POWer:SWEep :STEP[:INCRement]**?

Description: Queries the sweep step size.

Parameters: None

Response: <NR2>

Returned values: Sweep step size

Example: SOUR:POW:SWE:STEP:INCR?

[SOURce]:**POW**er:**SWE**ep:**STOP**

Description:	Sets the stop level for a power sweep.
Parameters:	<numeric_value>
Valid values:	<NRf>(dB) | MAXimum | MINimum
*RST sets:	MAX

[SOURce]:**POW**er:**SWE**ep:**STOP**?

Description:	Queries the final level for a power sweep.
Parameters:	None
Response:	<NR2>
Returned values:	Stop level

Pulse modulation commands

([SOURce][:SIGGen][:GENerator][:MODulation]:PULSe subsystem)

```
[SOURce]
    [:SIGGen]
        [:GENerator]
        (alias :SOURce)
            :PULSe
                :DELay\?
                :DOUBle
                    :DELay\?
                    :STATe\?
                :PERiod\?
                :WIDTh\?
```

[SOURce][:SIGGen][:GENerator]**:PULSe:DELay**

Description: Sets the time from the start of the period to the first edge of the pulse.

Parameters: <numeric_value>

Valid values: <NRf>(s)

Example: SOUR:SIGG:GEN:PULS:DEL 5s

[SOURce][:SIGGen][:GENerator]**:PULSe:DELay**?

Description: Queries the delay introduced to a pulsed waveform.

Parameters: None

Response: <NR2>

Returned values: Delay introduced to a pulsed waveform in seconds.

Example: SOUR:SIGG:GEN:PULS:DEL?

[SOURce][:SIGGen][:GENerator]**:PULSe:DOUBle:DELay**

Description: Sets the time from the start of the period to the first edge of the second pulse.

Parameters: <numeric_value>

Valid values: <NRf>(s)

Example: SOUR:SIGG:GEN:PULS:DOUB:DEL 5s

[SOURce][:SIGGen][:GENerator]**:PULSe:DOUBle:DELay**?

Description: Queries the delay introduced from the start of the period to the first edge of the second pulse.

Parameters: None

Response: <NR2>

Returned values: Delay introduced from the start of the period to the first edge of the second pulse.

Example: SOUR:SIGG:GEN:PULS:DOUB:DEL?

[SOURce][:SIGGen][:GENerator]**:PULS**e**:DOUB**le**:STAT**e

Description: This command sets the double-pulse mode ON or OFF.

Parameters: <Boolean>

Valid values: 0 | 1 |ON | OFF

Example: SOUR:SIGG:GEN:PULS:DOUB:STAT ON

[SOURce][:SIGGen][:GENerator]**:PULS**e**:DOUB**le**:STAT**e?

Description: Queries the pulse mode.

Parameters: None

Response: <Boolean>

Returned values: 0 | 1

Example: SOUR:SIGG:GEN:PULS:DOUB:STAT?

[SOURce][:SIGGen][:GENerator]**:PULS**e**:PER**iod

Description: Sets the period of a pulsed waveform.

Parameters: <numeric_value>

Valid values: <NRf>(s)

Example: SOUR:SIGG:GEN:PULS:PER 5s

[SOURce][:SIGGen][:GENerator]**:PULS**e**:PER**iod?

Description: Queries the period of a pulsed waveform.

Parameters: None

Response: <NR2>

Returned values: Period of pulsed waveform, in seconds.

Example SOUR:SIGG:GEN:PULS:PER?

[SOURce][:SIGGen][:GENerator]**:PULS**e**:WIDT**h

Description: Sets the width or duration of the pulse.

Parameters: <numeric_value>

Valid values: <NRf>(s)

Example: SOUR:SIGG:GEN:PULS:WIDT 2s

[SOURce][:SIGGen][:GENerator]**:PULS**e**:WIDT**h?

Description: Queries the width or duration of a pulsed waveform.

Parameters: None

Response: <NR2>

Returned values: Width of pulsed waveform, in seconds.

Example: SOUR:SIGG:GEN:PULS:WIDT?

Sweep commands

([SOURce]:SWEep subsystem)

```
[SOURce]
    :SWEep
        :ABORt
        :DOWN
        :INITiate
        :OPERation\?
        :PAUSe
        :TRIGger
            [:MODe]\?
        :UP
```

[SOURce]**:SWE**ep**:ABOR**t

Description: Stops the sweep immediately.

Parameters: None

[SOURce]**:SWE**ep**:DOWN**

Description: Sets to sweep downwards.

Parameters: None

[SOURce]**:SWE**ep**:INIT**iate

Description: Starts a sweep.

Parameters: None

[SOURce]**:SWE**ep**:OPER**ation

Description: Sets whether the sweep mode is single or continuous.

Parameters: <CPD>

Valid values: SINGle | CONTinuous

*RST sets: SING

[SOURce]**:SWE**ep**:OPER**ation?

Description: Returns whether the sweep mode is single or continuous.

Parameters: None

Response: <CRD>

Returned values: SING | CONT

[SOURce]**:SWE**ep**:PAUS**e

Description: Pauses the sweep.

Parameters: None

[SOURce]**:SWE**ep**:TRIG**ger[:MODe]

Description: Sets the trigger mode to off, start, start then stop, or step.

Parameters: <CPD>

Valid values: OFF | STARt | SSTOP | STEP

*RST sets: OFF

[SOURce]**:SWE**ep**:TRIG**ger[:MODe]?

Description: Queries the trigger mode for the sweep.

Parameters: None

Response: <CRD>

Returned values: OFF | STAR | SSTOP | STEP

[SOURce]**:SWE**ep**:UP**

Description: Sets to sweep upwards.

Parameters: None

Instrument system-level commands

([SOURce][:SIGGen][:GENerator]:SYSTem subsystem)

```
SYSTem
    :ERRor
        :ALL?
        [:NEXT]?
    :LANGuage\?
```

SYSTem**:ERR**or**:ALL**?

Description: Queries the error queue for all unread items, and removes them from the queue.

Parameters: None

Response: <NR1>,<CRD>

Returns a comma-separated list of number, string pairs in FIFO order. If the queue is empty, the response is *0,'No error'*.

Example: SYST:ERR:ALL?

SYSTem**:ERR**or[:NEXT]?

Description: Queries the error queue for the next unread item, and removes it from the queue.

Parameters: None

Response: <NR1>,<CRD>

Returns a number and string. If the queue is empty, the response is *0,'No error'*.

Example: SYST:ERR?
SYST:ERR:NEXT?

SYSTem**:LANG**uage

Description: Configures the instrument to function with either the SCPI-like command set or the 2023 command set and status reporting.

This command is only actioned once the EOM at the end of the message has been received and all outstanding query responses have been read.

Follow any change of language with *RST to clear status registers.

Parameters: <string program data>

Valid values: 'Default' | '2030' | '2031' | '2032' | '2040'

*RST sets: No effect on the language set.

SYSTem**:LANG**uage?

Description: Returns the command set that the instrument is to work with.

Parameters: None

Response: <string response data>

Returned values: 'Default' | '2030' | '2031' | '2032' | '2040'

Status commands

(STATus subsystem)

Commands for determining the state of the instrument

This subsystem controls the SCPI-defined status-reporting structures. SCPI defines QUEStionable, OPERation, Instrument SUMmary and INSTrument registers, in addition to those in *IEEE 488.2*. These registers conform to the *IEEE 488.2* specification, and each may consist of a condition register, an event register, an enable register, and negative and positive transition filters.

```
STATus
    <StatReg>
        :CONDition?
        :ENABle\?
        :EVENt?
        :NTRansition\?
        :PTRansition\?
    :PRESet
```

where **<StatReg>** is:

```
    :OPERation
    :OPERation:TRIGger
    :QUEStionable
    :QUEStionable:FREQuency
    :QUEStionable:MODulation
    :QUEStionable:POWer
```

STATus:<StatReg>:CONDition?

Description: Reads the contents of the status register.

Parameters: None.

Response: <NR1> Status register contents.

Example: STAT:OPER:COND?
STAT:QUES:COND?

STATus:<StatReg>:ENABle

Description: Sets the enable mask, which allows true conditions in the status event register to be reported in the summary bit. If a bit is ‘1’ in the enable register and its associated event bit makes a transition to true, a positive transition occurs in the associated summary bit.

Parameters: <NRf> Mask

Valid values: 0–7FFFH

Example: STAT:OPER:ENAB 2000
STAT:QUES:ENAB 1536

STATus:<StatReg>:ENABle?

Description: Reads the enable mask for the status register.

Parameters: [<NRf>] [Mask]

Response: <NR1> Mask

Returned values: 0–7FFFH

Example: STAT:OPER:ENAB?

STATus:<StatReg>:EVENt?

Description: Reads the contents of the event register associated with the operation status register.

Parameters: None.

Response: <NR1> Event register contents.

Returned values: 0–7FFFH

Example: STAT:OPER:EVEN?

STATus:<StatReg>:NTRansition

Description: Sets the negative transition filter in the status register. Setting a bit in the negative transition filter causes a 1 to 0 transition in the corresponding bit of the associated condition register, causing a '1' to be written in the associated bit of the corresponding event register.

Parameters: <NRf> Mask

Valid values: 0–7FFFH

Example: STAT:OPER:NTR 2000
STAT:QUES:NTR 2000

STATus:<StatReg>:NTRansition?

Description: Reads the negative transition mask for the status register.

Parameters: [<NRf>] [Mask]

Response: <NR1> Mask

Returned values: 0–7FFFH

Example: STAT:OPER:NTR?

STATus:<StatReg>:PTRansition

Description: Sets the positive transition filter in the status register. Setting a bit in the positive transition filter causes a 0 to 1 transition in the corresponding bit of the associated condition register, causing a '1' to be written in the associated bit of the corresponding event register.

Parameters: <NRf> Mask

Valid values: 0–7FFFH

Example: STAT:OPER:PTR 2000
STAT:QUES:PTR 2000

STATus:<StatReg>:PTRansition?

Description: Reads the positive transition mask for the status register.

Parameters: [<NRf>] [Mask]

Response: <NR1> Mask

Returned values: 0–7FFFH

Example: STAT:OPER:PTR?

Output control commands

(OUTPut subsystem)

```
OUTPut
    :MODulation
        [:STATe]\?
    [:POWer]
        [:STATe]\?
```

OUTPut**:MOD**ulation[:STATe]

Description: Enables or disables all the active modulation outputs.

When ON, this command causes each modulation output to adopt the state set by its relevant [SOURce][:MODulation]:<modn>:STATe command.

Corresponds to the MOD ON/OFF button.

Parameters: <Boolean>

Valid values: OFF | ON | 0 | 1

Example: OUT:MOD ON
OUT:MOD:STAT ON

OUTPut**:MOD**ulation[:STATe]?

Description: Queries the state of the active modulation outputs.

Parameters: None

Response: <Boolean>

Returned values: 0 | 1

Example: OUT:MOD?
OUT:MOD:STAT?

OUTPut[:POWer][:STATe]

Description: Turns the RF output on or off.

Parameters: <Boolean>

Valid values: OFF | ON | 0 | 1

*RST sets: OFF

OUTPut[:POWer][:STATe]?

Description: Queries whether the RF output is on (1) or off (0).

Parameters: None

Response: <Boolean>

Returned values: 0 | 1

IQ commands — ARB subsystem

([SOURce][:MODulation]:IQ:ARB subsystem)

```
[SOURce]
    [:MODulation]
        :IQ
            :ARB
                :ABORt
                :INITiate
                :MODe\?
                :REPeat\?
                :RESTart\?
                :TRIGger\?
                :WAVeform
                    :CATalog?
                    :DELete
                        :ALL
                        [:FILE]
                    :LOAD
                    :SELect\?
            :SOURce\?
            :STATe\?
```

[SOURce][:MODulation]**:IQ:ARB:ABOR**t

Description: Stops ARB generation.

Parameters: None

[SOURce][:MODulation]**:IQ:ARB:INIT**iate

Description: Starts ARB generation.

Parameters: None

[SOURce][:MODulation]**:IQ:ARB:MODE**

Description: Controls ARB generation. CONTinuous generates the selected waveform continuously. A SINGle command generates one cycle of the selected waveform. MULTiple outputs a set number of cycles.

Parameters: <CPD>

Valid values: SINGle | CONTinuous | MULTiple

*RST sets: CONT

[SOURce][:MODulation]**:IQ:ARB:MODE**?

Description: Returns the ARB generation mode.

Parameters: None

Response: <CRD>

Returned values: SING | CONT | MULT

[SOURce][:MODulation]**:IQ:ARB:REP**eat

Description: Only used when IQ:ARB:MODE is set to MULTiple. Defines the number of repeats of the waveform. The waveform outputs once, then repeats for the number of times defined.

Parameters: <NRf>

Valid values: 000 to 255

*RST sets: 000

[SOURce][:MODulation]**:IQ:ARB:REP**eat?

Description: Returns the number of repeats requested.

Parameters: None

Response: <NR1>

Returned values: Number of repeats

[SOURce][:MODulation]**:IQ:ARB:REST**art

Description: Defines whether a waveform already playing can be restarted by the trigger input.

Parameters: <CPD>

Valid values: ENABle | DISable

*RST sets: DIS

[SOURce][:MODulation]**:IQ:ARB:REST**art?

Description: Returns whether a waveform already playing can be restarted by the trigger input.

Parameters: <CPD>

Response: <CRD>

Returned values: ENAB | DIS

[SOURce][:MODulation]**:IQ:ARB:TRIG**ger

Description: Sets the trigger mode to immediate; start; start then stop; gated.

Parameters: <CPD>

Valid values: IMMediate | STARt | SSTOP | GATed

*RST sets: IMM

[SOURce][:MODulation]**:IQ:ARB:TRIG**ger?

Description: Returns the trigger mode.

Parameters: None

Response: <CRD>

Returned values: IMM | STAR | SSTOP | GAT

[SOURce][:MODulation]**:IQ:ARB:WAV**eform**:CAT**alog?

Description: Returns memory available and a list of files.

Parameters: None

Response: <numeric_value>,<numeric_value>,<numeric_value>{,<string>}

<Free narrow sectors>,<Free wide sectors>,<Memory available>,{File list}

The string for each file is <name> (in character data)

Returned values: Free narrow sectors: the number of sectors (and therefore the number of low sample-rate files) that can be stored.

Free wide sectors: the space left for larger high sample-rate files.

Memory available: number of samples that can be stored in the largest contiguous block.

File list: list of filenames, separated by commas.

Example: `:IQ:ARB:WAV:CAT? 5111808,’is95_1.aiq’,’is95_2.aiq’`

[SOURce][:MODulation]**:IQ:ARB:WAV**eform**:DEL**ete**:ALL**

Description: Deletes all the user files in the ARB, without removing calibration files.

Parameters: None

[SOURce][:MODulation]**:IQ:ARB:WAV**eform**:DEL**ete[:FILE]

Description: Deletes the named file.

Parameters: <string program data>

Valid values: ARB filename

Example: `:IQ:ARB:WAV:DEL ‘is95.aiq’`

[SOURce][:MODulation]**:IQ:ARB:WAV**eform**:LOAD**

Description: Loads the named file from the hard drive to the ARB memory.

Parameters: <string program data>

Valid values: ARB filename

Example: `:IQ:ARB:WAV:LOAD ‘is95.aiq’`

[SOURce][:MODulation]**:IQ:ARB:WAV**eform**:SEL**ect

Description: Selects the named file in order to play the waveform.

Parameters: <string program data>

Valid values: ARB filename

Example: `:IQ:ARB:WAV:SEL ‘is95.aiq’`

[SOURce][:MODulation]**:IQ:ARB:WAV**eform**:SEL**ect?

Description: Returns the name of the selected ARB file.

Parameters: None

Response: <string response data>

Returned values: ARB filename

Example: `:IQ:ARB:WAV:SEL? ‘is95.aiq’`

[SOURce][:MODulation]**:IQ:SOUR**ce

Description: Selects either an internal or external source to generate modulation.

Parameters: <CPD>

Valid values: EANalog | ARB

Example: :IQ:SOUR ARB

[SOURce][:MODulation]**:IQ:SOUR**ce?

Description: Returns the IQ modulation source.

Parameters: None

Response: <CRD>

Returned values: EAN | ARB

Example: :IQ:SOUR?

[SOURce][:MODulation]**:IQ:STAT**e

Description: Turns the ARB path on or off.

Parameters: <Boolean>

Valid values: OFF | ON | 0 | 1

Example: :IQ:STAT ON

[SOURce][:MODulation]**:IQ:STAT**e?

Description: Queries whether the ARB path is on (1) or off (0).

Parameters: None

Response: <Boolean>

Returned values: 0 | 1

Example: :IQ:STAT?

Example commands for playing an ARB waveform

This example demonstrates how to set up the instrument to play an ARB waveform.

Command	Result
IQ:SOURCE ARB	selects Digital mode
IQ:STATE ON	turns ARB paths on
OUTPUT:MOD ON	turns global modulation on
IQ:ARB:WAVEFORM:LOAD "*.aiq"	transfers waveform from HDD to ARB memory
IQ:ARB:WAVEFORM:SELECT "*.aiq"	selects file to be played
IQ:ARB:INITIATE	starts ARB generation

Calibration commands

(CALibration subsystem)

Most calibration commands are included in the Maintenance Manual, as they are likely to be used only at routine calibration intervals or after servicing. The following command may however be useful during everyday operation.

```
CALibration
        :IQUSer
            :ADJust
```

CALibration**:IQUS**er**:ADJ**ust

Description: Performs a user IQ calibration at the current settings.

AEROFLEX LIMITED
SOFTWARE LICENCE AND WARRANTY

This document is an Agreement between the user of this Licensed Software, the Licensee, and Aeroflex Limited (‘Aeroflex’), the Licensor. By installing or commencing to use the Licensed Software you accept the terms of this Agreement. If you do not agree to the terms of this Agreement do not use the Licensed Software.

1. DEFINITIONS

The following expressions will have the meanings set out below for the purposes of this Agreement:

Add-In Application Software	Licensed Software that may be loaded separately from time to time into the Designated Equipment to improve or modify its functionality
Computer Application Software	Licensed Software supplied to run on a standard PC or workstation
Designated Equipment	means either: the single piece of equipment or system supplied by Aeroflex upon which the Licensed Software is installed; or a computer that is connected to a single piece of equipment or system supplied by Aeroflex upon which computer the Licensed Software is installed
Downloaded Software	any software downloaded from an Aeroflex web site
Embedded Software	Licensed Software that forms part of the Designated Equipment supplied by Aeroflex and without which the Equipment cannot function
Licence Fee	means either the fee paid or other consideration given to Aeroflex for the use of the Licensed Software on the Designated Equipment
Licensed Software	all and any programs, listings, flow charts and instructions in whole or in part including Add-in, Computer Application, Downloaded and Embedded Software supplied to work with Designated Equipment
Licensee	the organization or individual that is the user of the Licensed Software
PXI Software	Licensed Software specific to Aeroflex’s 3000 Series PXI product range

2. LICENCE FEE

The Licensee shall pay the Licence Fee to Aeroflex in accordance with the terms of the contract between the Licensee and Aeroflex.

3. TERM

This Agreement shall be effective from the date of receipt or download (where applicable) of the Licensed Software by the Licensee and shall continue in force until terminated under the provisions of Clause 8.

4. LICENCE

4.1 The following rights and restrictions in this Article 4 apply to all Licensed Software unless otherwise expressly stated in other Articles of this Agreement.

4.2 Unless and until terminated, this Licence confers upon the Licensee the non-transferable and non-exclusive right to use the Licensed Software on the Designated Equipment.

4.3 Neither the Licensed Software nor any information provided by Aeroflex to the Licensee nor any licence and rights granted hereunder, may be sold, leased, assigned, sublicensed, electronically distributed, timeshared or otherwise transferred, in whole or in part by the Licensee other than as specified in this Licence without the prior written consent of Aeroflex. Such consent may be withheld at Aeroflex's sole discretion.

4.4 The Licensee may not use the Licensed Software on other than the Designated Equipment, unless written permission is first obtained from Aeroflex and until the appropriate additional Licence Fee has been paid to Aeroflex.

4.5 The Licensee may not amend or alter the Licensed Software and shall have no right or licence other than that stipulated herein.

4.6 Except as specifically permitted elsewhere in this Agreement the Licensee may make not more than two copies of the Licensed Software (but not the Authoring and Language Manuals) in machine-readable form for operational security and shall ensure that all such copies include Aeroflex's copyright notice, together with any features which disclose the name of the Licensed Software and the Licensee. Furthermore, the Licensee shall not permit the Licensed Software or any part to be disclosed in any form to any third party and shall maintain the Licensed Software in secure premises to prevent any unauthorized disclosure. The Licensee shall notify Aeroflex immediately if the Licensee has knowledge that any unlicensed party possesses the Licensed Software. The Licensee's obligation to maintain confidentiality shall cease when the Licensed Software and all copies have been destroyed or returned. The copyright in the Licensed Software shall remain with Aeroflex. The Licensee will permit Aeroflex at all reasonable times to audit the use of the Licensed Software.

4.7 The Licensee will not disassemble or reverse engineer the Licensed Software, nor sub-license, lease, rent or part with possession or otherwise transfer the whole or any part of the Licensed Software.

5 ADDITIONAL LICENCE RIGHTS SPECIFIC TO PXI SOFTWARE

5.1 Definitions for PXI Software

The following expressions will have the meanings set out below for the purposes of the supplementary rights granted in this Article.

PXI Drivers
All 3000 Series PXI module device drivers including embedded firmware that are installed at runtime

PXI Executable Applications
All executable applications supplied with each 3000 Series PXI module including:-

PXI Studio

Soft Front Panels (manual operation graphical user interfaces)

Utilities including: RF Investigator, PXI Version Information and Self Test

PXI Spectrum Analysis Library
The spectrum analysis measurement suite library .dll software supplied with each 3000 Series PXI module

PXI Optional Application Library
Individual measurement suite available from a range of optional .dll application libraries

5.2 PXI Drivers, PXI Executable Applications and PXI Spectrum Analysis Library Licence Rights

Subject to the licence granted in Article 4 hereof notwithstanding the limitations on number of copies in Clause 4.5 hereof, the Licensee is entitled to make and distribute as many copies of the PXI Drivers and PXI Executable Applications as necessary for use with 3000 Series PXI modules acquired by the Licensee from Aeroflex or its authorized distributor or reseller provided that the Licensee may not sell or charge a fee for the PXI Drivers and PXI Executable Applications.

5.3 PXI Optional Application Library Licence Rights

Subject to the licence granted in Article 4 hereof notwithstanding the limitations on number of copies in Clause 4.5 hereof, the Licensee is entitled to distribute as many copies of any PXI Optional Application Library as necessary for use with 3000 Series PXI modules acquired by the Licensee from Aeroflex or its authorized distributor or reseller provided that:

5.3.1 copies of the applicable PXI Optional Application Library are used solely with 3000 Series PXI modules which the customer has purchased with the corresponding option or part number for the applicable PXI Optional Application Library; and

5.3.2 the Licensee may not sell or charge a fee for the PXI Optional Application Library.

6 WARRANTY

6.1 Aeroflex certifies that the Licensed Software supplied by Aeroflex will at the time of delivery function substantially in accordance with the applicable Software Product Descriptions, Data Sheets or Product Specifications published by Aeroflex.

6.2 The warranty period (unless an extended warranty for Embedded Software has been purchased) from date of delivery in respect of each type of Licensed Software is:

PXI Drivers	24 months
Embedded Software	12 months
Add-In Application Software	90 days
Computer Application Software	90 days
Downloaded Software	No warranty

6.3 If during the appropriate Warranty Period the Licensed Software does not conform substantially to the Software Product Descriptions, Data Sheets or Product Specifications Aeroflex will provide:

6.3.1 In the case of Embedded Software and at Aeroflex’s discretion either a fix for the problem or an effective and efficient work-around.

6.3.2 In the case of Add-In Application Software and Computer Application Software and at Aeroflex’s discretion replacement of the software or a fix for the problem or an effective and efficient work-around.

6.4 Aeroflex does not warrant that the operation of any Licensed Software will be uninterrupted or error free.

6.5 The above Warranty does not apply to:

6.5.1 Defects resulting from software not supplied by Aeroflex, from unauthorized modification or misuse or from operation outside of the specification.

6.5.2 Third party produced proprietary software ('Third Party Software') which Aeroflex may deliver with its products (where such Third Party Software carries a more limited warranty than the above warranty). In such case Aeroflex will provide a remedy for any non-conformance of the Third Party Software commensurate with the third party's warranty to Aeroflex (if any).

6.6 The remedies offered above are sole and exclusive remedies and to the extent permitted by applicable law are in lieu of any implied conditions, guarantees or warranties whatsoever and whether statutory or otherwise as to the Licensed Software all of which are hereby expressly excluded.

7. INDEMNITY

7.1 Aeroflex shall defend, at its expense, any action brought against the Licensee alleging that the Licensed Software infringes any patent, registered design, trademark or copyright, and shall pay all Licensee's costs and damages finally awarded up to an aggregate equivalent to the Licence Fee provided the Licensee shall not have done or permitted to be done anything which may have been or become any such infringement and shall have exercised reasonable care in protecting the same failing which the Licensee shall indemnify Aeroflex against all claims costs and damages incurred and that Aeroflex is given prompt written notice of such claim and given information, reasonable assistance and sole authority to defend or settle such claim on behalf of the Licensee. In the defense or settlement of any such claim, Aeroflex may obtain for the Licensee the right to continue using the Licensed Software or replace it or modify it so that it becomes non-infringing.

7.2 Aeroflex shall not be liable if the alleged infringement:

7.2.1 is based upon the use of the Licensed Software in combination with other software not furnished by Aeroflex, or

7.2.2 is based upon the use of the Licensed Software alone or in combination with other software in equipment not functionally identical to the Designated Equipment, or

7.2.3 arises as a result of Aeroflex having followed a properly authorized design or instruction of the Licensee, or

7.2.4 arises out of the use of the Licensed Software in a country other than the one disclosed to Aeroflex as the intended country of use of the Licensed Software at the commencement of this Agreement.

7.3 Aeroflex shall not be liable to the Licensee for any loss of use or for loss of profits or of contracts arising directly or indirectly out of any such infringement of patent, registered design, trademark or copyright.

7.4 Other than as may be covered by the indemnity provisions of Clauses 7.1, 7.2 and 7.3 Aeroflex shall not, under any circumstances, be liable to the Licensee for any direct, indirect, special, consequential and/or contingent loss or damage (and such loss or damage shall include without limitation loss of use or profit, loss of revenue, loss of product, liquidated damages or penalties, economic loss, delay in operations, loss of contracts or loss of business) whether or not the same are foreseeable and whether arising out of breach of contract, tort, statutory duty or otherwise. The total liability of Aeroflex and its employees, in contract, tort, or otherwise (including negligence, warranty, indemnity, and strict liability) howsoever arising out of this Licence shall be limited to the total amount of the Licence Fee and total support fees actually paid to Aeroflex by the Licensee.

8. TERMINATION

8.1 Notwithstanding anything herein to the contrary, this Licence shall forthwith determine if the Licensee:

8.1.1 As an individual has a Receiving Order made against him or is adjudicated bankrupt or compounds with creditors or as a corporate body, compounds with creditors or has a winding-up order made against it or

8.1.2 Parts with possession of the Designated Equipment.

8.2 This Licence may be terminated by notice in writing to the Licensee if the Licensee shall be in breach of any of its obligations hereunder and continue in such breach for a period of 21 days after notice thereof has been served on the Licensee.

8.3 On termination of this Agreement for any reason, Aeroflex may require the Licensee to return to Aeroflex all copies of the Licensed Software in the custody of the Licensee and the Licensee shall, at its own cost and expense, comply with such requirement within 14 days and shall, at the same time, certify to Aeroflex in writing that all copies of the Licensed Software in whatever form have been obliterated from the Designated Equipment.

9. THIRD PARTY LICENCES

9.1 The Licensed Software or part thereof may be the proprietary property of third party licensors. In such an event such third party licensors (as may be referenced on the software media, or any on screen message on start up of the software or on the order acknowledgement) and/or Aeroflex may directly enforce the terms of this Agreement and may terminate the Agreement if the Licensee is in breach of the conditions contained herein.

9.2 If any third party software supplied with the Licensed Software is supplied with, or contains or displays the third party’s own licence terms then the Licensee shall abide by such third party licence terms (for the purpose of this Article the term “third party” shall include other companies within the Aeroflex group of companies).

10. EXPORT REGULATIONS

The Licensee undertakes that where necessary the Licensee will conform with all relevant export regulations imposed by the Governments of the United Kingdom and/or the United State of America.

11. U.S. GOVERNMENT RESTRICTED RIGHTS

The Licensed Software and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFAR 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software-Restricted Rights at 48 CFR 52.227-19, as applicable.

12. NOTICES

Any notice to be given by the Licensee to Aeroflex shall be addressed to:

Aeroflex Limited, Longacres House, Six Hills Way, Stevenage, SG1 2AN, UK.

13. LAW AND JURISDICTION

This Agreement shall be governed by the laws of England and shall be subject to the exclusive jurisdiction of the English courts. This agreement constitutes the whole agreement between the parties and may be changed only by a written agreement signed by both parties.

CHINA Beijing
Tel: [+86] (10) 6539 1166
Fax: [+86] (10) 6539 1778

CHINA Shanghai
Tel: [+86] (21) 2028 3588
Fax: [+86] (21) 2028 3558

CHINA Shenzhen
Tel: [+86] (755) 3301 9358
Fax: [+86] (755) 3301 9356

FINLAND
Tel: [+358] (9) 2709 5541
Fax: [+358] (9) 804 2441

FRANCE
Tel: [+33] 1 60 79 96 00
Fax: [+33] 1 60 77 69 22

GERMANY
Tel: [+49] 89 99641 0
Fax: [+49] 89 99641 160

HONG KONG
Tel: [+852] 2832 7988
Fax: [+852] 2834 5364

INDIA
Tel: [+91] 80 [4] 115 4501
Fax: [+91] 80 [4] 115 4502

JAPAN
Tel: [+81] (3) 3500 5591
Fax: [+81] (3) 3500 5592

KOREA
Tel: [+82] (2) 3424 2719
Fax: [+82] (2) 3424 8620

SCANDINAVIA
Tel: [+45] 9614 0045
Fax: [+45] 9614 0047

SINGAPORE
Tel: [+65] 6873 0991
Fax: [+65] 6873 0992

TAIWAN
Tel: [+886] 2 2698 8058
Fax: [+886] 2 2698 8050

UK Stevenage
Tel: [+44] (0) 1438 742200
Fax: [+44] (0) 1438 727601
Freephone: 0800 282388

USA
Tel: [+1] (316) 522 4981
Fax: [+1] (316) 522 1360
Toll Free: (800) 835 2352

As we are always seeking to improve our products, the information in this document gives only a general indication of the product capacity, performance and suitability, none of which shall form part of any contract. We reserve the right to make design changes without notice. All trandemarks are acknowledged. Parent company Aeroflex, Inc.

web: **www.aeroflex.com**   Email: **info-test@aeroflex.com**

April 2012