System Two GPIB Programmer's Reference Manual

Audio precision®

System Two GPIB Programmer's Reference Guide

Revision 3 for GPIB Firmware Version 2.002
June 2000
Audio Precision PN 8211.0046

Published by:

Audio Precision, Inc.
PO Box 2209
Beaverton, Oregon 97075-2209
U.S. Toll Free: 1-800-231-7350
Tel: (503) 627-0832 Fax: (503) 641-8906
email: techsupport@audioprecision.com
Web: www.audioprecision.com

Printed in the United States of America

Contents

1. General Information

Scope of This Manual

This Programmer’s Reference Manual describes how to control System Two via the GPIB Interface.

This manual provides information on the following key topics:

- Establishing GPIB communications
- GPIB programming and command reference
- Example programs

Even though you may be familiar with GPIB programming concepts, we recommend that you first read the **Establishing GPIB Communications** portions of this manual.

This manual does not provide a detailed description of instrument operation. Summary descriptions are provided to explain the basic function of the GPIB commands. Refer to the **APWIN User’s Manual for System Two** or other documentation listed below for detailed information about operation of the instrument.

The GPIB firmware in your instrument implements features of APWIN Version 1.52a. Appendix E, "Firmware Information," describes features of APWIN that are not available in the GPIB interface firmware at this time. This appendix also provides additional information about specific versions of the firmware.

Related Documentation

APWIN User’s Manual 8211.0006
Contains a comprehensive description of the full capabilities of APWIN software for System Two.

APWIN BASIC (procedure language) User’s Manual and Programmer’s Reference
Volume 1 (Introduction and Language Reference) 8211.0002
Volume 2 (Command Extensions A – R) 8211.0012
Volume 3 (Command Extensions S – Z) 8211.0025
These three volumes provide detailed descriptions and syntax for all APWIN commands.

APWIN Simplified Tutorial, Multitone, & Procedures for System Two 8211.0014
Designed to lead you through your first operating session with APWIN and System Two.

APWIN Installation & Getting Started for System Two 8211.0026
Guides hardware and software installation for APWIN and System Two setup.

System Two GPIB Option Description and Installation Manual 8211.0051
Describes basic hardware considerations for installing the components, such as power line voltage settings, fuse information, and GPIB control interface setup.

System Two Service Manual .... AP Part No. 8210.S201
Contains detailed System Two information, including adjustment procedures, diagnostic procedures, and drawings of electrical and mechanical parts. This manual is not required for routine understanding or operation and must be purchased separately.

Audio Measurement Handbook .... AP Part No. 8200.AMH1

Intended as a practical, hands-on assistance for workers in all phases of the audio field. Describes general measurement techniques and includes a glossary of specific audio terminology and test definitions.

System Two Overview

The System Two is a comprehensive audio test set that provides analog and digital audio generator and analyzer measurement functions. System Two may be controlled with APWIN software running in a Windows 95/98/NT operating system or with a GPIB controller and application software. The control method is determined by the setting of a rear panel switch.

The System Two GPIB Software Development Process

A practical working knowledge of System Two is required in order to develop effective GPIB software for it. Figure 1-1 illustrates a typical software development scenario in which Audio Precision APWIN software is used to develop expertise with the System Two before attempting to integrate it into a larger system with your GPIB interface software programs.

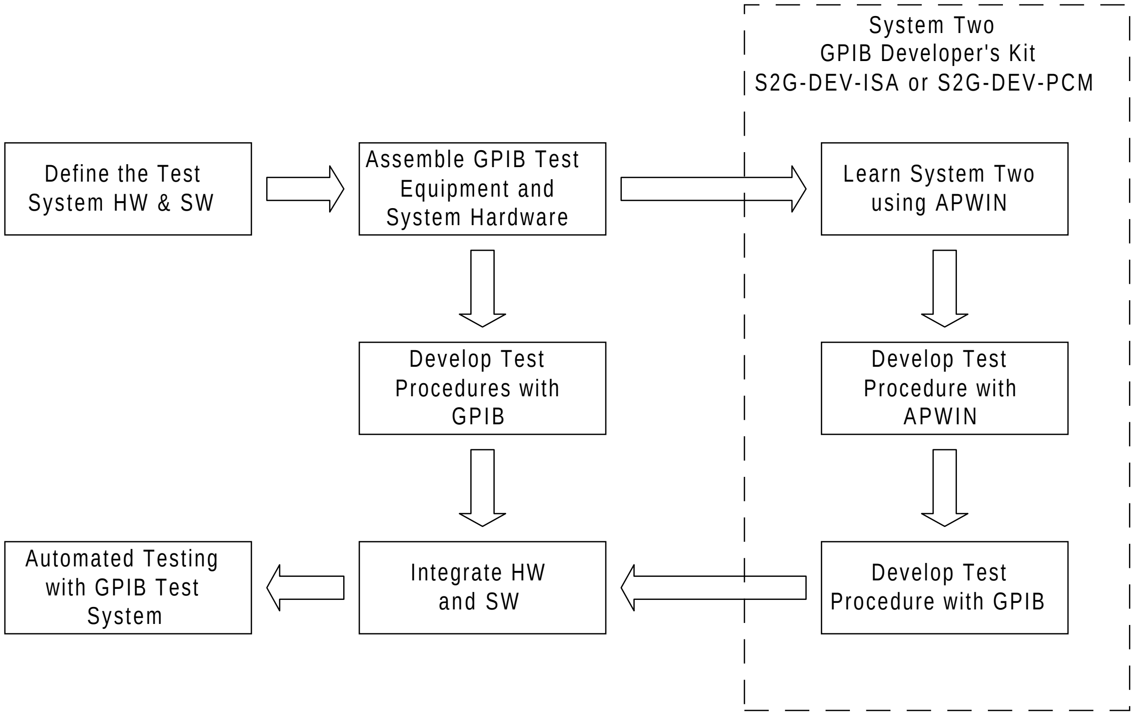
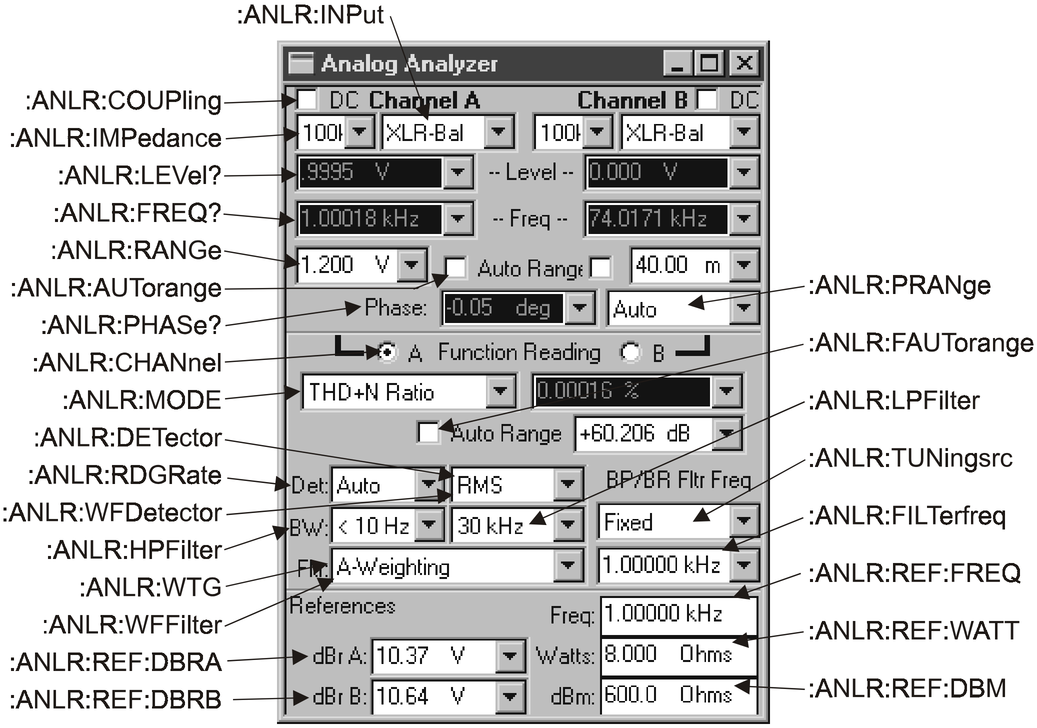

System Two
GPIB Developer's Kit
S2G-DEV-ISA or S2G-DEV-PCM
Define the Test System HW & SW
Assemble GPIB Test Equipment and System Hardware
Learn System Two using APWIN
Develop Test Procedures with GPIB
Develop Test Procedure with APWIN
Automated Testing with GPIB Test System
Integrate HW and SW
Develop Test Procedure with GPIB

Figure 1-1. Recommended GPIB software development process using APWIN

Audio Precision provides two software developer's kits that provide everything you need to begin developing audio tests for the System Two: the S2G-DEV-ISA and the S2G-DEV-PCM. Both of these kits provide an APIB interface for your computer, the APWIN manuals and this manual, and a CD ROM with the APWIN software and GPIB Sample Software for the System Two. APWIN software provides the interactive "front panel" for System Two. APWIN is designed to run on WIN 95, WIN 98, or Win NT 4.0 with an interface card that interfaces to the APIB port connector of the System Two. The GPIB programming commands for System Two are modeled directly on the APWIN user interface, thus knowledge of the APWIN user interface is a prerequisite for efficient use of the GPIB commands.

The fastest development scenario for GPIB software involves using APWIN with SystemTwo. Use APWIN Basic and the Learn Mode button on the tool bar to develop APWIN Basic procedures that perform the tests you wish to develop for GPIB. If you use APWIN to develop your audio tests you will quickly learn how to take the best advantage of System Two. With the interactive environment of APWIN you will quickly develop test methods for your device under test and build confidence that your methods are correct. Once you have done this, the task of developing equivalent code for the GPIB port will be greatly simplified.

The command description sections of this manual show APWIN control panels for each of the major subsections of the instrument. Each panel is illustrated with GPIB command call-outs to help you convert your APWIN setup to GPIB commands. Figure 1-2 illustrates this for the Analog Analyzer panel in Section 5.



Analog Analyzer

☐ DC Channel A Channel B ☐ DC

100k XLR-Bal 100k XLR-Bal

.9995 V -- Level -- 0.000 V

1.00018 kHz -- Freq -- 74.0171 kHz

1.200 V ☐ Auto Range ☐ 40.00 m

Phase: -0.05 deg Auto

● A Function Reading ○ B

THD+N Ratio 0.00016 %

☐ Auto Range +60.206 dB

Det: Auto RMS BP/BR Fltr Freq

BW: < 10 Hz 30 kHz Fixed

Fltr: A-Weighting 1.00000 kHz

References Freq: 1.00000 kHz

dBr A: 10.37 V Watts: 8.000 Ohms

dBr B: 10.64 V dBm: 600.0 Ohms

Figure 1-2. APWIN Analog Analyzer Panel with GPIB Command Call-outs

GPIB and APIB Control Modes for Software Development

Your SystemTwo with GPIB option installed has an APIB port and a GPIB port on the rear panel. The rear panel switches provide a means to configure either port to be used to control the instrument. During the software development process you may switch between the two modes in order to take advantage of the interactive development environment of APWIN.

Only one port may be used as the control port. The red LED on the rear panel labeled GPIB adjacent to the GPIB connector indicates which port is in control. If it is ON then it indicates that the GPIB port is in control and the APIB controller card in your computer must be disconnected from the APIB port on the instrument. Note that the instrument uses the APIB port to control other APIB products from Audio Precision. Figure 1-3 illustrates how to connect these products to the System Two and how to connect an APIB controller in a computer to this system when the GPIB port is disabled.

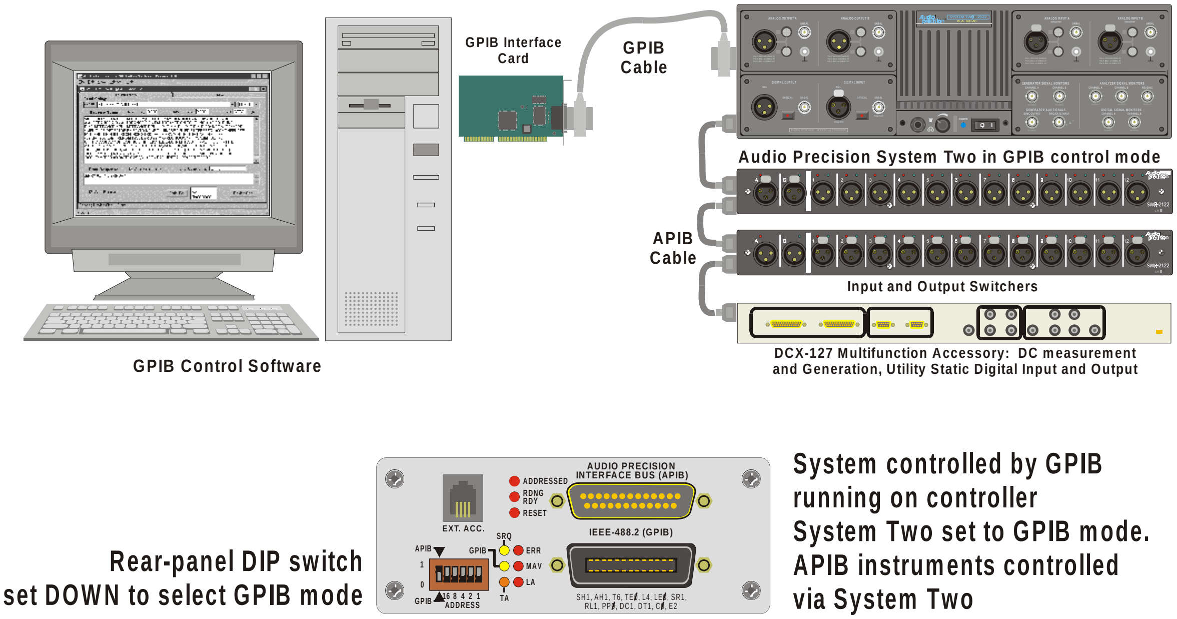

GPIB Interface Card
GPIB Cable
Audio Precision System Two in GPIB control mode
APIB Cable
Input and Output Switchers
DCX-127 Multifunction Accessory: DC measurement and Generation, Utility Static Digital Input and Output
GPIB Control Software
EXT. ACC.
ADDRESSED
RDNG RDY
RESET
AUDIO PRECISION INTERFACE BUS (APIB)
IEEE-488.2 (GPIB)
SRQ
ERR
MAV
LA
TA
APIB
GPIB
1
0
16 8 4 2 1
ADDRESS
SH1, AH1, T6, TE0, L4, LE0, SR1, RL1, PP0, DC1, DT1, C0, E2
Rear-panel DIP switch set DOWN to select GPIB mode
System controlled by GPIB running on controller
System Two set to GPIB mode.
APIB instruments controlled via System Two

Figure 1-3. APIB connections to System Two with GPIB option in GPIB control mode. SWR-2122 Switcher and DCX-127 connected to the APIB connector of the System Two. Computer APIB cable not connected.

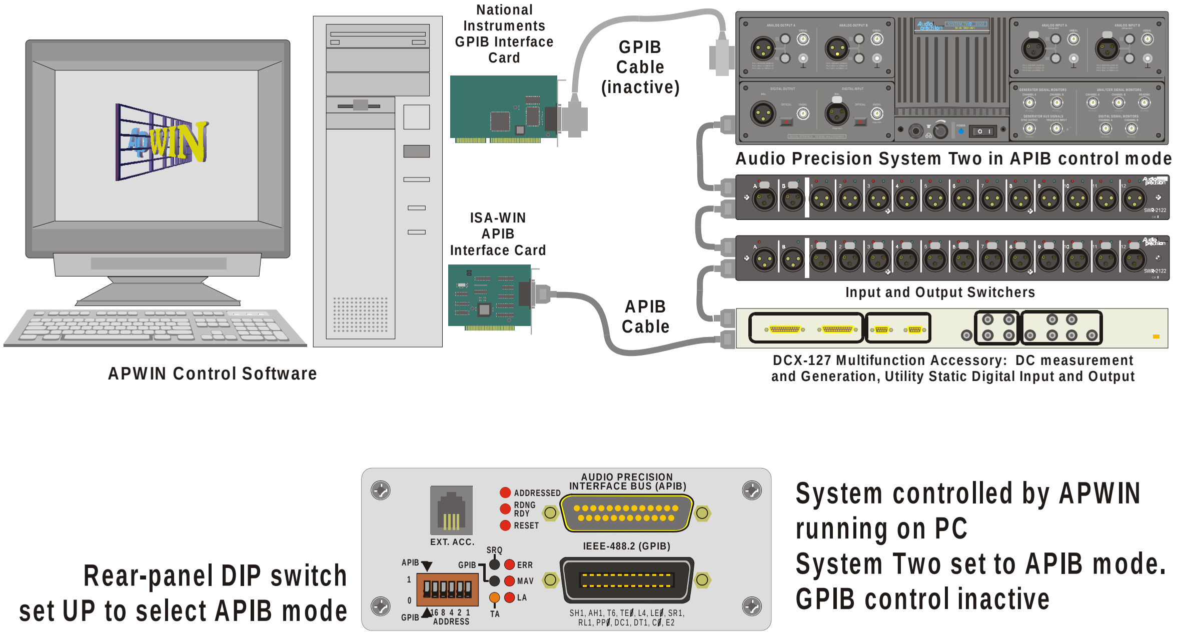

National Instruments GPIB Interface Card
GPIB Cable (inactive)
Audio Precision System Two in APIB control mode
ISA-WIN APIB Interface Card
APIB Cable
Input and Output Switchers
DCX-127 Multifunction Accessory: DC measurement and Generation, Utility Static Digital Input and Output
APWIN Control Software
APWIN
EXT. ACC.
ADDRESSED
RDNG RDY
RESET
AUDIO PRECISION INTERFACE BUS (APIB)
IEEE-488.2 (GPIB)
SRQ
ERR
MAV
LA
TA
APIB
GPIB
1
0
16 8 4 2 1
ADDRESS
SH1, AH1, T6, TE0, L4, LE0, SR1, RL1, PP0, DC1, DT1, C0, E2
Rear-panel DIP switch set UP to select APIB mode
System controlled by APWIN running on PC
System Two set to APIB mode.
GPIB control inactive

Figure 1-4. APIB controller connections, System Two with GPIB option in APIB control mode. Computer APIB cable connected to DCX-127, then to SWR-2122, then to System Two.

Establishing GPIB Communication

GPIB Connection

The System Two with GPIB option installed has a 24-pin GPIB-compatible connector on the rear panel. This D-shell connector conforms to the mechanical requirements of IEEE-488.1-1987. The instrument is connected to the instrument controller via a GPIB cable. The instrument controller (a computer) must have a corresponding GPIB interface port. The GPIB cables are designed so they can be stacked if needed to connect multiple instruments into your GPIB system.

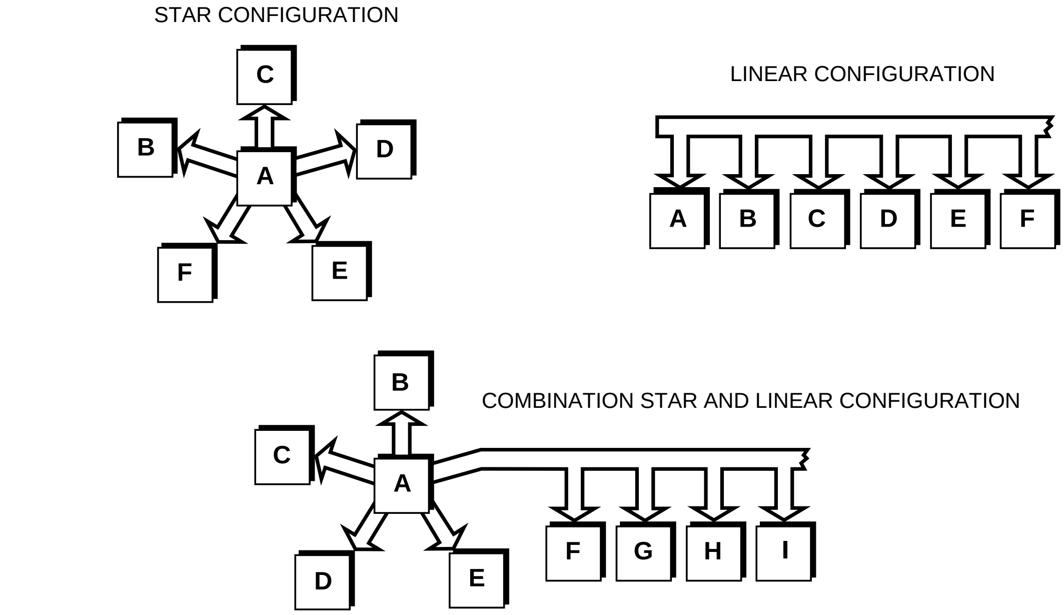


Figure 1-5. GPIB devices may be connected in star, linear, or combination star/linear configurations

When connecting instruments into a GPIB system, observe the following rules:

- Connect and disconnect instruments from the bus only when the power to all instruments in the system is off.
- Assign a unique GPIB address to each instrument (device) on the bus.
- Devices may be connected in a star or linear configuration (see Figure 1-5), or a combination of star and linear configuration.
- Do not attach more than 15 devices (including the controller) to one bus.
- One device must be attached to the bus for every two meters (6 feet) of cable.
- Total cable length must not exceed 20 meters (66 feet).
- At least two-thirds of the devices on the bus must be powered up for operation.

GPIB Address and I/O Mode Switch

The instrument must be set to a unique GPIB address, one that is not used by any other device on the bus.

An address select switch with six slide switches sets the GPIB address and the I/O Mode. The switch is shown below in Figure 1-6. The five switches on the right labeled 1 to 16 comprise the 5-bit binary primary address of the instrument. Legal addresses are 0 through 30. Set each switch up for a binary 1 or down for a binary 0.

The left bit of the select switch sets the I/O mode independent of the GPIB address switch settings. Set the switch to 0 (down) to set the I/O mode to GPIB. The APIB port must not be connected to an Audio Precision APIB interface card when the GPIB mode is selected. Set the switch to 1 (up) to set the I/O mode to APIB. The APIB mode disables the GPIB interface (high-impedance state) and permits an Audio Precision APIB interface card in a computer to control the System Two.

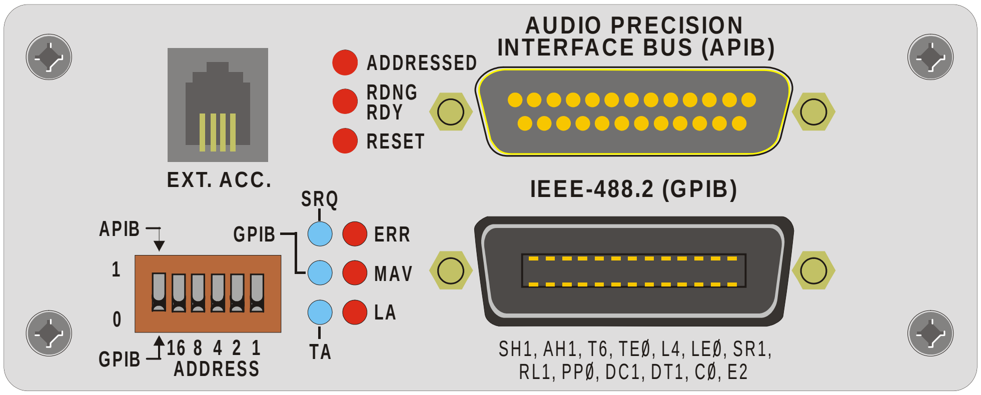


Figure 1-6. System Two Rear Panel Connector – Address Switch, Status LEDs, GPIB Port, and APIB Port

GPIB Status LEDs

The GPIB Status LEDs, indicate the current status of the GPIB bus. See figure 1-6 above. Status indication is as follows when the LED is illuminated:

SRQ The instrument has asserted the SRQ line in order to request service from the GPIB controller (SRQ interrupt).

GPIB The instrument is under the control of the GPIB interface according to the setting of the I/O Mode switch. The APIB bus connector will be driven internally by the instrument GPIB interface board.

TA Talk Addressed – The instrument is talk addressed by the GPIB controller.

LA Listen Addressed – The instrument is listen addressed by the GPIB controller.

MAV The instrument has one or more bytes in the output queue and has asserted the MAV bit in the Status Byte Register.

ERR The instrument has detected an error condition that has not been reported to the system controller. This LED goes off when the error condition has been reported.

GPIB Program Message Terminators <PMT>

Two program message terminators are supported when the instrument is addressed as a listener: EOI line asserted with the NL character (ASCII 10, the linefeed character), or EOI line asserted with the last byte of a message. The NL character alone without EOI is not supported.

The program message terminator when the instrument is addressed as a talker is the NL character sent with the EOI line asserted.

Command Structure and Syntax

Program Messages

Control from a GPIB controller is accomplished by sending program messages to the instrument and receiving measurements and status messages from the instrument. A program message is a sequence of program message units separated by the semicolon (;) character.

Program Message Units

Program message units are set commands or query commands (usually referred to as commands and queries). Set commands are used to change instrument settings. Query commands cause the instrument to respond with measurements, instrument settings, or status information.

A query command is distinguished by a trailing question mark. For example :AGEN:OUTPUT AB is the command used to turn on the analog generator outputs. The query :AGEN:OUTPUT? requests generator output setting.

Character Encoding

Commands are formed with characters described by the American Standard Code for Information Interchange (ASCII) character encoding. Appendix A shows the ASCII code chart.

Case

Commands can be formed in either upper or lower case. Commands are shown in mixed upper and lower case in this manual to show the minimum required form (see Abbreviations for more details). The instrument always responds to queries with upper case characters.

White Space

White space is used to delimit certain syntactic elements in a command. White space is defined as a single ASCII-encoded byte in the range ASCII 0 to 32 (decimal) with the exception of ASCII 10, the Line Feed (LF) character. The instrument ignores white space except as a syntactic delimiter.

Special Characters

The Line Feed (LF) character, also known as the New Line (NL) character, (ASCII 10) and all characters in the range of ASCII 127 to 255 are defined as special characters. These characters are used in arbitrary block data only; using these characters in normal ASCII commands may yield unpredictable results.

Abbreviations

All instrument commands (and queries) are 12 characters or less in length. Those that are longer than 4 characters have a long and a short form. The long form consists of the whole command. Through out this manual the short form and long form are distinguished by the user of upper and lower case letters. The short form is shown in upper case in the instrument command section of this manual. The long form consists of the short form plus the remaining lower case letters. For example, the analog analyzer coupling command would appear as ":ANLR:COUPling". The short for is ":ANLR:COUP". The long form is ":ANLR:COUPLING". The instrument does not distinguish between upper and lower case. Therefore the instrument will interpret the command ":ANLR:COUPLING" the same as ":anlr:coupling".

The instrument will only accept the short or the long form of a command mnemonic. For example, the first two commands listed below are valid, but the third command is invalid and will generate a command error.

1.	:ANLR:COUPLING	valid long form
2.	:anlr:coup	valid short form
3.	:ANLR:COUPLI	invalid form

Syntax Notation

The following symbols used in this manual describe the syntax of instrument GPIB commands:

< >	Defined element
|	Exclusive OR
{ }	Group, where one element is required
[ ]	Optional, may be omitted
…	Previous element or elements may be repeated

Syntactic Delimiters

Syntactic elements in a program message are delimited with commas, colons, white space, and semicolons.

Comma (,)	Delimits command arguments.
Colon (:)	Delimits compound command headers.
White Space ( )	Delimits a command header from its argument.
Semicolon (;)	Delimits message units.

Message Unit Syntax

A message unit is comprised of a header and associated argument(s). Commands have the syntax:

[:]<Header>[<white space><Arguments>]

Compound Command Headers

A message unit compound command header contains multiple header mnemonics delimited with colons (:). Compound command headers are used for most instrument settings and queries.

Syntax:

[:]<Header>:<Header>[<white space><Arguments>]

For example the compound header command :DGEN:WFM SINE,DUAL sets the digital generator waveform to dual sinewave.

Queries

Queries cause the instrument to return information about its status or settings. Queries may consist of simple command headers or compound command headers with a trailing question mark (?) character, followed by query arguments.

Compound Header Syntax: [:]<Header>:<Header>? [<white space><Arguments>]

Query Response Headers

You can control whether the instrument returns headers in query responses. Use the :HEADer command to control this feature. With :HEADer ON, the query response returns command headers and is formatted as a complete command that may be sent back to the instrument unaltered. With :HEADer OFF, a query response includes only the argument but no header. Query responses without headers are often desired when the response is a measurement value which is easier to convert to a number when a header string is omitted from the response. Some of the IEEE-488.2 common commands are queries that respond without headers regardless of the state of the :HEADer command.

Input / Output Deadlocks & Query Errors

The System Two has 1024 bytes (characters) of input queue memory space for commands and queries. The output queue has 1024 bytes of memory space for query responses. Multiple queries will be processed in the order received until either the input queue is full, the output queue is full, or both queues are full.

An I/O deadlock can occur if the both the input queue and the output queue become full. The output queue is cleared automatically and a query error is generated if an I/O deadlock occurs or if the output queue is full or becomes full when another query command is processed.

A Query Error will be generated if an attempt is made to read an empty output queue. This type of query error can be avoided by checking the MAV bit in the status byte register prior to an attempt to read the output queue.

A Query Error will be generated if a query terminated with the <END> message results in a response in the output queue and then another query is received before the entire response to the first query is read from the output queue. The output queue will be deleted and the QYE bit (bit 2) will be set in the Event Status Register to indicate that a query error has occurred.

Verbose and Terse Query Responses

The VERBOSE OFF command specifies the abbreviated form for query response headers and arguments. The VERBOSE ON command (default) specifies that query response headers and arguments will be in the long form of the command. The VERBOSE OFF form has the advantage of reducing the number of characters sent in response to queries, thus reducing the amount of time to complete the GPIB data transfer.

Concatenating Message Units

Command messages may be constructed of multiple message units that are delimited from each other with the semicolon character (;). The instrument executes concatenated message units in the order received.

When concatenating message units into a complete command message, you should follow these rules:

1. Separate completely different headers by a semicolon. For example, the commands :AGEN:OUTPUT AB and :ANLR:INPUT A,GENMON can be concatenated into a single command message: :AGEN:OUTPUT AB;:ANLR:INPUT A,GENMON.

2. If concatenated message units have compound headers that differ by only the last mnemonic, you can eliminate the duplicate compound header mnemonic. For example, you can concatenate the commands :AGEN:OUTPUT AB and :AGEN:FREQ 1000HZ into a single command: :AGEN:OUTPUT AB;FREQ 1000HZ instead of :AGEN:OUTPUT AB;:AGEN:FREQ 1000HZ.

3. With concatenated queries, the responses to all the queries are concatenated into a single response message. For example, the concatenated query :HEADER ON;:AGEN:OUTPUT?;FREQ? HZ will respond with :AGEN:OUTPUT AB;:AGEN:FREQ 1000HZ and the command string :HEADER OFF;:AGEN:OUTPUT?;FREQ? HZ will respond with AB;1000HZ.

4. Commands and queries may be concatenated in the same command message. For example: :ANLR:INPUT A,GENMON;:AGEN:OUTPUT? is a valid message that sets the analyzer input source to genmon and queries the generator output state. Concatenated commands and queries are executed in the order received.

* Common Commands

Commands and queries that have an asterisk (*) preceding the header are common commands. Common commands are defined by the IEEE-488.2 standard and some are required by all devices that support the standard. The System Two implements all required common commands and many optional common commands.

Command Arguments

A command argument (sometimes called program data) is a quantity, quality, restriction, or limit associated with the command header. The argument is one of the following types depending upon the command header:

- Mnemonic

Decimal Numeric

Arbitrary Block Data

Mnemonic

A mnemonic is an ASCII coded alpha-numeric string of 12 characters or less that represents one of the possible argument values. Mnemonics always begin with an alpha character.

Decimal Numeric

The instrument defines a decimal numeric argument expressed as <nrf> (Numeric Representation flexible). An nrf formatted number may be any of the formats shown in Table 1-1.

Table 1-1. nrf-formatted number formats

Format	Example
implicit point <nr1>	1, +3, -2, +10
explicit point unscaled <nr2>	1, +2.8, -0.021, +98.65
explicit point scaled <nr3>	1E+3, 9.8 E-3, +1.53E+2,-6.005E+3

Arbitrary Block Data

Arbitrary block data is used for command macro definitions or DSP waveform data. This data may be formatted as binary data or as ASCII data. The arbitrary block data format uses a # character followed by a byte count field followed by the data bytes. The byte count field encodes the number of bytes in the data to follow. The byte count field consists of two fields, a one byte decimal digit that encodes the number of byte count digits to follow, then the byte count digits. This type of arbitrary block data is called <definite length arb block data> because the byte count must indicate the exact number of data bytes.

For example, send a command to define a macro that sets the analog generator outputs on with a <definite length arb block data> format :

```
*DMC “AGEN_ON”,#215:AGEN:OUTPUT AB.
```

The 2 following the # character indicates that two byte count digits follow (15). The digits 15 indicate that 15 data bytes are to follow. Additional message units may be concatenated to the end of the <definite length arb block data> argument.

An alternative to the <definite length arbitrary block data> format is the <indefinite length arbitrary block data> format which requires a 0 byte count but must be terminated with the <END> message (LF with EOI asserted).

For example, send a command to define a macro that sets the analog generator outputs on with an <indefinite length arb block data> format: :*DMC “AGEN_ON”, #0:AGEN:OUTPUT AB<*END*>. This type of format does not allow additional message units to be concatenated to the end of the <indefinite length arb block data> argument.

Binary Representation of Single Precision Floating-point Numbers

The :DSP:BATCH? query, the :DSP:TABLE command, and the :DSP:TABLE? query support a 4-byte binary representation of single precision floating-point numbers as data within <definite length arb block data> and <indefinite length arb block data> message units. The format is specified in IEEE Std 754-1985 and in IEEE Std 488.2-1992. Please see Appendix F for a detailed description of this specification.

NAN - Not-A-Number Number - 9.91E+37

A measurement query response may return the Not-A-Number number (NAN) to indicate an invalid measurement condition. A NAN number is the value 9.91E+37. For example, the phase measurement query for the analog analyzer, :ANLR:PHASE?, may respond with :ANLR:PHASE 9.91E+37,1 if the inputs have no signal.

IEEE-488.1 Interface Functions

The instrument supports the IEEE-488.1 standard interface functions shown in Table 1-2.

Table 1-2. IEEE-488.1 Interface Functions

SH1	Source Handshake	complete capability
AH1	Acceptor Handshake	complete capability
T6	Talker	basic talker, untalk if listen address received, no talk only, serial poll capability (serial poll status byte defined by IEEE-488.2 standard for Status Byte Register) supports the UNL, UNT, SPD and SPE interface messages
TE0	Talker Extended	no capability, no secondary addressing
L4	Listener	basic listener, no listen only, unlisten if talk address received
LE0	Listener Extended	no capability, no secondary addressing
SR1	Service Request	complete capability, asserts SRQ interface line when service is required (if enabled by Service Request Enable Register)
RL0	Remote Local	no capability, not applicable
PP0	Parallel Poll	no capability does not support the PPC, PPD, PPE, or PPU interface messages
DC1	Device Clear	complete capability, clear input and output queues, halt internal sweep and selftest supports the DCL and SDC interface messages
DT1	Device Trigger	complete capability, triggers measurements if the GET interface message is received with TRIGGER ON supports the GET interface message
C0	Controller	no capability does not support the TCT interface message
E2	Tri-State I/O Drivers	high impedance with power off or with GPIB interface disabled by rear panel address switch

Determining Instrument Status

The system Two provides a status and event reporting system for the GPIB interface. This system informs you of certain significant events that occur within the instrument. The status handling system consists of three status registers and three status enable registers. This section describes these registers and explains how the event handling system operates.

Status Registers and Enable Registers

The registers in the event handling system fall into two functional groups:

- Status Registers contain information about the status of the instrument. They include the Standard Event Status Register (SESR), the Status Byte Register (SBR), and the AP Event Status Register (AESR).
- Enable Registers determine whether selected types of events are propagated through the event handling system. They include the Event Status Enable Register (ESER), the Service Request Enable Register (SRER), and the AP Event Status Enable Register (AESER).

Event Handling Sequence

Figure 1-7 and Figure 1-8 show the relationship of the status and enable registers in the event handling system. The IF / THEN statements below illustrate the behavior of the event handling system. The numbers in parentheses in the description below refer to the number notations in the figures.

1. IF an IEEE-488 defined event occurs THEN an event bit is set in the SESR (1).

IF an AP defined event occurs THEN an event bit is set in the AESR (7).

IF the IEEE-488 defined event in step 1 is enabled (set to one) in the ESER (2) THEN the ESB bit (3) in the SBR (10) is set to one.

IF the AP event in step 2 is enabled (set to one) in the AESER (8) THEN the AESB bit (9) in the SBR (10) is set to one.

IF the Output Queue is not empty (one or more bytes in the output queue), THEN the MAV bit (4) in the SBR (10) is set to one.

IF a bit in the SBR (10) is set to one AND the corresponding bit in the SRER (5) is set to one THEN the MSS bit (6) in the SBR is set to one AND a service request is generated.

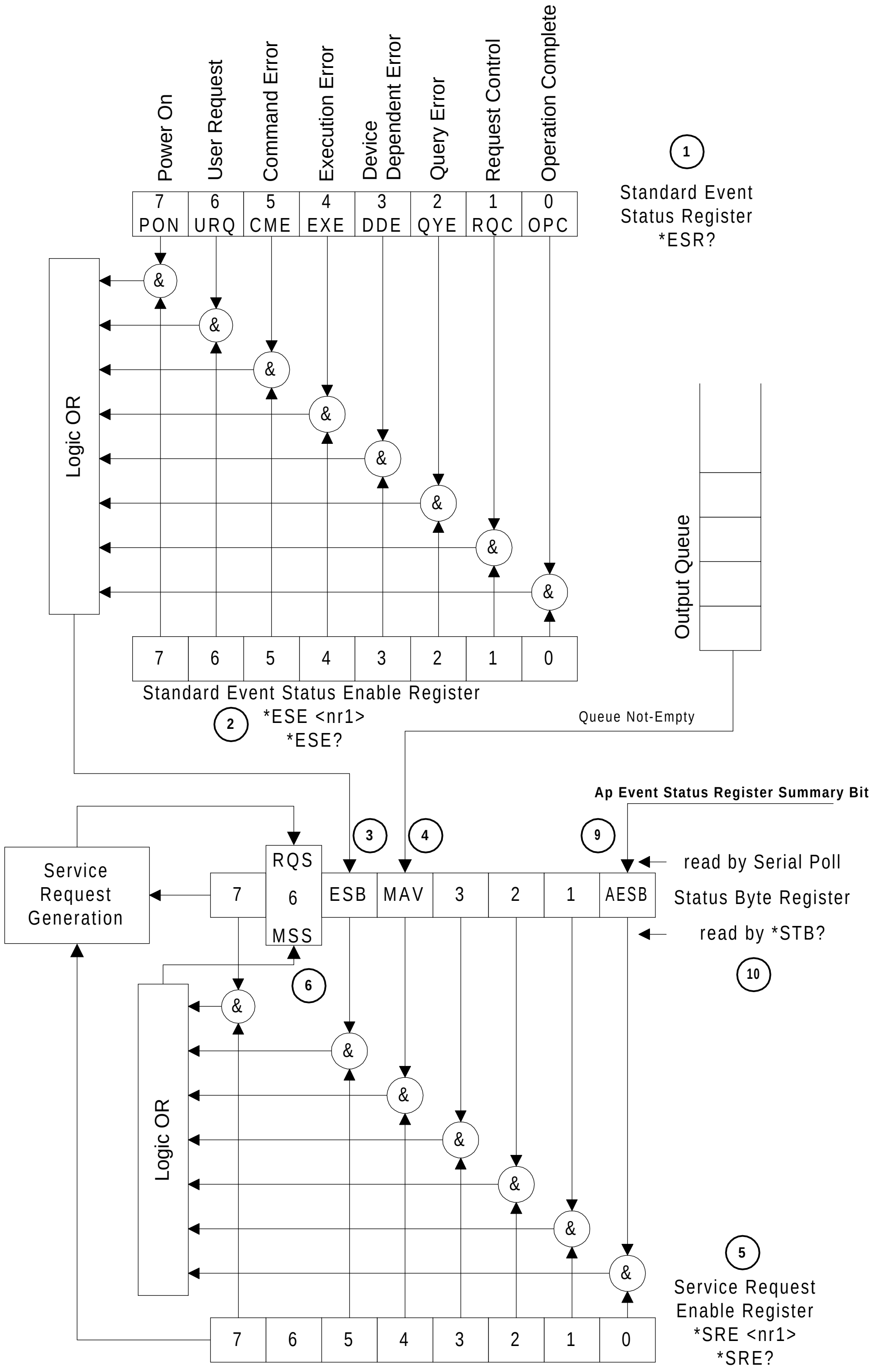


Figure 1-7. IEEE-488 Status Registers and Status Enable Registers

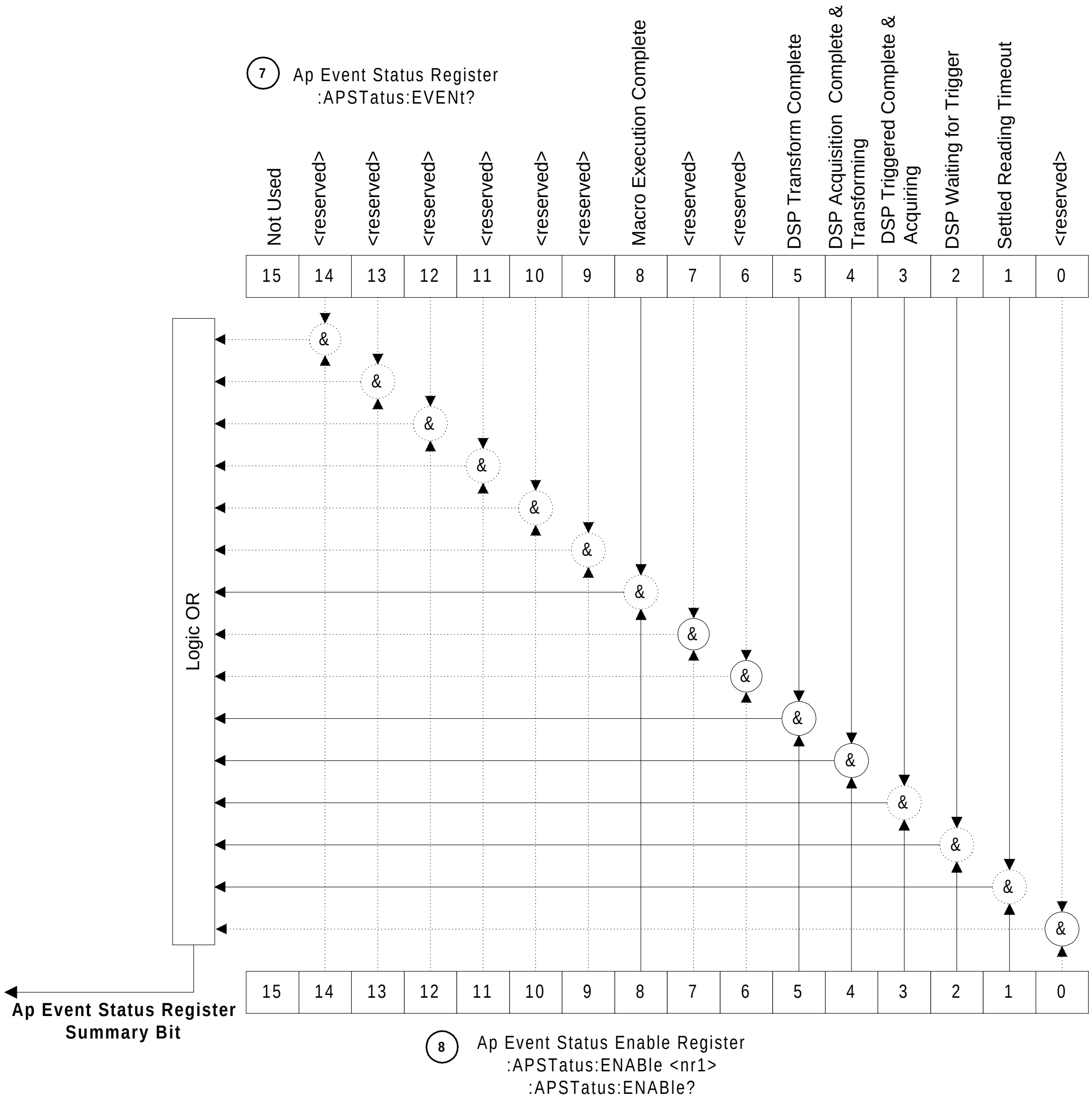


Figure 1-8. AP Event Status Register and Event Status Enable Register

Status Registers

The Standard Event Status Register (SESR), the Status Byte Register (SBR), and the AP Event Status Register (AESR) record certain types of events that may occur while the instrument is in use. IEEE Std 488.2-1987 defines the SESR and SBR registers. Audio Precision has added the AESR register. Each bit in a Status Register records a particular type of event, such as an execution error or service request. When an event occurs the instrument sets the bit that represents that type of event to a value of one. Reading the status registers tells you what types of events have occurred.

The Standard Event Status Register (SESR) & *ESR?

The SESR records eight types of events that can occur within the instrument. Use the *ESR? query to read the SESR register. Reading the register clears the bits of the register so that the register can accumulate information about new events.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Bit	Function
7 (MSB)	**PON** (Power On). The instrument was powered on.
6	**URQ** (User Request). Not Implemented.
5	**CME** (Command Error). An error occurred while the instrument was parsing a command or query.
4	**EXE** (Execution Error). An error occurred while the instrument was executing a command or query.
3	**DDE** (Device Error). A device error occurred, such as a conflict in settings that cannot be resolved in hardware.
2	**QYE** (Query Error). Indicates one of two possible scenarios that result in a Query Error. An attempt was made to read the Output Queue when no data was present or pending. A new command was received but the output queue contained a complete or partial response data from a query message terminated with <END>. The output queue was deleted because the data was not read.
1	**RQC** (Request Control). Not implemented.
0 (LSB)	**OPC** (Operation Complete). An operation complete event occurred. This bit is set when an *OPC command is executed.

Figure 1-9. Standard Event Status Register (SESR) Bit Functions

The Status Byte Register (SBR) & *STB?

The Status Byte Register (SBR) records whether output is available in the Output Queue, whether the instrument requests service, whether the Standard Event Status Register (SESR) has recorded any events that were enabled in the Standard Event Status Enable Register (ESER), or whether the AP Event Status Register (AESR) has recorded any events that were enabled in the AP Event Status Enable Register (AESER).

Use a Serial Poll or the *STB? query to read the contents of the SBR. The bits in the SBR are set and cleared depending upon the contents of the Standard Event Status Register (SESR), the Event Status Enable Register (ESER), and the Output Queue. When you use a Serial Poll to obtain the SBR, bit 6 is the Request Service (RQS) bit which indicates whether or not the instrument asserted the SRQ interface line on the GPIB. When you use the *STB? query to obtain the SBR, bit 6 is the Master Summary Status (MSS) bit which indicates that either or both of the Event Status Bit or the Message Available bit were set and enabled by the Service Request Enable Register (SRER). The SBR is cleared only by reading it with a Serial Poll. An *STB? query will not clear the SBR.

7	6	5	4	3	2	1	0
—	RQS	ESB	MAV	—	—	—	AESB
	MSS						

Bit	Dec Value	Function
7		Not used.
6	**64**	**RQS** (Request Service), obtained from a serial poll. Shows that the instrument requests service from the GPIB controller.
		MSS (Master Summary Status), obtained from *STB? Query. Summarizes bits 0 - 5 in the SBR.
5	**32**	**ESB** (Event Status Bit). An enabled event occurred in the SESR.
4	**16**	**MAV** (Message Available). A response message is available in the Output Queue.
3		Not used.
2		Not used.
1		Not used.
0		**AESB** (Ap Event Status Bit). An enabled event occurred in the AESR.

Figure 1-10. Status Byte Register (SBR) Bit Functions

The AP Event Status Register (AESR) & :APSTatus:EVENt?

The AESR records events that can occur within the instrument. Use APSTatus:EVENt? to read the AESR register. Reading the register clears the bits of the register so that the register can accumulate information about new events. Refer to Table 1-3.

Table 1-3. Ap Event Status Register (AESR) Bit Functions

Bit	Function
15 (MSB)	Not used
14	<reserved>
13	<reserved>
12	<reserved>
11	<reserved>
10	<reserved>
9	<reserved>
8	Macro Execution Complete
7	<reserved>
6	<reserved>
5	DSP Transform Complete
4	DSP Acquisition Complete and Transforming
3	DSP Trigger Complete & Acquiring
2	DSP Waiting for Trigger
1	Settled Reading Timeout
0 (LSB)	<reserved>

Enable Registers

The enable registers (ESER, SRER, and AESER) allow you to select the events in the status registers that are reported to the Status Byte Register in the ESB and AESB bits. Each bit in an enable register corresponds to a bit in a status register. In order for an event to be reported in the Status Byte Register, the corresponding bit in an enable register must be set to one. If the bit in the enable register is set to zero, the event is not reported. The Enable Registers and the commands used to set them are described below.

The Event Status Enable Register (ESER)

This register determines the types of events summarized by the Event Status Bit (ESB) in the SBR. Use the *ESE command to set the bits in the ESER. Use the *ESE? query to read the ESER.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Figure 1-11. The Event Status Enable Register (ESER)

The Service Request Enable Register (SRER)

This register determines which bits in the SBR generate a Service Request and are summarized by the Master Summary Status (MSS) bit.

Use the *SRE command to set the SRER. Use the *SRE? query to read the SRER. The RQS bit remains set to one until either the Status Byte Register is read with a Serial Poll or the MSS bit changes back to a zero.

7	6	5	4	3	2	1	0
—	—	ESB	MAV	—	—	—	AESB

Figure 1-12. The Service Request Enable Register (SRER)

The AP Event Status Enable Register (AESER)

This register determines the events summarized by Bit 0 in the SBR. Use the APSTAtus:ENABle command to set the bits in the AESER. Use the :APSTatus:ENABle? query to read the AESER.

Bit	Function
15 (MSB)	Not used
14	<reserved>
13	<reserved>
12	<reserved>
11	<reserved>
10	<reserved>
9	<reserved>
8	Macro Execution Complete
7	<reserved>
6	<reserved>
5	DSP Transform Complete
4	DSP Acquisition Complete and Transforming
3	DSP Trigger Complete & Acquiring
2	DSP Waiting for Trigger
1	Settled Reading Timeout
0 (LSB)	<reserved>

Figure 1-13. AP Event Status Enable Register (AESER) Bit Functions

Error Codes and Error Messages

Programming error status may be reported through two methods. The first method uses the *ESR? query to read the Standard Event Status Register. Each bit is set by a particular class of errors or events defined below. Bit 1 is not set by the instrument and is always 0. Multiple errors can be determined by evaluating the number returned in response to the *ESR? query.

A second method involves use of the :ERRMessage? query to report an error code (module, error number) and a quoted string explaining the error. Many error conditions may be associated with a given bit in the Standard Event Status Register. Appendix C lists all error messages that the System Two may generate in response to error conditions. Errors described in Appendix C will cause a Command Error or an Execution Error.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Event	Power On	User Request	Command Error	Execution Error	Device Dependent Error	Query Error	Not Used	Operation Complete
Decimal Value	128	64	32	16	8	4	2	1

Figure 1-14. Standard Event Status Register Error Codes

Synchronization Methods

Although most GPIB commands are completed almost immediately after being received by the instrument, some commands start a process that requires more time. For example, a settled measurement may take several seconds if the signal is noisy or if the input circuit is auto ranging due to a change in the input signal level. The status reporting system may be used to synchronize a GPIB read of an expected measurement query response with the availability of the data in the output queue in order to avoid GPIB read timeouts of the system controller.

Using the *OPC Command

An operation complete event will occur when a *OPC command is executed in the input queue. This will set the OPC bit in the Standard Event Status Register. If the Standard Event Status Enable Register also has the OPC enable bit set, then the Event Status Bit in the Status Byte Register will be set. Thus synchronization is achieved when the ESB bit changes from a 0 to a 1, indicating that the input queue has been executed up to and including the *OPC command.

Serial Poll Method

Enable the OPC bit in the Event Status Enable Register (ESER) using the *ESE command. When the operation is complete, the OPC bit in the Standard Event Status Register (SESR) will be set and the Event Status Bit (ESB) in the Status Byte Register will be set.

The command sequence below written in Visual Basic serial polls the SBR to read the status byte and then tests the ESB bit before reading the measurement query response. The keywords SENDS2G, READS2G, and SERIALPOLLS2G are pseudo-code names for subroutines that you must write that send data to the instrument, read data from the instrument, and read a status byte from the status byte register with a serial poll.

```
' Enable the status registers

SENDS2G “*ESE 1;*SRE 0”

' Send a level measurement query for the channel A analog
analyzer input level

SENDS2G “:ANLR:LEVEL? A,V;*OPC”

' Execute a DO LOOP to serial poll and test status byte until
the ESB bit is set.

WaitForESB = False

Do

    ' Read the System Two status byte iSPR with a Serial Poll.

    iSPR = SERIALPOLLS2G
    WaitForESB = iSPR And 32  ' True if ESB bit is set

Loop While WaitForESB = False   ' Loop While no ESB bit.
' Read the measurement data
ANLRLEVEL$ = READS2G
' Clear the event status register to clear the OPC bit.
SENDS2G “*CLS”
```

Service Request Method

Enable the OPC bit in the Event Status Enable Register (ESER) using the *ESE command. Also enable service requests by setting the ESB bit in the Service Request Enable Register (SRER) using the *SRE command. When the operation is complete, a Service Request will be generated.

The command sequence below written in Visual Basic uses the *OPC command to cause the System Two to assert a Service Request Interrupt to indicate a measurement is available. The status byte is then tested to determine that the ESB bit is set (which indicates that the OPC event caused the service request). If the ESB bit is set then the measurement response is read from the output queue. The keywords SENDS2G, READS2G, SERIALPOLLS2G, and WaitForSRQ are pseudo-code names for subroutines that you must write that send data to the instrument, read data from the instrument, read a status byte from the status byte register with a serial poll, and wait for an SRQ interrupt.

```
' Enable the Event Status Enable Register for OPC and Service
Request if ESB is set in status byte register.
SENDS2G “*ESE 1;*SRE 32”
' Send a level measurement query for the channel A analog
analyzer input level
SENDS2G “:ANLR:LEVEL? A,V;*OPC”
' Wait for a Service Request Interrupt
WaitForSRQ
' Read System Two status byte iSPR with a Serial Poll.

iSPR = SERIALPOLLS2G
' Read the measurement data if the ESB bit was set in the
```

```
status byte.
If iSPR And 32 = True Then ANLRLEVEL$ = READS2G
' Clear the event status register to clear the OPC bit.
SENDS2G “*CLS”
```

This technique is more efficient but requires a more sophisticated software program capable of handling a Service Request Interrupt from the GPIB controller interface.

Using the SBR MAV Bit To Acquire Query Responses

The MAV bit (Message Available) in the Status Byte Register may also be used to detect when a query response is available in the instrument output queue. For example, settled measurements may be acquired as they become available in response to measurement queries. However the MAV bit is set only if one or more bytes of a response are available in the output queue. The MAV bit does not guarantee that all pending query responses are available, thus is it possible to read a partial response and not read all expected responses. Use the *OPC method to guarantee that all expected responses are available in the output queue.

The command sequence below written in Visual Basic serial polls the SBR to read the status byte and then tests the MAV bit before reading the measurement query response. The keywords SENDS2G, READS2G, and SERIALPOLLS2G are pseudo-code names for subroutines that you must write that send data to the instrument, read data from the instrument, and read a status byte from the status byte register with a serial poll.

```
' Enable the status registers
SENDS2G “*ESE 0;*SRE 16”
' Send a level measurement query for the channel A analog
analyzer input level
SENDS2G “:ANLR:LEVEL? A,V”
' Execute a DO LOOP to serial poll and test status byte until
the MAV bit is set.
WaitForMAV = False

Do

    ' Read the System Two status byte iSPR with a Serial Poll.

    iSPR = SERIALPOLLS2G
    WaitForMAV = iSPR And 16  ' True if MAV bit is set.

Loop While WaitForMAV = False   ' Loop While no MAV bit.
' Read the measurement data
ANLRLEVEL$ = READS2G
```

2. Programming Examples

This section provides practical guidelines for writing programs that use unique features of the System Two. Each topic is supported with System Two GPIB command examples in pseudo code or actual Visual Basic code examples.

This Visual Basic code is provided in two files on the CD ROM included with this manual: “S2GSample2.bas” and “S2G_GPIB_IO_2.bas”. A complete executable program with a VB user interface is also included to simplify running each of the code examples shown in this section. Select “S2G Sample 2” from your Programs menu in your “Start” menu after you have installed the software from the CD ROM. A Visual Basic project file “S2GSample2Project.vbp” is also provided with a form file “frmS2Gsample2.frm.” In order to run the code from the VB editor, you will have to add the two National Instruments modules Vbib-32.bas and Niglobal.bas from the directories where you have installed the National Instruments GPIB driver files (typically C:\Program Files\National Instruments\Gpib\Ni488\LangInt\VBasic\Ver5).

Error Handling

The System Two provides information about programming errors and hardware execution errors. This error information is very helpful in the process of debugging test programs. The error queue holds up to 16 errors. When an error causes the queue to transition from empty to non-empty, appropriate bits in the Standard Event Status Register (depending on the class of error) will be set. Error classes are summarized in the section of the manual describing the status registers.

Three queries provide specific error information in their response messages: ERRN?, ERRM?, and ERRS?. The ERRN? query responds with the number of unread errors in the error queue. When an error queue overflow has occurred, the last error message will be replaced with the error message “Too Many Errors” and the error count will remain the same. For example, this response to the ERRN? query results when the error queue contains three error messages: **`:ERRN 3`**

The ERRM? query responds with the oldest error in the error queue. The response data provides the numerical error codes and a descriptive string. For example, this error results when the analog generator output frequency is set to a frequency beyond the valid range: **`505,14,":AGEN:FREQ, AGEN, ABOVE MAXIMUM FREQUENCY."`**

The ERRS? query responds with all errors in the error queue in FIFO (First In First Out) order. Each error in the response message is separated by a semicolon delimiter. For example, if the command string **`:AGEN:FREQ 2HZ;AMPL A,40;OUTPUT AB;:ERRS?`** is sent, these errors are reported in response to the **`ERRS?`** query: **`:ERRS 505,13,":AGEN:FREQ, AGEN, BELOW MINIMUM FREQUENCY.";502,6,":AGEN:AMPL, NOT ENOUGH PARAMETERS -OR- MISSING UNIT SUFFIX.";502,2,":OUTPUT, COMMAND NOT FOUND."`** Note that the **`2HZ`** argument in the **`:AGEN:FREQ 2HZ`** command is out of range and the 40 argument to the **`AMPL A,40`** is lacking a unit (such a V or DBV) for the number, but the **`OUTPUT AB`** command seems valid. The preceding errors cause the **`OUTPUT AB`** command to be interpreted at the root of the command tree (not associated with the**`:AGEN:`** compound header), therefore generating a command error.

The Visual Basic function code in Figure 2-1 demonstrates a technique for checking errors in the System Two. This function returns the error string `:ERRS 0,0,"NO ERROR"` if no errors are queued. The function uses the :ERRS? query to acquire all errors in the error queue and displays them if the “no errors” string “0,0” is found. If one of more errors exists in the response string, then the string “0,0” will not be found. For example, the response `:ERRS 502,2,":FOO, COMMAND NOT FOUND."; 505,12,":AGEN:AMPL, AGEN, ABOVE MAXIMUM AMPLITUDE."` indicates two errors. This code is provided on the CD ROM in file S2G_GPIB_IO_2.bas. Appendix C provides additional information about specific errors.

```
Public Function S2G_Errors() As String
  ' Check for errors in the System Two.
  ' If errors Then
  '   Display errors in a message box,
  '   If user selects message box Cancel button Then end program,
  '   Else return error string.
  ' Else return an empty string.
  S2GSend ":ERRS?"
  S2G_Errors = S2GRcv
  If InStr(1, S2G_Errors, "0,0") Then ' No errors found.
    S2G_Errors = ""
  Else
    If MsgBox("System Two GPIB Error" & vbCrLf & vbCrLf & S2G_Errors, vbOKCancel,
      MSGBOX_TITLE) = vbCancel Then End
  End If
End Function
```

Figure 2-1. Error Handling with Visual Basic function S2G_Errors

Saving and Recalling Settings with the *SAV and *RCL Commands

The System Two provides volatile memory to store nine instrument hardware states for recall at any time. The *SAV command is used to save the current hardware setup into one of nine registers. The *RCL command is used to recall any one of the nine saved registers and reset the instrument to that state. The *RCL command is much faster than sending commands for all hardware states; therefore, it provides a significant speed benefit when used to maximum advantage.

The typical scenario for use of saved settings involves an initial setup for a series of tests with the instrument. This setup may be acquired as the response to the *LRN? query. This *LRN? query response may be saved in a file or data structure and then sent directly to the instrument and followed by a *SAV command to save the entire state in memory. The *RCL command may then be used to recall that state when needed. This avoids unnecessary GPIB bus traffic and greatly decreases the time necessary to set up the hardware for the next test.

The *SAV command saves all instrument settings with a few exceptions. Waveform registers and waveform data stored in the DSP processor acquisition buffers, transform buffers, and digital generator buffers are not saved.

The *RCL command recalls all settings stored in a register with a few exceptions. The arbitrary waveforms stored with the :DGEN:ARBWFM command are erased if the :DGEN:ARBSIZE command is executed with a size that is different from the current size. In this event, the digital generator waveform and the analog generator waveform will be set to SINE,SINE if the waveform had been set to ARBITRARY or DAARBITRARY when the setting was saved with the *SAV command.

Register 0 is a special register that cannot be used with *SAV but can be used with *RCL. The *RCL 0 command sets the instrument to factory default settings and executes faster than *RST. Note that *RCL 0 is different from *RST in some respects. *RCL 0 will invoke all of the settings in Appendix B, Table B-1. The *RCL 0 command also does the following:

No macros are disturbed.

The state of macro enable is unchanged.

The state of the output queue is unchanged.

The event enable registers (SESE and ASESE) are unchanged.

The event registers (SESR and ASESR) are unchanged.

The Service request enable register (SRER) is unchanged.

The *SAV/*RCL registers are unchanged.

The Filter slot configuration is not scanned

The instrument GPIB interface may not always respond immediately when *SAV or *RCL is in progress. Therefore, it is a good practice to use the **`*ESE 1;*OPC`** commands at the end of a *SAV or *RCL command sequence. Read the status byte register with a serial poll and test the ESB bit in the status byte until the bit is true in order to determine that the operation is complete.

The Visual Basic program below illustrates the use of the *SAV and *RCL commands in two functions included on the CD ROM in file “S2Gsample2.bas”.

```
Sub SAV()
  ' Using the *SAV command to save instrument hardware settings.
  ' Send *SAV with user's input (value of combo control cmbSAV_RCL) appended to it. Error
      if not a valid number in the range 0 to 9.
  S2GSend "*SAV " & frmS2GSample2.cmbSAV_RCL
  ' Check for errors.
  S2G_Errors
End Sub

Sub RCL()
  ' Using the *RCL command to recall saved instrument hardware settings.
  ' Send *RCL with user's input (value of combo control cmbSAV_RCL) appended to it. Error
      if not a valid number in the range 0 to 9.
  S2GSend "*RCL " & frmS2GSample2.cmbSAV_RCL
  ' Check for errors.
  S2G_Errors
End Sub
```

*Figure 2-2. Visual Basic subprograms for *SAV and *RCL*

Using Macro Commands

Macros provide a convenient means of defining and executing special strings of commands that may be given unique labels. Common sequences of setups or measurements may be pre-defined as macros and saved in memory until needed, thus reducing bus traffic and time required for transmission during test execution.

The *DMC command is used to define a macro with a label and a string of GPIB commands to be executed when the macro label is encountered in the input queue. The *EMC command enables or disables the execution of macros. The *GMC? query responds with the current definition of a labeled macro. The *LMC? query responds with a list of all defined macro labels. The *PMC command deletes all defined macros and labels. The *RMC deletes a specific macro. Use of macros takes on three distinct phases: Definition, Verification, and Execution.

Macro Definition

The Definition phase occurs upon receipt of a *DMC command. The macro and its label are inserted into the macro table. Once a macro has been inserted into the macro table, another macro with the same label may not be inserted until the first one has been removed through the use of the *RMC or *PMC commands. The label is checked for length and validity of the characters it contains. It must begin with a letter, and must be no longer than 12 characters. All characters in the label must be chosen from the ASCII character set ranges from 48-57 ("0"-"9"), 65-90 ("a"-"z"), or 97-122 ("A"-"Z"). Although it is valid to specify macro labels with either upper or lower (or both) case letters, case is not considered when parsing the input stream, so "macrolabel", "MacroLabel" and "MACROLABEL" would all refer to the same macro. Thus, any attempt to define macros with labels differing only in case will result in an error. After the label is validated, the arbitrary block containing the macro itself is read. The size of the arbitrary block is compared against the free space in the macro table to ensure there is enough room to store it. The size of the macro table is 10240 characters. Each entry in the macro table requires sufficient memory to hold the macro string itself plus 20 bytes of overhead. A special macro contains the response to a Device Trigger (*TRG or a 'GET' GPIB message). Unlike other macros, this macro is defined by a special command (*DDT), and does not have a label. Also unlike normal macros, the DDT macro is limited to a length of 1024 bytes. Other macros can be as large as free space in the macro table will allow. Error messages produced during the Macro Definition phase are generally Parser Errors, so they will fall into error message category 502.

Macro Verification

Macros are verified as they are being inserted into the macro table if the *EMC command has been used to enable macro expansion. Macro verification will not occur when the *DMC command is received if the *EMC command has disabled macro expansion (***EMC 0). You should enable macro expansion with the *EMC command when you send *DMC commands to define macros in order to force verification. The following requirements must be met for the macro to be valid:

- The macro must not contain certain commands: *DDT, *EMC, *GMC, *LMC, *PMC, *DMC, *TRG and *RMC.
- The macro must not include references to other macros (no chained or recursive macros).
- Macro parameters cannot be used to insert sub-strings into command headers or parameter fields.

For example, ':AGEN:FREQ $1' is valid, but ':AGEN:FREQ $1HZ' is not valid. Error messages produced during the Macro Verification phase are identified as message category 503.

Macro Execution

To enable macro expansion, the *EMC 1 command is issued (the default condition for *EMC is disabled or '0'). Enabling macro expansion incurs a slight penalty in execution time for all commands, because as each command header is parsed, the macro table must be scanned for a match before the standard list of commands is consulted. Except for extremely time-critical applications, this delay should be negligible. Once the *EMC command has been used to enable macro expansion, and a macro has been defined and verified, it is considered to exist at the top level of the command hierarchy, along with other commands such as :HEADer, :DELay, and :ERRS?. As a general rule, macros should be invoked with a leading ":" to prevent the parser from attempting to compound the macro label with a command header in effect from an earlier command in the same program message. When a macro label is encountered in the command stream, the macro string is copied into a "macro expansion buffer". In the process, arguments (if any) to the macro invocation are substituted into the original macro string wherever a $<n> is encountered to produce a command stream that is injected into the parser input. No further commands are accepted at the GPIB input until all commands are used from the macro expansion buffer. Any parser errors that are encountered during this process belong to category 504, macro execution errors.

Macro Execution Complete Events

You can use the Macro Execution Complete event in the AP Event Status Register to cause an SRQ interrupt when a macro has completed normally, or you can poll the serial poll status byte in a loop code structure (for instance, a DO/WHILE loop) until the AESB Summary bit is set in the status byte.

AESB Bit 8 will be set if the macro terminates normally. However, if a SDC or DCL halts the macro during its execution, then AESB Bit 8 will NOT be set. Note that if a macro containing queries is terminated prematurely, then the OUTPUT queue may not have all query responses; that is, unexecuted queries will have no query responses.

Macro Examples

To illustrate, a macro may be defined that sets the analog generator frequency and level and then acquires an analyzer level and THD+N ratio measurements. The *DMC command will define a macro with the label “GENFRQLVLTHD” and the macro will use optional variables for the generator frequency and level and send appropriate measurement queries. The variable $1 will specify the analog analyzer response frequency setting for optimum auto reading rates. The variable $2 will specify the analog generator output frequency. The variable $3 will specify the generator channel A output amplitude. Note that the values used as arguments for the macro when it is executed must be complete arguments for the commands within the macro. Thus the first command argument for the macro requires no unit suffix for the :ANLR:RESPONSE command, but the second and third arguments for the :AGEN:FREQ and :AGEN:AMPL commands require unit suffixes.

Define a macro labeled “GENFRQLVLTHD”:

```
*EMC 1;*DMC "GENFRQLVLTHD",#3189:ANLR:CHANNEL A;MODE
THDRATIO;INPUT A,GENMON;RDGRATE
AUTO,LEVEL,FREQ,FUNCMETER;TUNINGSRC AGEN;RESPONSE
$1;:AGEN:OUTPUT AB;FREQ $2;AMPL A,$3;:DELAY 0.05;:ANLR:LEV?
A,DBV;FREQ? A,HZ;FUNC? PCT
```

Now execute the macro with parameter values as arguments:

```
GENFRQLVLTHD 1000,1000HZ,2DBV
```

Now read the macro measurement query responses:

```
:ANLR:LEVEL 1.9975DBV,0;:ANLR:FREQ 1000.2HZ,0;:ANLR:FUNCMETER
0.000741988PCT,0
```

Now delete the macro:

```
*RMC "GENFRQLVLTHD"
```

Now disable the use of all macros:

```
*EMC 0
```

A macro label may be used anywhere within a command message. However, the macro label must be preceded by a colon character ":" if the position within the command message does not place the macro label at the root of the command tree. For example, the command message **:AGEN:OUTPUT AB;:GENFRQLVLTHD 1000,1000HZ,2DBV** is valid because the macro label **GENFRQLVLTHD** is preceded by a colon. However the command message **:AGEN:OUTPUT AB;GENFRQLVLTHD 1000,1000HZ,2DBV** is invalid and will cause an error because the macro label **GENFRQLVLTHD** is assumed to be part of the **:AGEN:** branch of the command tree.

Triggering Events with the *DDT Command

The *DDT command may be used to execute a predefined string of commands whenever a GPIB Group Execute Trigger (GET) command or a *TRG command is received by the instrument. Receipt of a trigger executes the commands defined by the *DDT command. These commands can be used to specify a sequence of commands for the System Two, including measurements.

A series of measurement queries may be defined by the *DDT command. When a trigger is received the commands will execute and the results will be place in the output queue.

The example below defines a string that sets the generator frequency to two different values and measures the analog analyzer level, frequency, and THD+N at each of the frequencies:

Send:

```
*DDT #3228:AGEN:OUTPUT AB;AMPL A,10DBV;FREQ 500HZ;:ANLR:RDGR
AUTO,LEVEL,FREQ,FUNCMETER;TUN AGEN;RESP 500;:DELAY
0.05;:ANLR:LEV? A,DBV;FREQ? A,HZ;FUNC? PCT;:AGEN:FREQ
100HZ;:ANLR:RESP 100;:DELAY 0.05;:ANLR:LEVEL? A,DBV;FREQ?
A,HZ;FUNC? PCT
```

Send:

```
*TRG
```

Receive:

```
:ANLR:LEV 9.99168DBV,0;:ANLR:FREQ 500.077HZ,0;:ANLR:FUNC
0.00100461PCT,0;:ANLR:LEV 6.53936DBV,0;:ANLR:FREQ
100.023HZ,0;:ANLR:FUNC 0.00100868PCT,0
```

Saving and Recalling Waveforms with Batch Mode DSP Programs

Waveforms acquired with the DSP programs are not stored in permanent memory and thus are volatile. Waveforms are latched into the DSP acquisition waveform memory for processing until the DSP program is changed, the instrument is turned off, or another acquisition is started. Acquired waveforms may be transferred from the instrument to the GPIB controller and saved in a disk file until needed, then download later and reprocessed by the DSP program. This feature makes it possible to compare waveforms acquired under different conditions, or reprocess the waveforms with different DSP parameters for detailed study.

The :DSP:DATA? query is used to retrieve the contents of the two DSP acquisition buffers or the two DSP transform buffers. The response is formatted as a complete GPIB command that includes the **`:DSP:DATA`** header (if the command :HEADER:ON has already been sent),channel number, and data in a definite length arbitrary block format. The recommended practice is to send this command with program headers turned on in query responses. Thus the command **`:HEADER ON;:DSP:DATA? A1`** would result in a response of **`:DSP:DATA A1,#555873`** followed by 55873 bytes of unprintable binary waveform data. The response will be delimited with a Newline EOI (linefeed with EOI line asserted) if it is the last program message unit in a terminated program message. This response data may be sent directly back to the instrument and then reprocessed if the instrument hardware is set up exactly as it was when the data was originally acquired.

See the subsection entitled "**Spectrum Sweeps with the FFT DSP Program**" later in this section for an example of how to save and recall waveforms when using the FFT DSP program.

Using Arbitrary Waveforms with the Analog and Digital Generators

The analog and digital generators have the ability to output stereo arbitrary waveforms generated by the DSP processor. The DSP processor generates the arbitrary waveforms using digital sample points contained in a waveform memory structure. The source of each channel of the stereo waveforms is a file stored on disk that must be downloaded into the DSP memory. The DSP processor generates the stereo arbitrary waveform signal simultaneously on both the analog generator and the digital generator when the waveform type is set to ARBITRARY for the digital generator and DAARBITRARY for the analog generator. The same signal is provided to both generator outputs and either or both of the outputs may be changed to use a waveform other than arbitrary without affecting the other. Thus, the two generators share the same set of stereo arbitrary waveforms. You can change the arbitrary waveform at any time.

Many waveform files are furnished with the APWIN software provided on the CD ROM for this manual. These arbitrary waveform files have been developed for applications such as audio instrument calibration, multitone signal generation for production test, broadcast system testing, and low bit rate coder testing. Custom arbitrary waveform files may be created by an external program called MAKEWAV2.EXE that is provided with the APWIN software on the CD ROM (included with this manual). Refer to the APWIN User's Manual for System Two, section 13, "Multitone Generation from Files," and a subsection entitled "Creating Waveform Files with MAKEWAV2," for more information about creating custom arbitrary waveform files.

The process of creating a multitone waveform with MAKEWAV2 results in waveform file and a file containing a list of the exact sinewave frequencies created in the waveform. Note that these frequencies are correct only if the waveform is generated at the sample rate specified on the MAKEWAV2 command line. In order to use the multitone arbitrary waveform with the System Two via the GPIB interface, you must know the frequencies contained in the arbitrary waveform file and you must load the waveform file into the System Two waveform memory.

Managing Generator Waveform Memory

The System Two GPIB processor provides memory space to store many downloaded arbitrary waveforms. The waveforms may be stored in this memory space and then loaded into the DSP processor as needed for different tests. This memory space may be configured to optimize the number of waveforms that may be stored, depending on the size of the largest waveform desired. The **:DGEN:ARBCOUNT?** query returns the number of generator arbitrary waveform registers than can be defined in the available memory space for a given size of waveform (specified in sample points). The sample size may be 256, 512, 1024, 2048, 4096, or 8192. Thus given a sample size of 8192, the System Two GPIB processor can store 16 waveforms. This means that 8 stereo pairs of waveforms with a size of 8192 samples may be downloaded and stored in memory until needed by the DSP processor.

The arbitrary waveform memory may be configured to optimize the number of waveforms with the **:DGEN:ARBSIZE** command. This command reconfigures the memory for the size of waveform specified as an argument. Thus the command **:DGEN:ARBSIZE 2048** provides space for 62 waveforms. Each waveform may be no larger than 2048 but may be smaller. Waveforms that may already be stored in this memory space will be deleted when this command executes if the size is changed. Thus it is possible to clear all waveforms from memory by “resizing” a different size and then to the original size.

The table below shows the byte count for different sizes of waveforms created by MAKEWAV2 and the maximum number of registers that may be specified with the ARBSIZE command for each waveform sample size.

Table 2-1. Arbitrary Waveform Byte Count & Register Count

Waveform Sample Size	Waveform Byte Count	Maximum Registers
8192	24832	16
4096	12544	31
2048	6400	62
1024	3328	119
512	1794	221
256	1024	388

Loading Arbitrary Waveforms into Generator Waveform Memory Registers

A waveform becomes available for loading into one of the DSP generator buffers when it has been downloaded into a waveform register. Arbitrary waveforms are downloaded from the GPIB system controller to a System Two GPIB waveform register with the **:DGEN:ARBLOAD** command. The first command argument specifies the waveform register to receive the waveform data. The second command argument is a definite length arbitrary block that contains the waveform data. This data is the data contained in a waveform file created by the MAKEWAV2 program. The exact size of the file in bytes is the byte count that must be specified in the arbitrary block byte count field. This byte count may be determined by looking at the file size shown in the file properties dialog box in WIN 95. In order to build the **:DGEN:ARBLOAD** command properly for a waveform of 1024 samples for waveform register 5, the literal string “:DGEN:ARBLOAD 5,#43328” may be concatenated to the beginning of another string containing the waveform file data.

For example, a waveform with 1024 samples would be loaded into register 5 with the following command in which the data in the waveform file is represented by “bb…” :

```
:DGEN:ARBLOAD 5,#43328bb…
```

Loading Generator Waveform Memory Registers into the DSP Generator

The :DGEN:ARBWFM command is used to load a waveform from one of the generator waveform registers into one or both of the two DSP generator buffers. For example, to load waveform register 5 into DSP generator buffer 1 and waveform register 6 into DSP generator buffer 2, send the following command:

```
:DGEN:ARBWFM 5,6
```

If the same waveform is desired for both DSP generator buffers, then the same generator waveform register may be specified for both channels, or one of the channels may have register 0 specified. Register 0 for either channel causes both channels to be loaded with the non-zero register number, for example, the command **`:DGEN:ARBWFM 0,6`** loads both channels with the waveform from register 6.

Note that arbitrary waveforms will not be present at the analog generator output or the digital generator output unless the ARB waveform is selected with the **`:AGEN:WFM DAARBITRARY,NONE`** or **`:DGEN:WFM ARBITRARY,NONE`** commands after the **`:DGEN:ARBWFM`** command has loaded waveforms into the DSP digital generator buffers. The error message "**`:ERRS 507,7,":DGEN:WFM, DGEN, DSP DOES NOT CONTAIN ARB."`**" will result if arbitrary waveforms are not loaded into the DSP digital generator waveform buffers first with the :DGEN:ARBWFM command.

The example below loads two arbitrary waveforms from files "C:\APWIN\Waveform\Xtlkleft.agm" and "C:\APWIN\Waveform\Xtlkrigh.agm" into arbitrary waveform registers 1 and 2, and then loads both DSP digital generator buffers with those waveforms. These two waveforms each contain frequencies not present in the other in order to permit simultaneous level, distortion, noise, and crosstalk tests on a stereo device. The example does not show how the file is transferred to the GPIB port of the controller. Your program code should assemble a command string in this manner, inserting the file contents after the arbitrary block byte count. The Audio Precision "GPIB Talker/Listener Program" provided on the CD ROM is designed to accept this string exactly as it appears below and then send the file contents when it encounters the file name within the < and > characters.

Set the size of arbitrary waveform storage registers to 8192 bytes and load the waveform data from files "C:\APWIN\Waveform\Xtlkleft.agm" and "C:\APWIN\Waveform\Xtlkrigh.agm" into registers 1 and 2:

```
:DGEN:ARBSIZE 8192;ARBLOAD 1,#524832<C:\APWIN\Waveform\Xtlklef
t.agm>;ARBLOAD 2,#524832<C:\APWIN\Waveform\Xtlkrigh.agm>
```

The next example sets the output sample rate to 48 kHz in order to generate the correct frequencies in the arbitrary waveforms, and sets up the digital and analog outputs in order to generate the arbitrary waveforms. The headphone monitor is configured to monitor the analog generator output to demonstrate that the arbitrary waveform is properly loaded. Both analog and digital generators create the same arbitrary waveforms.

```
:DOUT:RATE 48000 HZ;:DGEN:ARBWFM 1,2;OUTPUT AB;WFM
ARBITRARY,NONE;AMPL A,-10DBFS;AMPL B,-10DBFS;DITHERTYPE
TRI;:AGEN:OUTPUT AB;AMPL A,-10DBV;AMPL B,-10DBV;WFM
DAARBITRARY,NONE;:MON:MODE STEREO;STEREO GENMON
```

Acquiring Measurements

Measurements may be acquired from measurement meters in the following modules of System Two:

Analyzer (analog audio analyzer)

DCX-127 Multifunction Module

Digital Domain Audio Analyzer

FFT Spectrum Analyzer

Digital Interface Analyzer

Multitone Audio Analyzer

Digital I/O Parameters

Digital Synch Reference Input

Digital I/O Status Bits

Measurements are provided in response to specific measurement queries. A list of measurement queries is shown in Table 2-2 for each of the measurement meters. Most measurement meters have a settling capability. Most measurement queries require a channel argument and a measurement unit argument.

Meters with settling capability provide measurement responses with two argument values. The numerical measurement data with a unit suffix is the first response argument. The second response argument indicates the settled state of the measurement if settling has been enabled for that measurement.

For example, the measurement query for the analog analyzer channel A level meter in DBV units is: **`:ANLR:LEVEL? A,DBV`**. The response is: **`:ANLR:LEVEL 12.0322DBV,0`**. The response header may be omitted with the use of the **`HEADER OFF`** command. The measurement query above may be set up for no measurement query response header with: **`:HEADER OFF;:ANLR:LEVEL? A,DBV`**. The result is: **`12.0322DBV,0`**. The 0 in the second argument indicates that the measurement was either settled or that settling was not enabled. The response will be a 1 if settling was enabled and the settling algorithm timed out in the process of acquiring a settled measurement.

Table 2-2. Measurement Queries

Measurement Meter	Measurement Query	Settling Available
Analog Analyzer Level A	:ANLR:LEVEL? A,V	Yes
Analog Analyzer Level B	:ANLR:LEVEL? B,V	Yes
Analog Analyzer Frequency A	:ANLR:FREQ? A,HZ	Yes
Analog Analyzer Frequency B	:ANLR:FREQ? B,HZ	Yes
Analog Analyzer Phase	:ANLR:PHASE?	Yes
Analog Analyzer Function	:ANLR:FUNCMETER? V	Yes
DCX–127 DMM DC Volt Meter	:DCX:DMM? V	Yes
DCX-127 DMM Ohm Meter	:DCX:DMM? O	Yes
DSP Audio Analyzer II Level A	:DSP:DANLR:LEVEL? A,FFS	Yes
DSP Audio Analyzer II Level B	:DSP:DANLR:LEVEL? B,FFS	Yes
DSP Audio Analyzer II Frequency A	:DSP:DANLR:FREQ? A,HZ	Yes
DSP Audio Analyzer II Frequency B	:DSP:DANLR:FREQ? B,HZ	Yes
DSP Audio Analyzer II Function Meter	:DSP:DANLR:FUNCMETER? FFS	Yes
DSP FFT Spectrum Analyzer Peak Meter 1	:DSP:FFT:PEAK? 1,FFS	No
DSP FFT Spectrum Analyzer Peak Meter 2	:DSP:FFT:PEAK? 2,FFS	No
DSP FASTTEST Multitone Audio Analyzer Peak Meter 1	:DSP:FASTTEST:PEAK? 1,FFS	No
DSP FASTTEST Multitone Audio Analyzer Peak Meter 2	:DSP:FASTTEST:PEAK? 2,FFS	No
DSP MLS Quasi-anechoic Acoustical Tester Peak Meter 1	:DSP:MLS:PEAK? 1,FFS	No
DSP MLS Quasi-anechoic Acoustical Tester Peak Meter 2	:DSP:MLS:PEAK? 2,FFS	No
DSP Bittest Data Meter A	:DSP:BITTEST:DATA? A,HEX	No
DSP Bittest Data Meter B	:DSP:BITTEST:DATA? B,HEX	No
DSP Bittest Error Counter A	:DSP:BITTEST:ERROR? A,HEX	No
DSP Bittest Error Counter B	:DSP:BITTEST:ERROR? B,HEX	No
Digital Input Peak-to-Peak Interface Signal Amplitude	:DIN:AMPL?	Yes
Digital Input Active Bits or Data Bits A	:DIN:BITS? A	No
Digital Input Active Bits or Data Bits B	:DIN:BITS? B	No
Digital Input Jitter	:DIN:JITTER?	Yes
Digital Input Delay From Generator Output	:DIN:ODELAY?	Yes
Digital Input Audio Peak Level A	:DIN:PEAK? A,FFS	No
Digital Input Audio Peak Level B	:DIN:PEAK? B,FFS	No
Digital Input Sample Rate	:DIN:RATE? HZ	Yes
Digital Sync - Reference Input Frequency	:SYNC:IFReq?	No
Digital Sync - Input Delay from Reference Input	:SYNC:IDELay?	Yes

Measurement Settling

Many measurements may be made with the internal settling algorithms enabled. These algorithms and their parameters process a series of measurements in order to provide a single settled meter measurement in response to the measurement query. The settling parameters for each meter are individually specified with the settling commands.

When measurements (throughout this document, “measurements” and “readings” are used synonymously) are taken there are three possible settling results:

Nonsettled Reading - settling is turned off or not implemented.

Unsettled Reading - settling is enabled, but the series of measurements did not settle in time. An unsettled reading will return the average of the readings in the settling queue.

Settled Reading - settling is enabled, and a series of measurements meet the settling criteria within the time specified by the settling timeout period.

Responses to measurement queries consist of headers and arguments. The headers indicate the meter. Arguments contain the measurement value and the settling status. For example, **`:ANLR:FREQ? A,DHZ`** requests a frequency measurement (in delta Hz) from the channel A frequency meter (Analog Analyzer). The response could have two forms: **`:ANLR:FREQ 1000.17HZ,0`** or **`:ANLR:FREQ 1883.06HZ,1`**.

The second argument indicates whether this was a non-settled or settled reading (0), or an unsettled reading (1) due to a settling timeout.

Settling Parameters

The settling algorithm acquires measurements in a queue until the settling criteria have been fulfilled or until a time-out value has been exceeded. The parameters of the settling commands are ALGORITHM, FLOOR, TOLERANCE, POINTS, DELAY, TIMEOUT, and TRIGGER.

ALGORITHM specifies one of several algorithms for processing the statistical differences between measurements in the settling queue. Choices for ALGORITHM are NONE, FLAT, EXP, and AVG.

NONE turns off the settling algorithm, disables the settling process, and causes a measurement to be acquired immediately and returned in the measurement query response immediately.

The FLAT algorithm applies the same TOLERANCE to all successive measurements in the queue. The FLAT algorithm guarantees that measurement transients have been settled to the specified TOLERANCE for some time, which tends to take longer than the EXP algorithm.

The EXP algorithm applies exponential weighting to the TOLERANCE used for each pair of measurements in the queue. The tolerance limit for the most recent pair of measurements is the TOLERANCE parameter. The tolerance for the second and third oldest measurements in the queue is a factor of 2 times the TOLERANCE parameter. The tolerance for the fourth and third oldest measurements in the queue is a factor of 4 times the TOLERANCE parameter, and so forth for the number of measurements in the queue specified by POINTS.

The AVERAGE algorithm acquires the number of consecutive measurements specified by the POINTS parameter, computes their mathematical average, and returns the average in the measurement query response. TOLERANCE and FLOOR values are ignored when the

AVERAGE algorithm is specified. The AVERAGE algorithm is particularly useful when the signal is fundamentally noisy and might never settle within a practical TOLERANCE.

FLOOR sets a lower numerical bound, below which measurements in the settling queue whose difference is less than this value will be considered settled. This sets a limit on measurement resolution for purposes of comparing measurements.

POINTS specifies the number of measurements to be used for settling calculations (size of the settling queue).

TOLERANCE is the percent of change between successive measurements in the settling queue.

DELAY is a delay in seconds to wait before beginning to acquire measurements into the settling queue. This parameter is provided to allow a stimulus signal to stabilize before measurement acquisition begins.

TIMEOUT specifies the maximum length of time that measurements will be accumulated into the settling queue. The settling algorithm will process measurements in the settling queue until a measurement meets the settling criteria or until this time is exceeded.

TRIGGER restarts the internal measurement process when a measurement query is received. The current measurement cycle is aborted and a new measurement cycle is started. This synchronizes the internal measurement cycle with the receipt of a measurement query.

The settling algorithm begins by waiting the time specified by Delay, starting a timer, acquiring the first measurement and placing it into the settling queue, and testing elapsed time against the timeout parameter. If a timeout has occurred, then the average of the readings in the settling queue is the response to the measurement query and the timeout argument is set to 1.

If POINTS is two, a second measurement is acquired and placed into the measurement queue. This measurement is considered settled if the percent difference between it and the previous reading is less than the tolerance or it is less than the Resolution parameter. If the second reading meets the settling criteria, then it is returned as the settled measurement with 0 for the second response parameter. If the second reading is not settled, another measurement is acquired and compared to the previous measurement in like manner until either a measurement meets the settling criteria or a timeout occurs. The sequence is repeated and each new measurement is placed into the settling queue and compared with the previous measurement in the queue for Tolerance and Resolution.

If POINTS is three, a third measurement is acquired and placed into the settling queue. This measurement is compared to the other two measurements in the queue for settling criteria in like manner. If the third reading is settled with respect to both the first reading and the second reading, it is returned as the settled reading. If a time-out has not occurred and a settled reading has not been found, another reading is acquired and compared to the previous two readings. This same process is repeated for as many measurements specified by POINTS for the measurement meter. The process is repeated until either a settled reading is found or a timeout occurs.

Synchronizing Measurements and Events with the *OPC Command

Measurements and events may be synchronized with the GPIB system controller in order to control when the output queue is to be read after commands have been sent to the instrument. The most basic technique involves setting a long timeout for the system controller when a read of the System Two output queue is attempted. The length of the timeout must be longer than the expected time for a response from the instrument in order to use this technique successfully. This technique will read one or more query responses from the System Two output queue but does not guarantee that all responses that are expected will be read. A better technique is available for this application: the Operation Complete event (OPC).

The Operation Complete event sets the Operation Complete event bit in the Standard Event Status Register when the *OPC occurs in the message sent to System Two. Message units in the System Two input queue are executed in the order received; therefore, all messages prior to the instance of a *OPC message must be executed before the *OPC message will cause the OPC bit to be set. The *ESE 1 command must be used to enable the OPC bit in the Standard Event Status Register to cause the Event Status Bit to be set in the Status Byte Register. Thus, when *ESE 1 has been sent and when the *OPC is executed in the System Two input queue, the ESB bit will be set in the Status Byte Register. The application program may acquire the status byte by performing a serial poll of the System Two. The status byte may be compared with a bit mask to determine that the ESB bit is set and then take appropriate action.

The Visual Basic code in Figure 2-3 illustrates an algorithm for serial-polling the status byte register and comparing the bits in the serial poll response with a mask to determine if the ESB bit (Event Status Bit) has been set. This algorithm includes a timeout parameter in order to allow the application programmer to handle a timeout as an error. This code is provided in the "S2G_GPIB_IO_2.bas" file on the CD Rom provided with this manual. Note that the **ibrsp** serial poll routine is provided by National Instruments Inc.

```
Public Function WaitForESB(fTimeout As Single) As Boolean
  ' Wait for the Event Status Bit to be set in the Status Byte Register
  ' Serial Poll until ESB bit in status byte is True or until timeout
  ' Serial Poll repetition rate is 0.1 of timeout interval or 0.055 seconds, whichever is
     larger.
  Static iSPR As Integer, fStartTime As Single, fDelay As Single
  fDelay = fTimeout / 10
  If fDelay < 0.055 Then fDelay = 0.055
  fStartTime = Timer
  WaitForESB = False
  Do
    Delay fDelay
    ' Read the System Two status byte iSPR with a Serial Poll.
    ibrsp S2G, iSPR
    WaitForESB = iSPR And 32 ' True if ESB bit is set, else False
    ' Let other Windows tasks run too.
    DoEvents
  ' Loop While no ESB bit and no timeout.
  Loop While WaitForESB = False And Timer - fStartTime < fTimeout
End Function
```

Figure 2-3. Visual Basic WaitForESB Function in file S2G_GPIB_IO_2.bas

The OPC bit must be cleared from the Standard Event Status Register before another OPC event will be recognized. This may be accomplished by reading the Standard Event Status Register with the *ESR? query or by sending the *CLS command. Note, however, that the

*CLS command will clear any query responses in the output queue, therefore it must be used carefully.

The MAV bit (Message AVailable) may also be used to detect that query responses are available in the System Two output queue. The MAV bit is set in the status byte register whenever one or more bytes are available in the output queue. However, this does not guarantee that all query responses are available. Thus, OPC is a more reliable indicator that all query responses are ready to be read from the output queue.

Stimulus and Response Measurement Sweeps

System Two GPIB commands simplify the process of sweeping a stimulus parameter and acquiring measurements for each stimulus parameter value. The Visual Basic program examples below illustrate how to implement different types of measurement sweeps.

The program examples are written for Visual Basic 5.0 to illustrate the appropriate System Two GPIB commands and measurement synchronization techniques. This code is provided in the “S2Gsample2.bas” file on the CD ROM provided with this manual. The Operation Complete (OPC) bit in the Standard Event Status Register is used to indicate when a sequence of commands have been executed prior to reading measurement results. This is particularly important when it is desired to read multiple measurement query responses in the System Two GPIB output queue.

Frequency Response Sweep – Analog Generator Frequency Sweeps with Analog Analyzer Measurements

To sweep the analog generator output frequency and measure analog analyzer level and THD+N, set up both the generator and analyzer for initial conditions. Then begin a loop structure, such as a FOR LOOP, that increments the generator frequency and the analyzer response frequency and acquires the settled measurements. The frequency of the distortion notch filter is automatically set to the frequency of the analog generator by the **:ANLR:TUNINGSRC AGEN** and **:AGEN:FREQ** commands.

A single message with multiple occurrences of a command macro invocation (for each frequency) may be used if a fixed set of frequencies is desired. The Visual Basic program example below sweeps 16 frequencies from 20 KHz to 20 Hz and measures analyzer channel A THD+N Ratio in % units, Level in dBr units with reference to the level at 1000 Hz, and frequency in HZ units. The test results are for a 15 band graphic equalizer with XLR inputs and outputs. This code is provided in the “S2Gsample2.bas” file on the CD ROM provided with this manual.

(Start of Figure 2-4)

```
Sub AGEN_ANLR_FreqRespSwp()
  ' Frequency Response Sweep - Analog Generator Frequency Sweeps with Analog Analyzer
      Measurements
  TEST_TITLE = "AGEN_ANLR_FreqRespSwp"
  If IsAnalog = False Then
    MsgBox "System Two Analog option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      INPUT;:AGEN:OUTPUT AB;AMPL A,1V;FREQ 1E3HZ;WFM SINE,SINE;:ANLR:INPUT AB,XLR;CHANNEL
      A;MODE THDRATIO;LPF F30K;TUNINGSRC AGEN;RDGRATE AFAST,LEVEL,FREQ,FUNCMETER;RESPONSE
      1E3;:SETTLING:ANLR AFREQ,1,0.01HZ,2,0.01,FLAT,4,1;ANLR ALEVEL,5,1E-
      06V,3,0.1,EXP,4,1;ANLR THDRATIO,5,5E-05PCT,3,0.1,EXP,4,1"
  ' Send a message that purges all macro definitions, enables all macros, and defines a
      sweep macro for generator output frequency, analyzer reading rate response
      frequency, and measurement queries
  S2GSend "*PMC;*EMC 1;*DMC ""SWPLVLTHD"",#270:ANLR:RESPONSE $1;:AGEN:FREQ $2;:ANLR:FREQ?
      A,HZ;LEV? A,DBRA;FUNC? PCT"
  ' Send the measurement acquisition commands.
  S2GSend ":AGEN:FREQ 1E3HZ;:DELAY 0.2;:ANLR:SETREFAUTO DBR;:AGEN:FREQ 20E3HZ;:DELAY
      0.2;:SWPLVLTHD 20E3,20E3HZ;SWPLVLTHD 16E3,16E3HZ;SWPLVLTHD 10E3,10E3HZ;SWPLVLTHD
      6.3E3,6.3E3HZ;SWPLVLTHD 4E3,4E3HZ;SWPLVLTHD 2.5E3,2.5E3HZ;SWPLVLTHD
      1.6E3,1.6E3HZ;SWPLVLTHD 1E3,1E3HZ;SWPLVLTHD 630,630HZ;SWPLVLTHD 400,400HZ;SWPLVLTHD
      250,250HZ;SWPLVLTHD 160,160HZ;SWPLVLTHD 100,100HZ;SWPLVLTHD 63,63HZ;SWPLVLTHD
      40,40HZ;SWPLVLTHD 20,20HZ;:AGEN:FREQ 1E3HZ;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(60) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the measurement data.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-4. Visual Basic subprogram AGEN_ANLR_FreqRespSwp code

```
AGEN_ANLR_FreqRespSwp

19999.1HZ,0;9.3738DBRA,0;0.00564884PCT,0;16002.3HZ,0;10.7447DBRA,0;0.0122445PCT,0;9999.13
HZ,0;8.59601DBRA,0;0.0454303PCT,0;6302.94HZ,0;6.75496DBRA,0;0.0523172PCT,0;3997HZ,0;5.087
42DBRA,0;0.0657679PCT,0;2501.13HZ,0;3.63876DBRA,0;0.0714291PCT,0;1600.47HZ,0;2.39538DBRA,
0;0.0617398PCT,0;1000.06HZ,0;-0.000665789DBRA,0;0.0486728PCT,0;630.386HZ,0;-
2.06875DBRA,0;0.0418493PCT,0;399.74HZ,0;-4.03281DBRA,0;0.035284PCT,0;250.163HZ,0;-
6.07738DBRA,0;0.0442731PCT,0;160.056HZ,0;-8.03881DBRA,0;0.0454996PCT,0;100.012HZ,0;-
9.68786DBRA,0;0.0487011PCT,0;63.0322HZ,0;-11.1055DBRA,0;0.0529194PCT,0;39.9885HZ,0;-
12.0542DBRA,0;0.060396PCT,0;20.0027HZ,0;-11.6529DBRA,0;0.058004PCT,0
```

Figure 2-5. Visual Basic subprogram AGEN_ANLR_FreqRespSwp measurement results

Gain Linearity Sweep – Analog Generator Amplitude Sweeps with Analog Analyzer Measurements

A gain linearity sweep is accomplished by sweeping the analog generator output amplitude and measuring analog analyzer input amplitude with a bandpass filter for each generator amplitude setting. The bandpass filter is used in order to measure gain below the wideband noise floor of the device under test. Set up both the generator and analyzer for initial conditions, then execute a loop structure, such as a FOR LOOP, that increments or decrements the generator output amplitude and acquires the settled measurement. If a fixed set of output amplitudes is desired, then a single command message may be used that contains multiple generator output amplitude settings and associated measurement queries for each.

The Visual Basic code example in figure 6 below sweeps 10 output amplitude levels in –10 dB steps from 10 dBV to –90 dBV and measures analyzer channel A Bandpass amplitude in DBGA units with generator frequency set to 1 kHz. The DBGA unit provides the gain of the device with respect to the generator output amplitude (the level on the device's input). The test results are for a 15 band graphic equalizer with XLR inputs and outputs with all band controls set to midrange amplitude. This code is provided in the "S2Gsample2.bas" file on the CD ROM provided with this manual.

```
Sub AGEN_ANLR_GainLinSwp()
  ' Gain Linearity Sweep - Analog Generator Amplitude Sweeps with Analog Analyzer
      Measurements
  TEST_TITLE = "AGEN_ANLR_GainLinSwp"
  If IsAnalog = False Then
    MsgBox "System Two Analog option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      INPUT;:AGEN:OUTPUT AB;AMPL A,10DBV;FREQ 1E3HZ;WFM SINE,SINE;:ANLR:AUTORANGE
      AB,ON;INPUT AB,XLR;CHANNEL A;MODE BP;LPF F30K;TUNINGSRC FIXED;FILTERFREQ
      1E3HZ;RDGRATE APRECISE;RESPONSE 1E3;:SETTLING:ANLR BP,0.1,1E-06V,4,0.3,FLAT,4,1"
  ' Send the measurement acquisition commands.
  S2GSend ":AGEN:AMPL AB,10DBV;:ANLR:FUNC? DBGA;:AGEN:AMPL AB,0DBV;:ANLR:FUNC?
      DBGA;:AGEN:AMPL AB,-10DBV;:ANLR:FUNC? DBGA;:AGEN:AMPL AB,-20DBV;:ANLR:FUNC?
      DBGA;:AGEN:AMPL AB,-30DBV;:ANLR:FUNC? DBGA;:AGEN:AMPL AB,-40DBV;:ANLR:FUNC?
      DBGA;:AGEN:AMPL AB,-50DBV;:ANLR:FUNC? DBGA;:AGEN:AMPL AB,-60DBV;:ANLR:FUNC?
      DBGA;:AGEN:AMPL AB,-70DBV;:ANLR:FUNC? DBGA;:AGEN:AMPL AB,-80DBV;:ANLR:FUNC?
      DBGA;:AGEN:AMPL AB,-90DBV;:ANLR:FUNC? DBGA;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(60) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the measurement data.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-6. Visual Basic subprogram AGEN_ANLR_GainLinSwp code

```
AGEN_ANLR_GainLinSwp

4.22223DBGA,0;4.23746DBGA,0;4.21388DBGA,0;4.21494DBGA,0;4.23324DBGA,0;4.26561DBGA,0;4.263
69DBGA,0;4.20791DBGA,0;4.52924DBGA,0;4.48045DBGA,0;4.60453DBGA,0
```

Figure 2-7. Visual Basic subprogram AGEN_ANLR_GainLinSwp measurement results

Noise Sweep – Analog Analyzer Bandpass Filter Frequency Sweeps

Swept-bandpass noise measurements may be made by changing the center frequency of the analog analyzer bandpass filter and then acquiring a measurement. The settling parameters for noise measurements of this type should be set up to use the AVG algorithm and number of measurement points in order to provide an average of a series of noise measurements. Note that the settling parameters in the example below specify four measurements to be averaged (the fourth argument parameter) and a settling delay of 0.3 second. This delay assures sufficient time for the bandpass filter to be tuned to the desired center frequency before measurements are acquired into the settling queue.

The reading rate and bandpass filter center frequency are automatically set with the **RESPONSE** command when used in conjunction with the **:ANLR:RDGRATE AFAST,FUNCMETER** and **:ANLR:TUNINGSRC FIXED** commands. The **:ANLR:RDGRATE AFAST,FUNCMETER** command specifies auto-fast reading rate for the function meter (the measurement of audio level at the output of the bandpass filter). The **:ANLR:TUNINGSRC FIXED** command specifies fixed tune mode for the bandpass filter. These two commands in conjunction with the **:ANLR:RESPONSE** command will automatically select the best reading rate for the frequency band with a center frequency set to the argument of the **RESPONSE** command. The measurement commands may be sent as a single string of multiple analyzer response frequency settings and function meter measurement queries, as shown below. The sweep may also be implemented from within a simple FOR/NEXT loop that computes the response frequency, formats and sends each instance of **`:ANLR:RESPONSE n;FUNC? V`**, and then reads each query response.

The Visual Basic program example below sweeps 16 bandpass center frequencies from 20 KHz to 20 Hz and measures analyzer channel A noise in volts. The test results are for a 15 band graphic equalizer with XLR inputs and outputs and gain at 1 kHz set to midrange. This code is provided in the “S2Gsample2.bas” file on the CD ROM provided with this manual.

```
Sub AGEN_ANLR_BPFRNoiseSwp()
  ' Noise Sweep - Analog Analyzer Bandpass Filter Frequency Sweeps
  TEST_TITLE = "AGEN_ANLR_BPFRNoiseSwp"
  If IsAnalog = False Then
    MsgBox "System Two Analog option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE MONO;MONO
      FUNCMETER;:AGEN:OUTPUT OFF;AMPL A,0V;WFM SINE,SINE;:ANLR:INPUT AB,XLR;CHANNEL
      A;MODE BP;LPF F30K;TUNINGSRC FIXED;RDGRATE AFAST,FUNCMETER;:SETTLING:ANLR BP,1,1E-
      07V,4,0.3,AVG,4,1"
  ' Send the measurement acquisition commands.
  S2GSend ":ANLR:RESPONSE 20E3;FUNC? V;RESPONSE 16E3;FUNC? V;RESPONSE 10E3;FUNC?
      V;RESPONSE 6.3E3;FUNC? V;RESPONSE 4E3;FUNC? V;RESPONSE 2.5E3;FUNC? V;RESPONSE
      1.6E3;FUNC? V;RESPONSE 1E3;FUNC? V;RESPONSE 630;FUNC? V;RESPONSE 400;FUNC?
      V;RESPONSE 250;FUNC? V;RESPONSE 160;FUNC? V;RESPONSE 100;FUNC? V;RESPONSE 63;FUNC?
      V;RESPONSE 40;FUNC? V;RESPONSE 20;FUNC? V;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(60) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the measurement data.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-8. Visual Basic subprogram AGEN_ANLR_BPFRNoiseSwp code

```
AGEN_ANLR_BPFRNoiseSwp

8.44193E-05V,0;8.81805E-05V,0;7.63397E-05V,0;3.83492E-05V,0;2.75154E-05V,0;1.97716E-
05V,0;1.58529E-05V,0;1.13268E-05V,0;1.26066E-05V,0;9.95731E-06V,0;2.33116E-
05V,0;2.79789E-05V,0;2.28219E-05V,0;2.56138E-05V,0;2.95138E-06V,0;3.5584E-06V,0
```

Figure 2-9. Visual Basic subprogram AGEN_ANLR_BPFRNoiseSwp measurement results

Digital Generator Frequency Sweeps with DSP Analyzer Measurements

To sweep the digital generator output frequency and measure digital analyzer level and THD+N, set up both the digital generator and digital analyzer for initial conditions. A loop structure, such as a FOR LOOP, may be used to increment the generator frequency and the analyzer response frequency and acquire the settled measurements. The frequency of the distortion notch filter is automatically set to the frequency of the digital generator by the **`:DSP:DANLR:TUNINGSRC DGEN`** and **`:DGEN:FREQ`** commands.

A single message with multiple occurrences of a command macro invocation (for each frequency) may be used if a fixed set of frequencies is desired. The example below sweeps

16 frequencies from 20 KHz to 20 Hz and measures analyzer channel A THD+N Ratio in dB units, Gain (Level A) in dBr units with reference to the digital input level (DBR1), and frequency in HZ units. The test results are for a stereo digital audio processor. This code is provided in the “S2Gsample2.bas” file on the CD ROM included with this manual.

```
Sub DGEN_DANLR_FreqRespSwp()
  ' Digital Generator Frequency Sweeps with DSP Analyzer Measurements
  TEST_TITLE = "DGEN_DANLR_FreqRespSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;FRQ1 1E3HZ;WFM SINE,SINE"
  S2GSend ":DIN:FORMAT XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM
      OFF;CMOUTPUT OFF;INVALID 0;JWFM NONE;NOUTPUT OFF;PREEMPHASIS OFF;RATE
      48000HZ;RESOLUTION 24;RFENABLE FIXED;FORMAT XLR"
  S2GSend ":DSP:PROGRAM DANLR;REF:DBR1 -10DBFS;DBR2 -10DBFS;:DSP:DANLR:INPUT
      DIGITAL;FAUTORANGE ON;DETECTOR FRMS;COUPLING AB,AC;CHANNEL A;MODE THDRATIO;LPF
      FS_2;TUNINGSRC DGEN;RDGRATE AUTO,LEVEL,FREQ,FUNCMETER;RESPONSE 1E3;:DELAY
      0.1;:SETTLING:DANLR:FREQ A,1,0.001HZ,2,0.01,FLAT,4,1;LEVEL CHAD,FRMS,5,1E-
      06FFS,3,0.1,EXP,4,1;FUNC THDRATIO,FRMS,5,5E-05PCT,3,0.1,EXP,4,1"
  ' Send a message that purges all macro definitions, enables all macros, and defines a
      sweep macro for generator output frequency, analyzer reading rate response
      frequency, and measurement queries.
  S2GSend "*PMC;*EMC 1;*DMC ""DIGSWPLVLTHD"",#279:DSP:DANLR:RESPONSE $1;:DGEN:FRQ1
      $2;:DSP:DANLR:FREQ? A,HZ;LEV? A,DBR1;FUNC? DB"
  ' Send the measurement acquisition commands.
  S2GSend ":DGEN:FRQ1 20E3HZ;:DIGSWPLVLTHD 20E3,20E3HZ;DIGSWPLVLTHD
      16E3,16E3HZ;DIGSWPLVLTHD 10E3,10E3HZ;DIGSWPLVLTHD 6.3E3,6.3E3HZ;DIGSWPLVLTHD
      4E3,4E3HZ;DIGSWPLVLTHD 2.5E3,2.5E3HZ;DIGSWPLVLTHD 1.6E3,1.6E3HZ;DIGSWPLVLTHD
      1E3,1E3HZ;DIGSWPLVLTHD 630,630HZ;DIGSWPLVLTHD 400,400HZ;DIGSWPLVLTHD
      250,250HZ;DIGSWPLVLTHD 160,160HZ;DIGSWPLVLTHD 100,100HZ;DIGSWPLVLTHD
      63,63HZ;DIGSWPLVLTHD 40,40HZ;DIGSWPLVLTHD 20,20HZ;:DGEN:FRQ1 1E3HZ;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(10) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the measurement data.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-10. Visual Basic subprogram DGEN_DANLR_FreqRespSwp code

```
DGEN_DANLR_FreqRespSwp

20000.5HZ,0;-8.58719DBR1,0;-92.0705DB,0;16000HZ,0;-6.05109DBR1,0;-
50.1342DB,0;9999.62HZ,0;-5.03042DBR1,0;-99.6664DB,0;6299.99HZ,0;-4.80581DBR1,0;-
96.8183DB,0;4000HZ,0;-4.72112DBR1,0;-101.573DB,0;2500HZ,0;-4.66836DBR1,0;-
103.117DB,0;1600HZ,0;-4.57838DBR1,0;-103.077DB,0;999.999HZ,0;-4.3343DBR1,0;-
103.124DB,0;629.991HZ,0;-3.77563DBR1,0;-92.2192DB,0;399.991HZ,0;-2.70159DBR1,0;-
96.0298DB,0;250.013HZ,0;-0.95235DBR1,0;-95.6729DB,0;160.008HZ,0;1.0205DBR1,0;-
85.0419DB,0;100.024HZ,0;2.84032DBR1,0;-104.4DB,0;62.9339HZ,0;3.99448DBR1,0;-
106.995DB,0;39.9628HZ,0;4.5229DBR1,0;-114.104DB,0;19.9957HZ,0;4.45285DBR1,0;-111.451DB,0
```

Figure 2-11. Visual Basic subprogram DGEN_DANLR_FreqRespSwp measurement result

Sweeps and Measurements with the BITTEST Digital Data Analyzer

The BITTEST Digital Data Analyzer DSP program measures digital audio signals for bit errors in conjunction with certain specific signals that the digital audio generator creates. The audio test signal may be a pseudo-random noise sequence, constant valued samples ("digital dc"), a sinewave of selectable amplitude and frequency, or walking bit patterns. Generated word width must generally be equal to or greater than measurement word width. The measurements provide both real-time received data and errors in the received data sequence. Any amount of delay between transmitted and received signals is permissible, allowing testing of devices and transmission links with large amounts of delay or even recorder-reproducers. Digital generator dither cannot be used during BITTEST analysis except with sine waveforms.

BITTEST is useful for investigating the integrity of digital audio data links, recorders, etc. It is also invaluable for design test of digital interfaces. Each waveform measured by BITTEST has a specific testing application. BITTEST operates only with digital domain input and output.

The VB5 sample program listing below illustrates how to use BITTEST to make digital data analysis measurements at five different digital generator amplitude settings using the CONSTANT waveform (digital DC), followed with a measurement using the “Freeze Data On Error” mode (provided by the :DSP:BITTEST:FREEZE command) to capture an error condition. This demonstrates how to use the :DSP:BITTEST:FROZEN? query to automatically determine when an error has caused a frozen condition. In this program example, the digital generator dither has been deliberately set to TRIANGULAR instead of NONE in order to cause an error condition with the CONSTANT waveform.

(Start of Figure 2-12)

```
Sub DGEN_BITTEST()
  Dim sDataChA As String, sDataChB As String, sErrorChA As String, sErrorChB As String,
     bFrozen As Boolean
  ' Digital Generator Constant Waveform Amplitude Sweep with BITTEST Analyzer
  TEST_TITLE = "DGEN_BITTEST"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. NOTE: *RCL 0 DOES NOT PROPERLY INITIALIZE THE DSP.
     *RST DOES PROPERLY INITIALIZE THE DSP.
  S2GSend "*RST;*ESE 0;*SRE 0;:APST:ENAB 256;:HEADER OFF;:DGEN:OUTPUT AB;AMPL
     AB,0FFS;DITHERTYPE NONE;WFM SPCIAL,CONSTANT"
  S2GSend ":DIN:FORMAT XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM
     OFF;CMOUTPUT OFF;INVALID 0;JWFM NONE;NOUTPUT OFF;PREEMPHASIS OFF;RATE
     48000HZ;RESOLUTION 24;RFENABLE FIXED;FORMAT XLR"
  S2GSend ":DSP:PROGRAM BITTEST;:DSP:BITTEST:MODE NORMAL;RDGRATE R16;WFM CONSTANT;FREEZE
     OFF"
  ' Send a message that purges all macro definitions, enables all macros, and defines a
     sweep macro for generator output, and analyzer measurement queries.
  S2GSend "*PMC;*EMC 1;*DMC ""DIGSWPBIT1"",#0:DGEN:AMPL AB,$1;:DELAY
     .5;:DSP:BITTEST:CLEAR;DATA? A,HEX;ERROR? A,HEX;DATA? B,HEX;ERROR? B,HEX"
  ' Send the measurement acquisition commands.
  S2GSend ":DIGSWPBIT1 1FFS;DIGSWPBIT1 .5FFS;DIGSWPBIT1 0FFS;DIGSWPBIT1 -.5FFS;DIGSWPBIT1
     -1FFS"
  ' Wait for AESB bit in Status Byte
  If WaitForAESB(5) = False Then
    UserErrMsg "System Two Measurement Timeout in module " & TEST_TITLE  ' If AESB time
      out then exit.
    S2GSend "*CLS"   ' Clear status registers.
    Exit Sub ' Quit, do not read response.
  End If
  ' Read the measurement data.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set Freeze Data On Error mode and set digital generator dither to triangular in order
      to cause a data error
  S2GSend ":DGEN:AMPL AB,0.0FFS;:DELAY 0.2;:DSP:BITTEST:CLEAR;FREEZE ON;:DELAY
      0.2;:DGEN:DITHERTYPE TRI"
  ' Wait for errors to occur, get frozen data and errors for channels A and B
  bFrozen = WaitforBittestFreeze(10, "HEX", sDataChA, sErrorChA, sDataChB, sErrorChB)
  If bFrozen Then
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & vbCrLf &
      "BITTEST Frozen on Error"
  Else
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & vbCrLf & "No
      BITTEST Error after 10 seconds"
  End If
  S2GSend ":DSP:BITTEST:FREEZE OFF"
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & "  A Data: " &
      sDataChA & vbCrLf & "  A Error: " & sErrorChA & vbCrLf & "  B Data: " & sDataChB &
      vbCrLf & "  B Error: " & sErrorChB & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit.
End Sub

Function WaitforBittestFreeze(fTimeout As Single, sUnit As String, sDataA As String,
      sErrorA As String, sDataB As String, sErrorB As String) As Boolean
  ' Wait for a frozen condition to occur or until timeout value is exceeded if FREEZE is
      ON.
  ' Read BITTEST channels A and B, DATA and ERROR measurements if data is frozen before
      the timeout period has elapsed.
  ' Return the last measurement value for Channel A DATA, Channel A ERROR, Channel B
      DATA, and Channel B ERROR that is valid (not a NAN).
  ' Return function value of True if FROZEN, else return False.
  Dim sDataAFrz As String, sErrorAFrz As String, sDataBFrz As String, sErrorBFrz As
      String, fStartTime As Single, sErr As String, sFreeze As String
  sUnit = UCase(sUnit)
  ' Exit function if BITTEST units are invalid
  If Not (sUnit = "HEX" Or sUnit = "DEC") Then Exit Function
  ' Check that BITTEST is already setup for Freeze Data On Error
  S2GSend ":DSP:BITTEST:FREEZE?"
  sFreeze = S2GRcv
  ' Exit function if BITTEST is not setup for Freeze Data On Error
  If InStr(sFreeze, "ON") = 0 Then Exit Function  ' Exit with False if BITTEST is not in
      "Freeze Data On Error" mode.
  If sUnit = "HEX" Then sErr = "80000000HEX"    ' hexadecimal NAN (not-a-number)
  If sUnit = "DEC" Then sErr = "2147483648DEC"  ' decimal NAN (not-a-number)
  fStartTime = Timer
  Do
    ' Check if data is frozen because of error
    S2GSend ":DSP:BITTEST:FROZEN?"
    WaitforBittestFreeze = Val(S2GRcv)
    If WaitforBittestFreeze Then
      ' Channel A Data
      S2GSend ":DSP:BITTEST:DATA? A," & sUnit
      sDataAFrz = S2GRcv
      Do While InStr(sDataAFrz, sErr) = 0   ' Read while response is not NAN
        sDataA = sDataAFrz
        S2GSend ":DSP:BITTEST:DATA? A," & sUnit
        sDataAFrz = S2GRcv
      Loop
      ' Channel A Error
      S2GSend ":DSP:BITTEST:ERROR? A," & sUnit
      sErrorAFrz = S2GRcv
      Do While InStr(sErrorAFrz, sErr) = 0   ' Read while response is not NAN
        sErrorA = sErrorAFrz
        S2GSend ":DSP:BITTEST:ERROR? A," & sUnit
        sErrorAFrz = S2GRcv
      Loop
      ' Channel B Data
      S2GSend ":DSP:BITTEST:DATA? B," & sUnit
      sDataBFrz = S2GRcv
      Do While InStr(sDataBFrz, sErr) = 0   ' Read while response is not NAN
```

```
        sDataB = sDataBFrz
        S2GSend ":DSP:BITTEST:DATA? B," & sUnit
        sDataBFrz = S2GRcv
      Loop
      ' Channel B Error
      S2GSend ":DSP:BITTEST:ERROR? B," & sUnit
      sErrorBFrz = S2GRcv
      Do While InStr(sErrorBFrz, sErr) = 0   ' Read while response is not NAN
        sErrorB = sErrorBFrz
        S2GSend ":DSP:BITTEST:ERROR? B," & sUnit
        sErrorBFrz = S2GRcv
      Loop
    End If
  ' Loop while data is not frozen and timer has not exceeded timeout value.
  Loop While WaitforBittestFreeze = False And Timer - fStartTime < fTimeout
End Function
```

Figure 2-12. Visual Basic code for subprogram DGEN_BITTEST and function WaitforBittestFreeze.

```
DGEN_BITTEST

7FFFFFHEX;0HEX;7FFFFFHEX;0HEX;400000HEX;0HEX;400000HEX;0HEX;0HEX;0HEX;0HEX;0HEX;FFC00000H
      EX;0HEX;FFC00000HEX;0HEX;FF800001HEX;0HEX;FF800001HEX;0HEX

BITTEST Frozen on Error
  A Data: FFFFFFFFHEX
  A Error: 1HEX
  B Data: 0HEX
  B Error: 0HEX
```

Figure 2-13. Visual Basic measurement results for subprogram DGEN_BITTEST and function WaitforBittestFreeze.

DSP Measurement Sweeps with FFT, INTERVU, FASTTEST, and MLS

The DSP programs FFT, FASTTEST, and MLS acquire and process stereo waveforms from either the digital audio inputs or from the analog to digital converters. The INTERVU program digitizes and processes the serial audio pulse waveform on the AES/EBU or SPDIF or TOSLINK inputs and provides jitter measurements and pulse measurements.

The DSP programs operate in two states: SETUP and READING. The SETUP state is required in order to load a DSP program and specify its parameters prior to acquiring and processing a signal. The READING state specifies the type of measurements to be provided after acquisition, transforming, or reprocessing has occurred. The :DSP:OPSTate command specifies either SETUP or READING state.

The process of acquisition and digital signal processing requires preliminary setup of the DSP with the :DSP:SRCParams command in SETUP state, followed by one of three commands that start the DSP process in READING state: ACQX?, XFRM?, and REPROCESS? commands. The results of acquisition and processing is one or two arrays of measurement data values stored within the System Two DSP engine. The SRCPARAMS command specifies which type of data values are to be provided in response to specific queries, the range of measurement frequencies or time intervals, the number of steps within the range, the intervals between points to be measured (LOG, LIN, or ARBITRARY table), and the use of peak-picking for spectrum measurements. In the case of the FFT DSP program, two types of data are available, time domain and frequency domain. The SRCPARAMS command specifies which domain and the units of the parameter. This may be thought of in terms of specifying the abscissa or x-axis values on a coordinate plane; for example, the frequency of a spectrum bin or the time value of a waveform sample.

The ACQX? command initiates acquisition of the signal(s) into the waveform buffer(s), followed by additional processing depending on the DSP program. The XFRM? query performs an FFT transform or other DSP program-specific processing of the waveform(s) already acquired in the input waveform buffer(s). The REPROCESS? command causes measurement data to be extracted from the data already transformed or processed by the DSP. Each command responds with a numeric value when the process has completed or when a timeout value is exceeded. The timeout period is specified with the :DSP:TIMeout command. If the response to the ACQX?, XFRM?, or REPROCESS? commands is 0, then the process has completed normally and data is ready to be extracted with the appropriate query commands. If the response is 1, then the process has been aborted due to a timeout and no measurements will be available. Thus, the response to these queries must be checked with measurement queries before proceeding.

Two techniques may be used to read the data available from the DSP engine after a successful acquire, transform, or reprocess: DSP program-specific measurement queries, or the faster :DSP:BATCh? query. Each DSP program has a specific set of measurement queries that provide one measurement for each query. Each query specifies the source parameter x-axis value to be read and the specific meter to be read. For example to read the 1.000 kHz FFT spectrum bin on channel 1, the correct query would be :DSP:FFT:AMPL? 1,FFS,1000 if the :DSP:OPSTate READing,AMP1 was already sent and an acquisition had already been made with the :DSP:ACQX? Command. Thus, many queries are required in order to read a large array of data using these individual measurement queries because each query may only read one spectrum bin.

The :DSP:BATCh? query automatically sweeps the x-axis values according to the :DSP:SRCPARAMS parameters and reads the values specified by the :DSP:OPSTate READ parameters. It returns the data as sets of data for each x-axis value bundled in a single arbitrary block data response. This method is much faster because the internal microprocessor in the System Two does all the computations for the sweep x-axis values or uses one of the two sweep tables you may specify with the :DSP:TABLe command.

Three possible formats are provided for the :DSP:BATCh? query: ASCII, BINARY, and RBINARY. The ASCII format is comma separated ASCII coded floating-point numbers. The ASCII format can require as many as 14 bytes per data point, thus requiring more bytes for the transfer of data. The ASCII format response of the BATCH? query is about three times faster than the equivalent number of DSP program specific measurement queries because the System Two processor does all the work of computing the x-axis values within the range of the parameters in the :DSP:SRCPARAMS command.

The BINARY format for the :DSP:BATCh? query is single precision floating-point numbers grouped in four-byte words with the most significant bit sent first, encoded according to IEEE standard 754-1985 specified in the IEEE-488.2 GPIB interface standard. This is the internal numeric format normally used in Motorola microprocessors called “big endian”. Each four byte group must be decoded according to the standard. Refer to appendix F for a detailed definition of the bit coding for this format.

The RBINARY format is a byte-reversed version of the BINARY format, typically called “little endian”. This RBINARY format is more convenient for use with Intel processors used in personal computers. Both binary coded formats result in faster reads of data because the System Two GPIB processor can pass its internal computer representation of the data directly to the GPIB output queue and does not waste time formatting the data as ASCII strings. The binary format of data requires only four bytes of data per data point. The BINARY and RBINARY formats are about eight times faster for reads of data into computer arrays than the equivalent read done with iterations of the DSP program specific measurement queries.

The set of data values within the arbitrary block in the response to the :DSP:BATCh? query may contain one or more measurements that include the value of the x-axis value for each set of measurements. For example, if the commands ":DSP:OPSTate SETUP;SRCPARAMS FREQ, HZ, 10, 1000, 1, LOG;OPSTATE READ, AMP1, AMP2, PHA2; ACQX?" has been sent followed by ":DSP:BATCh? ASCII, AMP1, DBV, AMP2, DBV, PHA2, DEG", then the first set of data within the arbitrary block will start with the source parameter frequency of 10 Hz, followed by three comma separated measurements for channel 1 amplitude, channel 2 amplitude, and channel 2 phase. You can see the format of this data in the figures later in this section that show the measurement data responses from :DSP:BATCh? queries.

When re-sweeping data previously acquired and when you desire to change the SRCPARAMS sweep parameters command and OPSTate READing command parameters, you must use the REPROCESS? query prior to reading the data with the :DSP:BATCh? Query.

The acquisition process is as follows:

1. Load and set up the DSP program.
2. With DSP OPSTATE in SETUP mode, specify the sweep source parameters with the SRCPARAMS command. This is the parameter to be swept, its unit, its sweep range, the number of sweep steps, and the sweep type (whether LOG or LIN spacing or ARB for table values loaded by the :DSP:TABLe command).
3. Specify the desired measurement type with :DSP:OPSTate READing.
4. Acquire the signal with :DSP:ACQX?.
5. Read the response to ACQX? and test it to determine that the acquisition was successful. If successful then proceed, else halt.
6. Use the appropriate DSP program's measurement queries or the :DSP:BATCh? query to read the measurements. These specify the value to query for the type of source parameter previously set by the SRCPARAMS command. These must be compatible with the parameters used with the :DSP:OPSTate READing command.
7. Set the DSP OPSTATE back to the SETUP mode.

The Visual Basic program examples shown in the following sections illustrate how the commands described above should be used with each of the DSP programs.

Spectrum Peak-Picking with the SRCParams Command

The last parameter of the SRCParams command specifies the use of a peak-picking algorithm for frequency domain spectrum sweeps. This parameter is ignored if a time sweep is specified. The arguments for the last parameter may be ARB, LIN, or LOG. ARB must be used to disable the peak-pick algorithm and report the magnitude of the requested bin frequency specified with the `:FFT:AMPL?` command. Use ARB if the application requires measurements of specific bins only or if you intend to use a sweep table with the :DSP:BATCh? query.

The peak-picking algorithm is enabled with the use LIN or LOG. The peak-picking algorithm provides a means of automatically reporting the highest amplitude of all frequency bins between the last bin frequency measured and the current one requested. This feature makes it possible to find the highest bin amplitude within a span of frequency bins without knowing the exact bin frequency of that highest amplitude signal. Thus if it is desired to know the highest signal amplitude within a frequency span, use the LIN or LOG argument and query for the amplitude of the bins at the lower and upper ends of the frequency span. If the lowest frequency is the start frequency in the SRCPARAMS parameter list, then the next frequency requested will result in a measurement of the highest signal found in the frequency span between the start frequency and the second one requested. To illustrate this, Figure 2-14 shows a spectrum of a 1010.74 Hz signal driving an analog device at the clipping point. Note that distortion components appear at the expected 2021.48 Hz second harmonic, that intermodulation products are present, and that there is an interference signal at 1048.83 Hz.

If the message **`:DSP:OPSTATE SETUP;SRCPARAMS FREQ,HZ,1010.74,2021.48,2,ARB;OPSTATE READ,AMP1;:DSP:ACQX?`** is used to acquire the signal followed by **`:DSP:FFT:AMPL? 1,1010.74,DBV;AMP? 1,2021.48,DBV`** to measure the fundamental and second harmonic of 1010.74 Hz, then the response will be the amplitudes for the bins at those two frequencies: **`18.414;-50.635`**. However, if the SRCPARAMS command message uses LIN or LOG as the last argument instead of ARB in the command above, then the response will be quite different due to the peak-pick algorithm: the first measurement response will be the same, 18.414 dBV, but the second measurement response will be the highest bin amplitude found above the 1010.74 Hz bin and equal to or below the bin at 2021.48 Hz. By inspection of Figure 2-15, it is obvious that this will be the bin at 1048.83 Hz, the interference signal at a higher level than the second harmonic. Therefore, the second measurement response will be –31.281 dBV and not the amplitude of the second harmonic.

The peak-pick algorithm must know the frequency of the first spectrum bin in a span of bins in order to be able to search for the highest bin in the span. During a sweep, the algorithm assumes the sweep proceeds in the same direction and never reverses. The order of search is determined by the relationship of the start frequency and the stop frequency. The start frequency may be above the stop frequency, provided the sweep queries proceed from high to low. Note that the peak-pick algorithm will operate in reverse for a sweep from high to low and will search for peaks above the next lower frequency in a sweep. Thus, a spectrum sweep with peak-picking enabled with LIN or LOG may result in different measurements, depending on the direction of the sweep and the nature of the signal.

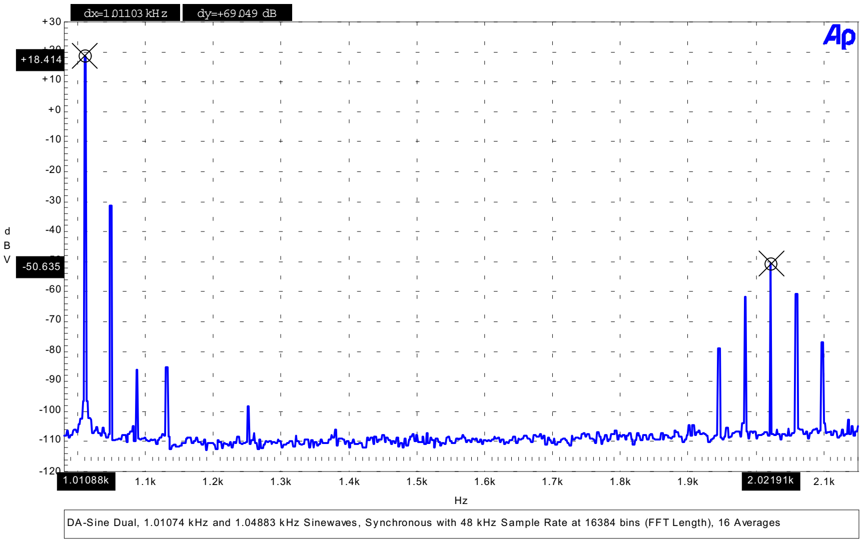


Figure 2-14. Spectrum of signal with distortion components, fundamental at 1010.74 Hz (cursor 1) with second harmonic at 2021.48 Hz (cursor 2)

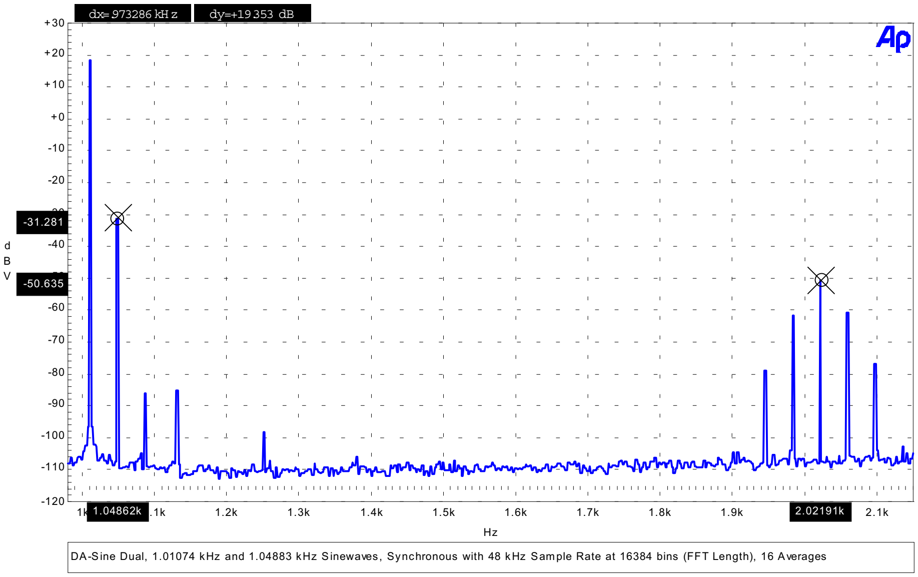


Figure 2-15. Spectrum of signal with distortion components, 1048.83 Hz interference signal at -31.281 dBV (cursor 1) is the highest signal between the fundamental at 1010.74 Hz and its second harmonic at 2021.48 Hz (cursor 2)

Spectrum Sweeps with the FFT DSP Program

DSP-based frequency spectra and time waveforms may be acquired with the FFT DSP program. The FFT DSP program acquires and measure signals in a "batch" process. The results of an acquisition may be read as a series of measurements. In the case of a spectrum, each spectral line (bin) is read by sending a query for amplitude at a specified frequency for the specified channel. The process requires an initial setup, followed by the acquisition of the signal, and then followed by a series of queries and measurement responses.

The example below sets up the instrument for a digital input and output, loads the FFT program into the digital signal processor, sets up the FFT parameters for an acquisition, acquires the signal, and then reads the amplitude of the FFT bins at the specified bin frequencies. The spectrum of interest is 984 Hz to 1017 Hz in order to look at the 1 kHz fundamental signal from the System Two digital generator.

The program reads a raw acquisition waveform from the DSP channel 1 buffer and saves it into a file for later use. Then the digital generator waveform is switched off and a new acquisition is started in order to erase the original signal in the acquisition buffers. The spectrum of the new input signal is processed and displayed to prove that the original signal is gone.

The original acquisition waveform is then read from the file, written back into the original waveform buffer, and then transformed to produce a spectrum identical to the original spectrum. See the section entitled "Saving and Recalling Waveforms with Batch Mode DSP Programs," above, for more information about the process of saving and recalling acquisition waveforms. This code is provided in the "S2Gsample2.bas" file on the CD ROM included with this manual.

(Start of Figure 2-16)

```
Sub FFT_SpectrumSwp()
  ' Spectrum Sweeps with the FFT DSP Program
  TEST_TITLE = "FFT_SpectrumSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  sIO = ""
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;FRQ1 1E3HZ;WFM SINE,SINE;:DIN:FORMAT
      XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID
      0;JWFM NONE;NOUTPUT OFF;PREEMPHASIS OFF;RATE 48000HZ;RESOLUTION 24;RFENABLE FIXED"
  S2GSend ":DSP:PROGRAM FFT;:DSP:FFT:INPUT DIGITAL;ACQLENGTH XLENGTH;AVGS 1;AVGTYPE
      1;COUPLING DC;DELAY 0;MODE INTRPOLATE;SRC1 A;SRC2 B;INPUT DIGITAL;START 0;TRIGGER
      FREE;TSLOPE POS;WINDOW EQR;XLENGTH 16384;SENS 0.001FFS;PKTRIG 1;:DELAY 0.2"
  ' Setup the DSP source parameters for a frequency sweep of the FFT bins and then
      acquire the signal on channel 1 (A) and process for a frequency domain spectrum.
  S2GSend ":DSP:OPSTATE SETUP;TIMEOUT 2;SRCP FREQ,HZ,0,0,0,ARB;OPSTATE
      READ,AMP1;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout"  ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send amplitude measurement queries for channel 1 spectrum bins.
  S2GSend ":DSP:FFT:AMPL? 1,DBFS,984;AMPL? 1,DBFS,987;AMPL? 1,DBFS,990;AMPL?
      1,DBFS,993;AMPL? 1,DBFS,996;AMPL? 1,DBFS,999;AMPL? 1,DBFS,1002;AMPL?
      1,DBFS,1005;AMPL? 1,DBFS,1008;AMPL? 1,DBFS,1011;AMPL? 1,DBFS,1014;AMPL?
      1,DBFS,1017"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      New FFT Acquisition #1." & vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  SaveWaveform 1, App.Path & "\Wav1.aas"   ' Save waveform in FFT acquisition buffer 1 to
      a file.
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
  ' Acquire again with no digital generator output.
  S2GSend ":HEADER OFF;:DGEN:OUTPUT OFF;:DSP:OPSTATE READ,AMP1;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout"   ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send amplitude measurement queries for channel 1 spectrum bins.
  S2GSend ":DSP:FFT:AMPL? 1,DBFS,984;AMPL? 1,DBFS,987;AMPL? 1,DBFS,990;AMPL?
      1,DBFS,993;AMPL? 1,DBFS,996;AMPL? 1,DBFS,999;AMPL? 1,DBFS,1002;AMPL?
      1,DBFS,1005;AMPL? 1,DBFS,1008;AMPL? 1,DBFS,1011;AMPL? 1,DBFS,1014;AMPL?
      1,DBFS,1017"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      New FFT Acquisition #2." & vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  ' Recall the DSP acquisition waveform from the file it was saved in and transform it
      spectrum.
  RecallWaveform App.Path & "\Wav1.aas"   ' Recall waveform from a file and put into DSP
      acquisition buffer.
  ' Transform the waveform in the DSP acquisition buffers.
  S2GSend ":DSP:OPSTATE READ,AMP1;XFRM?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the XFRM? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout"   ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send amplitude measurement queries for channel 1 spectrum bins.
  S2GSend ":DSP:FFT:AMPL? 1,DBFS,984;AMPL? 1,DBFS,987;AMPL? 1,DBFS,990;AMPL?
      1,DBFS,993;AMPL? 1,DBFS,996;AMPL? 1,DBFS,999;AMPL? 1,DBFS,1002;AMPL?
      1,DBFS,1005;AMPL? 1,DBFS,1008;AMPL? 1,DBFS,1011;AMPL? 1,DBFS,1014;AMPL?
      1,DBFS,1017"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      Original FFT Acquisition #1 recalled from file." & vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-16. Visual Basic subprogram FFT_SpectrumSwp code

```
FFT_SpectrumSwp New FFT Acquisition #1.

-99.7931;-60.9017;-37.7837;-22.9687;-14.1175;-10.2539;-11.0212;-16.4867;-27.1785;-
44.3928;-71.3917;-121.765

FFT_SpectrumSwp New FFT Acquisition #2.

-181.809;-178.012;-173.179;-173.188;-174.464;-174.257;-176.526;-183.85;-176.871;-
174.942;-178.491;-185.663

FFT_SpectrumSwp Original FFT Acquisition #1 recalled from file.

-99.7931;-60.9017;-37.7837;-22.9687;-14.1175;-10.2539;-11.0212;-16.4867;-27.1785;-
44.3928;-71.3917;-121.765
```

Figure 2-17. Visual Basic subprogram FFT_SpectrumSwp measurement results.

The BATCH? ASCII Query Reads FFT Spectrum Data 3X Faster

Another technique using the :DSP:BATCh? query simplifies the process of sweeps with batch mode DSP programs by utilizing the parameters of the :DSP:SRCParams command and the :DSP:OPSTate READing command to control an internal sweep of the DSP data. The data is returned in the response to the BATCH? query in three possible formats: ASCII, BINARY, and RBINARY. The Visual Basic code below for the subprogram Batch_FFT_SpectrumSwp shows this alternative code. This type of sweep is approximately three times faster than the method shown above. This example returns the data in ASCII format.

(start of Figure 2-18)

```
Sub Batch_FFT_SpectrumSwp()
  ' Spectrum Sweeps with the FFT DSP Program
  ' Acquire the signal on digital audio input channel 1.
  TEST_TITLE = "Batch_FFT_SpectrumSwp"
  If IsDigital = False Then
```

```
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  sIO = ""
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;FRQ1 1E3HZ;WFM SINE,SINE;:DIN:FORMAT
      XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID
      0;JWFM NONE;NOUTPUT OFF;PREEMPHASIS OFF;RATE 48000HZ;RESOLUTION 24;RFENABLE FIXED"
  S2GSend ":DSP:PROGRAM FFT;:DSP:FFT:INPUT DIGITAL;ACQLENGTH XLENGTH;AVGS 1;AVGTYPE
      1;COUPLING DC;DELAY 0;MODE INTRPOLATE;SRC1 A;SRC2 B;INPUT DIGITAL;START 0;TRIGGER
      FREE;TSLOPE POS;WINDOW EQR;XLENGTH 16384;SENS 0.001FFS;PKTRIG 1;:DELAY 0.2"
  ' Setup the DSP source parameters for a frequency sweep of the FFT bins and then
      acquire the signal on channel 1 (A) and process for a frequency domain spectrum.
  ' Setup the BATCH sweep parameters to sweep from 10 Hz to 10 KHz with 9 steps (10 bins)
      with LIN spacing.
  S2GSend ":DSP:OPSTATE SETUP;TIMEOUT 2;SRCP FREQ,HZ,10,10000,9,LIN;OPSTATE
      READ,AMP1;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send amplitude measurement queries for channel 1 spectrum bins.
  S2GSend ":DSP:BATCh? ASCII,AMP1,DBFS"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & "New FFT Acquisition #1 with LIN bin freq spacing (with peak-
      picking)."
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & "Format = FREQ,
      AMP1" & vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Setup the BATCH sweep parameters to sweep from 10 Hz to 10 KHz with 9 steps (10 bins)
      with LIN spacing.
  S2GSend ":DSP:OPSTATE SETUP;SRCP FREQ,HZ,10,10000,9,LOG;OPSTATE
      READ,AMP1;REPROCESS?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send amplitude measurement queries for channel 1 spectrum bins.
  S2GSend ":DSP:BATCh? ASCII,AMP1,DBFS"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & "Reprocess FFT
      Acquisition #1 with LOG bin freq spacing (with peak-picking)."
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & "Format = FREQ,
      AMP1" & vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-18. Visual Basic subprogram Batch_FFT_SpectrumSwp code

```
Batch_FFT_SpectrumSwp

New FFT Acquisition #1 with LIN bin freq spacing (with peak-picking).
Format = FREQ, AMP1
```

```
ASCII,10,10.0021,-170.025,1120,-10.2539,2230,-161.673,3340,-161.3,4450,-168.531,5560,-
166.133,6670,-168.398,7780,-167.654,8890,-168.079,10000,-168.426

Reprocess FFT Acquisition #1 with LOG bin freq spacing (with peak-picking).
Format = FREQ, AMP1

ASCII,10,10.0021,-170.025,21.5435,-170.387,46.4172,-166.733,99.9985,-165.625,215.441,-
166.438,464.161,-165.308,1000,-10.2539,2154.44,-11.0212,4641.59,-161.3,10000,-166.133
```

Figure 2-19. Visual Basic subprogram Batch_FFT_SpectrumSwp measurement results using the ASCII format for the BATCH? query.

The BATCH? Binary Formats Read FFT Spectrum Data 8X Faster

Binary formats with the :DSP:BATCh? query provide more speed, the RBINARY and BINARY formats. The data is returned in the response to the BATCH? as 4-byte IEEE single precision floating point numbers that must be byte-ordered and converted into the internal numeric representation of the controller's processor. The Visual Basic code below for the subprogram Batch_RBinary_To_Array_FFT_StereoSpectrumSwp shows code that acquires stereo spectrum data. This type of sweep is approximately eight times faster than the :DSP:FFT:AMPL? method shown above. The subprogram ReadBatch2 performs the read and parsing of the RBINARY formatted response into a dynamic single float array structure that contains the bin frequency and magnitudes of analog input channels A and B. The output display shows the magnitudes and bin frequencies of the first and last bins in the array data and the frequency and magnitude of the highest spectrum bin found in the channel A data.

(start of Figure 2-20)

```
Sub Batch_RBinary_To_Array_FFT_StereoSpectrumSwp()
  ' Spectrum Sweeps with the FFT DSP Program and places responses into a numeric array.
  ' Acquire the signal on analog analyzer channels A and B.
  ' Define dynamic two dimensional single float array to hold source frequencies, chan 1
      values, and chan 2 values
  ReDim fSrcCh1Ch2(0 To 3, 0 To 1) As Single  ' Size of second dimension will be changed
      to fit available data
  Dim bSuccess As Boolean, lPoints As Long, lIndex As Long, lMaxIndex As Long, fMaxPoint
      As Single
  Dim Src As Long, Ch1 As Long, Ch2 As Long
  Src = 0: Ch1 = 1: Ch2 = 2
  TEST_TITLE = "Batch_RBinary_To_Array_FFT_StereoSpectrumSwp"
  If IsAnalogPlusDSP = False Then
    MsgBox "System Two DSP option not installed. Test Aborted.", vbOKOnly, MSGBOX_TITLE &
      TEST_TITLE
    Exit Sub
  End If
  sIO = ""
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;:AGEN:AMPL
      A,1V;AMPL B,1V;WFM SINE,SINE;OUTPUT AB;:ANLR:INPUT A,XLR;INPUT B,XLR;AUTORANGE
      A,ON;AUTORANGE B,ON"
  S2GSend ":DSP:PROGRAM FFT;:DSP:FFT:INPUT AD1X;XLENGTH 2048;ACQLENGTH XLENGTH;AVGS
      1;AVGTYPE 1;COUPLING SUBH;DELAY 0;MODE INTRPOLATE;SRC1 A;SRC2 B;START 0;TRIGGER
      FREE;TSLOPE POS;WINDOW EQR;SENS 0.001FFS;PKTRIG 1;:DELAY 0.2"
  ' Wait for instrument outputs and inputs to settle before acquiring the spectrum.
  ' Delay 2
  ' Setup the DSP source parameters for a frequency sweep of the FFT bins and then
      acquire the signal on channel 1 (A) and process for a frequency domain spectrum.
  ' Setup the BATCH sweep parameters to sweep from 10 Hz to 22.5 KHz with 999 steps (1000
      bins) with LOG spacing.
  S2GSend ":DSP:OPSTATE SETUP;TIMEOUT 2;SRCP FREQ,HZ,10,22500,999,LOG;OPSTATE
      READ,AMP1,AMP2;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send batch mode amplitude measurement queries for channel 1 and channel 2 spectrum
      bins and read the measurement query responses.
  bSuccess = ReadBatch2("RBINARY", "AMP1", "DBV", "AMP2", "DBV", fSrcCh1Ch2())
  ' Set breakpoint on next line to enter VB debug mode in order to "watch" values in
      array fSrcCh1Ch2()
  If bSuccess Then
    lPoints = UBound(fSrcCh1Ch2, 2)   ' get last index
    ' Find array index of maximum value in Ch1 array data
    fMaxPoint = -999
    For lIndex = 0 To lPoints Step 1
      If fSrcCh1Ch2(Ch1, lIndex) > fMaxPoint Then
        fMaxPoint = fSrcCh1Ch2(Ch1, lIndex)
        lMaxIndex = lIndex
      End If
    Next lIndex
    ' Display data for first frequency bin, for largest signal found in Ch1 spectrum, and
      for last frequency bin.
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & "FFT - Spectrum acquisition complete (with peak-picking):" &
      vbCrLf & "Acquired 1000 LOG frequency spaced measurements of Bin Freq (SRC), Ch1, &
      Ch2." & vbCrLf
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & "First element
      array values are: " & vbCrLf & "fSrcCh1Ch2(SRC, 0) = " & fSrcCh1Ch2(Src, 0)
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf &
      "fSrcCh1Ch2(CH1, 0) = " & fSrcCh1Ch2(Ch1, 0)
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf &
      "fSrcCh1Ch2(CH2, 0) = " & fSrcCh1Ch2(Ch2, 0)
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & "Highest
      Signal Level found on Channel 1:" & vbCrLf & "fSrcCh1Ch2(SRC, " & lMaxIndex & ") =
      " & fSrcCh1Ch2(Src, lMaxIndex)
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf &
      "fSrcCh1Ch2(CH1, " & lMaxIndex & ") = " & fSrcCh1Ch2(Ch1, lMaxIndex)
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf &
      "fSrcCh1Ch2(CH2, " & lMaxIndex & ") = " & fSrcCh1Ch2(Ch2, lMaxIndex)
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & "Last element
      array values are: " & vbCrLf & "fSrcCh1Ch2(SRC, " & lPoints & ") = " &
      fSrcCh1Ch2(0, lPoints)
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf &
      "fSrcCh1Ch2(CH1, " & lPoints & ") = " & fSrcCh1Ch2(Ch1, lPoints)
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf &
      "fSrcCh1Ch2(CH2, " & lPoints & ") = " & fSrcCh1Ch2(Ch2, lPoints) & vbCrLf
  Else
    ' Display error message
    frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & "FFT - Spectrum acquisition failed." & vbCrLf
  End If
  ' Set display to last line
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-20. Visual Basic subprogram Batch_RBinary_To_Array_FFT_StereoSpectrumSwp code

```
Batch_RBinary_To_Array_FFT_StereoSpectrumSwp

FFT - Spectrum acquisition complete (with peak-picking):
Acquired 1000 LOG frequency spaced measurements of Bin Freq (SRC), Ch1, & Ch2.

First element array values are:
fSrcCh1Ch2(SRC, 0) = 10.00214
fSrcCh1Ch2(CH1, 0) = -113.4391
fSrcCh1Ch2(CH2, 0) = -136.5694
Highest Signal Level found on Channel 1:
fSrcCh1Ch2(SRC, 596) = 999.773
fSrcCh1Ch2(CH1, 596) = -0.2413335
fSrcCh1Ch2(CH2, 596) = -0.2377162
Last element array values are:
fSrcCh1Ch2(SRC, 999) = 22500
fSrcCh1Ch2(CH1, 999) = -122.3953
fSrcCh1Ch2(CH2, 999) = -117.4754
```

Figure 2-21. Visual Basic subprogram Batch_RBinary_To_Array_FFT_StereoSpectrumSwp measurement results.

Waveform Acquisition Sweeps with the FFT DSP Program

DSP based time waveforms may be acquired with the FFT DSP program. The FFT DSP program acquires and measure signals in a “batch” process. The results of an acquisition may be read as a series of measurements. In the case of a waveform in the time domain, each sample is read by sending an amplitude query at each specified time point in the waveform record for the specified channel. The process requires an initial setup, followed by the acquisition of the signal, and then followed by a series of queries and measurement responses.

The examples below set up the instrument for a digital input and output, and load the FFT program into the digital signal process. The FFT program parameters are set up for a waveform acquisition and the signal is acquired. After successful acquisition the amplitude of the samples are read at the specified point within the acquisition record. The time period of interest begins at 1 ms after the trigger point and ends 2.1 ms later in order to analyze the second cycle of the 1 kHz fundamental signal from the System Two digital generator. For this example, a coarse measurement time interval was selected for simplicity. The two subprograms show the two methods for reading data from the FFT DSP program after the acquisition is complete, using the :DSP:FFT:AMPL? query method and the :DSP:BATCh? query method with ASCII formatted response data. This code is provided in the “S2Gsample2.bas” file on the CD ROM included with this manual.

(Start of Figure 2-22)

```
Sub FFT_WaveformSwp()
  ' Waveform Acquisition Sweeps with the FFT DSP Program
  TEST_TITLE = "FFT_WaveformSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,0DBFS;WFM SINE,SINE;:DIN:FORMAT XLR;SCALEFREQBY
      MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID 0;JWFM
      NONE;NOUTPUT OFF;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE FIXED"
  S2GSend ":DSP:PROGRAM FFT;:DSP:FFT:INPUT DIGITAL;ACQLENGTH L24K;AVGS 1;AVGTYPE
      1;COUPLING DC;DELAY 0;MODE RAW;SRC1 A;SRC2 B;INPUT DIGITAL;START 0.001;TRIGGER
      DGEN;TSLOPE POS;WINDOW EQR;XLENGTH 1024;SENS 0.001FFS;PKTRIG 1;:DELAY 0.2"
  ' Setup the DSP source parameters for a time domain sweep of the waveform sample points
      and then acquire the signal on channel 1 (A.
  S2GSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,0,0,0,LIN;OPST READ,AMP1;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send amplitude measurement queries for channel 1 waveform samples from 1 mS to 2.1 mS
      in 0.1 mS intervals.
  S2GSend ":DSP:FFT:AMPL? 1,FFS,0.001;AMPL? 1,FFS,0.0011;AMPL? 1,FFS,0.0012;AMPL?
      1,FFS,0.0013;AMPL? 1,FFS,0.0014;AMPL? 1,FFS,0.0015;AMPL? 1,FFS,0.0016;AMPL?
      1,FFS,0.0017;AMPL? 1,FFS,0.0018;AMPL? 1,FFS,0.0019;AMPL? 1,FFS,0.002;AMPL?
      1,FFS,0.0021"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub

Sub Batch_FFT_WaveformSwp()
  ' Waveform Acquisition Sweeps with the FFT DSP Program
  TEST_TITLE = "Batch_FFT_WaveformSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,0DBFS;WFM SINE,SINE;:DIN:FORMAT XLR;SCALEFREQBY
      MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID 0;JWFM
      NONE;NOUTPUT OFF;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE FIXED"
  S2GSend ":DSP:PROGRAM FFT;:DSP:FFT:INPUT DIGITAL;ACQLENGTH L24K;AVGS 1;AVGTYPE
      1;COUPLING DC;DELAY 0;MODE RAW;SRC1 A;SRC2 B;INPUT DIGITAL;START 0.001;TRIGGER
      DGEN;TSLOPE POS;WINDOW EQR;XLENGTH 1024;SENS 0.001FFS;PKTRIG 1;:DELAY 0.2"
  ' Setup the DSP source parameters for a time domain sweep of the waveform sample points
      and then acquire the signal on channel 1 (A.
  ' Sweep start time of 1.0 mS, sweep stop time of 2.1 mS, 0.1 mS intervals (11 steps, 12
      points), with linear spacing.
  S2GSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,0.001,.0021,11,LIN;OPST
      READ,AMP1;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send BATCH? query for channel 1 waveform samples in FFS units.
  S2GSend ":DSP:BATCh? ASCII,AMP1,FFS"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-22. Visual Basic code for subprograms FFT_WaveformSwp and Batch_FFT_WaveformSwp.

```
FFT_WaveformSwp

-0.591869;0.388695;1.05931;1.39704;1.20248;0.613005;-0.366183;-1.0437;-1.39323;-1.21462;-
0.63398;0.34356

Batch_FFT_WaveformSwp

ASCII,12,0.00100004,-
0.545619,0.00109995,0.437011,0.00120004,1.0921,0.00129996,1.404,0.00140005,1.17513,0.0014
9996,0.567096,0.00160006,-0.414736,0.00169997,-1.07711,0.00180006,-1.40101,0.00189998,-
1.18798,0.00200007,-0.588427,0.00209999,0.392337
```

Figure 2-23. Visual Basic subprograms FFT_WaveformSwp and Batch_FFT_WaveformSwp measurement results.

Digital Interface Eye Pattern Sweeps with the Intervu DSP Program

The Intervu DSP program may be used to analyze the digital waveform on the AES/EBU input connector for jitter and amplitude distortion. This may be measured with the eye pattern measurement capability. The Intervu DSP program acquires and measure the digital interface signal in a "batch" process. The results of an acquisition may be read as a series of measurements. In the case of an eye pattern waveform in the time domain, each sample is read by sending a query for amplitude at each specified time point in the upper eye waveform and in the lower eye waveform. The process requires an initial setup, followed by the acquisition of the signal, and then followed by a succession of queries and measurement responses.

The examples below setup the instrument for a digital input and output, and load the Intervu program into the digital signal processor. The Intervu DSP program parameters are setup for an eye pattern waveform acquisition and the signal is acquired. After successful acquisition the amplitude of the eye pattern samples are read at the specified time points for the upper and lower eye patterns. The time period of interest begins at 0.0 ns and ends 82 ns later in order to analyze one pattern. For this example a linear time interval of 15 ns per point was selected. This code is provided in the "S2Gsample2.bas" file on the CD ROM included with this manual. The first subprogram "INTERVU_EyePatternSwp" reads the upper and lower eye data using a series of :DSP:INTERVU:LOWER? and UPPER? queries imbedded in a macro. The second subprogram "Batch_INTERVU_EyePatternSwp" reads the upper and lower eye data using a :DSP:BATCh? query, resulting in a faster response if many data points are required.

(start of Figure 2-24)

```
Sub INTERVU_EyePatternSwp()
  ' Digital Interface Eye Pattern Sweeps with the Intervu DSP Program
  TEST_TITLE = "INTERVU_EyePatternSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;WFM SINE,SINE;:DIN:FORMAT XLR;SCALEFREQBY
      MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID 0;JWFM
      NONE;NOUTPUT OFF;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE FIXED"
  S2GSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE EYE;MON
      JITTER;TRIGGER ARCV;WINDOW BH;:DELAY 0.5"
  ' Define a measurement macro with lower and upper eye pattern measurement queries.
  S2GSend "*PMC;*EMC 1;*DMC ""EYE"",#232:DSP:INT:LOWER? V,$1;UPPER? V,$1"
  ' Setup the DSP source parameters for a time domain sweep of the upper and lower eye
      pattern waveform sample points and then acquire the signal.
  S2GSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,0,0,0,LIN;OPST
      READ,LOWER,UPPER;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send measurement queries for each time point in the eye pattern using the EYE macro
      (defined above).
  S2GSend "EYE 0;EYE 5E-9;EYE 10E-9;EYE 15E-9;EYE 20E-9;EYE 25E-9;EYE 30E-9;EYE 35E-9;EYE
      40E-9;EYE 45E-9;EYE 50E-9;EYE 55E-9;EYE 60E-9;EYE 65E-9;EYE 70E-9;EYE 75E-9;EYE
      80E-9;EYE 85E-9;EYE 90E-9;EYE 95E-9;EYE 100E-9;EYE 105E-9;EYE 110E-9;EYE 115E-9;EYE
      120E-9;EYE 125E-9;EYE 130E-9;EYE 135E-9;EYE 140E-9; EYE 145E-9;EYE 150E-9;EYE 155E-
      9;EYE 160E-9;EYE 165E-9;EYE 170E-9;EYE 175E-9;EYE 180E-9;EYE 185E-9;EYE 190E-9;EYE
      195E-9;EYE 200E-9"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub

Sub Batch_INTERVU_EyePatternSwp()
  ' Digital Interface Eye Pattern Sweeps with the Intervu DSP Program
  TEST_TITLE = "Batch_INTERVU_EyePatternSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;WFM SINE,SINE;:DIN:FORMAT XLR;SCALEFREQBY
      MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID 0;JWFM
      NONE;NOUTPUT OFF;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE FIXED"
  S2GSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE EYE;MON
      JITTER;TRIGGER ARCV;WINDOW BH;:DELAY 0.5"
  ' Setup the DSP source parameters for a time domain sweep of the upper and lower eye
      pattern waveform sample points and then acquire the signal.
  ' Sweep start time of 0.0 Sec, sweep stop time of 200 nS, intervals of 5 nS (40 steps,
      41 points), with linear spacing.
  S2GSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,0,200E-9,40,LIN;OPST
      READ,LOWER,UPPER;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send BATCH? query for upper and lower eye pattern waveform samples in Volt units.
  S2GSend ":DSP:BATCh? ASCII,LOWER,V,UPPER,V"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-24. Visual Basic subprograms INTERVU_EyePatternSwp and Batch_INTERVU_EyePatternSwp code

```
INTERVU_EyePatternSwp

0;0.0472487;-0.425246;0.330739;-1.22848;1.18122;-1.88997;1.79545;-2.26797;2.22068;-
2.45695;2.36244;-2.45695;2.36244;-2.45695;2.36244;-2.40972;2.36244;-2.45695;2.36244;-
2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-
2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-
2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-
2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-2.40972;2.36244;-2.40972;2.31518;-
2.40972;2.31518;-2.3152;2.22068;-1.98446;1.88994;-1.37023;1.22846;-
0.472492;0.377989;0;0.0472487;-0.425246;0.283493;-0.425246;0.283493;-1.18123;1.13397;-
1.84271;1.74819;-2.22071;2.17345

Batch_INTERVU_EyePatternSwp

ASCII,41,0,0,0.0472487,5.12228E-09,-0.472492,0.425239,9.77889E-09,-
1.32297,1.22846,1.49012E-08,-1.88997,1.84268,2.00234E-08,-2.3152,2.22068,2.51457E-08,-
2.3152,2.22068,2.98023E-08,-2.45695,2.36244,3.49246E-08,-2.45695,2.36244,4.00469E-08,-
2.45695,2.36244,4.51692E-08,-2.45695,2.36244,4.98258E-08,-2.40972,2.36244,5.4948E-08,-
2.40972,2.36244,6.00703E-08,-2.40972,2.31518,6.51926E-08,-2.40972,2.36244,6.98492E-08,-
2.40972,2.36244,7.49715E-08,-2.40972,2.36244,8.00938E-08,-2.40972,2.36244,8.5216E-08,-
2.40972,2.36244,8.98726E-08,-2.40972,2.36244,9.49949E-08,-2.40972,2.36244,1.00117E-07,-
2.40972,2.31518,1.04774E-07,-2.40972,2.36244,1.09896E-07,-2.40972,2.36244,1.15018E-07,-
2.40972,2.36244,1.20141E-07,-2.40972,2.36244,1.24797E-07,-2.40972,2.36244,1.2992E-07,-
2.40972,2.36244,1.35042E-07,-2.40972,2.31518,1.40164E-07,-2.40972,2.36244,1.44821E-07,-
2.40972,2.36244,1.49943E-07,-2.40972,2.31518,1.55065E-07,-2.26797,2.22068,1.60188E-07,-
1.93723,1.84268,1.64844E-07,-1.27572,1.18122,1.69966E-07,-0.377993,0.283493,1.75089E-
07,0,0.0472487,1.80211E-07,0,0.0472487,1.84868E-07,-0.472492,0.377989,1.8999E-07,-
1.22848,1.18122,1.95112E-07,-1.88997,1.74819,1.99769E-07,-2.22071,2.17345
```

Figure 2-25. Visual Basic measurement results for subprograms INTERVU_EyePatternSwp and Batch_INTERVU_EyePatternSwp

Digital Interface Jitter Probability Sweeps with the Intervu DSP Program

The Intervu DSP program may be used to analyze the digital waveform on the AES/EBU input connector for jitter probability. The numbers represent the probability of jitter as a function of jitter magnitude in seconds. The Intervu DSP program acquires and measures the digital interface signal in a "batch" process. The results of an acquisition may be read as a series of measurements. In order to measure jitter probability, a digital frame is processed for jitter and the resultant jitter magnitude data is processed to create probability values for each jitter magnitude of interest. The process requires an initial setup, followed by the acquisition of the signal, and then followed by a succession of jitter probability queries for specific jitter magnitudes.

The examples below set up the instrument for AES/EBU digital input and output, generate a 1-Vpp interference noise at the digital output, and load the Intervu program into the digital signal processor. The Intervu DSP program parameters are set up for probability processing and digital interface waveform acquisition. After successful acquisition the jitter probability values are queried at the jitter magnitudes specified in positive and negative time values (relative to the clock transitions through zero). The time period of interest begins at -5.0 ns and ends +5.0 ns. Measurements are queried at 0.5 ns intervals in this example. This code is provided in the "S2Gsample2.bas" file on the CD ROM included with this manual. The first subprogram "INTERVU_JitterProbabilitySwp" reads the probability data using a series of :DSP:INTERVU:PROBABILITY? queries. The second subprogram "Batch_INTERVU_JitterProbabilitySwp" reads the probability data using a :DSP:BATCh? query, resulting in a faster response if many data points are required.

(start of Figure 2-26)

```
Sub INTERVU_JitterProbabilitySwp()
  'Digital Interface Jitter Probability Sweeps with the Intervu DSP Program
  TEST_TITLE = "INTERVU_JitterProbabilitySwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;WFM SINE,SINE;:DIN:FORMAT XLR;SCALEFREQBY
      MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID 0;JWFM
      NONE;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE FIXED;NAMPL 1.0;NOUTPUT
      ON"
  S2GSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE INTRPOLATE;MON
      JITTER;TRIGGER ARCV;WINDOW BH;:DELAY 0.5"
  ' Setup the DSP source parameters for a probability sweep of the jitter magnitude and
      acquire the signal.
  S2GSend ":DSP:OPSTATE SET;TIMEOUT 2;SRCP JITTER,SEC,0,0,0,LIN;OPSTATE
      READ,PROBABILITY;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send probability measurement queries for each jitter magnitude.
  S2GSend "DSP:INTERVU:PROB? PCT,-5E-9;PROB? PCT,-4.5E-9;PROB? PCT,-4E-9;PROB? PCT,-3.5E-
      9;PROB? PCT,-3E-9;PROB? PCT,-2.5E-9;PROB? PCT,-2E-9;PROB? PCT,-1.5E-9;PROB? PCT,-
      1E-9;PROB? PCT,-0.5E-9;PROB? PCT,0;PROB? PCT,0.5E-9;PROB? PCT,1E-9;PROB? PCT,1.5E-
      9;PROB? PCT,2E-9;PROB? PCT,2.5E-9;PROB? PCT,3E-9;PROB? PCT,3.5E-9;PROB? PCT,4E-
      9;PROB? PCT,4.5E-9;PROB? PCT,5E-9"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub

Sub Batch_INTERVU_JitterProbabilitySwp()
  'Digital Interface Jitter Probability Sweeps with the Intervu DSP Program
  TEST_TITLE = "Batch_INTERVU_JitterProbabilitySwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;WFM SINE,SINE;:DIN:FORMAT XLR;SCALEFREQBY
      MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID 0;JWFM
      NONE;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE FIXED;NAMPL 1.0;NOUTPUT
      ON"
  S2GSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE INTRPOLATE;MON
      JITTER;TRIGGER ARCV;WINDOW BH;:DELAY 0.5"
  ' Setup the DSP source parameters for a probability sweep of the jitter magnitude and
      acquire the signal.
  ' Sweep start time of -5.0 nS, sweep stop time of 5 nS, intervals of 0.5 nS (20 steps,
      21 points), with linear spacing.
  S2GSend ":DSP:OPSTATE SET;TIMEOUT 2;SRCP JITTER,SEC,-5E-9,5E-9,20,LIN;OPSTATE
      READ,PROBABILITY;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send BATCH? query for jitter probability in percent units.
  S2GSend ":DSP:BATCh? ASCII,PROBABILITY,PCT"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-26. Visual Basic subprogram INTERVU_JitterProbabilitySwp code.

```
INTERVU_JitterProbabilitySwp

0;0.0362396;0.561714;4.96483;11.7416;11.4698;7.99084;3.50618;2.91729;2.3284;2.73609;2.518
65;2.79045;4.13132;4.99201;10.283;12.3668;11.2705;2.96259;0.280857;0.0181198

Batch_INTERVU_JitterProbabilitySwp

ASCII,21,-4.99934E-09,0,-4.50015E-09,0.0181198,-4.00096E-09,0.371456,-3.50177E-
09,2.62737,-3.00258E-09,9.73034,-2.5034E-09,16.036,-1.99676E-09,9.07803,-1.49757E-
09,3.68738,-9.98378E-10,3.05319,-4.99189E-10,2.56395,0,2.93541,4.99189E-
10,2.69985,9.98378E-10,2.65455,1.49757E-09,3.37934,1.99676E-09,4.41217,2.5034E-
09,7.42912,3.00258E-09,13.4993,3.50177E-09,10.9262,4.00096E-09,4.14944,4.50015E-
09,0.607014,4.99934E-09,0.0181198
```

Figure 2-27. Visual Basic measurement results for subprograms INTERVU_JitterProbabilitySwp and Batch_INTERVU_JitterProbabilitySwp

Digital Interface Jitter Waveform Sweeps with the Intervu DSP Program

The Intervu DSP program may be used to acquire the recovered jitter signal in the form of a waveform. INTERVU operates by first acquiring 256k samples (2 ^ 18 –1 = 262143) of the digital interface signal at a sample rate of 67.108864 MHz. Thus, the total time period of the acquisition is 0.00390625 seconds. The waveform sample points are specified in units of time relative to the beginning of the sample interval. The acquisition of jitter magnitude versus time may be triggered in many ways. The example subprograms below trigger on the source of jitter, the jitter generator of System Two. This code is provided in the "S2GSample2.bas" file on the CD ROM included with this manual. The first subprogram "INTERVU_JitterWaveformSwp" reads the jitter waveform data using a series of :DSP:INTERVU:JITTER? queries. The second subprogram "Batch_INTERVU_JitterWaveformSwp" reads the jitter waveform data using a :DSP:BATCh? query, resulting in a faster response if many data points are required.

Connect the System Two AES/EBU digital output to the digital input in order to acquire results similar to those shown below. The System Two digital generator is set up to produce a JTEST digital audio waveform and the digital output jitter generator is set up to produce a sinewave jitter waveform with a jitter frequency of 1 kHz and jitter magnitude of 0.3 UI (48.83 nS at 44.1 kHz sample rate). The measured jitter magnitude in UI (Unit Intervals) as a function of time relative to the trigger point is shown below the subprogram code. The measurements are acquired beginning at 1 ms and ending at 2.3 ms in increments of 50 us. The measurement interval may be smaller in order to measure higher frequencies of jitter.

(Start of Figure 2-28)

```
Sub INTERVU_JitterWaveformSwp()
  ' Digital Interface Jitter Waveform Sweeps with the Intervu DSP Program
  TEST_TITLE = "INTERVU_JitterWaveformSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  'Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,0DBFS;WFM SPCIAL,JTEST;:DIN:FORMAT XLR;SCALEFREQBY
      MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID 0;JWFM
      SINE;JAMPL 0.3UI;JFREQ 1000;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE
      FIXED;NAMPL 0.0;NOUTPUT OFF"
  S2GSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE RAW;MON
      JITTER;TRIGGER JITTER;WINDOW BH;:DELAY 0.5"
  ' Setup the DSP source parameters for a time domain sweep of the jitter waveform sample
      points and then acquire the signal.
  S2GSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,1E-3,2.3E-3,26,LIN;OPST
      READ,JITTER;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send jitter measurement queries for each time point in the jitter waveform.
  S2GSend ":DSP:INTERVU:JITT? UI,1E-3;JITT? UI,1.05E-3;JITT? UI,1.1E-3;JITT? UI,1.15E-
      3;JITT? UI,1.2E-3;JITT? UI,1.25E-3;JITT? UI,1.3E-3;JITT? UI,1.35E-3;JITT? UI,1.4E-
      3;JITT? UI,1.45E-3;JITT? UI,1.5E-3;JITT? UI,1.55E-3;JITT? UI,1.6E-3;JITT? UI,1.65E-
      3;JITT? UI,1.7E-3;JITT? UI,1.75E-3;JITT? UI,1.8E-3;JITT? UI,1.85E-3;JITT? UI,1.9E-
      3;JITT? UI,1.955E-3;JITT? UI,2E-3;JITT? UI,2.05E-3;JITT? UI,2.1E-3;JITT? UI,2.15E-
      3;JITT? UI,2.2E-3;JITT? UI,2.25E-3;JITT? UI,2.3E-3"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub

Sub Batch_INTERVU_JitterWaveformSwp()
  ' Digital Interface Jitter Waveform Sweeps with the Intervu DSP Program
  TEST_TITLE = "Batch_INTERVU_JitterWaveformSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  'Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,0DBFS;WFM SPCIAL,JTEST;:DIN:FORMAT XLR;SCALEFREQBY
      MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID 0;JWFM
      SINE;JAMPL 0.3UI;JFREQ 1000;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE
      FIXED;NAMPL 0.0;NOUTPUT OFF"
  S2GSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE RAW;MON
      JITTER;TRIGGER JITTER;WINDOW BH;:DELAY 0.5"
  ' Setup the DSP source parameters for a time domain sweep of the jitter waveform sample
      points and then acquire the signal.
  ' Sweep start time of 1 mS, sweep stop time of 2.3 mS, intervals of 50 uS (26 steps, 27
      points), with linear spacing.
  S2GSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,1E-3,2.3E-3,26,LIN;OPST
      READ,JITTER;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send BATCH? query for jitter waveform in UI units.
  S2GSend ":DSP:BATCh? ASCII,JITTER,UI"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-28. Visual Basic subprogram INTERVU_JitterWaveformSwp code

```
INTERVU_JitterWaveformSwp

-0.255192;-0.297146;-0.316755;-0.307113;-0.261011;-0.189934;-0.103086;-
      0.00473989;0.0929508;0.177169;0.2535;0.294348;0.318447;0.300149;0.260826;0.18996;0.
      108578;0.00879696;-0.0883547;-0.188351;-0.249467;-0.296676;-0.322859;-0.316182;-
      0.275334;-0.210832;-0.126613

Batch_INTERVU_JitterWaveformSwp

ASCII,27,0.001,-0.21793,0.00105,-0.265141,0.0011,-0.291321,0.00115,-0.283331,0.0012,-
      0.249057,0.00125,-0.191123,0.0013,-0.112161,0.00135,-
      0.0249778,0.0014,0.0736986,0.00145,0.147402,0.0015,0.225048,0.00155,0.269181,0.0016
      ,0.292287,0.00165,0.281875,0.0017,0.249123,0.00175,0.185165,0.0018,0.116926,0.00185
      ,0.0197746,0.0019,-0.0655498,0.00195,-0.152189,0.002,-0.225684,0.00205,-
      0.272893,0.0021,-0.293819,0.00215,-0.286806,0.0022,-0.256472,0.00225,-
      0.197227,0.0023,-0.122208
```

Figure 2-29. Visual Basic subprograms INTERVU_JitterWaveformSwp and Batch_INTERVU_JitterWaveformSwp measurement results

Digital Interface Waveform Sweeps with the Intervu DSP Program

The Intervu DSP program may be used to acquire the squarewave signal in the form of a waveform. INTERVU operates by first acquiring 256k samples (2 ^ 18 –1 = 262143) of the digital interface signal at a sample rate of 67.108864 MHz (14.9 nS per sample point). Thus the total time period of the acquisition is 0.00390625 seconds. The waveform sample points are specified in units of time relative to the beginning of the sample interval. This Visual Basic code is provided in the “S2Gsample2.bas” file on the CD ROM included with this manual. The first subprogram “INTERVU_WaveformSwp” reads the digital interface waveform data using a series of :DSP:INTERVU:AMPL? queries. The second subprogram “Batch_INTERVU_WaveformSwp” reads the digital interface waveform data using a :DSP:BATCh? query, resulting in a faster response if many data points are required.

Connect the System Two AES/EBU digital output to the digital input in order to acquire results similar to those shown below. The System Two digital generator is set up to produce a 1-kHz sinewave audio waveform and all digital output impairments are disabled. The pulse amplitude samples are measured in volts as a function of time relative to the trigger point. The measurements are acquired beginning at 0.0 ns and ending at 150 ns in increments of 15 ns.

(Start of Figure 2-30)

```
Sub INTERVU_WaveformSwp()
  ' Digital Interface Waveform Sweeps with the Intervu DSP Program
  TEST_TITLE = "INTERVU_WaveformSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,0DBFS;FRQ1 997 HZ;WFM SINE,SINE;:DIN:FORMAT
      XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID
      0;JWFM NONE;JAMPL 0.0UI;JFREQ 1000;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION
      24;RFENABLE FIXED;NAMPL 0.0;NOUTPUT OFF"
  S2GSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE INTRPOLATE;MON
      AUDIO;TRIGGER AXMT;WINDOW BH;:DELAY 0.1"
  ' Setup the DSP source parameters for a time domain sweep of the AES/EBU waveform
      sample points and then acquire the signal.
  S2GSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,0,0,0,LIN;OPST READ,AMPL;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send Intervu amplitude measurement queries for each time point in the AES/EBU
      waveform.
  S2GSend ":DSP:INTERVU:AMPL? V,1E-6;AMPL? V,1.015E-6;AMPL? V,1.03E-6;AMPL? V,1.045E-
      6;AMPL? V,1.06E-6;AMPL? V,1.075E-6;AMPL? V,1.09E-6;AMPL? V,1.105E-6;AMPL? V,1.12E-
      6;AMPL? V,1.135E-6;AMPL? V,1.15E-6;AMPL? V,1.165E-6;AMPL? V,1.18E-6;AMPL? V,1.195E-
      6;AMPL? V,1.21E-6;AMPL? V,1.225E-6;AMPL? V,1.240E-6;AMPL? V,1.255E-6;AMPL? V,1.27E-
      6;AMPL? V,1.285E-6;AMPL? V,1.3E-6;AMPL? V,1.315E-6;AMPL? V,1.33E-6;AMPL? V,1.345E-
      6;AMPL? V,1.360E-6;AMPL? V,1.375E-6;AMPL? V,1.39E-6;AMPL? V,1.405E-6;AMPL? V,1.42E-
      6;AMPL? V,1.435E-6;AMPL? V,1.45E-6;AMPL? V,1.465E-6;AMPL? V,1.48E-6;AMPL? V,1.495E-
      6;AMPL? V,1.51E-6"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub

Sub Batch_INTERVU_WaveformSwp()
  ' Digital Interface Waveform Sweeps with the Intervu DSP Program
  TEST_TITLE = "Batch_INTERVU_WaveformSwp"
  If IsDigital = False Then
    MsgBox "System Two Digital option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the initial setup commands. Note that a small delay is necessary to permit the
      DSP program to load before the start of an acquisition.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:MODE STEREO;STEREO
      DSP;:DGEN:OUTPUT AB;AMPL AB,0DBFS;FRQ1 997 HZ;WFM SINE,SINE;:DIN:FORMAT
      XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INVALID
      0;JWFM NONE;JAMPL 0.0UI;JFREQ 1000;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION
      24;RFENABLE FIXED;NAMPL 0.0;NOUTPUT OFF"
  S2GSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE INTRPOLATE;MON
      AUDIO;TRIGGER AXMT;WINDOW BH;:DELAY 0.1"
  ' Setup the DSP source parameters for a time domain sweep of the AES/EBU waveform
      sample points and then acquire the signal.
  ' Sweep start time of 1 uS, sweep stop time of 1.51 uS, intervals of 15 nS (34 steps,
      35 points), with linear spacing.
  S2GSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,1E-6,1.51E-6,34,LIN;OPST
      READ,AMPL;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the response to the ACQX? query.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send BATCH? query for jitter waveform in volts units.
  S2GSend ":DSP:BATCh? ASCII,AMPL,V"
  ' Read the measurement query responses.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors       ' Check for System Two errors.
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-30. Visual Basic subprogram INTERVU_WaveformSwp and Batch_INTERVU_WaveformSwp code

```
INTERVU_WaveformSwp

-2.42152;-2.46423;-2.50138;-2.45582;-2.45104;-2.41222;-2.44505;-2.40694;-2.46411;-
2.38248;-2.42329;-
1.8001;0.369482;2.11985;2.6353;2.54564;2.54745;2.54015;2.55387;2.48715;2.51976;2.48579;2.
50742;1.61303;-0.855053;-2.3178;-2.52976;-2.43408;-2.44566;-2.45984;-2.42064;-2.41071;-
2.4425;-2.36016;-2.37469

Batch_INTERVU_WaveformSwp

ASCII,35,9.99775E-07,-2.45695,1.01514E-06,-2.45855,1.03004E-06,-2.45263,1.04494E-06,-
2.40646,1.05985E-06,-2.41109,1.07521E-06,-2.3964,1.09011E-06,-2.41197,1.10501E-06,-
2.3941,1.11992E-06,-2.41614,1.13482E-06,-2.38738,1.15018E-06,-2.42296,1.16508E-06,-
1.79527,1.17999E-06,0.41628,1.19489E-06,2.11469,1.20979E-06,2.63753,1.22516E-
06,2.54478,1.24006E-06,2.54735,1.25496E-06,2.54124,1.26986E-06,2.55137,1.28523E-
06,2.50075,1.30013E-06,2.56264,1.31503E-06,2.47712,1.32993E-06,2.51233,1.34483E-
06,1.60884,1.3602E-06,-0.846195,1.3751E-06,-2.26056,1.39E-06,-2.49825,1.4049E-06,-
2.42852,1.4198E-06,-2.38981,1.43517E-06,-2.42258,1.45007E-06,-2.4313,1.46497E-06,-
2.36486,1.47987E-06,-2.36534,1.49477E-06,-2.36218,1.51014E-06,-2.36145
```

Figure 2-31. Visual Basic subprogram INTERVU_WaveformSwp and Batch_INTERVU_WaveformSwp measurement results

Multitone Signal Testing with FASTTEST DSP

The FASTTEST DSP program acquires and processes stereo multitone sinewave signals. The multitone signal must be loaded into the DSP digital generator as arbitrary waveforms in order for the FASTTEST DSP program to correctly process the input signal. Thus the digital generator waveform buffers and the FASTTEST DSP program must both be properly configured. Please see the APWIN User's Manual Chapter 13 for detailed operation concerning FASTTEST.

The FASTTEST DSP program provides superior performance in both speed and breadth of testing. The multitone testing technique uses a multitone sinewave signal generated by the digital signal processor to stimulate the device under test. The FASTTEST DSP program acquires this signal from the output of the device and analyzes it to provide measurements of stereo level, stereo distortion, stereo noise, crosstalk, and inter-channel phase or phase from input to output depending on signal routing. The process of generating the signal and analyzing it requires proper setup of both the DSP digital generator and the FASTTEST DSP analyzer program. The DSP digital generator must use an arbitrary waveform in order to create the multitone waveform.

Using FASTTEST DSP to Acquire and Process Multitone Signals

The FASTTEST DSP program must analyze the multitone arbitrary waveforms loaded into the DSP digital generator buffers before it can correctly acquire and process the multitone signal on its inputs. This analysis of the digital generator waveforms occurs automatically when the waveform is loaded into the DSP generator buffers with the :DGEN:ARBWFM command or when the FASTTEST DSP program is loaded if the digital generator arbitrary waveforms are already loaded.

The FASTTEST multitone analyzer can accept input either from the analog analyzer inputs via the analog to digital converters, or it can accept input directly from the digital inputs. In either case, part of the setup involves the inputs as well as the FASTTEST commands (:DSP:FASTTEST:).

FASTTEST will acquire the stereo waveform into its DSP acquisition memory and perform an FFT in order to create a spectrum of each of the two inputs. The resultant spectrum information is then processed in different ways depending on the type of measurement desired. The processing modes are selected by the MODE command to extract measurements for distortion, noise, response, spectrum, crosstalk, or psycho-acoustic masking curves (DISTORTION, NOISE, RESPONSE, SPECTRUM, XTALK, and MASKING). The process involves signal acquisition in one of the processing modes, usually RESPONSE, and then re-processing after changes in mode. Thus, the signal need only be acquired one time with the ACQX? command and then re-processed with the REPROCESS? command after changing processing modes with the MODE command.

In order to make measurements with FASTTEST, it is necessary to know the frequencies of the sinewave signals imbedded in the multitone signal. These frequencies will be used in measurement queries of FFT bins. The example below will use the ISO31.AGM arbitrary waveform file for the analog generator multitone waveform provided with the APWIN software. The frequencies of the sinewaves in this waveform are shown in Table 2-3 below. These frequencies will be used in measurement queries or to setup a DSP sweep table with the :DSP:TABLe and :DSP:TSELECT commands to extract measurements for response, distortion, noise, phase, and crosstalk when this waveform is used for the stimulus signal.

Table 2-3. Multitone Frequencies in Waveform File ISO31.AGM (48 kHz sample rate)

17.578125
23.4375
29.296875
41.015625
52.734375
64.453125
82.03125
99.609375
123.046875
158.203125
199.21875
251.953125
316.40625
398.4375
498.046875
632.8125
802.734375
1001.953125
1248.046875
1599.609375
1998.046875
2501.953125
3152.34375
4001.953125
4998.046875
6351.5625
7998.046875
10001.953125
12498.046875
16001.953125
19998.046875

The Visual Basic program examples below accomplish the following tasks:

1. Set up the FASTTEST DSP program for acquisition on the analog inputs with the A/D converters. Set processing mode to RESPONSE in order to make level measurements of the frequency bins specified in the macros above.
2. Load the arbitrary waveform ISO31.AGM into waveform register 1.
3. Load the arbitrary waveform from waveform register 1 into both DSP digital generator waveform buffers 1 and 2 (left and right).
4. Set up the internal sample rate for 48 kHz required for correct frequencies in arbitrary waveform IS031.AGM.
5. Set up the analog generator for a 1.0 Vrms (1.414 Vpk) multitone signal level, select Sine D/A Arbitrary waveform, and turn the output on.
6. Set up the analog analyzer for XLR inputs with auto-ranging enabled.
7. Set up the headphone monitor to hear the multitone signal on the analog analyzer inputs, in stereo mode.
8. In the subprogram FASTTEST_MultitoneSwp define a macro that queries for the measurements of the 31 frequency bins on channel 1. Define another macro that queries for the measurements of the 31 frequency bins on channel 2. Type of measurement will be set up separately.
9. In the subprograms Batch_FASTTEST_MultitoneSwp and Batch_FASTTEST_MultitoneSwp_RQS, define a table with the :DSP:TABLe and :DSP:TSELECT commands that specify the frequencies of the multitone signals to be measured using the :DSP:BATCh? query.
10. Set up for a sweep of the frequency bins from 17.57 Hz to 19998.04 Hz with 31 frequencies in the multitone signal. Acquire the multitone signal and process left and right channels for levels at 31 frequencies. Read the response to the ACQX? query and then read the 31 level measurements on both left and right channels. Read 31 phase measurements for channel 2.
11. Reprocess the multitone signal acquired above for distortion. Read the REPROCESS? query response and then read the 31 distortion measurements on both left and right channels.
12. Reprocess the multitone signal acquired above for noise. Read the REPROCESS? query response and then read the 31 noise measurements for both left and right channels.

This code is provided in the “S2Gsample2.bas” file on the CD ROM included with this manual. The first subprogram “FASTTEST_MultitoneSwp” reads the data using a series of :DSP:FASTTEST:AMPL? queries imbedded in macros for each channel and for channel 2 phase. The second subprogram “Batch_FASTTEST_MultitoneSwp” reads the data using a :DSP:BATCh? query, resulting in a faster response if many data points are required. The measurements shown below the subprogram code were acquired from the output of an analog stereo graphic equalizer. Both channels were set with all EQ controls set to mid-range with the low cut controls set at 150 Hz and the high cut controls set at 15 kHz.

The subprogram Batch_FASTTEST_MultitoneSwp_RQS sets up the System Two GPIB interface to assert the SRQ line when DSP acquisition is complete. The function WaitForSRQ waits for an SRQ interrupt event handler (GpibNotify1 control on form frmS2GSample2) to invoke a callback function (GpibNotify1_Notify in form frmS2GSample2) that serial polls the System Two. This happens when the System Two asserts the SRQ line on the GPIB interface when the DSP acquisition has been triggered by the multitone signal on the analog inputs. This technique may be used to wait for signal detection if other processes must be running while waiting for the signal to occur, such as in a broadcast link testing application. This code illustrates how to set up the AP Status Enable register to assert an SRQ when the DSP has completed an acquisition, and how to detect the SRQ in Visual Basic with the National Instruments GpibNotify callback function. This subprogram also shows how multiple DSP readings may be acquired with the :DSP:BATCh? query.

(Start of Figure 2-32)

```
Sub FASTTEST_MultitoneSwp()
  ' Multitone Signal Testing with FASTTEST DSP
  ' Revised 18-JUNE-1999
  TEST_TITLE = "FASTTEST_MultitoneSwp"
  If IsAnalogPlusDSP = False Then
    MsgBox "System Two Analog Plus DSP option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  Dim sLevel1 As String, sLevel2 As String, sDistortion1 As String, sDistortion2 As
      String
  Dim sNoise1 As String, sNoise2 As String, sPhase As String
  ' Initial Setup: set to defaults, enable OPC event reporting, turn off headers, set
      digital generator arbitrary waveform register size to 8192 samples per waveform.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:DGEN:ARBSIZE 8192"
  ' 1. FASTTEST DSP setup for signal acquisition and level sweep.
  S2GSend ":DSP:PROGRAM FASTTEST;FASTTEST:FREQRES 0;INPUT AD1X;SRC1 A;SRC2 B;LENGTH
      AUTO;MODE RESPONSE;PMODE ICHANNEL;PROCESS SYNC;TRGDELAY 0;TRIGGER NORMAL;:DELAY
      0.5"
  ' 2. Load the arbitrary waveform ISO31.AGM into waveform register 1.
  ArbLoad_ibwrtf 1, App.Path & "\ISO31.AGM"
  ' 3. Load the arbitrary waveform from waveform register 1 into both DSP digital
      generator waveform buffers 1 and 2 (left and right).
  ' 4. Setup the internal sample rate for 48 kHz required for correct frequencies in
      arbitrary waveform IS031.AGM.
  ' 5. Setup the analog generator for a 1.0 Vrms (1.414 Vpk) multitone signal level,
      select Sine D/A Arbitrary waveform, and turn the output on.
  ' 6. Setup the analog analyzer for XLR inputs with auto-ranging enabled. Delay 2
      seconds for autoranging, then disable autoranging to fix ranges.
  ' 7. Setup the headphone monitor to hear the multitone signal on the analog analyzer
      inputs, in stereo mode.
  S2GSend ":DGEN:ARBWFM 1,1;:DOUT:RATE 48000HZ;:AGEN:AMPL A,1V;AMPL B,1V;WFM
      DAARBITRARY,NONE;OUTPUT AB;:ANLR:INPUT A,XLR;INPUT B,XLR;AUTORANGE A,ON;AUTORANGE
      B,ON;:DELAY 2;:ANLR:AUTORANGE A,OFF;AUTORANGE B,OFF;:MON:STEREO INPUT;MODE STEREO"
  '8. Define three macros to sweep the ISO31 frequency bins for FASTTEST channels 1 and
      2.
  S2GSend "*PMC;*EMC 1;*DMC ""SWPCH1_ISO31"",#0:DSP:FASTTEST:AMPL? 1,DBV,17.57;AMPL?
      1,DBV,23.43;AMPL? 1,DBV,29.29;AMPL? 1,DBV,41.01;AMPL? 1,DBV,52.73;AMPL?
      1,DBV,64.45;AMPL? 1,DBV,82.03;AMPL? 1,DBV,99.6;AMPL? 1,DBV,123.04;AMPL?
      1,DBV,158.2;AMPL? 1,DBV,199.21;AMPL? 1,DBV,251.95;AMPL? 1,DBV,316.4;AMPL?
      1,DBV,398.43;AMPL? 1,DBV,498.04;AMPL? 1,DBV,632.81;AMPL? 1,DBV,802.73;AMPL?
      1,DBV,1001.95;AMPL? 1,DBV,1248.04;AMPL? 1,DBV,1599.6;AMPL? 1,DBV,1998.04;AMPL?
      1,DBV,2501.95;AMPL? 1,DBV,3152.34;AMPL? 1,DBV,4001.95;AMPL? 1,DBV,4998.04;AMPL?
      1,DBV,6351.56;AMPL? 1,DBV,7998.04;AMPL? 1,DBV,10001.95;AMPL? 1,DBV,12498.04;AMPL?
      1,DBV,16001.95;AMPL? 1,DBV,19998.04"
  S2GSend "*DMC ""SWPCH2_ISO31"",#0:DSP:FASTTEST:AMPL? 2,DBV,17.57;AMPL?
      2,DBV,23.43;AMPL? 2,DBV,29.29;AMPL? 2,DBV,41.01;AMPL? 2,DBV,52.73;AMPL?
      2,DBV,64.45;AMPL? 2,DBV,82.03;AMPL? 2,DBV,99.6;AMPL? 2,DBV,123.04;AMPL?
      2,DBV,158.2;AMPL? 2,DBV,199.21;AMPL? 2,DBV,251.95;AMPL? 2,DBV,316.4;AMPL?
      2,DBV,398.43;AMPL? 2,DBV,498.04;AMPL? 2,DBV,632.81;AMPL? 2,DBV,802.73;AMPL?
      2,DBV,1001.95;AMPL? 2,DBV,1248.04;AMPL? 2,DBV,1599.6;AMPL? 2,DBV,1998.04;AMPL?
      2,DBV,2501.95;AMPL? 2,DBV,3152.34;AMPL? 2,DBV,4001.95;AMPL? 2,DBV,4998.04;AMPL?
      2,DBV,6351.56;AMPL? 2,DBV,7998.04;AMPL? 2,DBV,10001.95;AMPL? 2,DBV,12498.04;AMPL?
      2,DBV,16001.95;AMPL? 2,DBV,19998.04"
  S2GSend "*DMC ""SWPPHA2ISO31"",#0:DSP:FASTTEST:PHASE? 2,DEG,17.57;PHASE?
      2,DEG,23.43;PHASE? 2,DEG,29.29;PHASE? 2,DEG,41.01;PHASE? 2,DEG,52.73;PHASE?
      2,DEG,64.45;PHASE? 2,DEG,82.03;PHASE? 2,DEG,99.6;PHASE? 2,DEG,123.04;PHASE?
      2,DEG,158.2;PHASE? 2,DEG,199.21;PHASE? 2,DEG,251.95;PHASE? 2,DEG,316.4;PHASE?
      2,DEG,398.43;PHASE? 2,DEG,498.04;PHASE? 2,DEG,632.81;PHASE? 2,DEG,802.73;PHASE?
      2,DEG,1001.95;PHASE? 2,DEG,1248.04;PHASE? 2,DEG,1599.6;PHASE? 2,DEG,1998.04;PHASE?
```

```
2,DEG,2501.95;PHASE? 2,DEG,3152.34;PHASE? 2,DEG,4001.95;PHASE? 2,DEG,4998.04;PHASE?
     2,DEG,6351.56;PHASE? 2,DEG,7998.04;PHASE? 2,DEG,10001.95;PHASE?
     2,DEG,12498.04;PHASE? 2,DEG,16001.95;PHASE? 2,DEG,19998.04"
 ' 9. Level and Phase Sweep.
 S2GSend ":DSP:OPSTATE SETUP;TIMEOUT 5;SRCP
     FREQ,HZ,17.578125,19998.046875,31,ARB;OPSTATE READ,AMP1,AMP2,PHA2;:DSP:ACQX?"
 ' Read ACQX? query response.
 sIO = S2GRcv
 If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
   UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
     time out then exit.
   Exit Sub
 End If
 S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
 ' Send FASTTEST channel 1 measurement queries.
 S2GSend "SWPCH1_ISO31"
 ' Read Multitone Level measurements.
 sLevel1 = S2GRcv
 ' Send FASTTEST channel 2 measurement queries.
 S2GSend "SWPCH2_ISO31"
 ' Read Multitone Level measurements.
 sLevel2 = S2GRcv
 ' Send FASTTEST channel 2 phase measurement queries.
 S2GSend "SWPPHA2ISO31"
 ' Read Multitone channel 2 phase measurements.
 sPhase = S2GRcv
 ' 10. Distortion Sweep, setup for distortion processing.
 S2GSend ":DSP:OPSTATE SETUP;FASTTEST:MODE DISTORTION;:DELAY 1.1;:DSP:OPSTATE
     READING,AMP1,AMP2;REPROCESS?"  ' original line
 ' Read REPROCESS? query response.
 sIO = S2GRcv
 If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
   UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
     time out then exit.
   Exit Sub
 End If
 ' Send FASTTEST channel 1 measurement queries.
 S2GSend "SWPCH1_ISO31"
 ' Read Multitone channel 1 distortion measurements.
 sDistortion1 = S2GRcv
 ' Send FASTTEST channel 2 measurement queries.
 S2GSend "SWPCH2_ISO31"
 ' Read Multitone channel 2 distortion measurements.
 sDistortion2 = S2GRcv
 ' 11. Noise Sweep, setup for noise processing.
 S2GSend ":DSP:OPSTATE SETUP;FASTTEST:MODE NOISE;:DELAY 0.1;:DSP:OPSTATE
     READING,AMP1,AMP2;REPROCESS?" ' original line
 ' Read REPROCESS? query response.
 sIO = S2GRcv
 If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
   UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
     time out then exit.
   Exit Sub
 End If
 ' Send FASTTEST channel 1 measurement queries.
 S2GSend "SWPCH1_ISO31"
 ' Read Multitone channel 1 noise measurements.
 sNoise1 = S2GRcv
 ' Send FASTTEST channel 2 measurement queries.
 S2GSend "SWPCH2_ISO31"
 ' Read Multitone channel 2 noise measurements.
 sNoise2 = S2GRcv
 sIO = "Level 1" & vbCrLf & sLevel1 & vbCrLf & "Level 2" & vbCrLf & sLevel2
 frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
     vbCrLf & vbCrLf & sIO & vbCrLf
 sIO = "Interchannel Phase (Phase 2)" & vbCrLf & sPhase
 frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & sIO & vbCrLf
 sIO = "Distortion 1" & vbCrLf & sDistortion1 & vbCrLf & "Distortion 2" & vbCrLf &
     sDistortion2
 frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & sIO & vbCrLf
 sIO = "Noise 1" & vbCrLf & sNoise1 & vbCrLf & "Noise 2" & vbCrLf & sNoise2
 frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & sIO & vbCrLf
 frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
 S2GSend ":DSP:OPSTATE SETUP"
 S2G_Errors
```

```
End Sub

Sub Batch_FASTTEST_MultitoneSwp()
  ' Multitone Signal Testing with FASTTEST DSP
  TEST_TITLE = "Batch_FASTTEST_MultitoneSwp"
  If IsAnalogPlusDSP = False Then
    MsgBox "System Two Analog Plus DSP option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  Dim sLevel1 As String, sLevel2 As String, sDistortion1 As String, sDistortion2 As
      String
  Dim sNoise1 As String, sNoise2 As String, sPhase As String
  ' Setup: set to defaults, enable OPC event reporting, turn off headers, set digital
      generator arbitrary waveform register size to 8192 samples per waveform.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:DGEN:ARBSIZE 8192"
  ' 1. FASTTEST DSP setup for signal acquisition and level sweep.
  S2GSend ":DSP:PROGRAM FASTTEST;FASTTEST:FREQRES 0;INPUT AD1X;SRC1 A;SRC2 B;LENGTH
      AUTO;MODE RESPONSE;PMODE ICHANNEL;PROCESS SYNC;TRGDELAY 0;TRIGGER NORMAL;:DELAY
      0.5"
  ' 2. Load the arbitrary waveform ISO31.AGM into waveform register 1.
  ArbLoad_ibwrtf 1, App.Path & "\ISO31.AGM"
  ' 3. Load the arbitrary waveform from waveform register 1 into both DSP digital
      generator waveform buffers 1 and 2 (left and right).
  ' 4. Setup the internal sample rate for 48 kHz required for correct frequencies in
      arbitrary waveform IS031.AGM.
  ' 5. Setup the analog generator for a 1.0 Vrms (1.414 Vpk) multitone signal level,
      select Sine D/A Arbitrary waveform, and turn the output on.
  ' 6. Setup the analog analyzer for XLR inputs with auto-ranging enabled. Delay 2
      seconds for autoranging, then disable autoranging to fix ranges.
  ' 7. Setup the headphone monitor to hear the multitone signal on the analog analyzer
      inputs, in stereo mode.
  S2GSend ":DGEN:ARBWFM 1,1;:DOUT:RATE 48000HZ;:AGEN:AMPL A,1V;AMPL B,1V;WFM
      DAARBITRARY,NONE;OUTPUT AB;:ANLR:INPUT A,XLR;INPUT B,XLR;AUTORANGE A,ON;AUTORANGE
      B,ON;:DELAY 2;:ANLR:AUTORANGE A,OFF;AUTORANGE B,OFF;:MON:STEREO INPUT;MODE STEREO"
  '8. Define a sweep table to sweep the ISO31 frequency bins for FASTTEST channels 1 and
      2.
  S2GSend ":DSP:TABLE
      2,ASCII,31,#017.57,23.43,29.29,41.01,52.73,64.45,82.03,99.6,123.04,158.2,199.21,251
      .95,316.4,398.43,498.04,632.81,802.73,1001.95,1248.04,1599.6,1998.04,2501.95,3152.3
      4,4001.95,4998.04,6351.56,7998.04,10001.95,12498.04,16001.95,19998.04"
  S2GSend ":DSP:TSELECT 2"
  ' 9. Level and Phase Sweep.
  S2GSend ":DSP:OPSTATE SETUP;TIMEOUT 5;SRCP
      FREQ,HZ,17.578125,19998.046875,31,ARB;OPSTATE READ,AMP1,AMP2,PHA2;:DSP:ACQX?"
  ' Read ACQX? query response.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"   ' Clear Event Status Register to clear OPC bit
  ' Send FASTTEST channel 1 measurement query.
  S2GSend ":DSP:BATCh? ASCII,AMP1,DBV"
  ' Read Multitone Level measurements.
  sLevel1 = S2GRcv
  ' Send FASTTEST channel 2 measurement query.
  S2GSend ":DSP:BATCh? ASCII,AMP2,DBV"
  ' Read Multitone Level measurements.
  sLevel2 = S2GRcv
  ' Send FASTTEST channel 2 measurement query.
  S2GSend ":DSP:BATCh? ASCII,PHA2,DEG"
  ' Read Multitone channel 2 phase measurements.
  sPhase = S2GRcv
  ' 10. Distortion Sweep, setup for distortion processing.
  S2GSend ":DSP:OPSTATE SETUP;FASTTEST:MODE DISTORTION;:DELAY 1.1;:DSP:OPSTATE
      READING,AMP1,AMP2;REPROCESS?"  ' original line
  ' Read REPROCESS? query response.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send FASTTEST channel 1 measurement query.
  S2GSend ":DSP:BATCh? ASCII,AMP1,DBV"
  ' Read Multitone channel 1 distortion measurements.
  sDistortion1 = S2GRcv
  ' Send FASTTEST channel 2 measurement query.
  S2GSend ":DSP:BATCh? ASCII,AMP2,DBV"
  ' Read Multitone channel 2 distortion measurements.
  sDistortion2 = S2GRcv
  ' 11. Noise Sweep, setup for noise processing.
  S2GSend ":DSP:OPSTATE SETUP;FASTTEST:MODE NOISE;:DELAY 0.1;:DSP:OPSTATE
      READING,AMP1,AMP2;REPROCESS?" ' original line
  ' Read REPROCESS? query response.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  ' Send FASTTEST channel 1 measurement query.
  S2GSend ":DSP:BATCh? ASCII,AMP1,DBV"
  ' Read Multitone channel 1 noise measurements.
  sNoise1 = S2GRcv
  ' Send FASTTEST channel 2 measurement query.
  S2GSend ":DSP:BATCh? ASCII,AMP2,DBV"
  ' Read Multitone channel 2 noise measurements.
  sNoise2 = S2GRcv
  ' Format data for output to user
  sIO = "Level 1" & vbCrLf & "Format = FREQ, AMP1, [...]" & vbCrLf & vbCrLf & sLevel1 &
      vbCrLf & vbCrLf & "Level 2" & vbCrLf & "Format = FREQ, AMP2, [...]" & vbCrLf &
      vbCrLf & sLevel2
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  sIO = "Interchannel Phase (Phase 2)" & vbCrLf & "Format = FREQ, PHA2, [...]" & vbCrLf &
      vbCrLf & sPhase
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & sIO & vbCrLf
  sIO = "Distortion 1" & vbCrLf & "Format = FREQ, AMP1, [...]" & vbCrLf & vbCrLf &
      sDistortion1 & vbCrLf & vbCrLf & "Distortion 2" & vbCrLf & "Format = FREQ, AMP2,
      [...]" & vbCrLf & vbCrLf & sDistortion2
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & sIO & vbCrLf
  sIO = "Noise 1" & vbCrLf & "Format = FREQ, AMP1, [...]" & vbCrLf & vbCrLf & sNoise1 &
      vbCrLf & vbCrLf & "Noise 2" & vbCrLf & "Format = FREQ, AMP2, [...]" & vbCrLf &
      vbCrLf & sNoise2
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors
End Sub

Sub Batch_FASTTEST_MultitoneSwp_RQS()
  ' Multitone Signal Testing with FASTTEST DSP using GPIB SRQ ReQuest Service (RQS)
      interrupt.
  ' Use this method to wait for FASTTEST acquisition triggering when a very long wait may
      be necessary
  ' and the GPIB bus must be available for other instruments. Use the AP Event Status
      Register (AESR)to cause an SRQ interrupt
  ' when the DSP Transform Complete bit becomes set (bit 5).
  TEST_TITLE = "Batch_FASTTEST_MultitoneSwp_RQS"
  If IsAnalogPlusDSP = False Then
    MsgBox "System Two Analog Plus DSP option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  Dim s2ChAmp_Phase2 As String, sDistortion As String, sNoise As String
  ' Setup: set to defaults, turn off headers, set digital generator arbitrary waveform
      register size to 8192 samples per waveform.
  S2GSend "*RCL 0;:HEADER OFF;:DGEN:ARBSIZE 8192"
  ' 1. FASTTEST DSP setup for signal acquisition and level sweep.
  S2GSend ":DSP:PROGRAM FASTTEST;FASTTEST:FREQRES 0;INPUT AD1X;SRC1 A;SRC2 B;LENGTH
      AUTO;MODE RESPONSE;PMODE ICHANNEL;PROCESS SYNC;TRGDELAY 0;TRIGGER NORMAL;:DELAY
      0.5"
  ' 2. Load the arbitrary waveform ISO31.AGM into waveform register 1.
  ArbLoad_ibwrtf 1, App.Path & "\ISO31.AGM"
  ' 3. Load the arbitrary waveform from waveform register 1 into both DSP digital
      generator waveform buffers 1 and 2 (left and right).
```

```
' 4. Setup the internal sample rate for 48 kHz required for correct frequencies in
    arbitrary waveform IS031.AGM.
' 5. Setup the analog generator for a 1.0 Vrms (1.414 Vpk) multitone signal level,
    select Sine D/A Arbitrary waveform, and turn the output on.
' 6. Setup the analog analyzer for XLR inputs with auto-ranging enabled. Delay 2
    seconds for autoranging, then disable autoranging to fix ranges.
' 7. Setup the headphone monitor to hear the multitone signal on the analog analyzer
    inputs, in stereo mode.
S2GSend ":DGEN:ARBWFM 1,1;:DOUT:RATE 48000HZ;:AGEN:AMPL A,1V;AMPL B,1V;WFM
    DAARBITRARY,NONE;OUTPUT AB;:ANLR:INPUT A,XLR;INPUT B,XLR;AUTORANGE A,ON;AUTORANGE
    B,ON;:DELAY 2;:ANLR:AUTORANGE A,OFF;AUTORANGE B,OFF;:MON:STEREO INPUT;MODE STEREO"
' 8. Define a sweep table to sweep the ISO31 frequency bins for FASTTEST channels 1 and
    2.
S2GSend ":DSP:TABLE
    2,ASCII,31,#017.57,23.43,29.29,41.01,52.73,64.45,82.03,99.6,123.04,158.2,199.21,251
    .95,316.4,398.43,498.04,632.81,802.73,1001.95,1248.04,1599.6,1998.04,2501.95,3152.3
    4,4001.95,4998.04,6351.56,7998.04,10001.95,12498.04,16001.95,19998.04"
S2GSend ":DSP:TSELECT 2"
' 9. Level and Phase Sweep. Use SRQ interrupt to detect that DSP Transform is complete
    (bit 5 in AESER).
S2GSend "*SRE 1;:APST:ENAB 32;:DSP:OPSTATE SETUP;TIMEOUT 5;SRCP
    FREQ,HZ,17.578125,19998.046875,31,ARB;OPSTATE READ,AMP1,AMP2,PHA2;ACQX?"
' Wait for SRQ interrupt with timeout.
If WaitForSRQ(10) = False Then
  ' If DSP Acquisition time out (no SRQ interrupt detected within timeout period) then
    exit option.
  If MsgBox("System Two SRQ Timeout or SRQ Interrupt Handler Not Active", vbOKCancel,
    MSGBOX_TITLE & TEST_TITLE) = vbCancel Then End
End If
' Read ACQX? query response.
sIO = S2GRcv
If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
    time out then exit.
  Exit Sub
End If
' Clear Event Status Register and AP Event Status Register
S2GSend "*CLS"
' Send FASTTEST measurement query for channels 1 & 2 and interchannel phase (channel
    2).
S2GSend ":DSP:BATCh? ASCII,AMP1,DBV,AMP2,DBV,PHA2,DEG"
' Read Multitone Level measurements for channels 1 & 2 and interchannel phase (channel
    2).
s2ChAmp_Phase2 = S2GRcv
' 10. Distortion Sweep, setup for distortion processing.
S2GSend ":DSP:OPSTATE SETUP;FASTTEST:MODE DISTORTION;:DELAY 1.1;:DSP:OPSTATE
    READING,AMP1,AMP2;REPROCESS?"  ' original line
' Read REPROCESS? query response.
sIO = S2GRcv
If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
    time out then exit.
  Exit Sub
End If
' Send FASTTEST channel 1 measurement query.
S2GSend ":DSP:BATCh? ASCII,AMP1,DBV,AMP2,DBV"
' Read Multitone channel 1 distortion measurements.
sDistortion = S2GRcv
' 11. Noise Sweep, setup for noise processing.
S2GSend ":DSP:OPSTATE SETUP;FASTTEST:MODE NOISE;:DELAY 0.1;:DSP:OPSTATE
    READING,AMP1,AMP2;REPROCESS?" ' original line
' Read REPROCESS? query response.
sIO = S2GRcv
If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
    time out then exit.
  Exit Sub
End If
' Send FASTTEST channel 1 measurement query.
S2GSend ":DSP:BATCh? ASCII,AMP1,DBV,AMP2,DBV"
' Read Multitone channel 1 noise measurements.
sNoise = S2GRcv
' Format data for output to user
sIO = "Level Channels 1 & 2 and Phase Channel 2" & vbCrLf & "Format = FREQ, AMP1, AMP2,
    PHA2, [...]" & vbCrLf & vbCrLf & s2ChAmp_Phase2
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE &
      vbCrLf & vbCrLf & sIO & vbCrLf
  sIO = "Distortion Channels 1 & 2" & vbCrLf & "Format = FREQ, AMP1, AMP2, [...]" &
      vbCrLf & vbCrLf & sDistortion
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & sIO & vbCrLf
  sIO = "Noise Channels 1 & 2" & vbCrLf & "Format = FREQ, AMP1, AMP2, [...]" & vbCrLf &
      vbCrLf & sNoise
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2GSend ":DSP:OPSTATE SETUP;*SRE 0;:APST:ENAB 0"   ' disable SRQ interrupts from S2G
  S2G_Errors
End Sub
```

Figure 2-32. Visual Basic subprogram FASTTEST_MultitoneSwp code

(start of Figure 2-33)

```
FASTTEST_MultitoneSwp

Level 1
-42.9424;-38.2173;-34.6673;-29.8147;-26.4423;-24.1771;-22.0062;-20.6496;-19.4897;-
18.5004;-17.9625;-17.6498;-17.4698;-17.3952;-17.2956;-17.252;-17.1605;-17.1474;-17.2465;-
17.4769;-17.4692;-17.5156;-17.5284;-17.647;-17.6916;-17.8438;-17.9653;-18.2617;-18.6954;-
19.5602;-20.6741
Level 2
-41.3994;-36.3739;-32.8579;-28.19;-25.2475;-23.2647;-21.2973;-20.0899;-19.124;-18.3968;-
17.9915;-17.828;-17.669;-17.6314;-17.473;-17.3836;-17.1146;-16.9409;-17.0153;-17.2348;-
17.3217;-17.4675;-17.443;-17.4968;-17.5197;-17.6693;-17.8454;-18.2162;-18.7152;-19.7107;-
20.8929

Interchannel Phase (Phase 2)
-
1.49414;0.175781;2.72461;5.44922;7.03125;6.63574;6.37207;6.19629;6.15234;5.31738;4.39453;
3.20801;2.06543;1.05469;0.0878906;-0.703125;-1.4502;-0.878906;-
0.219727;0.439453;1.01074;0.65918;0.131836;0.351563;0.74707;1.27441;1.88965;2.19727;2.548
83;2.68066;2.54883

Distortion 1
-112.498;-118.868;-120.324;-115.916;-118.946;-108.685;-112.883;-113.83;-100.68;-102.557;-
101.438;-104.305;-104.319;-103.629;-101.266;-99.3658;-101.65;-100.723;-98.5286;-95.4917;-
95.6226;-95.4748;-94.5816;-93.8307;-92.6371;-93.1258;-92.9882;-92.0951;-91.8866;-
89.0529;-90.0698
Distortion 2
-79.8156;-120.324;-118.868;-109.749;-104.856;-88.3405;-102.297;-107.338;-97.7232;-
100.861;-101.902;-101.27;-101.692;-100.7;-99.5606;-99.0137;-100.947;-102.095;-100.336;-
97.2091;-96.6447;-95.567;-95.287;-94.8386;-94.2195;-93.2856;-92.2624;-91.4244;-90.3055;-
86.9315;-88.5324

Noise 1
-124.188;-118.868;-120.324;-119.782;-121.865;-113.87;-118.427;-127.068;-101.402;-
116.428;-110.077;-114.492;-113.048;-113.473;-112.041;-111.341;-112.48;-109.36;-108.645;-
108.118;-107.365;-106.827;-105.089;-104.212;-103.685;-102.958;-101.668;-101.22;-100.897;-
99.1562;-98.7734
Noise 2
-87.414;-120.324;-118.868;-111.242;-105.744;-91.5421;-104.73;-112.68;-98.1335;-115.942;-
112.582;-112.551;-110.067;-110.402;-109.809;-109.973;-108.942;-107.736;-108.929;-
108.022;-106.357;-106.84;-105.215;-104.314;-103.728;-101.465;-100.611;-100.845;-100.095;-
98.5558;-98.5055

Batch_FASTTEST_MultitoneSwp

Level 1
Format = FREQ, AMP1, [...]

ASCII,31,17.5724,-42.9477,23.4318,-38.2212,29.2912,-34.6706,41.0099,-29.8173,52.7287,-
26.444,64.4474,-24.1783,82.0313,-22.0065,99.5979,-20.6496,123.041,-19.4895,158.197,-
18.5002,199.207,-17.9623,251.947,-17.6497,316.401,-17.4698,398.432,-17.3951,498.041,-
17.2956,632.813,-17.2519,802.729,-17.1605,1001.95,-17.1474,1248.04,-17.2466,1599.6,-
17.477,1998.04,-17.4692,2501.95,-17.5156,3152.34,-17.5284,4001.95,-17.647,4998.04,-
17.6915,6351.56,-17.8438,7998.04,-17.9653,10001.9,-18.2617,12498,-18.6954,16001.9,-
19.5602,19998,-20.6741

Level 2
Format = FREQ, AMP2, [...]
```

```
ASCII,31,17.5724,-41.4007,23.4318,-36.3733,29.2912,-32.858,41.0099,-28.1901,52.7287,-
25.2462,64.4474,-23.2644,82.0313,-21.2969,99.5979,-20.0897,123.041,-19.1237,158.197,-
18.3964,199.207,-17.9913,251.947,-17.828,316.401,-17.669,398.432,-17.6313,498.041,-
17.473,632.813,-17.3836,802.729,-17.1146,1001.95,-16.941,1248.04,-17.0153,1599.6,-
17.2348,1998.04,-17.3217,2501.95,-17.4676,3152.34,-17.443,4001.95,-17.4968,4998.04,-
17.5197,6351.56,-17.6693,7998.04,-17.8456,10001.9,-18.2167,12498,-18.7156,16001.9,-
19.7107,19998,-20.8928

Interchannel Phase (Phase 2)
Format = FREQ, PHA2, [...]

ASCII,31,17.5724,-
1.53809,23.4318,0.219727,29.2912,2.72461,41.0099,5.44922,52.7287,7.03125,64.4474,6.67969,
82.0313,6.37207,99.5979,6.19629,123.041,6.15234,158.197,5.31738,199.207,4.39453,251.947,3
.25195,316.401,2.06543,398.432,1.05469,498.041,0.0878906,632.813,-0.703125,802.729,-
1.4502,1001.95,-0.878906,1248.04,-
0.219727,1599.6,0.483398,1998.04,1.01074,2501.95,0.65918,3152.34,0.175781,4001.95,0.35156
3,4998.04,0.74707,6351.56,1.31836,7998.04,1.88965,10001.9,2.19727,12498,2.54883,16001.9,2
.68066,19998,2.54883

Distortion 1
Format = FREQ, AMP1, [...]

ASCII,31,17.5724,-110.718,23.4318,-127.242,29.2912,-135.693,41.0099,-112.182,52.7287,-
120.471,64.4474,-111.456,82.0313,-112.149,99.5979,-113.687,123.041,-101.979,158.197,-
101.9,199.207,-102.213,251.947,-105.081,316.401,-105.341,398.432,-102.089,498.041,-
101.878,632.813,-98.455,802.729,-100.812,1001.95,-100.403,1248.04,-98.5164,1599.6,-
95.2956,1998.04,-95.6347,2501.95,-95.7013,3152.34,-94.5884,4001.95,-93.5409,4998.04,-
92.7535,6351.56,-93.2472,7998.04,-92.6531,10001.9,-92.4806,12498,-91.9594,16001.9,-
89.1197,19998,-89.9796

Distortion 2
Format = FREQ, AMP2, [...]

ASCII,31,17.5724,-79.8835,23.4318,-116.532,29.2912,-118.222,41.0099,-108.556,52.7287,-
106.187,64.4474,-89.0068,82.0313,-103.687,99.5979,-113.082,123.041,-96.8844,158.197,-
100.651,199.207,-101.262,251.947,-101.22,316.401,-101.63,398.432,-100.408,498.041,-
100.578,632.813,-99.278,802.729,-100.04,1001.95,-102.22,1248.04,-100.072,1599.6,-
96.8239,1998.04,-96.9811,2501.95,-96.0971,3152.34,-95.048,4001.95,-94.5328,4998.04,-
94.1047,6351.56,-93.6909,7998.04,-92.5064,10001.9,-90.8462,12498,-90.3307,16001.9,-
87.0041,19998,-88.4716

Noise 1
Format = FREQ, AMP1, [...]

ASCII,31,17.5724,-123.537,23.4318,-127.242,29.2912,-135.693,41.0099,-139.373,52.7287,-
128.959,64.4474,-112.857,82.0313,-118.999,99.5979,-123.691,123.041,-102.844,158.197,-
116.487,199.207,-110.279,251.947,-115.296,316.401,-115.307,398.432,-112.498,498.041,-
109.874,632.813,-112.636,802.729,-109.622,1001.95,-110.251,1248.04,-109.768,1599.6,-
108.787,1998.04,-108.186,2501.95,-106.241,3152.34,-104.728,4001.95,-104.527,4998.04,-
103.675,6351.56,-102.418,7998.04,-101.548,10001.9,-101.414,12498,-100.511,16001.9,-
99.2117,19998,-98.955

Noise 2
Format = FREQ, AMP2, [...]

ASCII,31,17.5724,-87.5601,23.4318,-116.532,29.2912,-118.222,41.0099,-111.268,52.7287,-
107.073,64.4474,-91.9974,82.0313,-105.941,99.5979,-116.828,123.041,-97.3867,158.197,-
114.814,199.207,-114.157,251.947,-117.414,316.401,-107.206,398.432,-113.341,498.041,-
107.508,632.813,-111.363,802.729,-110.953,1001.95,-107.967,1248.04,-108.621,1599.6,-
106.575,1998.04,-107.271,2501.95,-107.049,3152.34,-104.574,4001.95,-103.73,4998.04,-
102.913,6351.56,-101.271,7998.04,-101.125,10001.9,-100.908,12498,-100.595,16001.9,-
98.4646,19998,-98.7052

Batch_FASTTEST_MultitoneSwp_RQS

Level Channels 1 & 2 and Phase Channel 2
Format = FREQ, AMP1, AMP2, PHA2, [...]

ASCII,31,17.5724,-42.9482,-41.399,-1.49414,23.4318,-38.222,-36.375,0.219727,29.2912,-
34.6711,-32.8595,2.72461,41.0099,-29.8181,-28.1902,5.44922,52.7287,-26.4445,-
25.2476,7.03125,64.4474,-24.179,-23.2662,6.67969,82.0313,-22.007,-
21.2974,6.37207,99.5979,-20.65,-20.0901,6.19629,123.041,-19.4898,-
19.1241,6.15234,158.197,-18.5006,-18.3969,5.31738,199.207,-17.9625,-
17.9916,4.39453,251.947,-17.6498,-17.8282,3.25195,316.401,-17.4699,-
```

```
17.669,2.06543,398.432,-17.3952,-17.6314,1.05469,498.041,-17.2957,-
17.4732,0.0878906,632.813,-17.252,-17.3837,-0.703125,802.729,-17.1606,-17.1146,-
1.4502,1001.95,-17.1476,-16.941,-0.878906,1248.04,-17.2466,-17.0153,-0.219727,1599.6,-
17.477,-17.2348,0.439453,1998.04,-17.4693,-17.3217,1.05469,2501.95,-17.5157,-
17.4676,0.65918,3152.34,-17.5284,-17.443,0.175781,4001.95,-17.647,-
17.4968,0.351563,4998.04,-17.6916,-17.5197,0.703125,6351.56,-17.8438,-
17.6694,1.27441,7998.04,-17.9653,-17.8457,1.8457,10001.9,-18.2617,-
18.2168,2.15332,12498,-18.6954,-18.7156,2.50488,16001.9,-19.5601,-19.7107,2.68066,19998,-
20.674,-20.8927,2.54883

Distortion Channels 1 & 2
Format = FREQ, AMP1, AMP2, [...]

ASCII,31,17.5724,-108.268,-113.399,23.4318,-131.839,-122.076,29.2912,-123.106,-
117.06,41.0099,-116.157,-110.85,52.7287,-117.943,-105.444,64.4474,-117.043,-
88.8798,82.0313,-113.671,-101.276,99.5979,-110.655,-113.305,123.041,-100.997,-
97.805,158.197,-102.493,-100.15,199.207,-101.967,-100.384,251.947,-106.252,-
104.838,316.401,-104.737,-100.506,398.432,-102.59,-100.489,498.041,-101.419,-
99.8968,632.813,-98.3763,-99.0792,802.729,-100.643,-99.9772,1001.95,-100.892,-
102.808,1248.04,-98.6244,-99.9164,1599.6,-95.1944,-96.7672,1998.04,-95.7918,-
96.3786,2501.95,-95.6491,-95.7053,3152.34,-94.6245,-95.0013,4001.95,-93.9558,-
94.4678,4998.04,-92.6333,-94.0396,6351.56,-92.9983,-93.5294,7998.04,-92.6832,-
92.2846,10001.9,-92.4704,-91.0281,12498,-91.9604,-90.3295,16001.9,-89.0709,-
86.9581,19998,-90.0072,-88.3712

Noise Channels 1 & 2
Format = FREQ, AMP1, AMP2, [...]

ASCII,31,17.5724,-116.298,-116.909,23.4318,-131.839,-122.076,29.2912,-123.106,-
117.06,41.0099,-119.505,-113.473,52.7287,-123.038,-107.187,64.4474,-117.323,-
91.9553,82.0313,-120.182,-103.645,99.5979,-117.47,-115.274,123.041,-102.003,-
98.1207,158.197,-115.838,-117.86,199.207,-110.814,-115.481,251.947,-114.399,-
113.015,316.401,-113.822,-108.916,398.432,-111.993,-113.291,498.041,-110.439,-
112.266,632.813,-112.216,-110.734,802.729,-112.539,-110.191,1001.95,-110.45,-
109.314,1248.04,-109.861,-107.185,1599.6,-107.752,-106.896,1998.04,-106.666,-
106.674,2501.95,-107.264,-105.699,3152.34,-106.106,-105.049,4001.95,-104.944,-
104.222,4998.04,-102.332,-103.577,6351.56,-102.58,-102.132,7998.04,-101.456,-
100.828,10001.9,-101.086,-100.86,12498,-100.533,-100.518,16001.9,-98.9458,-
98.4079,19998,-98.9694,-98.4324
```

Figure 2-33. Visual Basic measurement results for subprograms FASTTEST_MultitoneSwp, Batch_FASTTEST_MultitoneSwp, and Batch_FASTTEST_MultitoneSwp_RQS

Acoustic Measurement Sweeps with the MLS DSP Program

The examples below test a monitor speaker with a calibrated microphone placed 10 inches from the speaker grill. The microphone preamplifier is connected directly to the analog analyzer channel A XLR input and is driven by a built-in amplifier with line input connected to the analog generator channel A XLR output. The acoustic delay for the first arrival impulse may be seen in the measurement data for the time domain impulse response data to be about 0.975 ms. This value is used to set the MLS delay parameter prior to acquiring phase measurements.

This code is provided in the “S2Gsample2.bas” file on the CD ROM included with this manual. The first subprogram “MLS_ImpulseRespSwp” reads the data using a series of :DSP:MLS:AMPL? queries and PHASE? queries for channel 1. The second subprogram “Batch_MLS_ImpulseRespSwp” reads the data using a :DSP:BATCh? query for the same parameters, resulting in a faster response if many data points are required. The third subprogram shows how to acquire two channels of measurement data using the :DSP:BATCh? query technique. This illustrates how the BATCH? query is capable of returning complex sets of measurement data in a single response.

(Start of Figure 2-34)

```
Sub MLS_ImpulseRespSwp()
  ' Acoustic Measurement Sweeps with the Maximum Length Sequence (MLS) DSP Program
  TEST_TITLE = "MLS_ImpulseRespSwp"
  If IsAnalogPlusDSP = False Then
    MsgBox "System Two Analog Plus DSP option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  Dim sAMPLQuery As String, dTimebase As Double
  sAMPLQuery = ":DSP:MLS:"
  ' Send the setup string to load the MLS DSP program and set its parameters for an
      impulse response sweep.
  ' Setup the analog generator for an MLS pink noise #1 waveform with output amplitude of
      1 Vrms (1.414 Vpk).
  ' Setup analog analyzer for XLR input with input ranges fixed after an autoranging for
      2 seconds,
  ' stereo headphone monitor of the analyzer inputs, sweep source parameters setup for
      MLS impulse response time sweep
  ' from 0 seconds to 55 milliseconds, and acquisition of the impulse response.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:STEREO INPUT;MODE
      STEREO;:DSP:PROGRAM MLS;:DSP:MLS:DELAY 0SEC;ETWINDOW NONE;INPUT AD1X;SRC1 A;SRC2
      B;MODE PEAK;START NONE;STOP NONE;TIME IMPULSE;TRIGGER AGEN"
  S2GSend ":DOUT:RATE 48000HZ;:AGEN:WFM DAMLS,P1;AMPL A,1V;AMPL B,1V;OUTPUT
      AB;:ANLR:INPUT A,XLR;INPUT B,XLR;AUTORANGE A,ON;AUTORANGE B,ON;:DELAY
      2;:ANLR:AUTORANGE A,OFF;AUTORANGE B,OFF"
  S2GSend ":DSP:OPSTATE SETUP;TIMEOUT 3;SRCPARAMS
      TIME,SEC,0.0,0.055,16383,ARB;:DSP:OPSTATE READING,AMP1;ACQX?;:AGEN:OUTPUT OFF;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(10) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Receive the response to the ACQX? query
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  ' Send the MLS impulse response measurement sweep queries for time points from 0 mSec
      to 1.4 mSec in 50 uSec intervals
  ' to acquire the first impulse.
  ' Create MLS Impulse amplitude query string for sweep from 0 to 1.4 mS in 50 nS
      intervals.
  For dTimebase = 0 To 0.0014 Step 0.00005
    sAMPLQuery = sAMPLQuery & "AMPL? 1,DBV," & Format(dTimebase, "#.00000") & ";"
  Next dTimebase
  sAMPLQuery = sAMPLQuery & "*OPC"
  S2GSend sAMPLQuery    ' Send the MLS Impulse amplitude query string
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the impulse response measurements noting the first impulse peak.
  sIO = S2GRcv
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      MLS Impulse Response (0 to 1.4 mS)" & vbCrLf & vbCrLf & sIO & vbCrLf
  ' Set the MLS Delay parameter to the time of the first impulse peak in the impulse
      response measurements above.
  ' Set up MLS to retransform the signal in the time interval between the first impulse
      arrival and the second arrival.
  ' Set the MLS Delay parameter to the time of the first impulse peak in the impulse
      response measurements above.
  ' Set up MLS to retransform the signal in the time interval between the first impulse
      arrival and the second arrival (0.975mS to 23 mS):
  S2GSend ":DSP:OPSTATE SETUP;MLS:DELAY 0.975E-3SEC;:DSP:SRCPARAMS TIME,SEC,0.975E-3,23E-
      3,16383,ARB;OPSTATE READING,AMP1;XFRM?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the XFRM? query response.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  ' Read at least one impulse response measurement after changing the SRCPARAMS time span
      in order to set MLS time span for frequency domain processing.
  S2GSend ":DSP:MLS:AMPL? 1,DBV,1E-3"
  ' Read the impulse response measurement.
  sIO = S2GRcv
  ' Setup MLS to retransform the time interval to the frequency domain for amplitude and
      phase measurements.
  S2GSend ":DSP:OPSTATE SETUP;SRCPARAMS FREQ,HZ,20,2E+4,31,ARB;OPSTATE
      READING,AMP1,PHA1;XFRM?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the XFRM? query response.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  ' Query the MLS Amplitude vs Frequency measurements for channel 1.
  S2GSend ":DSP:MLS:AMPL? 1,DBV,20;AMPL? 1,DBV,25.17;AMPL? 1,DBV,31.7;AMPL?
      1,DBV,39.9;AMPL? 1,DBV,50.2;AMPL? 1,DBV,63.2;AMPL? 1,DBV,79.6;AMPL?
      1,DBV,100.2;AMPL? 1,DBV,126.2;AMPL? 1,DBV,158.9;AMPL? 1,DBV,200;AMPL?
      1,DBV,251.8;AMPL? 1,DBV,317;AMPL? 1,DBV,399;AMPL? 1,DBV,502.4;AMPL?
      1,DBV,632.4;AMPL? 1,DBV,796.2;AMPL? 1,DBV,1002.4;AMPL? 1,DBV,1261.9;AMPL?
      1,DBV,1588.7;AMPL? 1,DBV,2000;AMPL? 1,DBV,2517.9;AMPL? 1,DBV,3169.8;AMPL?
      1,DBV,3991;AMPL? 1,DBV,5024.8;AMPL? 1,DBV,6324.5;AMPL? 1,DBV,7962.1;AMPL?
      1,DBV,10024;AMPL? 1,DBV,12619;AMPL? 1,DBV,15887;AMPL? 1,DBV,20000;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the MLS Amplitude vs Frequency measurements.
  sIO = S2GRcv
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      MLS Amplitude vs Frequency (17 Hz to 20 kHz)" & vbCrLf & vbCrLf & sIO & vbCrLf
  ' Query the MLS Phase vs Frequency measurements for channel 1.
  S2GSend ":DSP:MLS:PHASE? 1,DEG,17.57;PHASE? 1,DEG,23.43;PHASE? 1,DEG,29.29;PHASE?
      1,DEG,41.01;PHASE? 1,DEG,52.73;PHASE? 1,DEG,64.45;PHASE? 1,DEG,82.03;PHASE?
      1,DEG,99.6;PHASE? 1,DEG,123.04;PHASE? 1,DEG,158.2;PHASE? 1,DEG,199.21;PHASE?
      1,DEG,251.95;PHASE? 1,DEG,316.4;PHASE? 1,DEG,398.43;PHASE? 1,DEG,498.04;PHASE?
      1,DEG,632.81;PHASE? 1,DEG,802.73;PHASE? 1,DEG,1001.95;PHASE? 1,DEG,1248.04;PHASE?
      1,DEG,1599.6;PHASE? 1,DEG,1998.04;PHASE? 1,DEG,2501.95;PHASE? 1,DEG,3152.34;PHASE?
      1,DEG,4001.95;PHASE? 1,DEG,4998.04;PHASE? 1,DEG,6351.56;PHASE? 1,DEG,7998.04;PHASE?
      1,DEG,10001.95;PHASE? 1,DEG,12498.04;PHASE? 1,DEG,16001.95;PHASE?
      1,DEG,19998.04;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the MLS Phase vs Frequency measurements.
  sIO = S2GRcv
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      MLS Phase vs Frequency (20 Hz to 20 kHz)" & vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors
End Sub
Sub Batch_MLS_ImpulseRespSwp()
  ' Acoustic Measurement Sweeps with the Maximum Length Sequence (MLS) DSP Program
  TEST_TITLE = "Batch_MLS_ImpulseRespSwp"
  If IsAnalogPlusDSP = False Then
    MsgBox "System Two Analog Plus DSP option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the setup string to load the MLS DSP program and set its parameters for an
      impulse response sweep.
  ' Setup the analog generator for an MLS pink noise #1 waveform with output amplitude of
      1 Vrms (1.414 Vpk).
  ' Setup analog analyzer for XLR input with input ranges fixed after an autoranging for
      2 seconds,
  ' stereo headphone monitor of the analyzer inputs, sweep source parameters setup for
      MLS impulse response time sweep
  ' from 0 seconds to 55 milliseconds, and acquisition of the impulse response.
```

```
S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:STEREO INPUT;MODE
    STEREO;:DSP:PROGRAM MLS;:DSP:MLS:DELAY 0SEC;ETWINDOW NONE;INPUT AD1X;SRC1 A;SRC2
    B;MODE PEAK;START NONE;STOP NONE;TIME IMPULSE;TRIGGER AGEN"
S2GSend ":DOUT:RATE 48000HZ;:AGEN:WFM DAMLS,P1;AMPL A,1V;AMPL B,1V;OUTPUT
    AB;:ANLR:INPUT A,XLR;INPUT B,XLR;AUTORANGE A,ON;AUTORANGE B,ON;:DELAY
    2;:ANLR:AUTORANGE A,OFF;AUTORANGE B,OFF"
S2GSend ":DSP:OPSTATE SETUP;TIMEOUT 3;SRCPARAMS TIME,SEC,0.0,0.055,28,LIN;:DSP:OPSTATE
    READING,AMP1;ACQX?;:DELAY 1;:AGEN:OUTPUT OFF;*OPC"
' wait for ESB bit in Status Byte
If WaitForESB(10) = False Then Exit Sub ' Quit, do not read measurement response.
' Receive the response to the ACQX? query
sIO = S2GRcv
If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
    time out then exit.
  Exit Sub
End If
S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
' Send the MLS impulse response measurement sweep queries for time points from 0 mSec
    to 1.4 mSec in 50 uSec intervals
' to acquire the first impulse.
' Set sweep for start of 0 sec and stop of 1.4 mS with linear intervals of 50 uS (28
    steps, 29 points), use more steps if more time resolution is required.
S2GSend ":DSP:OPSTATE SETUP;SRCPARAMS TIME,SEC,0.0,0.0014,28,LIN;:DSP:OPSTATE
    READING,AMP1;REPROCESS?;*OPC"
' wait for ESB bit in Status Byte
If WaitForESB(10) = False Then Exit Sub ' Quit, do not read measurement response.
' Receive the response to the ACQX? query
sIO = S2GRcv
If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
    time out then exit.
  Exit Sub
End If
S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
S2GSend ":DSP:BATCh? ASCII,AMP1,DBV"
' Read the impulse response measurements noting the first impulse peak.
sIO = S2GRcv
frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
    MLS Impulse Response (0 to 1.4 mS)" & vbCrLf
frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & "   Format = time, ch1
    ampl, ..." & vbCrLf & vbCrLf & sIO & vbCrLf
' Set the MLS Delay parameter to the time of the first impulse peak in the impulse
    response measurements above.
' Set up MLS to retransform the signal in the time interval between the first impulse
    arrival and the second arrival.
' Set the MLS Delay parameter to the time of the first impulse peak in the impulse
    response measurements above.
' Set up MLS to retransform the signal in the time interval between the first impulse
    arrival and the second arrival (0.975mS to 23 mS), linear intervals
S2GSend ":DSP:OPSTATE SETUP;MLS:DELAY 0.975E-3SEC;:DSP:SRCPARAMS TIME,SEC,0.975E-3,23E-
    3,16383,ARB;OPSTATE READING,AMP1;XFRM?;*OPC"
' wait for ESB bit in Status Byte
If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
' Read the XFRM? query response.
sIO = S2GRcv
If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
    time out then exit.
  Exit Sub
End If
S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
' Read at least one impulse response measurement after changing the SRCPARAMS time span
    in order to set MLS time span for frequency domain processing.
S2GSend ":DSP:MLS:AMPL? 1,DBV,1E-3"
' Read the impulse response measurement.
sIO = S2GRcv
' Setup MLS to retransform the time interval to the frequency domain for amplitude and
    phase measurements.
S2GSend ":DSP:OPSTATE SETUP;SRCPARAMS FREQ,HZ,20,2E+4,30,LOG;OPSTATE
    READING,AMP1,PHA1;XFRM?;*OPC"
' wait for ESB bit in Status Byte
If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
' Read the XFRM? query response.
sIO = S2GRcv
```

```
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  ' Query the MLS Amplitude vs Frequency measurements for channel 1.
  S2GSend ":DSP:BATCh? ASCII,AMP1,DBV,PHA1,DEG"
  ' Read the MLS Amplitude vs Frequency measurements.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      MLS Amplitude & Phase vs Frequency (20 Hz to 20 kHz)" & vbCrLf
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & "   Format = freq, ch1
      ampl, ch1 phase, ..." & vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors
End Sub
Sub Batch_Stereo_MLS_ImpulseRespSwp()
  ' Acoustic Measurement Sweeps with the Maximum Length Sequence (MLS) DSP Program
  TEST_TITLE = "Batch_Stereo_MLS_ImpulseRespSwp"
  If IsAnalogPlusDSP = False Then
    MsgBox "System Two Analog Plus DSP option not installed. Test Aborted.", vbOKOnly,
      MSGBOX_TITLE & TEST_TITLE
    Exit Sub
  End If
  ' Send the setup string to load the MLS DSP program and set its parameters for an
      impulse response sweep.
  ' Setup the analog generator for an MLS pink noise #1 waveform with output amplitude of
      1 Vrms (1.414 Vpk).
  ' Setup analog analyzer for XLR input with input ranges fixed after an autoranging for
      2 seconds,
  ' stereo headphone monitor of the analyzer inputs, sweep source parameters setup for
      MLS impulse response time sweep
  ' from 0 seconds to 55 milliseconds, and acquisition of the impulse response.
  S2GSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:STEREO INPUT;MODE
      STEREO;:DSP:PROGRAM MLS;:DSP:MLS:DELAY 0SEC;ETWINDOW NONE;INPUT AD1X;SRC1 A;SRC2
      B;MODE PEAK;START NONE;STOP NONE;TIME IMPULSE;TRIGGER AGEN"
  S2GSend ":DOUT:RATE 48000HZ;:AGEN:WFM DAMLS,P1;AMPL A,1V;AMPL B,1V;OUTPUT
      AB;:ANLR:INPUT A,XLR;INPUT B,XLR;AUTORANGE A,ON;AUTORANGE B,ON;:DELAY
      2;:ANLR:AUTORANGE A,OFF;AUTORANGE B,OFF"
  S2GSend ":DSP:OPSTATE SETUP;TIMEOUT 3;SRCPARAMS TIME,SEC,0.0,0.055,28,LIN;:DSP:OPSTATE
      READING,AMP1,AMP2;ACQX?;:DELAY 1;:AGEN:OUTPUT OFF;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(10) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Receive the response to the ACQX? query
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  ' Send the MLS impulse response measurement sweep queries for time points from 0 mSec
      to 1.4 mSec in 50 uSec intervals to acquire the first impulse.
  ' Set sweep for start of 0 sec and stop of 1.4 mS with linear intervals of 50 uS (28
      steps, 29 points), use more steps if more time resolution is required.
  S2GSend ":DSP:OPSTATE SETUP;SRCPARAMS TIME,SEC,0.0,0.0014,28,LIN;:DSP:OPSTATE
      READING,AMP1,AMP2;REPROCESS?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Receive the response to the ACQX? query
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  S2GSend ":DSP:BATCh? ASCII,AMP1,DBV,AMP2,DBV"
  ' Read the impulse response measurements noting the first impulse peak.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      MLS Impulse Response (0 to 1.4 mS)" & vbCrLf
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & "   Format = time, ch1
      ampl, ch2 ampl, ..." & vbCrLf & vbCrLf & sIO & vbCrLf
  ' Set the MLS Delay parameter to the time of the first impulse peak in the impulse
      response measurements above.
  ' Set up MLS to retransform the signal in the time interval between the first impulse
      arrival and the second arrival.
  ' Set the MLS Delay parameter to the time of the first impulse peak in the impulse
      response measurements above.
  ' Set up MLS to retransform the signal in the time interval between the first impulse
      arrival and the second arrival (0.975mS to 23 mS), linear intervals
  S2GSend ":DSP:OPSTATE SETUP;MLS:DELAY 0.975E-3SEC;:DSP:SRCPARAMS TIME,SEC,0.975E-3,23E-
      3,16383,ARB;OPSTATE READING,AMP1,AMP2;XFRM?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the XFRM? query response.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  ' Read at least one impulse response measurement after changing the SRCPARAMS time span
      in order to set MLS time span for frequency domain processing.
  S2GSend ":DSP:MLS:AMPL? 1,DBV,1E-3;AMPL? 2,DBV,1E-3"
  ' Read the impulse response measurement.
  sIO = S2GRcv
  ' Setup MLS to retransform the time interval to the frequency domain for amplitude and
      phase measurements.
  S2GSend ":DSP:OPSTATE SETUP;SRCPARAMS FREQ,HZ,20,2E+4,30,LOG;OPSTATE
      READING,AMP1,PHA1,AMP2,PHA2;XFRM?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then Exit Sub ' Quit, do not read measurement response.
  ' Read the XFRM? query response.
  sIO = S2GRcv
  If sIO <> "0" Then   ' System Two DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "System Two DSP ACQX or XFRM or REPROCESS Timeout" ' If DSP Acquisition
      time out then exit.
    Exit Sub
  End If
  S2GSend "*CLS"    ' clear Event Status Register to clear OPC bit.
  ' Query the MLS Amplitude vs Frequency measurements for channel 1.
  S2GSend ":DSP:BATCh? ASCII,AMP1,DBV,PHA1,DEG,AMP2,DBV,PHA2,DEG"
  ' Read the MLS Stereo Amplitude & Phase vs Frequency measurements.
  sIO = S2GRcv
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & vbCrLf & TEST_TITLE & "
      MLS Amplitude & Phase vs Frequency (20 Hz to 20 kHz)" & vbCrLf
  frmS2GSample2.txtOutput.Text = frmS2GSample2.txtOutput.Text & "   Format = freq, ch1
      ampl, ch1 phase, ch2 ampl, ch2 phase, ..." & vbCrLf & vbCrLf & sIO & vbCrLf
  frmS2GSample2.txtOutput.SelStart = Len(frmS2GSample2.txtOutput.Text)
  S2GSend ":DSP:OPSTATE SETUP"
  S2G_Errors
End Sub
```

Figure 2-34. Visual Basic code for subprograms MLS_ImpulseRespSwp, Batch_MLS_ImpulseRespSwp, and Batch_Stereo_MLS_ImpulseRespSwp

(start of Figure 2-35)

```
MLS_ImpulseRespSwp MLS Impulse Response (0 to 1.4 mS)

-84.9527;-87.1154;-85.4807;-88.8834;-89.4443;-85.2562;-86.3423;-84.4613;-79.6146;-
74.6291;-73.6306;-69.1528;-64.7204;-61.6665;-55.9202;-52.8587;-49.2953;-44.3533;-
20.9272;-20.9481;-38.9558;-33.1008;-32.3733;-35.0377;-42.6671;-49.1298;-34.8198;-37.216;-
34.4325

MLS_ImpulseRespSwp MLS Amplitude vs Frequency (17 Hz to 20 kHz)

-24.1017;-24.8266;-23.8627;-20.8211;-19.0043;-18.6482;-19.945;-17.642;-14.5924;-12.5905;-
11.5462;-12.1588;-18.7266;-24.9595;-19.1604;-21.1105;-24.1271;-25.6627;-23.7261;-
22.7837;-30.5421;-18.9036;-21.8946;-24.3922;-18.6146;-24.8418;-22.72;-20.8955;-16.5849;-
13.7429;-16.1138

MLS_ImpulseRespSwp MLS Phase vs Frequency (20 Hz to 20 kHz)
```

```
-19.9512;-7.91016;16.5674;27.5977;15.0293;11.0742;-1.31836;37.5293;13.0957;-1.18652;-
19.9512;-43.8135;-69.6533;-33.4424;-45.8789;-49.043;-68.4229;-40.957;-54.668;-58.5791;-
117.686;-1089.98;-1147.94;-1823.2;-1924.63;-2370.28;-2799.18;-3162.08;-3314.53;-4272.32;-
4733.22

Batch_MLS_ImpulseRespSwp MLS Impulse Response (0 to 1.4 mS)
   Format = time, ch1 ampl, ...

ASCII,29,0,-88.5176,4.99674E-05,-92.2881,9.99349E-05,-89.4897,0.000150065,-
88.6172,0.000200033,-88.8993,0.00025,-85.5274,0.000299967,-88.8807,0.000349935,-
84.5252,0.000400065,-80.9076,0.000450033,-75.4446,0.0005,-73.6806,0.000549967,-
69.6482,0.000599935,-64.4756,0.000650065,-61.7144,0.000700033,-56.2585,0.00075,-
53.2176,0.000799967,-49.6137,0.000849935,-44.6392,0.000900065,-21.2204,0.000950033,-
20.9347,0.001,-39.2705,0.00104997,-33.3476,0.00109993,-31.9969,0.00115007,-
35.3411,0.00120003,-42.7163,0.00125,-48.8011,0.00129997,-34.9175,0.00134993,-
37.8264,0.00140007,-34.6242

Batch_MLS_ImpulseRespSwp MLS Amplitude & Phase vs Frequency (20 Hz to 20 kHz)
   Format = freq, ch1 ampl, ch1 phase, ...

ASCII,31,19.9986,-18.3655,23.5986,25.177,-17.8831,10.0195,31.7001,-
18.0136,3.69141,39.9055,-18.9669,-6.85547,50.2396,-21.8531,-11.1182,63.2458,-
20.8236,19.0723,79.6223,-20.1708,14.5459,100.239,-18.8802,41.9678,126.194,-
13.989,14.1943,158.867,-12.6085,1.66992,200.003,-11.2678,-20.6982,251.787,-12.1878,-
56.8213,316.978,-18.9927,-72.4658,399.05,-24.5798,-25.1367,502.378,-19.3192,-
48.7354,632.458,-21.3458,-47.2412,796.217,-23.9064,-66.4893,1002.38,-25.724,-
45.3076,1261.92,-24.1438,-56.3818,1588.66,-22.9653,-55.9863,2000,-30.4487,-
117.422,2517.85,-18.8821,-730.635,3169.78,-21.8737,-787.017,3990.53,-24.6525,-
1463.95,5023.77,-18.4109,-1565.38,6324.55,-25.0054,-2008.52,7962.14,-22.7759,-
2440.24,10023.7,-20.928,-2803.05,12619.1,-16.502,-2956.46,15886.6,-13.8047,-
3915.22,20000,-16.2083,-4335.03

Batch_Stereo_MLS_ImpulseRespSwp MLS Impulse Response (0 to 1.4 mS)
   Format = time, ch1 ampl, ch2 ampl, ...

ASCII,29,0,-94.241,-126.131,4.99674E-05,-92.249,-95.8485,9.99349E-05,-88.7337,-
101.383,0.000150065,-87.3276,-94.9472,0.000200033,-89.9596,-100.932,0.00025,-94.7049,-
98.5821,0.000299967,-85.7536,-113.465,0.000349935,-82.002,-99.0844,0.000400065,-86.5141,-
94.0417,0.000450033,-77.1634,-95.8418,0.0005,-75.8575,-104.003,0.000549967,-71.2409,-
105.351,0.000599935,-65.2917,-97.5596,0.000650065,-62.7128,-120.814,0.000700033,-
58.3055,-102.728,0.00075,-55.4971,-92.6842,0.000799967,-51.8528,-96.9397,0.000849935,-
46.7082,-95.5434,0.000900065,-22.7866,-103.329,0.000950033,-21.0122,-92.9669,0.001,-
38.9634,-111.02,0.00104997,-35.0987,-92.2705,0.00109993,-30.6643,-97.7428,0.00115007,-
36.9525,-99.766,0.00120003,-42.8994,-97.3776,0.00125,-48.7097,-93.3355,0.00129997,-
35.1105,-95.2276,0.00134993,-37.9154,-97.1298,0.00140007,-35.6044,-93.3993

Batch_Stereo_MLS_ImpulseRespSwp MLS Amplitude & Phase vs Frequency (20 Hz to 20 kHz)
   Format = freq, ch1 ampl, ch1 phase, ch2 ampl, ch2 phase, ...

ASCII,31,19.9986,-19.5868,23.6426,-76.6002,-86.0449,25.177,-18.9145,14.8535,-78.6424,-
116.807,31.7001,-18.9552,7.11914,-80.9549,-155.566,39.9055,-19.8828,0.966797,-82.122,-
229.966,50.2396,-20.7737,3.38379,-79.7932,-291.138,63.2458,-20.4206,7.64648,-77.6218,-
357.275,79.6223,-21.4742,8.39355,-79.4122,-403.682,100.239,-18.5006,37.4414,-84.3707,-
417.085,126.194,-14.2683,10.6348,-85.6948,-472.236,158.867,-12.5797,-0.351563,-84.7788,-
753.794,200.003,-12.2987,-42.1436,-88.6508,-1057.19,251.787,-20.2603,-88.9453,-96.1683,-
1300.78,316.978,-26.6668,-75.1025,-101.657,-1690.58,399.05,-24.9885,-25.9717,-82.2045,-
2272.28,502.378,-21.3902,-40.0781,-85.6353,-2489.15,632.458,-25.1499,-85.2539,-82.1653,-
3327.54,796.217,-35.6763,-85.957,-86.6817,-4236.11,1002.38,-28.441,-44.5605,-80.3717,-
4981.03,1261.92,-25.2672,-9.71191,-84.8277,-6205.43,1588.66,-31.2352,-102.085,-85.3097,-
7709.63,2000,-33.8366,-722.681,-77.1412,-8957.64,2517.85,-25.8695,-1142.09,-75.327,-
11038.2,3169.78,-30.0966,-1114.19,-80.7241,-13001.6,3990.53,-25.0704,-1474.19,-76.4034,-
14995.8,5023.77,-36.7425,-1979.25,-68.5362,-19595.8,6324.55,-26.1484,-2384.87,-69.0127,-
25783.7,7962.14,-28.3465,-2782.27,-79.8778,-32743.4,10023.7,-22.2809,-2884.04,-66.6194,-
40970.7,12619.1,-29.0938,-3442.06,-73.7138,-51780.2,15886.6,-18.5072,-4002.8,-66.7187,-
65930.4,20000,-29.7395,-4393.78,-71.2181,-82194.9
```

Figure 2-35. Visual Basic measurement results for subprograms MLS_ImpulseRespSwp, Batch_MLS_ImpulseRespSwp, and Batch_Stereo_MLS_ImpulseRespSwp

Loading New Firmware into the Instrument

The System Two GPIB interface provides the capability of downloading new firmware from the GPIB controller into the flash memory within the instrument. This simplifies the process of upgrading the firmware when new features become available from Audio Precision. Two GPIB commands are available to start the update process and restart the firmware. The process may be restarted if interrupted; however, the instrument will not be usable until firmware has been completely loaded and the processor has restarted. Thus, do not attempt to load new firmware into the instrument unless you have created a program that is known to work correctly. The CD ROM provided with this manual contains two executable programs that will download the firmware successfully. The “FTU.EXE” runs only on WIN95/98/NT and requires a National Instruments GPIB interface card with appropriate driver software. This program provides a full-featured Windows interface that monitors the progress of the firmware download and provides a progress monitor. The “FTU.EXE” program should be used whenever possible for downloading new firmware into the System Two. A detailed description of the “FTU.EXE” program is provided in Section 3, “Utility Programs,” in this manual.

Firmware Download Process

System Two GPIB goes through four separate stages in the process of downloading firmware. These stages are: 1) IDLE, 2) ERASE (old firmware), 3) TRANSFER (new firmware), and 4) START (new firmware).

IDLE Stage

In this stage the instrument will accept any valid GPIB commands and behave normally. The only GPIB command that is not valid in the IDLE stage is the GO command. The IDLE stage is indicated by a serial poll response of 0 (hex 00, binary 0000 0000).

ERASE Stage

The erase stage is started when the instrument receives the **`:FWUPDATE`** command with the correct numeric arguments (e.g. **`:FWUPDATE 0.01,2.001`**). The two arguments must be the current firmware boot code version number and the current firmware version number. The current firmware version number may be verified by inspecting the version number in the response to the *IDN? query. The firmware turns control over to the boot code when the correct **`:FWUPDATE`** command has been received. The boot code controls the process until the new firmware has been loaded and started (via the GO command).

The first action of the boot code is to erase the existing firmware. During the ERASE stage the serial poll response will be 2 (hex 02, binary 0000 0010). The rear panel LEDs next to the ADDRESS switch will slowly turn ON and OFF in a counter-clockwise direction during the ERASE stage.

TRANSFER Stage

When the ERASE stage is complete, all six GPIB LEDS will remain ON and the boot code will change the serial poll response to 134 (hex 86, binary 1000 0110). The file containing the new firmware may now be transferred via the GPIB interface to the instrument. During the firmware file transfer, the boot code will turn the six GPIB LEDs ON and OFF in a fast clockwise pattern (depending on the transfer rate).

START Stage

The boot code will verify that it has received the complete firmware file by inspecting the firmware data as it is received. When the complete firmware file has been received, the boot code will change the serial poll response to 128 (hex 80, binary 1000 0000), turn only one LED ON, and wait to receive the GO command to transfer control to the new firmware. When the GO command is received, the firmware takes control of the instrument, immediately resets the instrument to factory power on default settings., and returns to the IDLE stage.

Error Recovery

The instrument firmware cannot be started unless a download has been completed successfully (completion of the START stage). Control of the System Two is passed irreversibly to the boot code when the FWUP command has been received. The boot code is capable of recovering in the event of a power interruption or other failure to transfer the firmware file. The boot code will restart in the ERASE stage if the file transfer is interrupted or stopped while in the ERASE or TRANSFER stage. In order to recover, the application software must detect the ERASE or TRANSFER stages with a serial poll of the status byte register, compare the status byte with the values discussed above, and then re-start transfer of firmware data at the correct time.

VB Sample Code for GPIB Firmware Transfer - "miniFtu.bas"

A simplified version of the FTU.EXE program with no user interface, "miniFtu.bas", is also included on the CD ROM. Its source code, shown below, illustrates the firmware transfer process described above. You may use this as a starting point for developing your own download utility if you must use a different operating system, programming language, or GPIB controller interface.

(Start of Figure 2-36)

```
' System Two GPIB Sample Program for Windows
' miniFtu.bas, version 1.0, 23-Nov-1998
' Firmware Transfer Utility for System Two GPIB.
' Requires Microsoft Visual Basic 5.0 or later version.
' Requires National Instruments NI-488.2 GPIB Interface Board and WIN95/98/NT drivers.
' Requires National Instrument Visual Basic Interface Library files Niglobal.bas & Vbib-
      32.bas.
' Copyright (c) 1998, Audio Precision Inc.
' VB 5.0 Startup Object is Main()
Option Explicit
Const VERSION = 1#          ' version number of miniFTU
Const EOI_OFF = 0           ' don't send EOI with last byte of data
Const EOI_ON = 1            ' send EOI with last byte of data
Const IDLE_SPR = &H0        ' serial poll response when S2G idle
Const ERASE_SPR = &H2       ' serial poll response when S2G erasing
Const XFER_SPR = &H86       ' serial poll response when s2G downloading new firmware
Const DONE_SPR = &H80       ' serial poll response when download done, waiting for "GO"
Const ERASE_TIMEOUT = 20    ' max time (Secs) to wait for erase
Const GPIB_ADDR = 2         ' assumed GPIB address of S2G
Const TIME_OUT = 12         ' index of max time to wait for GPIB I/O
Const BLOCK_SIZE = 32767    ' size of block of data read from FW file and transferred
Const FW_FILE = "S2G.HEX"   ' name of firmware file
Const MSGBOX_TITLE = "Audio Precision System Two GPIB miniFTU"

Sub Main()
  Dim iSPR As Integer, iBoard As Integer, iS2G As Integer, iStatus As Integer
  Dim iNdx As Integer, iListener As Integer
  Dim l3rdComma As Long, lFileSize As Long
  Dim StartTime As Single, DelayStartTime As Single
  Dim sResponse As String, sFwVer As String, sBlock As String, sMsg As String
  Dim sBootVer As String, sFwFile As String, sPath As String

  iBoard = ilfind("GPIB0")      ' ilfind automatically places board on-line
  iStatus = ilsic(iBoard)       ' interface clear

  sPath = App.Path                    ' this next piece of code is because of a 'defect'
  If Right(sPath, 1) = "\" Then       ' in the path property.  If the current directory
      is
      sFwFile = sPath & FW_FILE       ' the root 'c:' then the path property has a
      trailing '\'.
  Else                                ' Any other directory the path does not have a
      sFwFile = sPath & "\" & FW_FILE ' trailing '\', so we need to add it.
  End If

  ' put up a message of assumed HW & SW configuration
  iStatus = MsgBox("To use this program, verify that..." & vbCrLf & vbCrLf & _
      "1) The System Two is in GPIB mode." & vbCrLf & _
      "2) The System Two is at GPIB address " & GPIB_ADDR & "." & vbCrLf & _
      "3) The firmware file to be transferred is called '" & FW_FILE & "'" & vbCrLf & _
      "4) and is located in the same directory as this utility." & vbCrLf & vbCrLf & _
      "To observe progress/activity, watch rear panel GPIB LED's.", _
      vbOKCancel + vbExclamation, MSGBOX_TITLE & ", Ver. " & VERSION)
  If iStatus = vbCancel Then Exit Sub                 ' if user clicked CANCEL then abort

  iS2G = ildev(0, GPIB_ADDR, NO_SAD, TIME_OUT, 1, 0)  ' get handle to GPIB_ADDR
  iStatus = illn(iS2G, GPIB_ADDR, NO_SAD, iListener)  ' is there a device there?
  If iListener = 0 Then                               ' no, flag error and abort
    iStatus = MsgBox("No Listener detected @ address " & GPIB_ADDR, vbOKOnly,
      MSGBOX_TITLE & "  Communication Error")
    Exit Sub
  End If

  ' first do a serial poll to see if bit 1 is set (indicates in update in process)
  iStatus = ilrsp(iS2G, iSPR)
  If (((iSPR And ERASE_SPR) <> ERASE_SPR) And ((iSPR And XFER_SPR) <> XFER_SPR)) Then
    ' S2G is idle
    sResponse = Space(100)
    sMsg = "*IDN?"
    iStatus = ilwrt(iS2G, sMsg, Len(sMsg))          ' what instrument is this?
    iStatus = ilrd(iS2G, sResponse, Len(sResponse)) ' get response
    If InStr(1, sResponse, "SYSTEM TWO") > 0 Then   ' is "SYSTEM TWO" in response?
      sResponse = Left(sResponse, ibcnt)            ' trim trailing chars from response
      buffer
      For iNdx = 1 To 3                             ' find 3rd comma
        l3rdComma = InStr(l3rdComma + 1, sResponse, ",", vbTextCompare)
      Next iNdx
      sFwVer = Trim(Right(sResponse, Len(sResponse) - l3rdComma)) ' grab the rest of the
      line after the 3rd comma
    Else
      iStatus = MsgBox("Could not find System Two at address ", vbOKOnly, MSGBOX_TITLE &
      "  ERROR MESSAGE")
      Exit Sub
    End If

    sMsg = "HEAD ON;FWUP?"                          ' make sure headers are on and get
      boot version
    iStatus = ilwrt(iS2G, sMsg, Len(sMsg))          ' write query for current boot
      version
    sBootVer = Space(100)
    iStatus = ilrd(iS2G, sBootVer, Len(sBootVer))   ' get reply
    sBootVer = Left(sBootVer, ibcnt)                ' trim excess chars from response -
      problem with NI
    If (Right(sBootVer, 1) = Chr$(10)) Then         ' if trailing "\n" trim it
      sBootVer = Left(sBootVer, Len(sBootVer) - 1)
    End If
    If (Right(sFwVer, 1) = Chr$(10)) Then           ' get firmware version from *IDN?
      response
      sFwVer = Left(sFwVer, Len(sFwVer) - 1)        ' if trailing "\n" trim it
    End If

    sMsg = sBootVer$ & "," & sFwVer                 ' build command to start download
      cycle
    iStatus = ilwrt(iS2G, sMsg, Len(sMsg))          ' send update command

    If (iStatus And &H8000) = &H8000 Then           ' if a command error, exit
      iStatus = MsgBox("Error in writing FWUPdate command.", vbOKOnly, MSGBOX_TITLE & "
      ERROR MESSAGE")
      Exit Sub                                      ' abort program
    End If

  End If
  ' at this time the instrument should be in download cycle; either erasing or waiting
      for transfer of new firmware
  StartTime = Timer()                               ' total process time
  Do                                                ' wait for erase cycle to complete,
      or timeout
    iStatus = ilrsp(iS2G, iSPR)                     ' perform serial poll
    DelayStartTime = Timer()                        ' these 4 statements are just
    Do While Timer() - DelayStartTime < 1           ' a manual delay
      DoEvents
    Loop
    iStatus = ilclr(iS2G)                           ' send device clear
  Loop While ((iSPR And XFER_SPR) <> XFER_SPR) And ((Timer() - StartTime) <
      ERASE_TIMEOUT)

  If ((iSPR And XFER_SPR) <> XFER_SPR) Then       ' not yet in xfer mode, we must have
      timed out
    sResponse = MsgBox("Serial Poll Response: " & "0x" & Hex(iSPR), vbOKOnly,
      MSGBOX_TITLE & "  Erase Cycle Timeout")
    Exit Sub                                    ' abort program
  End If

  ' the erase cycle has completed successfully,  the next stage is to transfer
  ' the new firmware, first we turn of EOI and send the file in blocks
  ' (size determined by BLOCK_SIZE).  When done, we turn EOI back on and send
  ' GO command to start new firmware.
  iStatus = ileot(iS2G, EOI_OFF)                  ' turn off EOI

  lFileSize = FileLen(sFwFile)                    ' get size of file

  If lFileSize <= 0 Then                          ' check that file is not empty
    MsgBox "Firmware file is empty", vbOKOnly, MSGBOX_TITLE & "  File Error"
    Exit Sub                                    ' abort program
  End If

  Open sFwFile For Input As #1                    ' open s-record file
  For iNdx = 1 To lFileSize \ BLOCK_SIZE          ' transferring any whole blocks
    sBlock = Input(BLOCK_SIZE, 1)               ' read upto blocksize bytes from #1
    iStatus = ilwrt(iS2G, sBlock, Len(sBlock))  ' write block to instrument

    If (iStatus And &H8000) = &H8000 Then
      Close
      MsgBox "Aborting transfer, status = " & "0x" & Hex(iStatus), vbOKOnly, MSGBOX_TITLE
      & "  GPIB Error"
      Exit Sub                                ' abort program
    End If

    iStatus = ilrsp(iS2G, iSPR)                 ' serial poll instrument
    DoEvents
  Next iNdx
  If (lFileSize Mod BLOCK_SIZE) > 0 Then          ' any partial block left over?
    sBlock = Input(lFileSize Mod BLOCK_SIZE, 1) ' read last partial block from #1
    iStatus = ilwrt(iS2G, sBlock, Len(sBlock))  ' write block to instrument
    If (iStatus And &H8000) = &H8000 Then
      Close                                   ' close s-record file
      MsgBox "Aborting transfer, status = " & "0x" & Hex(iStatus), vbOKOnly, MSGBOX_TITLE
      & "  GPIB Error"
      Exit Sub                                ' abort program
    End If
    iStatus = ilrsp(iS2G, iSPR)                 ' serial poll instrument
  End If
  Close                                           ' close s-record file
```

```
  iStatus = ileot(iS2G, EOI_ON)                   ' turn EOI back ON
  If ((iSPR And DONE_SPR) = DONE_SPR) Then        ' if serial poll = DONE
    iStatus = ilwrt(iS2G, "GO", 2)              ' tell instrument to reboot
  Else                                            ' an error must have occurred
    sResponse = MsgBox("Serial Poll Response:" & Str$(iSPR), vbOKOnly, MSGBOX_TITLE & "
      Xfer Cycle Error")
  End If
End Sub
```

Figure 2-36. Visual Basic miniFTU.bas source code

Loading New Firmware with the GPIB Talker-Listener Software

It is possible to load new instrument firmware into the System Two GPIB interface board using the Audio Precision GPIB Talker-Listener software provided on the CD ROM. A complete description of this program is provided in Section 3, “Utility Programs”, later in this manual.

Follow the sequence of steps below to load new firmware into the System Two with the Audio Precision GPIB Talker Listener software.

1. Make sure Auto Receive is ON
2. Turn OFF Auto Error Query
3. Enter **`*IDN?`** in the Send String text box and press the Enter key.

   Receive the response: **`AUDIO PRECISION,SYSTEM TWO,0,2.001`** in the Receive String text box.

   Highlight the last comma and argument (firmware version number) and press Ctrl C to copy this text to the Windows clipboard.

4. Enter **`FWUP?`** In the Send String text box and press the Enter key.

   4a. Receive the response: **`:FWUPDATE 0.01`** in the Receive String text box.

   4b. Put cursor at end of response and press Ctrl V to paste the firmware version as the second argument to the response.

   4c. Highlight whole response and press Ctrl C to copy the response to the Windows clipboard.

5. Click the mouse cursor in the Send String text box, the entire string should be selected.

   5a. Press Ctrl V to paste the clipboard into the Send String text box. The string should look something like: **`:FWUPDATE 0.01,2.001`**

   5b. Press the Enter key.

6. The System Two should start the ERASE stage.

   6a. The rear panel GPIB LEDs should cycle slowly counter clockwise.

   6b. When complete all 6 LEDs should light at same time.

7. With the cursor in the Send String text box, press Ctrl-F Ctrl-F to bring up the file browser.

   7a Navigate the file browser to the directory where the System Two GPIB *.hex firmware file is located.

7b Select the *.hex firmware file and click Open.

7c The Send String text box should show only the filename with its file path specification, for example **<C:\Program Files\FTU\S2G2001.hex>**.

7d Press the Enter key.

7e The System Two rear panel GPIB LEDs should cycle quickly clockwise.

7f When the firmware file transfer is complete only one LED will be ON continuously.

8. Enter **GO** in the Send String text box and press the Enter key.

8a The System Two rear panel GPIB LED should be ON, all other LEDs should be OFF.

8b The System Two relays should click as the instrument firmware starts and sets the hardware to factory default settings.

The new firmware has been loaded and started, and the instrument is ready to receive commands.

3. Utility Software

Audio Precision GPIB Talk & Listen Utility

This utility allows you to interact with the Audio Precision System Two/G via a convenient windows interface menu. The source code for this utility provides an example of how to implement programmable control of the Audio Precision System Two/G. This utility is optimized for control of the Audio Precision System Two/G, but supports communication with most other GPIB instruments as well.

This utility is coded to function only with instruments controlled with one GPIB controller board addressed as board GPIB0. If you need to control instrumentation on multiple buses, then you will need to make appropriate modifications to the source code.

This utility was written in Visual BASIC 5.0, and uses the following components

Microsoft Common Dialog Control 5.0 (SP2)

Microsoft Rich Textbox Control 5.0 (SP2)

Microsoft Tabbed Dialog Control 5.0 (SP2)

Microsoft Windows Common Controls 5.0 (SP2)

The interface consists of a main window with zero or more panels that communicate with GPIB instruments. An instrument panel can only communicate with one instrument. But it is possible to have more than one instrument panel communicate with the same instrument. However this must be done with caution because only one of the panels will display information about the current state of the instrument. The remaining panels that communicate with the same instrument *may* show stale query responses.

One of the capabilities of this utility is to "record" the current session. This is accomplished by enabling logging and either entering a comment, sending commands or queries, and receiving responses. When logging is enabled the user is presented with a file browser and must specify the log file name. Logging will continue until disabled. The log file is an ASCII text file. All entries except comments will be on a single line. A comment can span multiple lines.

There are three headers inserted into the log file to indicate comments, send data, and receive responses.

A comment will start with:

-COMMENT-<space>

where <space> is the space character (ASCII code 32 decimal).

The line that contains commands/queries sent to the instrument will start with:

<addr>:SEND>>

where <addr> is the GPIB address of the instrument.

Responses from the instrument will start with:

<addr>:READ<<

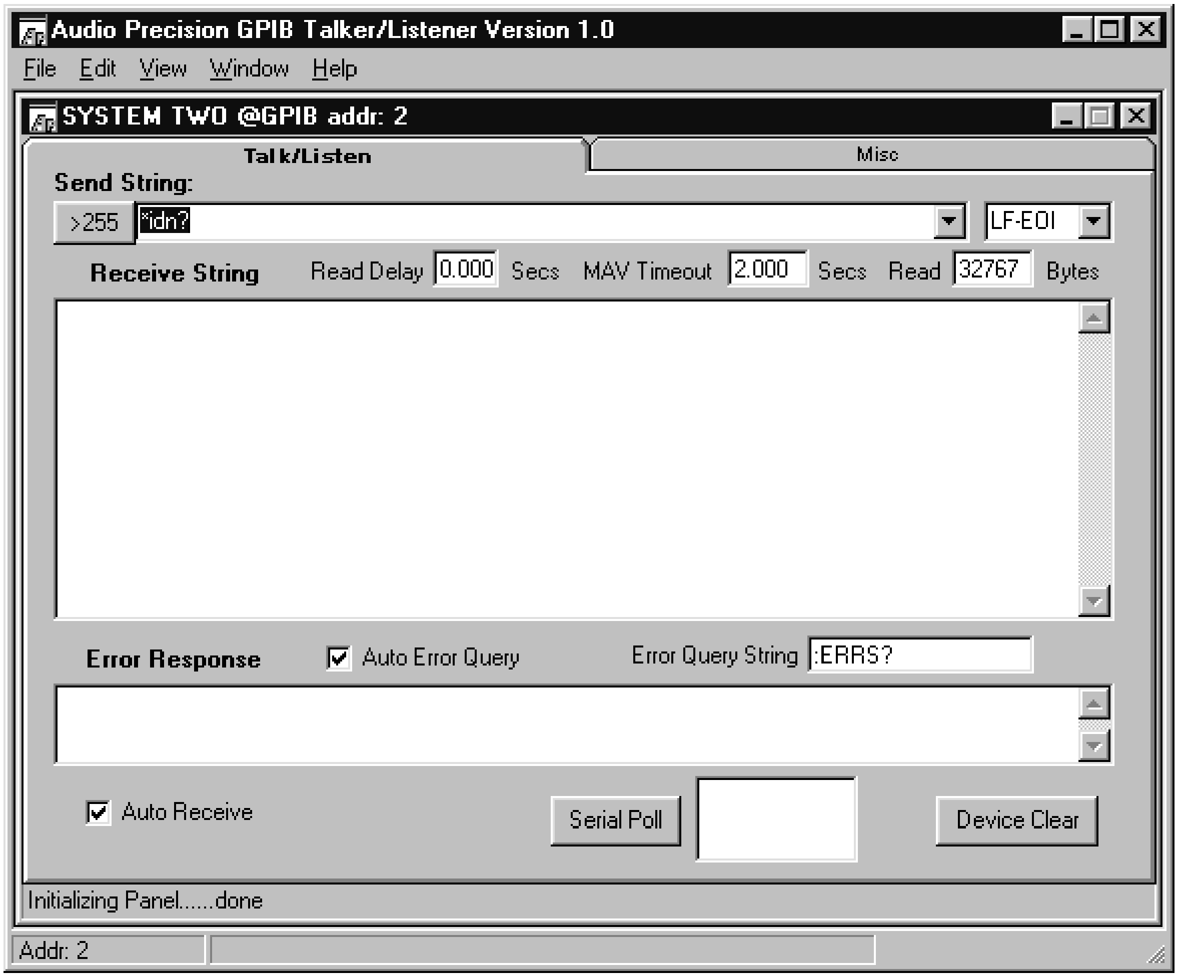
There are many panels, each with multiple controls (text boxes, check boxes, list boxes, etc.).

The basic process for using this utility is to enter commands and queries into the Send String control on the instrument panel, press <Enter>, and observe the results in the Receive String control. The reading of responses and checking for errors is automatic by default.

The above process will work for all Audio Precision GPIB instruments (System One, System Two, Portable One family and ATS-1 family). You may desire to have direct control over the sending of commands and reading of responses. Therefore, it is possible to disable the automatic features of this utility and exercise manual control over sending of commands and receiving responses.

Main Panel

The menu bar for the main panel provides control of the settings and actions that are non-instrument specific. For example, the File menu presents choices to start and stop data logging, insert a comment into the log file and exit the program.



Send String: *idn? LF-EOI

Read Delay 0.000 Secs MAV Timeout 2.000 Secs Read 32767 Bytes

☑ Auto Error Query Error Query String :ERRS?

☑ Auto Receive

Initializing Panel......done

Addr: 2

Figure 3-1. GPIB Main Panel

Menu Bar

File Menu

- ***Start/Stop Log*** – Initially this menu item is shown as Start Log. Once logging has been started this menu item changes to Stop Log.
- ***Comment*** – This menu item is enable when logging has been started. Selecting this menu item will display the Comment Panel. Text entered into the Comment text box will be appended to the end of the log file.
- ***Exit*** – Exit the utility.

Edit Menu

- ***Copy*** – Copy the highlighted portion of the active text control to the clipboard.
- ***Paste*** – Paste the contents of the clipboard to the currently active control. If any portion or all of the active control contents are highlighted, the highlighted portion will be replaced (overwritten) by the contents of the clipboard.
- ***GPIB Settings*** – Display the GPIB Bus Config Panel. The only GPIB setting that can be controlled from the Config Panel is the GPIB Bus Timeout.
- ***Preferences*** – At the present time this menu item is still under development and has been disabled.

View Menu

- ***Config*** - Check all GPIB addresses (1 - 30) to determine what instruments are currently on the bus. However, even if the configuration of instruments has changed, the visible panels will remain. You must delete unattached visible panels if you have removed instruments from the bus and invoke valid panels from the configuration panel.

Window Menu - this menu is only displayed when there are one or more instrument panels visible.

- ***Arrange Icons*** - Instrument panels may be displayed as an icon (minimized) or at their normal size. There may be more than one panel displayed but covered by another panel. If the main panel is enlarged or maximized, it is possible to see more than one panel at a time (depending on display resolution).
- ***Instrument Panels*** – When the Window menu item is selected, a list of the currently open instrument panels is displayed (immediately under the Arrange Icons item). Clicking on one of the entries will put the focus on that panel (and bring it to the front).

Help Menu

- ***About*** – This menu presents an “About” panel that shows the application name, version, disclaimer, and copyright notice.

Status Bar

- ***Current Instrument Address*** – The first pane in the status bar indicates the GPIB address of the instrument receiving commands and sending responses.
- ***Status messages*** – This pane provides general status messages about utility operation.

Comment Panel

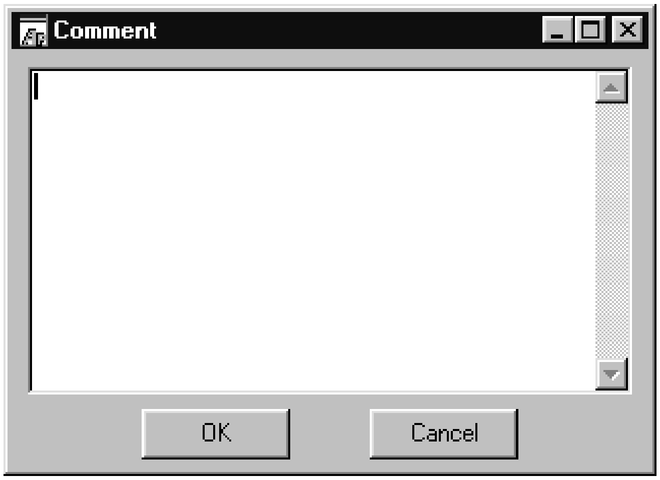
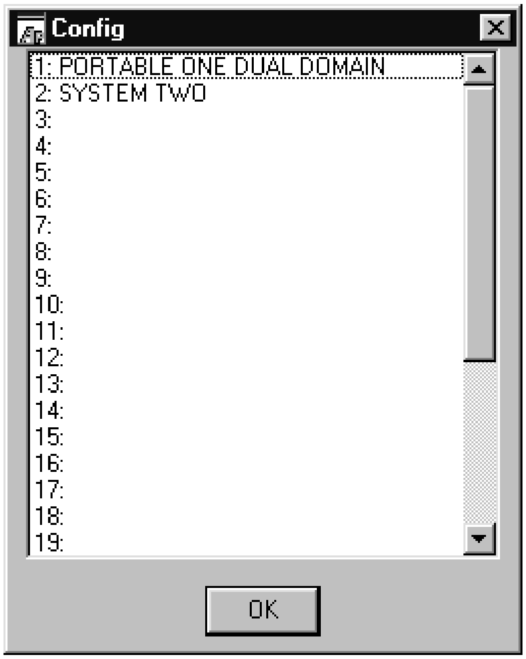


Figure 3-2. Comment panel

- ***Comment*** – Enter your log-file comments here (see Figure 3-2). The comment will be written to the log file when the OK button has been activated, either by clicking it with the mouse or by moving to the control with the tab key and pressing <Enter>.
- ***OK*** – Activating this control will cause the contents of the Comment text control to be written to the log file. The user comment will be preceded by "—COMMENT—" in the log file.
- ***Cancel*** – This will return to the main/instrument panel without writing a comment to the log file.

Config Panel

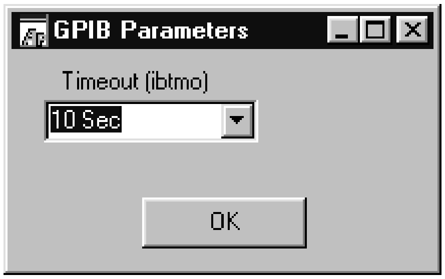

Config

1: PORTABLE ONE DUAL DOMAIN
2: SYSTEM TWO
3:
4:
5:
6:
7:
8:
9:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:

Figure 3-3. Config panel (typical)

- ***Config*** – Select the instrument/GPIB address for which a panel will be displayed. See Figure 3-3 for a typical display.
- ***OK*** – Display the instrument panel for the currently selected address. If there was no instrument detected at that address, a generic panel will appear. If an instrument panel is already available for the selected address, another panel will be displayed. While it is possible to have more than one panel for any given address, this should be done with caution.

GPIB Settings Panel

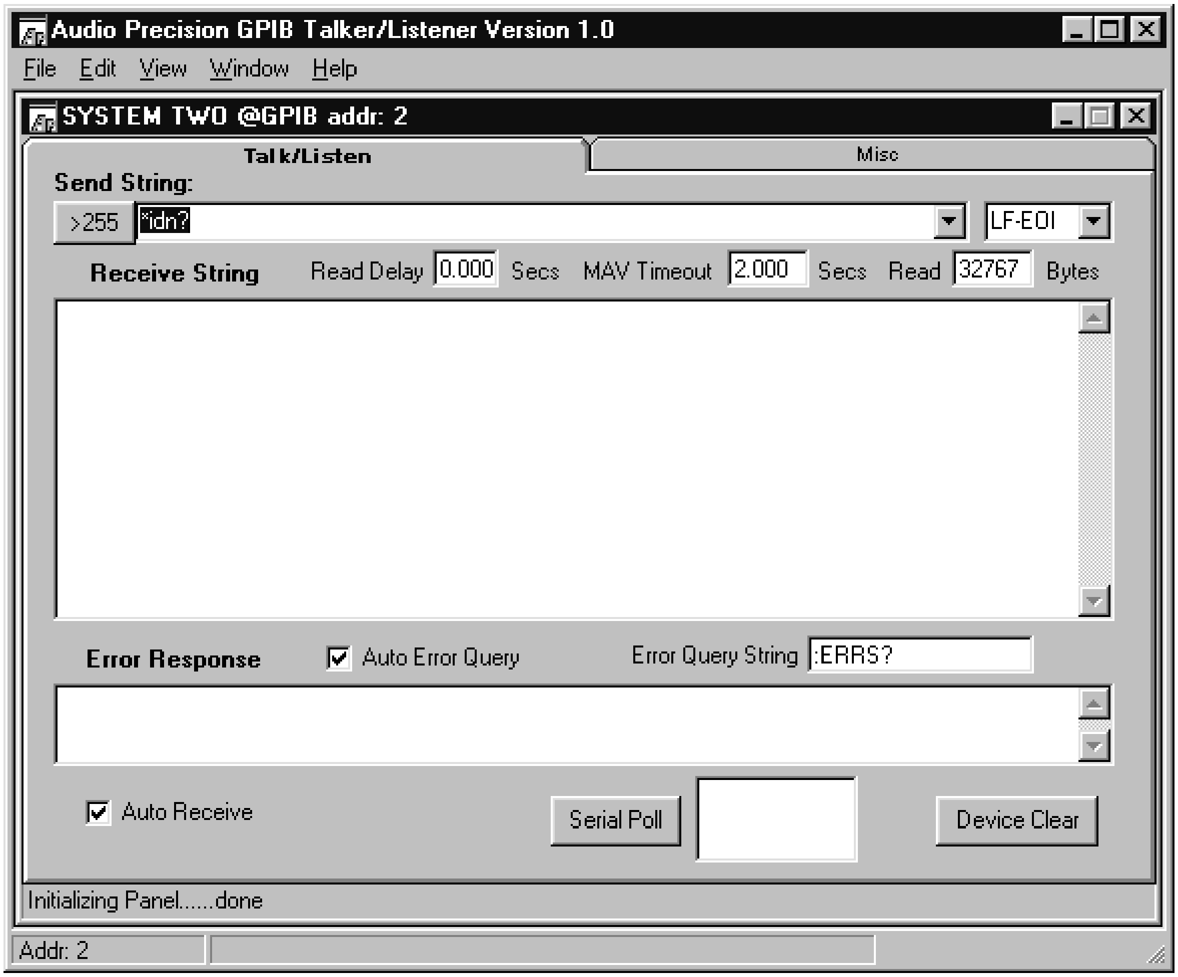

GPIB Parameters

Timeout (ibtmo): 10 Sec

Figure 3-4. GPIB Parameters (Timeout) Panel

- ***Timeout*** – See Figure 3-4. This control determines the GPIB timeout for this bus (GPIB 0). You may wish to change this if the default 10 seconds is too long. An instrument will time out if the Receive button is selected but no query has been sent.
- ***OK*** – This button will make the current setting(s) active.

Instrument Panel(s)



Send String: *idn? LF-EOI

Read Delay 0.000 Secs MAV Timeout 2.000 Secs Read 32767 Bytes

☑ Auto Error Query Error Query String :ERRS?

☑ Auto Receive

Initializing Panel......done

Addr: 2

Figure 3-5. GPIB Main Panel with System Two instrument panel

- ***>255*** – This button will display a panel that supports command strings greater than 255 characters. The Send String control on the instrument panel retains a history of previously commands executed but limits the length of each command line to 255 characters. To send a command string that is longer than 255 characters the you must either click this button or enter ^g (control-g) in the Send String control. Commands and queries sent to the instrument using this edit box will not appear in the Send String history.
- ***Send String*** – This control is the primary means of entering ASCII command strings to send to an instrument. The contents of this control can be one or more instrument commands or a filename reference. A referenced file will have its contents sent to the instrument in place of the filename reference. Consequently, the referenced file must contain command(s), queries, or portions thereof that will be syntactically correct in the context of the other data in the Send String control.
- **^a** (Select All) – This control key sequence will cause all of the contents of the Send String control to be selected (highlighted). This makes it is easy to copy or paste over the contents.
- **^f^a** (Insert Arb Block File Reference) – This control key sequence will insert a reference to a file containing arbitrary block data. If you have already

entered the arb block header (indicated by the presence of a '#' in the previous 7 chars) an arbitrary block header will not be created. However, if an arb block header has not been specified then one will be inserted based on the file size.

- **^f^f** (Insert File Reference) – This control key sequence will insert a file reference into the Send String control. This file reference must contain ASCII data that forms valid commands and/or queries in the context of the other ASCII characters in the Send String control.
- **^g** (Send String > 255) – The Send String control maintains a history of the commands & queries that have been sent to the associated instrument. However, each line has a maximum length of 255 characters. Sometimes it is desirable or necessary to send a longer string to the instrument. Control-g performs the same function as clicking the >255 button on the instrument panel.
- **<Enter>** – Acts on the contents of the Send String control and sends the contents to the instrument.

Terminator

- **No Term** – No Program Message terminator is sent.
- **LF Only** – The Program Message terminator is a linefeed (ASCII decimal 10).
- **LF-EOI** – The Program Message terminator is a linefeed with EOI asserted.
- **EOI Only** – The Program Message terminator is EOI asserted with the last character of the Program Message.

Receive String

- **^a** (Select All) – This control key sequence will select all of the data in the Receive String control. This is usually done in conjunction with ^c (control-c) or the File-Copy menu item to copy the contents of the Receive String control into the clipboard.
- **^s** (Search) – This control key sequence starts the search mode. In search mode the utility will find the first occurrence (from the starting cursor position) that matches the specified search string. When the user types ^s, a message is displayed in the instrument panel status bar that indicates the current search string. Initially the search string is empty. As the user types in the characters of the search string, the utility will highlight the first occurrence of the search string. At any time the user stops entering new characters in the search string and types ^s, the utility will locate the next occurrence of the search string. If the end of the Receive String is reached a message is display stating the search string was not found. Another ^s will then cause the utility to start searching at the beginning of the Receive String control.
- **^f^a** (Receive Arb Block Data) – This control key sequence displays a file browser. Select a file or enter a new filename. This utility will talk address the instrument, receive the response string, extract the first arbitrary block data that it finds, and copy the data into the file you specified. Use this to acquire arbitrary block data messages from the instrument that cannot be viewed as ordinary ASCII text with the Receive Data control. This utility will inspect the string looking for an arbitrary block data response message. If it finds one then it will extract the data using the arbitrary block byte count and copy the

data to the file you specified. The remainder of the data is discarded. If no arbitrary block data is found, then the response message is discarded. Use this only if you have sent a query that will return a response containing an arbitrary block data message unit.

- **^f^f** (Receive Data) - This control key sequence displays a file browser. Select a file or enter a new filename. This utility will talk address the instrument, receive the response string, and copy the response string into the file you specified. Use this to copy complete instrument setup commands by first sending the *LRN? query and then this Receive Data function. This is useful for data logging large response messages.

Read Delay – The time delay (in seconds) between writing the contents of the Send String control and the time an automatic read is performed. This setting is valid only if Auto Receive mode is enabled.

MAV Timeout –The length of time the utility will wait for the instrument to assert MAV (Message AVailable) before timing out. This setting is valid only if Auto Receive mode is enabled.

Read (Bytes) – The number of bytes to read from the instrument. This setting will be used when the instrument does a read (either manual or Auto Receive mode). Typically, if this is less than the number of characters the instrument is expected to return, the Auto Receive is disabled.

Error Response – Display the instruments response to the Error Query String. This can happen either in Auto Receive mode with Auto Error Query enabled, or when the Error button has been pressed when the utility is not in Auto Receive mode,

Auto Error Quer*y* – Specify whether the utility should automatically query the instrument for errors using the Error Query String. This setting is valid only if Auto Receive mode is enabled.

Error Query String – The query used to interrogate the instrument for any outstanding errors. This query may result in one or more error messages. The response to this string is displayed in the Error Response text control.

Auto Receive – Specify whether the utility should attempt to interpret the commands/queries sent to the instrument to determine if there is an expected response. If there is an expected response the utility will wait and flag a MAV timeout if no response is detected. The actions that will cause a read to be attempted are; MAV bit asserted, “?” in the Send String, or an empty Send String. The later is necessary for those situations where the Read (Bytes) size is less than the length of the instrument response. This allows the user to control how much of the string will be read (leaving the remainder of the response unread).

Serial Poll – Serial poll the instrument and display the result. The result will be displayed in three formats (decimal, hexadecimal and binary). The serial poll display will also be updated whenever the utility performs a serial poll to determine the status of the instrument when the instrument panel is in Auto Receive mode.

Device Clear – Send a selective device clear (SDC) bus message to the instrument.

Status Ba*r* – The instrument panel status bar displays various messages about the current actions of the utility on this specific instrument. The type of messages includes how many data bytes were sent, how many data bytes were read, error messages, timeout messages, etc. When using LF-EOI termination the count of types sent or read will be one more that the number of bytes in either the Send String control or the Receive String control. This extra byte the LF (linefeed) that is automatically appended (Send) or removed (Receive).

Send String >255 Characters Panel

Send String – This control serves the same function as the Instrument Panel Send String control. This control responds to the same inputs as the Instrument Panel Send String control with the exception of <Enter> and ^g. The primary advantage of this control is that it is not limited to the 255 characters as is the Instrument Panel Send String control.

- **^a** (Select All) – same as ^a for Instrument Panel Send String control.
- **^f^a** (Insert Arb Block File Reference) – same as ^f^a for Instrument Panel Send String control.
- **^f^f** (Insert File Reference) – same as ^f^f for Instrument Panel Send String control.

Send – Send the contents of the Send String to the instrument,

Cancel – Close this panel without sending the contents of the Send String control to the instrument.

System Two GPIB Firmware Transfer Utility

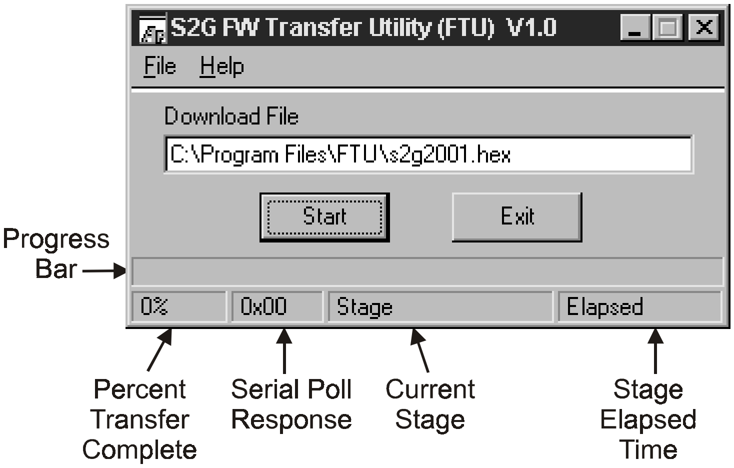
This utility transfers new firmware from a file supplied by Audio Precision into any of the System Two models with the GPIB option. Use this utility to download new versions of firmware as they become available from Audio Precision. Please note that the instrument firmware already loaded into the instrument will be erased before the new firmware is downloaded. The CD ROM provided with this manual contains the current version of the firmware that was downloaded into your System Two when it was shipped from the factory.

Please see Section 2 “Loading New Firmware into the Instrument” for a detailed description of the firmware download process and warnings about interrupting the process. If the process is not completed successfully the instrument will be inoperable.

Hardware Requirements

The FTU software from Audio Precision operates only with Windows 95/98/NT and requires a National Instruments GPIB interface board with software drivers for Windows 95/98/NT.

Main Panel Description



Download File

C:\Program Files\FTU\s2g2001.hex

Figure 3-6. GPIB firmware transfer utility main panel

The main panel of the GPIB firmware transfer utility is shown in Figure 3-6. This panel includes indicators, controls and menu items that allow you to control file selection and GPIB parameters, and view the firmware transfer status.

Panel Indicators

Download File - This field shows the path and filename of the file selected for download to the System Two. This indicator is located just above the Start and Exit buttons.

Progress Bar - This indicator shows how much of the firmware file that has been downloaded relative to the size of the file. The progress bar is located immediately above the status bar.

Status Bar Indicators

- **Percent Transfer Complete** – This status bar indicator shows the same information as the Progress Bar in percentage of the firmware file transferred.
- **Serial Poll Response** – This status bar indicator displays the result of the most recent instrument serial poll. The software uses the serial poll response to determine when the instrument has completed one stage and is waiting for the next stage.
- **Current Stage** – This status bar indicator shows the current stage of the download process indicated by the serial poll response. If the instrument has valid firmware and is not executing any instructions, this indicator will display “Idle.” As the instrument progresses through the three stages of loading new firmware this indicator will show “Erase,” then “Transfer,” and finally “Start.” When the new firmware has been started, this control will show “Idle.”
- **Stage Elapsed Time** – This status bar indicator shows how much time has elapsed in the current stage (except idle stage). Typically, the Erase and Start elapsed times are too short to be visible in the status bar. The Transfer stage elapsed time may be under 60 seconds for faster computers to more than 90 seconds for slower computers.

Menu Bar

File

- **Open** – This menu entry is used to select the file containing the instrument firmware. The firmware file will have a ".hex" extension. The file name will start with S2G followed by a four or five digit version number. The version number has 1 (for a 4-digit version number) or 2 digits (for a 5-digit version number) before the (implied) decimal point and 3 digits after the decimal point. It is possible to download any version firmware file. It is also possible to select a file by double-clicking on the Download File text control or by typing ^f (control-f) when focus is on the Download File control.
- **Properties** – This menu presents a panel that allows the user to change the GPIB address, timeout and transfer block size. The default settings are
  - **GPIB Address** - 2
  - **Timeout** - 3 seconds
  - **Block Size** - 32767
- **Exit** – Clicking this button exits the program

Help

- **About FTU** – This menu item presents information about the application, including version number, copyright and disclaimer.

Panel Controls

Start – The Start button is only enabled when a file has been selected. Once a file has been selected the Start button is enabled. Once the Start button has been pressed, the firmware download process cannot be stopped. The instrument will not respond to any of the commands listed in the Programmers Reference Manual until new firmware is installed and started.

Exit – The Exit button will exit the program. It is enabled whenever the firmware download process is not running.

Firmware Download Sequence Description

See “Loading New Firmware into the Instrument” in Section 2 for a complete description of the firmware download sequence.

4. System Commands

System commands apply to general communications, common system attributes (IEEE-488.2), system properties, and instrument status. The commands in this section are listed alphabetically within these groupings: common commands and queries (preceded by an asterisk *), APSTatus commands and queries, then other commands and queries.

*CLS

Clears the Standard Event Status register, the Status Byte register, and the AP Event Status Register. Clears the Unread Query Response if received in a subsequent PMT-terminated message.

Related Commands: *ESR?, *STB?, :APSTatus:AESR?

Command Syntax: *CLS

Example: `*CLS`

*DDT

Defines a series of instrument commands that are executed whenever a group execute trigger (GET) or a *TRG command is received.

This command definition is reset whenever a *RST is received. The instrument commands must be contained in an arbitrary block format. The *DDT macro definition may not contain any of the following commands (or the equivalent query forms, if applicable): *DDT, *DMC, *PMC, *RMC, *TRG or any macro labels.

The following command will clear the DDT command definition (where <PMT> is NL^END or ^END):

*DDT #10<PMT>	definite length format
*DDT #0NL^END	indefinite length format

The command argument is:

- **data** - <arbitrary block program data>

Related Commands: *RST, *TRG, *DDT?

Command Syntax: *DDT **data**

Example1: `*DDT #230:AGEN:AMPL A,2V;:ANLR:LEV? A,V`

Example2: `*DDT #0:AGEN:AMPL A,2V;:ANLR:LEV? A,V<PMT>`

*DDT?

Returns the command sequence to be executed when a GET or *TRG is received. The response will always be a definite length arbitrary block. The data contained in the arbitrary block will always be ASCII characters.

Response argument(s):

•**data** - <definite length arb block data>

Related Commands: *DDT

Command Syntax: *DDT?

Response Syntax: **data**

Example: *DDT?

Response 1: `#230:AGEN:AMPL A,2V;:ANLR:LEV? A,V`

Response 2: `#10` *`if not defined`*

*DMC

Assigns a sequence of zero or more instrument commands to a macro label. Macros are invoked by sending the macro label to the instrument. The following rules apply to macro labels and definitions:

- Macro labels may consist of any alphanumeric characters (ASCII decimal 48 - 57, 65 - 90, and 97 -122), and the underscore character (ASCII decimal 95) with the provision that the first character must be alpha.
- Valid macro label characters may be "0" - "9", "A" - "Z", "_", and "a" - "z". Macro labels are not case sensitive (lower and upper case are synonymous).
- The macro definition (data) may not include the commands: *DDT, *DMC, *PMC, *RMC, *TRG or macro labels (macros cannot invoke other macros).
- Place holders, for parameters passed to the macro during execution, are designated by $n, where n is 1 for the first passed parameter, 2 for the second passed parameter, etc. Place holders must appear as complete program data elements. A maximum of 9 place holders (1 - 9) may be used within a macro definition.

To illustrate the last point, the following sequence of commands enables the analog generator outputs, sets the frequency to 3000 Hz, channel A output to 1 volt, and channel B output to 2 volts:

```
:AGEN:OUTPUT AB;FREQ 3000HZ;AMPL
A,1V;AMPL B,2V
```

A macro that implements this series of commands would be:

```
*DMC "SETAGEN",#243:AGEN:OUTPUT AB;FREQ
$1;AMPL A,$2;AMPL B,$3
```

The macro invocation would be:

```
:SETAGEN 3E3HZ,1V,2V
```

The same macro might be used at some other time to set generator frequency to 1000 Hz and both amplitudes to 5DBV:

```
:SETAGEN 1000HZ,5DBV,5DBV
```

Macros may be deleted with either the *PMC or *RMC command. A macro cannot be redefined but may be deleted and then created again with a new definition.

The command arguments are:

- **label** - <string data>
- **data** - <arbitrary block program data>

Note that the arbitrary block program data must be terminated by a program message terminator, as indicated in Example 2 below. For further information, refer to Sections 1 and 2.

Related Commands: *DDT, *EMC, *EMC?, *GMC?, *LMC, *PMC, *RMC, *TRG

Command Syntax: *DMC **label, data**

Example 1:

```
*DMC "SETAGEN",#243:AGEN:OUTPUT AB;FREQ
$1;AMPL A,$2;AMPL B,$3
```

Example 2:

```
*DMC "SETAGEN",#0:AGEN:OUTPUT AB;FREQ
$1;AMPL A,$2;AMPL B,$3<PMT>
```

*EMC

Enables and disables all macro execution. An argument of 0 disables expansion of all macro labels. A non-zero argument that rounds to an integer value in the range of -32767 to +32767 enables macro execution. The default is 0, macro expansion disabled.

The command argument is:

- **enable** - <nr1> (range: ≥ -32767, ≤ 32767)

Default: 0

Related Commands: *DMC, *EMC?, *GMC?, *LMC, *PMC, *RMC

Command Syntax: *EMC **enable**

Example:

```
*EMC 1
```

*EMC?

Returns the state of macro execution capability. A response of 0 indicates macros are disabled. A response of 1 indicates macro execution is enabled.

Response argument(s):

•**emc** - <nr1> (0 or 1)

Related Commands: *EMC

Command Syntax: *EMC?

Response Syntax: **emc**

Example: `*EMC?`

Response: `1`

*ESE

Sets the bits in the Standard Event Status Enable Register. The range of values is 0 to 255. In the example the top 4 bits are set high and the lower 4 bits are set low.

The command argument is:

•**bitfield** - <nr1> (range: ≥ 0, ≤ 255)

Default: 0

Related Commands: *ESE?

Command Syntax: *ESE **bitfield**

Example: `*ESE 240`

*ESE?

Returns the contents of the Standard Event Status Enable Register.

Response argument(s):

•**bitfield** - <nr1>

Related Commands: *ESE

Command Syntax: *ESE?

Response Syntax: **bitfield**

Example: `*ESE?`

Response: `240`

*ESR?

Returns the contents of the Standard Event Status Register. This is a destructive read that clears the register. In the example response the 2nd LSB is set (bit 1).

Response argument(s):

- **bitfield** - <nr1>

Related Commands: *ESE, *ESE?

Command Syntax: *ESR?

Response Syntax: **bitfield**

Example: `*ESR?`

Response: `2`

*GMC?

Returns the current definition of a macro in definite length arbitrary block format.

The command argument is:

- **label** - <string data>

Response argument(s):

- **macro** - <definite length arb block data>

Related Commands: *DMC, *EMC, *EMC?, *LMC, *PMC, *RMC

Command Syntax: *GMC? **label**

Response Syntax: **macro**

Example: `*GMC? "SETAGEN"`

Response:

```
#243:AGEN:OUTPUT AB;FREQ $1;AMPL A,$2;AMPL
B,$3
```

*IDN?

Returns the manufacturer, product model, null serial number, and firmware version.

Response argument(s):

- **idn** - <arbitrary ASCII response data>

Related Commands: <None>

Command Syntax: *IDN?

Response Syntax: **idn**

Example: `*IDN?`

Response: `AUDIO PRECISION,SYSTEM TWO,0,1.001`

*LMC?

Returns the currently defined macro labels. If there are no defined macros labels then the response will consist of an empty quoted string (i.e. two consecutive double quotes).

Response argument(s):

•**macro_label** - <string data>

Related Commands: *DMC, *EMC, *EMC?, *GMC?, *PMC, *RMC

Command Syntax: *LMC?

Response Syntax: **macro_label, macro_label ...**

Example: `*LMC?`

Response1: `"MACRO1","MACRO2", ...`

Response2: `""    if no defined macros`

*LRN?

Returns the sequence of commands that define the current state of the instrument. This string will restore the instrument settings to the same state at a later time. Arbitrary waveforms, defined macros, *DDT definition, and acquisition waveforms within the DSP are not included in the response. The length of the string may be greater than 4 kBytes depending on the specific command settings, hardware configuration, installed options, and VERBOSE setting.

Response argument(s):

•**settings** - <response message unit> [<response message unit>] ...

Related Commands: :SET?

Command Syntax: *LRN?

Response Syntax: **settings**

Example: `*LRN?`

Response:

```
:DIN:REF 48000;BANDWIDTH 700;RESOLUTION
24;DETECTOR AVG;MODE ACTIVE;PKMODE
HALF;SCALEFREQBY MEASURED;DEEMPHASIS
OFF;FORMAT XLR;IMPEDANCE BNC,Z75;IMPEDANCE
XLR,Z110;:DOUT:FORMAT XLR...
```

*OPC

Sets the operation complete bit (bit 0) in the Standard Event Status Register when encountered in the input queue.

Use this behavior to indicate when commands sent prior to the *OPC have been executed. Set the OPC bit (bit 0) in the Event Status Enable register with *ESE 1 to enable this OPC event to set the Event Status Bit in the Status Byte Register. A serial poll of the Status Byte Register will read the Status Byte in order to check the value of the ESB bit that indicates an OPC event has occurred.

Related Commands: *OPC?

Command Syntax: *OPC

Example: `*OPC`

*OPC?

Returns a 1 in the output queue when encountered in the input queue. Use this behavior to indicate that commands sent prior to the OPC? have been executed (assuming none of the commands generated a response).

Response argument(s):

•**opc** - <nr1>

Related Commands: *OPC

Command Syntax: *OPC?

Response Syntax: **opc**

Example: `*OPC?`

Response: `1`

*OPT?

Returns the list of installed instrument options. The response consists of comma delimited fields containing numeric codes described in the table below when the corresponding option is installed. The response code will be 0 if an option is not installed.

Field No.	Response Code	Option Installed
0	Filter ID	Filter Slot 1
1	Filter ID	Filter Slot 2
2	Filter ID	Filter Slot 3
3	Filter ID	Filter Slot 4
4	Filter ID	Filter Slot 5
5	Filter ID	Filter Slot 6
6	Filter ID	Filter Slot 7
7	1	Analog Generator
8	1	Analog Analyzer
9	1	Digital Signal Processing
10	1	Digital I/O
11	1	Analog Intermodulation Distortion, IMD
12	1	Burst Generator, BUR
13	1	Wow & Flutter, W&F
14	1	DCX-127 Multifunction Module
15	1	SWR-122 Switcher Module(s), one or more

See **Appendix D - Optional Filters** for the list of Filter ID numbers.

The model number of System Two may be determined by inspecting the combinations of responses for fields 1 through 4, shown in the table below.

System Two Model	Field 7 Analog GEN	Field 8 Analog ANLR	Field 9 DSP	Field 10 DIO
SYS-2022	1	1	0	0
SYS-2222	1	1	1	0
SYS-2322	1	1	1	1
SYS-2300	0	0	1	1

Therefore, a SYS-2022 is a System Two, with Analog Generator and Analog Analyzer. A SYS-2300 is a System Two with DSP and DIO but no analog circuitry.

Response argument(s):

- **opt1 -** <nr1>
- **opt2 -** <nr1>
- **opt3 -** <nr1>
- **opt4 -** <nr1>
- **opt5 -** <nr1>
- **opt6 -** <nr1>
- **opt7 -** <nr1>
- **opt8 -** <nr1>
- **opt9 -** <nr1>
- **opt10** - <nr1>
- **opt11** - <nr1>
- **opt12** - <nr1>
- **opt13** - <nr1>
- **opt14** - <nr1>
- **opt15** - <nr1>
- **opt16** - <nr1>

Related Commands: <None>

Command Syntax: *OPT?

Response Syntax: **opt1, opt2, opt3, opt4, opt5, opt6, opt7, opt8, opt9, opt10, opt11, opt12, opt13, opt14, opt15, opt16**

Example: `*OPT?`

Response: `33,17,49,0,58,98,0,1,1,1,1,0,0,0,1,1<PMT>`

*PMC

Deletes all current macro definitions and macro labels.

Related Commands: *DMC, *EMC, *EMC?, *GMC?, *LMC?, *RMC

Command Syntax: *PMC

Example: `*PMC`

*RCL

Restores the instrument to an instrument setup created with the *SAV command. Nine complete instrument settings (numbered 1 - 9) may be saved with the *SAV command and recalled with this command. The scope of this command is the same as *LRN?, *SAV, and *RST.

*RCL deletes all arbitrary generator waveforms stored in volatile memory by the :DGEN:ARBLOAD command.

*RCL 0 resets the instrument to power on state. See appendix B for more information.

The command argument is:

- **setting** - <nr1> (range: ≥ 0 , ≤ 9)

Related Commands: *SAV

Command Syntax: *RCL **setting**

Example: `*RCL 5`

*RMC

Removes the specified macro label and macro definition from memory. Use of a macro label removed by this command will cause a command error. The command argument is a quoted macro label string.

The command argument is:

- **label** - <string data>

Related Commands: *DMC, *EMC, *EMC?, *GMC?, *LMC?, *PMC

Command Syntax: *RMC **label**

Example: `*RMC "MACRO1"`

*RST

Sets the instrument to factory default settings (defined for each command in the line labeled Default) with the following exceptions:

1. Will not affect the output queue.
2. Will not affect the Event Enable Register, Event Register, AP Event Enable Register, or AP Event Register.
3. Will not affect any macro definitions.
4. Will not affect any stored settings.
5. Will not affect any arbitrary waveform buffers.

The scope of this command is the same as *LRN?, *RCL, and *SAV.

Related Commands: <None>

Command Syntax: *RST

Example: `*RST`

*SAV

Saves the current settings of the instrument in a local volatile memory register, replacing any previous settings saved in that register. These settings may be recalled later with the *RCL command (using the same argument used with this command). The scope of this command is the same as *LRN?, *RCL, and *RST. The range of valid arguments is 1 – 9.

The command argument is:

- **setting** - <nr1> (range: ≥ 1, ≤ 9)

Related Commands: *RCL

Command Syntax: *SAV **setting**

Example: `*SAV 5`

*SRE

Sets the Service Request Enable Register bits. The valid range of values for the argument is 0 through 255. However, since Bit 6 is always 0 the corresponding query (*SRE?) will only return values from 0 through 63 and 128 through 191.

The command argument is:

- **bitfield** - <nr1> (range: ≥ 0, ≤ 255)

Default: 0

Related Commands: *SRE?

Command Syntax: *SRE **bitfield**

Example: `*SRE 16`

*SRE?

Returns the current setting of the bits in the Service Request Enable Register. The range of responses is 0 through 63 and 128 through 191 (because Bit 6 is always 0).

Response argument(s):

- **bitfield** - <nr1>

Related Commands: *SRE

Command Syntax: *SRE?

Response Syntax: **bitfield**

Example: `*SRE?`

Response: `16`

*STB?

Returns the contents of the Status Byte Register. Bit 6 is the Master Summary Status bit. Bit 6 is not read if a serial poll is performed.

Response argument(s):

•**bitfield** - <nr1>

Related Commands: <None>

Command Syntax: *STB?

Response Syntax: **bitfield**

Example: `*STB?`

Response: `16`

*TRG

This command is the equivalent of the GET (Group Execute Trigger) message. *TRG triggers the actions defined by the *DDT command. *TRG may not be included in a macro definition.

Related Commands: *DDT

Command Syntax: *TRG

Example: `*TRG`

*TST?

Causes the instrument to perform a self-test and return a result code indicating what failures occurred. A return value of 0 (zero) indicates that no failures occurred (instrument passed self-test). A non-zero return value indicates which test failed.

The *RST command should be sent to the instrument upon completion of self-test to ensure that the instrument is in a known state.

The current instrument functionality tested by *TST is:

- Scan Filter ID values

Response argument(s):

- **tst** - <nr1>

Related Commands: <None>

Command Syntax: *TST?

Response Syntax: **tst**

Example: `*TST?`

Response: `0`

*WAI

Causes execution of overlapped commands to be delayed until the no-operation-pending flag is off (TRUE). This forces all overlapped commands to execute in a sequential mode. At this time System Two has no overlapped commands, therefore this command is reserved for future implementations. No action will be taken when this command is executed.

Related Commands: <None>

Command Syntax: *WAI

Example: `*WAI`

APSTatus

This set of commands sets or reads the AP Event Status Register and the AP Event Status Enable Register. The AP Event Status Enable Register provides a bit mask for specifying the status bits in the AP Event Status Register that will set the AESB bit in the Status Byte Register. These are 16 bit registers with the highest bit unused. The compound header for this set of commands is :APSTATUS:

The event definitions are:

Bit	Status Definition
15	NOT USED (forces response to be > 0)
14	Reserved
13	Reserved
12	Reserved
11	Reserved
10	Reserved
9	Reserved
8	Macro Execution Complete
7	Reserved
6	Reserved
5	Transform complete
4	Acquisition complete, Transforming
3	Trigger received, Acquiring
2	Waiting for Trigger
1	Settled Reading Timeout
0	Reserved

:APSTatus:ENABle

Sets the bits in the AP Event Status Enable Register. The range of values is 0 through 32767. This register determines which events are enabled to set bit 0 (0x01) in the Status Byte Register. An AP event may generate an SRQ (Service Request) if the appropriate bits are set in the AP Event Status Enable Register and the Standard Event Status Register. In the example below only Bit 5 has been set TRUE to enable the completion of a DSP Transform to generate an SRQ.

The command argument is:

•**bitfield** - <nr1> (range: ≥ 0, ≤ 32767)

Default: 0

Related Commands: :APSTatus:EVENt, :APSTatus:ENABle?

Command Syntax: :APSTatus:ENABle **bitfield**

Example: `:APSTATUS:ENABLE 32`

:APSTatus:ENABle?

Returns the contents of the AP Event Status Enable Register. The range of response values is 0 through 32767. The example indicates that Bit 5 is set to allow Service Requests whenever the DSP completes a transform.

Response argument(s):

•**bitfield** - <nr1> (range ≥ 0, ≤ 32767)

Related Commands: :APSTatus:ENABle

Command Syntax: :APSTatus:ENABle?

Response Syntax: :APSTatus:ENABle **bitfield**

Example: `:APSTATUS:ENABLE?`

Response: `:APSTATUS:ENABLE 32`

:APSTatus:EVENt?

Returns the contents of the AP Event Status Register. The range of response values is 0 through 32767. In the example, Bits 5 is set indicating the DSP has completed a transform.

Response argument(s):

•**bitfield** - <nr1> (range = 0, = 32767)

Related Commands: :APSTatus:ENABle, :APSTatus:ENABle?

Command Syntax: :APSTatus:EVENt?

Response Syntax: :APSTatus:EVENt **bitfield**

Example: `:APSTATUS:EVENT?`

Response: `:APSTATUS:EVENT 32`

:DELay

Causes command execution to be delayed for the specified period of time. Any commands that have already been started will continue to process. This delay period may be aborted with either a SDC (selective device clear) or DCL (device clear).

The command argument (in seconds) is:

•**time** - <nrf> (range: ≥ 0, ≤ 1E34)

Default: 0

Related Commands: <None>

Command Syntax: :DELay **time**

Example: `:DELAY 100`

:ERRMessage?

Returns the oldest error in the error queue (FIFO). The error queue will hold the first 16 errors. A complete list of error numbers is provided in Appendix C.

Response argument(s):

- **modulenum** - <nr1>
- **errnum** - <nr1>
- **errmsg** - "<error text>"

Related Commands: :ERRN?, :ERRS?

Command Syntax: :ERRMessage?

Response Syntax: :ERRM **modulenum,errnum,errmsg**

Example: `:ERRMESSAGE?`

Response(with verbose off)

```
:ERRM 505,13,":AGEN:FREQ, AGEN, BELOW
MINIMUM FREQUENCY."
```

Response(with verbose on)

```
:ERRMSG 505,13,":AGEN:FREQ, AGEN, BELOW
MINIMUM FREQUENCY."
```

:ERRN?

Returns the number of errors in the error queue (0 to 16). When an error queue overflow occurs the last error message will be replaced with the error message "Too Many Errors", but the error count will not change until at least one error message is read or the error queue is cleared.

Response argument(s):

- **numerrors** - <nr1> (range: ≥ 0, ≤ 16)

Related Commands: :ERRM?, :ERRS?

Command Syntax: :ERRN?

Response Syntax: :ERRN **numerrors**

Example: `:ERRN?`

Response: `:ERRN 0`

:ERRS?

Generates one error response for each error in the error queue (max of 16 errors) in FIFO order.

Response argument is:

- **response** - error[; error[; error...]]

Where error is:

- error - modulenum,errnum,errmsg

And error parameters are:

- modulenum - <nr1>
- errnum - <nr1>
- errmsg - "<<command header> | SYSTEM>, [<module name>,] <error text>"

Related Commands: :ERRM?, :ERRN?

Command Syntax: :ERRS?

Response Syntax: **response**

Example:

```
:ERRS?
```

Response:

```
:ERRS 502,7,":AGEN:FREQ, ILLEGAL PARAMETER
TYPE.";502,6,":ANLR:INPUT, NOT ENOUGH
PARAMETERS -OR- MISSING UNIT SUFFIX."
```

:FWUPdate

Initiates the reprogramming of the instrument flash ROM. The firmware is provided by Audio Precision.

The original version of the instrument firmware is factory installed and also provided on the CD ROM provided with this manual (in case this version needs to be reinstalled in the future).

The instrument goes through four stages during this reprogramming. The instrument responds to a serial poll with a specific status byte for each stage of the process according to the table below.

Stage	Action	Status Byte(s) Hex
1	Erasing Flash ROM	0x02
2	Transferring new firmware	0x86 or 0xC6
3	Transfer Complete (waiting for GO)	0x80 or 0xC0
4	Idle (normal operation), waiting for commands	0x00

When this command is received the instrument will erase all instrument firmware except for that portion stored in the boot sector. During the erase stage, a serial poll of the instrument will return a status byte of 0x02.

If the instrument power is cycled after the start of the erase cycle the instrument will no longer respond to any GPIB commands.

After the flash ROM has been erased the instrument will assert an SRQ, wait to be listen addressed, and expect to receive the new firmware. For this stage the status byte will be either 0x86 or 0xC6. The two possible values for the status byte depend on whether the RQS bit (bit 6) is set. If SRQ is asserted, the RQS bit is set in the status byte (0xC2). When the first serial poll (after entering stage 2) is performed, the SRQ is removed and the RQS bit is turned off (0x82).

The preferred method of downloading the firmware is to transfer the firmware file from hard disk using a computer program to control the GPIB data transfer. See the Utility Programs section of this manual for more information. The serial poll response will be either 0xC6 (RQS bit on) or 0x86 (RQS bit is off) during the transfer stage.

When the transfer of new firmware is complete the instrument will go to stage 3 (Transfer Complete) and the status byte will change to 0xC0 or 0x80. The instrument will remain in this state until it receives the "GO" command The instrument will reboot using the new firmware when the “GO” command is received.

The FWUPdate command arguments are: the boot sector version number (the FWUPdate? query response), and the current firmware version number (part of the *IDN? query response). The argument values must be exactly the same as the version numbers in the FWUPDATE? and *IDN? responses, otherwise an error will result and the firmware update process will not start.

The command arguments are:

- **boot_ver** - <nrf> (range: ≥ 0, ≤ 1E34)
- **fw_ver** - <nrf> (range: ≥ 0, ≤ 1E34)

Related Commands: *IDN?, GO, :FWUPdate?

Command Syntax: :FWUPdate **boot_ver, fw_ver**

Example: `:FWUPDATE 1.01, 1.001`

:FWUPdate?

Returns the current boot firmware version number. This version number must be used as the first argument to the FWUPdate command to initiate the firmware download process.

The response argument is:

- **boot_ver** - <nrf> (range: $\geq$ 0, $\leq$ 1E34)

Related Commands: :FWUPdate

Command Syntax: :FWUPdate?

Response Syntax: :FWUPdate **boot_ver**

Example: `:FWUPDATE?`

Response: `:FWUPDATE 1.01`

GO

Tells the boot firmware to start the newly downloaded firmware. The firmware update process consists of 3 steps: erasing flash ROM, transferring new firmware, and starting new firmware. After the new firmware has been successfully transferred, it may be started by sending the GO command.

This command will cause a command error except when sent at the end of the firmware update process (when the serial poll response is either 0x80 or 0xC0).

Related Commands: :FWUPdate

Command Syntax: GO

Example: `GO`

:HEADer

Specifies whether instrument queries return data with a command header (HEADER ON) or with no header(HEADER OFF).

The command argument is:

- **state** - { ON | OFF }

Default: ON

Related Commands: :HEADer?

Command Syntax: :HEADer **state**

Example: `:HEADER ON`

:HEADer?

Returns the setting of the header flag.

Response argument(s):

- **state** - { ON | OFF }

Related Commands: :VERBose, :HEADer

Command Syntax: :HEADer?

Response Syntax: :HEADer **state**

Example: :HEADER?

Response: `:HEADER ON`

:T1

Sets the GPIB Bus T1 delay for the instrument GPIB interface. This delay provides a means of controlling the GPIB bus settling time for multiline messages. When the GPIB has long cable lengths and/or many instruments on the bus, it is possible that the data being placed on the bus may take longer than normal to settle to a steady state. In that case, the T1 setting may need to be increased from its nominal setting (500 nSecs). Valid T1 delay settings are 350 nSec (T350), 500 nSecs (T500), 1100 nSecs (T1K) or 2000 nSecs (T2K).

The command argument is:

- **delay** - { T350 | T500 | T1K | T2K }

Default: T500

Related Commands: :T1?

Command Syntax: :T1 **delay**

Example: `:T1 T350`

:T1?

Returns the T1 delay setting.

The response argument is:

•**delay** - { T350 | T500 | T1K | T2K }

Related Commands: :T1

Command Syntax: :T1?

Response Syntax: :T1 **delay**

Example: :T1?

Response: `:T1 T350`

:VERBose

Sets query responses to either verbose (ON) or terse (OFF) mode. In terse mode the responses will be the first three of four characters of the command and arguments. In the command descriptions in this section the terse form of each command and argument is shown in upper case on the Syntax line. In verbose mode the full commands and arguments are returned in response to queries.

The command argument is:

•**state** - { ON | OFF }

Default: ON

Related Commands: :VERBose?

Command Syntax: :VERBose **state**

Example: `:VERBOSE ON`

:VERBose?

Returns the setting of the verbose flag.

The query argument is:

•**state** - { ON | OFF }

Related Commands: :VERBose

Command Syntax: :VERBose?

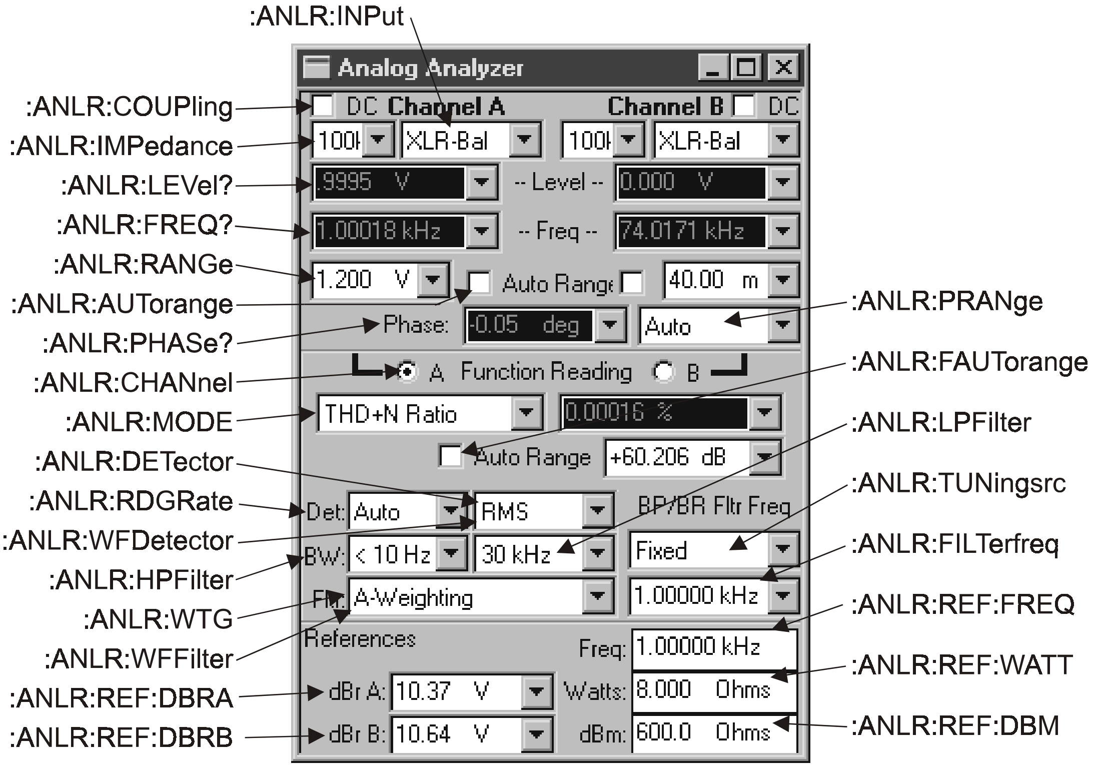
Response Syntax: :VERBose **state**

Example: `:VERBOSE?`

Response: `:VERBOSE ON`

5. Analog Analyzer Commands

The header path for the analog analyzer is :ANLR:.



Analog Analyzer

□ DC Channel A — Channel B □ DC

100k XLR-Bal — 100k XLR-Bal

.9995 V -- Level -- 0.000 V

1.00018 kHz -- Freq -- 74.0171 kHz

1.200 V □ Auto Range □ 40.00 m

Phase: -0.05 deg — Auto

● A Function Reading ○ B

THD+N Ratio — 0.00016 %

□ Auto Range +60.206 dB

Det: Auto — RMS — BP/BR Fltr Freq

BW: < 10 Hz — 30 kHz — Fixed

Fltr: A-Weighting — 1.00000 kHz

References — Freq: 1.00000 kHz

dBr A: 10.37 V — Watts: 8.000 Ohms

dBr B: 10.64 V — dBm: 600.0 Ohms

Figure 5-1. Analog Analyzer control panel showing relationship with GPIB commands

:ANLR:AUTorange

Enables/disables autoranging of the input attenuators and gain amplifiers for the indicated channel. The valid channel selections are channel A (A), channel B (B), or channels A and B (AB).

The valid settings for autoranging are OFF and ON. If the user issues the range command (:ANLR:RANGe) for either (or both) channel(s), autoranging will be turned off for the specified channel(s).

The command arguments are:

- **channel -** { A | B | AB }
- **state -** { OFF | ON }

Default: AB, ON

Related Commands: :ANLR:AUTorange?, :ANLR:RANGe

Command Syntax: :ANLR:AUTorange **channel**, **state**

Example: `:ANLR:AUTORANGE A,OFF`

:ANLR:AUTorange?

Returns the autorange setting for the indicated channel. The valid responses are OFF and ON.

Query argument:

- **channel -** { A | B }

Response argument(s):

- **channel -** { A | B }
- **state -** { OFF | ON }

Related Commands: :ANLR:AUTorange

Command Syntax: :ANLR:AUTorange? **channel**

Response Syntax: :ANLR:AUTorange **channel**, **state**

Example: `:ANLR:AUTORANGE? A`

Response: `:ANLR:AUTORANGE A,OFF`

:ANLR:CHANnel

Specifies which input channel to connect to the function meter. The possible inputs are channel A or channel B.

The command argument is:

- **channel -** { A | B }

Default: A

Related Commands: :ANLR:CHANnel?

Command Syntax: :ANLR:CHANnel **channel**

Example: `:ANLR:CHANNEL A`

:ANLR:CHANnel?

Returns the current function meter channel input.

Response argument(s):

- **channel -** { A | B }

Related Commands: :ANLR:CHANnel

Command Syntax: :ANLR:CHANnel?

Response Syntax: :ANLR:CHANnel **channel**

Example: `:ANLR:CHANNEL?`

Response: `:ANLR:CHANNEL A`

:ANLR:COUPling

Sets the coupling for the indicated channel(s). The valid settings for coupling are AC or DC.

The command arguments are:

- **channel -** { A | B | AB }
- **coupling -** { AC | DC }

Default: AB, AC

Related Commands: :ANLR:COUPling?

Command Syntax: :ANLR:COUPling **channel**, **coupling**

Example: `:ANLR:COUPLING A,AC`

:ANLR:COUPling?

Returns the coupling setting for the indicated channel.

The valid responses are AC and DC.

The command argument is:

- **channel -** { A | B }

Response argument(s):

- **response_channel -** { A | B }
- **coupling -** { AC | DC }

Related Commands: :ANLR:COUPling

Command Syntax: :ANLR:COUPling? **channel**

Response Syntax: :ANLR:COUPling **response_channel**, **coupling**

Example: `:ANLR:COUPLING? A`

Response: `:ANLR:COUPLING A,AC`

:ANLR:DETector

Sets the detector type for the function meter for all measurement functions except Wow & Flutter.

The command arguments are:

- **type -** { RMS | AVG | PEAK | QPEak | SPEak }

Default: RMS

Related Commands: :ANLR:DETector?

Command Syntax: :ANLR:DETector **type**

Example: `:ANLR:DETECTOR PEAK`

:ANLR:DETector?

Returns the detector type for the function meter.

Response argument(s):

- **type -** { RMS | AVG | PEAK | QPEak | SPEak }

Related Commands: :ANLR:DETector

Command Syntax: :ANLR:DETector?

Response Syntax: :ANLR:DETector **type**

Example: `:ANLR:DETECTOR?`

Response: `:ANLR:DETECTOR PEAK`

:ANLR:FAUTorange

Enables or disables the function meter autoranging. If the function meter range command (:ANLR:FRANge) is received, the function meter autoranging will be turned OFF.

The command argument is:

- **state -** { ON | OFF }

Default: ON

Related Commands: :ANLR:FRANge, :ANLR:FAUTorange?

Command Syntax: :ANLR:FAUTorange **state**

Example: `:ANLR:FAUTORANGE ON`

:ANLR:FAUTorange?

Returns the state of the function meter autoranging. The possible states are ON and OFF.

Response argument(s):

- **state -** { ON | OFF }

Related Commands: :ANLR:FAUTorange

Command Syntax: :ANLR:FAUTorange?

Response Syntax: :ANLR:FAUTorange **state**

Example: `:ANLR:FAUTORANGE ON`

:ANLR:FILTerfreq

Sets the frequency of the tunable bandpass or bandreject filter. The tuning source (:ANLR:TUNingsrc) will be forced to fixed (FIXed) when this command is received.

The command argument is:

- **frequency -** <nrf> (range: ≥ 9.9, ≤ 204775 HZ) { HZ | F_R | DHZ | PCTHz | CENT | OCTS | DECS | DPCT | DPPM }

Default: 1000HZ

Related Commands: :ANLR:TUNingsrc, :ANLR:FILTerfreq?

Command Syntax: :ANLR:FILTerfreq **frequency**

Example: `:ANLR:FILTERFREQ 1000HZ`

:ANLR:FILTerfreq?

Returns the current frequency setting of the analog analyzer bandpass or bandreject tuned filter. The setting will depend on the setting of the analog analyzer tuning source. For example, if the function meter is set to bandpass and the tuning source is set to AGEN (:ANLR:MODE BP;TUN AGEN) then this query will return the filter frequency set to the AGEN generator frequency.

If the TUNingsrc is changed to FIXed, the filter will be held to that frequency even though subsequently the generator may change frequency. The filter frequency may be directly set with the :ANLR:FILTerfreq command when the tuning source is fixed (:ANLR:TUN FIXed).

The command argument is:

- **unit -** { HZ | F_R | DHZ | PCTHz | CENT | OCTS | DECS | DPCT | DPPM }

Response argument(s):

- **frequency -** <nrf>{ HZ | F_R | DHZ | PCTHz | CENT | OCTS | DECS | DPCT | DPPM }

Related Commands: :ANLR:FILTerfreq, :ANLR:TUNingsrc

Command Syntax: :ANLR:FILTerfreq? **unit**

Response Syntax: :ANLR:FILTerfreq **frequency**

Example: `:ANLR:FILTERFREQ? HZ`

Response: `:ANLR:FILTERFREQ 1000HZ`

:ANLR:FRANge

Sets the function meter gain range. The range will be determined based on the chart below. The argument will be rounded down to the next lower range setting (with a maximum of 1024 X/Y or +60.206 dB). For example, an argument of 10DB will set the range to 0DB, 25DB will set the range to 24.082DB, 1000DB will set the range to +60.206DB, etc.

X/Y	dB
1	0
4	+12.041
16	+24.082
64	+36.124
256	+48.165
1024	+60.206

This command will set the the function meter autoranging mode to OFF (:ANLR:FAUTorange OFF).

The command argument is:

- **range -** <nrf> (range: see table above) { X_Y | DB }

Default: 16X_Y

Related Commands: :ANLR:FRANge?, :ANLR:FAUTorange

Command Syntax: :ANLR:FRANge **range**

Example: `:ANLR:FRANGE 256X_Y`

:ANLR:FRANge?

Returns the current function meter gain range setting.

The command argument is:

- **units -** { X_Y | DB }

Response argument(s):

- **range -** <nrf> { X_Y | DB }

Related Commands: :ANLR:FRANge, :ANLR:FAUTorange

Command Syntax: :ANLR:FRANge? **units**

Response Syntax: :ANLR:FRANge **range**

Example: `:ANLR:FRANGE? X_Y`

Response: `:ANLR:FRANGE 256X_Y`

:ANLR:FREQ?

Returns a frequency measurement from channel A or B in the specified units. The available units are Hertz, F/R, dHz, %Hz, cent, octs, decs, delta %, and delta PPM.

The first parameter in the response is the next available measurement.

The frequency meter settling is enabled or disabled by the algorithm parameter of the :SETTLING:ANLR command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

The command arguments are:

- **channel -** { A | B }
- **units -** { HZ | F_R | DHZ | PCTHz | CENT | OCTS | DECS | DPCT | DPPM }

Response argument(s):

- **freq -** <nrf> { HZ | F_R | DHZ | PCTHz | CENT | OCTS | DECS | DPCT | DPPM }
- **settle_timeout -** <nr1>

Related Commands: **:**SETTling:ANLR

Command Syntax: :ANLR:FREQ? **channel**, **units**

Response Syntax: :ANLR:FREQ **freq, settle_timeout**

Example: 1: `:ANLR:FREQ? A,HZ`

Response 1: `:ANLR:FREQ 11180.5HZ,1` *`(settling timed out)`*

Response 2: `:ANLR:FREQ 10871.2HZ,0` *`(no settling or measurement settled)`*

:ANLR:FUNCmeter?

Returns a measurement from the Analog Analyzer Function Meter in the specified units. The table below shows which units are legal for each function selected with the MODE command. An error will be generated if a unit is selected that is invalid for the current function mode.

The function meter has the following modes: Amplitude (AMPLitude), Bandpass (BP), Bandreject (BR), THD+N Amplitude (THDampl), THD+N Ratio (THDRatio), SMPTE (SMPTe), CCIF (CCIF), DIM (DIM), Wow & Flutter (WF), crosstalk (XTALk), DFD (DFD) and 2 Channel Ratio (RATio).

Function Mode	Valid Units
AMPLitude | BP | BR | THDampl	V | DBU | DBV | DBRA | DBRB | DBGA | DBGB | DBM | W
CCIF | DIM | SMPTe | THDRatio| WF | RATio | XTALk | DFD	DB | PCT | PPM | X_Y

The first parameter in the response is the next available measurement.

The function meter settling is enabled or disabled by the algorithm parameter of the :SETTLING:ANLR command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

The command argument is:

- **units -** { see table above }

Response argument(s):

- **rdg -** <nrf> { unit from table above }
- **settle_timeout -** <nr1>

Default: none

Related Commands: :ANLR:DETector, :ANLR:MODE, :SETTling:ANLR

Command Syntax: :ANLR:FUNCmeter? **units**

Response Syntax: :ANLR:FUNCmeter **rdg, settle_timeout**

Example: `:ANLR:FUNCMETER? V`

Response1: `:ANLR:FUNCMETER 0.534687V,1` *(settling timed out)*

Response2: `:ANLR:FUNCMETER 1.41306V,0` *(no settling or measurement settled)*

:ANLR:HPFilter

Sets the function meter high pass filter. There are four possible filter settings: <10 Hz (F10), 22.4 Hz (F22), 100 Hz (F100), and 400 Hz (F400).

If the function mode is SMPTE, CCIF, DIM, or Wow & Flutter then this setting will be preserved and used when the mode is changed to any one of the other functions.

The command argument is:

- **frequency -** { F10 | F22 | F100 | F400 }

Default: F10

Related Commands: :ANLR:HPFilter?, :ANLR:MODE

Command Syntax: :ANLR:HPFilter **frequency**

Example: `:ANLR:HPFILTER F100`

:ANLR:HPFilter?

Returns the high pass filter frequency setting. The query responses are F10, F22, F100, and F400.

Response argument(s):

- **frequency -** { F10 | F22 | F100 | F400 }

Related Commands: :ANLR:HPFilter

Command Syntax: :ANLR:HPFilter?

Response Syntax: :ANLR:HPFilter **frequency**

Example: `:ANLR:HPFILTER?`

Response: `:ANLR:HPFILTER F100`

:ANLR:IMPedance

Specifies the input impedance of the indicated channel(s). The possible argument values are: 300 ohms, 600 ohms, and high impedance (100k ohms).

The command arguments are:

- **channel -** { A | B | AB }
- **impedance -** {ZHI | Z300 | Z600 }

Default: ZHI

Related Commands: :ANLR:IMPedance?

Command Syntax: :ANLR:IMPedance **channel**, **impedance**

Example: `:ANLR:IMPEDANCE A,ZHI`

:ANLR:IMPedance?

Returns the input impedance of the indicated channel. The command argument is:

- **channel -** { A | B }

Response argument(s):

- **response_channel -** { A | B }
- **impedance -** {ZHI | Z300 | Z600 }

Related Commands: :ANLR:IMPedance

Command Syntax: :ANLR:IMPedance? **channel**

Response Syntax: :ANLR:IMPedance **response_channel**, **impedance**

Example: `:ANLR:IMPEDANCE? A`

Response: `:ANLR:IMPEDANCE A,ZHI`

:ANLR:INPut

Sets the input source for the indicated channel(s). The potential input sources are: XLR-Balanced, BNC-Unbalanced, and Generator Monitor.

The command arguments are:

- **channel -** { AB | A | B }
- **source -** { XLR | BNC | GENMon }

Default: AB, XLR

Related Commands: :ANLR:INPut?

Command Syntax: :ANLR:INPut **channel**, **source**

Example: `:ANLR:INPUT A,XLR`

:ANLR:INPut?

Returns the current input source of the indicated channel. The possible responses are: XLR-Balanced, BNC-Unbalanced, and Generator Monitor.

The command argument is:

- **channel -** { A | B }

Response argument(s):

- **response_channel -** { A | B }
- **source -** { XLR | BNC | GENMon }

Related Commands: :ANLR:INPut

Command Syntax: :ANLR:INPut? **channel**

Response Syntax: :ANLR:INPut **response_channel**, **source**

Example: `:ANLR:INPUT? A`

Response: `:ANLR:INPUT A,XLR`

:ANLR:LEVel?

Specifies the Analog Analyzer Level Meter channel and response units and returns the level meter reading in the indicated units.

The first parameter in the response is the next available measurement.

The level meter settling is enabled or disabled by the algorithm parameter of the :SETTLING:ANLR command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

The command arguments are:

- **channel -** { A | B }
- **units -** { DBGA | DBGB | DBM | DBRA | DBRB | DBU | DBV | V | W }

Response argument(s):

- **level -** <nrf> { DBGA | DBGB | DBM | DBRA | DBRB | DBU | DBV | V | W }
- **settle_timeout -** <nr1>

Related Commands: :SETTling:ANLR

Command Syntax: :ANLR:LEVel? **channel, units**

Response Syntax: :ANLR:LEVel **level, settle_timeout**

Example: `:ANLR:LEVEL? A,DBGA`

Response1: `:ANLR:LEVEL -11.5DBRA,1` *(settling timed out)*

Response2: `:ANLR:LEVEL 5.82109DBGA,0` *(no settling or measurement settled)*

:ANLR:LPFilter

Sets the function meter low pass filter frequency. There are four possible filter frequency settings: 22400 Hz (F22K), 30000 Hz (F30K), 80000 Hz (F80K), and 500000 Hz (F500k).

The command argument is:

- **frequency -** {F500k | F22K | F30K | F80K }

Default: F500k

Related Commands: :ANLR:LPFilter?

Command Syntax: :ANLR:LPFilter **frequency**

Example: `:ANLR:LPFILTER F30K`

:ANLR:LPFilter?

Returns the current function meter low pass filter frequency setting.

Response argument(s):

- **frequency -** {F500k | F22K | F30K | F80K }

Related Commands: :ANLR:LPFilter

Command Syntax: :ANLR:LPFilter?

Response Syntax: :ANLR:LPFilter **frequency**

Example: `:ANLR:LPFILTER?`

Response: `:ANLR:LPFILTER F30K`

:ANLR:MODE

Sets the function meter measurement mode. The possible settings are: amplitude, bandpass, bandreject, CCIF, DFD, DIM, SMPTE, THD+N amplitude, THD+N ratio, 2-channel, wow & flutter, and crosstalk.

The command argument is:

- **func -** { AMPLitude | BP | BR | CCIF | DFD | DIM | SMPTe | THDampl | THDRatio | RATio | WF | XTALk }

Default: AMPLitude

Related Commands: :ANLR:MODE?

Command Syntax: :ANLR:MODE **func**

Example: `:ANLR:MODE AMPLITUDE`

:ANLR:MODE?

Returns the function meter measurement mode.

Response argument(s):

- **func -** { AMPLitude | BP | BR | CCIF | DFD | DIM | SMPTe | THDampl |THDRatio | RATio | WF | XTALk }

Related Commands: :ANLR:MODE

Command Syntax: :ANLR:MODE?

Response Syntax: :ANLR:MODE **func**

Example: `:ANLR:MODE?`

Response: `:ANLR:MODE AMPLITUDE`

:ANLR:PHASe?

Returns a phase reading in degrees from the analog analyzer. The first parameter in the response is the next available measurement.

The phase meter settling is enabled or disabled by the algorithm parameter of the :SETTLING:ANLR command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

Response argument(s):

- **phase -** <nrf>
- **settle_timeout -** <nr1>

Related Commands: :ANLR:PRANge, :SETTling:ANLR

Command Syntax: :ANLR:PHASe?

Response Syntax: :ANLR:PHASe **phase, settle_timeout**

Example: `:ANLR:PHASE?`

Response1: `:ANLR:PHASE 28.8958,1` *(settling timed out)*

Response2: `:ANLR:PHASE 24.4611,0` *(no settling or measurement settled)*

:ANLR:PRANge

Sets the range of the phase meter.

The ranges are: auto (AUTO), -180/+180 (R180), -90/+270 (R270), 0/+360 (R360).

The command argument is:

- **range -** { AUTO | R180 | R270 | R360 }

Default: AUTO

Related Commands: :ANLR:PRANge?

Command Syntax: :ANLR:PRANge **range**

Example: `:ANLR:PRANGE R180`

:ANLR:PRANge?

Returns the phase range setting.

Response argument(s):

- **range -** { AUTO | R180 | R270 | R360 }

Related Commands: :ANLR:PRANge

Command Syntax: :ANLR:PRANge?

Response Syntax: :ANLR:PRANge **range**

Example: `:ANLR:PRANGE?`

Response: `:ANLR:PRANGE R180`

:ANLR:RANGe

Sets the input attenuators and selectable gain amplifiers for the indicated channel(s). The arguments of this command are the channel and the expected signal peak level. The System Two will determine the appropriate range setting from this value. The ranges (in Volts) are: 160.0, 80.0, 40.0, 20.0, 10.0, 5.0, 2.5, 1.2, 0.6, 0.3, 0.160, 0.080, 0.040. If a setting value (in Volts) is not an exact range value then the next higher range will be used. Any value larger than 160 (volts) will generate an Execution error and the range will remain unchanged.

This command will disable input autoranging, equivalent to sending the :ANLR:AUTorange OFF command.

The command arguments are:

- **channel -** { A | B | AB }
- **setting -** <nrf> (range: ≥ 0, ≤ 160 V) { DBU | DBV | V }

Related Commands: :ANLR:RANGe?, :ANLR:AUTorange

Command Syntax: :ANLR:RANGe **channel, setting**

Example: `:ANLR:RANGE A,4.35V`

:ANLR:RANGe?

Returns the channel input range in the specified units regardless if autoranging is enabled or disabled.

The command arguments are:

- **channel -** { A | B }
- **units -** { DBU | DBV | V }

Response argument(s):

- **response_channel -** { A | B }
- **setting -** <nrf> { DBU | DBV | V }

Related Commands: :ANLR:RANGe

Command Syntax: :ANLR:RANGe? **channel**, **units**

Response Syntax: :ANLR:RANGe **response_channel**, **setting**

Example: `:ANLR:RANGE? A,DBV`

Response: `:ANLR:RANGE A,13.9794DBV`

:ANLR:RDGRate

Sets the measurement rate for all six analog analyzer meters. The available rates are auto, 4/sec, 8/sec, 16/sec, 32/sec, 64/sec, 128/sec, auto-fast, and auto-precise. The choice of reading rate is dependent on the type of signal and its fundamental frequency.

The auto reading rate algorithm utilizes five pieces of information from four commands to determine the appropriate setting of the reading rate:

- :ANLR:RDGRate (which auto type),
- :ANLR:RDGRate (meter),
- :AGEN:WFM
- :ANLR:RESPonse, and
- :ANLR:HPFilter.

All six Analog Analyzer meters may make readings at the approximate rates of 4, 8, 16, 32, 64, and 128 readings per second plus the Auto-Fast and Auto-Precise modes. Slower reading rates provide more integration of noise and other variations and better resolution, but faster rates provide shorter test times. Reading rate is normally managed as a function of signal frequency by one of the Auto algorithms. The standard Auto selection is the most useful. Auto-Fast will provide speed increase with some loss in precision and Auto-Precise will improve measurement precision at some cost in speed. It may be sometimes desirable to force a faster reading rate to obtain greater testing speed at some cost in accuracy, or to force a slower rate to obtain greater resolution and noise integration at a cost in speed. The 128/second rate is useful above 600 Hz, the 64/second rate above 300 Hz, the 32/second rate will provide full specified accuracy for signal frequencies above 150 Hz, with 16/second valid above 30 Hz, 8/second valid above 20 Hz, and 4/second valid down to 10 Hz. With complex signals, the frequency to be concerned with is the smallest spacing between individual frequency components if that is less than the lowest absolute frequency.

Auto will not select measurement rates faster than 32 readings/sec. Auto-fast will not select measurement rates faster than 64 readings per second.

The three optional meter arguments are accepted only in AUTO, AFASt and APRecise modes. These optional arguments are used in the determination of the correct measurement rate for these automatic modes. For optimum measurement rate, the parameter corresponding to the type of reading to be taken must be specified. If the :ANLR:LEV? command is to be used, specify LEVel with the AUTO, AFASt or APRecise option. If :ANLR:FREQ? is to be used, select FREQ. If :ANLR:FUNC? is to be used, select FUNCmeter.

Multiple selections are valid if more than one meter type is to be used at this reading rate. If no meter type is specified, the reading rate will be valid, but may be slower than the fastest reading rate valid for that meter.

For example, assume that three measurements are desired with the AFAST measurement rate: Level, Frequency, and Function Meter. The correct combination of parameters would be :ANLR:RDGRATE AFAST,LEVEL,FREQ,FUNCMETER.

NOTE: Failure to specify the correct meter type with an automatic type reading rate may result in readings taken at a rate that is too fast for the input frequency and meter in use.

The command argument is:

- **rate** - { R8 | R4 | R16 | R32 | R64 | R128 | AFASt | APRecise | AUTO }
- **meter1 -** { FREQ | FUNCmeter | LEVel }
- **meter2 -** { FREQ | FUNCmeter | LEVel }
- **meter3 -** { FREQ | FUNCmeter | LEVel }

Default: R8

Related Commands: :ANLR:RDGRate?, :ANLR:RESPonse

Command Syntax: :ANLR:RDGRate **rate[, meter1[, meter2[, meter3]]]**

Example1: `:ANLR:RDGRATE AUTO,FUNCMETER,LEVEL`

Example2: `:ANLR:RDGRATE R32`

:ANLR:RDGRate?

Returns the analog analyzer reading rate. If one of the automatic reading rate modes is active, the assumed meter type(s) is(are) identified.

Response argument(s):

- **rate** - { R8 | R4 | R16 | R32 | R64 | R128 | AFASt | APRecise | AUTO }
- **meter1 -** { FREQ | FUNCmeter | LEVel }
- **meter2 -** { FREQ | FUNCmeter | LEVel }
- **meter3 -** { FREQ | FUNCmeter | LEVel }

Related Commands: :ANLR:RDGRate

Command Syntax: :ANLR:RDGRate?

Response Syntax: :ANLR:RDGRate **rate[, meter1[, meter2[, meter3]]]**

Example: `:ANLR:RDGRATE?`

Response: `:ANLR:RDGRATE R32`

:ANLR:REF

These commands set the parameters for the analog analyzer units that require reference values.

:ANLR:REF:DBM

Sets the DBM unit impedance reference value for the analog analyzer. The unit is ohm.

The command argument is:

- **impedance -** <nrf> (range: > 0 , ≤ 1E34)

Default: 600

Related Commands: :ANLR:REF:DBM?

Command Syntax: :ANLR:REF:DBM **impedance**

Example: `:ANLR:REF:DBM 900`

:ANLR:REF:DBM?

Returns the DBM unit impedance reference value for the analog analyzer. The response unit is ohms.

Response argument(s):

- **impedance -** <nrf>

Related Commands: :ANLR:REF:DBM

Command Syntax: :ANLR:REF:DBM?

Response Syntax: :ANLR:REF:DBM **impedance**

Example: `:ANLR:REF:DBM?`

Response: `:ANLR:REF:DBM 900`

:ANLR:REF:DBRA

Sets the analog analyzer DBRA unit reference value. This reference value can also be set using the :ANLR:SETRefauto command.

The command argument is:

- **setting -** <nrf> (table range: > 0, ≤ 1E34 V) { V | DBU | DBV }

Default: 0.3873V (-6 dBu)

Related Commands: :ANLR:REF:DBRA?, :ANLR:SETRefauto

Command Syntax: :ANLR:REF:DBRA **setting**

Example: `:ANLR:REF:DBRA 1DBV`

:ANLR:REF:DBRA?

Returns the current DBRA unit reference setting for the analog analyzer in the specified units.

The command arguments are:

- **units -** { V | DBU | DBV }

Response argument(s):

- **setting -** <nrf> { V | DBU | DBV }

Related Commands: :ANLR:REF:DBRA

Command Syntax: :ANLR:REF:DBRA? **units**

Response Syntax: :ANLR:REF:DBRA **setting**

Example: `:ANLR:REF:DBRA? V`

Response: `:ANLR:REF:DBRA 1.12202V`

:ANLR:REF:DBRB

Sets the analog analyzer DBRB unit reference value. This reference value can also be set using the :ANLR:SETRefauto command.

The command argument is:

- **setting -** <nrf> (table range: > 0, ≤ 1E34 V) { V | DBU | DBV }

Default: 0.3873V

Related Commands: :ANLR:REF:DBRB?, :ANLR:SETRefauto

Command Syntax: :ANLR:REF:DBRB **setting**

Example: `:ANLR:REF:DBRB 10DBV`

:ANLR:REF:DBRB?

Returns the current DBRB unit reference setting for the analog analyzer in the specified units.

The command arguments are:

- **units -** { V | DBU | DBV }

Response argument(s):

- **setting -** <nrf> { V | DBU | DBV }

Related Commands: :ANLR:REF:DBRB

Command Syntax: :ANLR:REF:DBRB? **units**

Response Syntax: :ANLR:REF:DBRB **setting**

Example: `:ANLR:REF:DBRB? V`

Response: `: :ANLR:REF:DBRB 3.16228V`

:ANLR:REF:FREQ

Sets the FREQ unit reference value for the analog analyzer. This reference value can also be set using the :ANLR:SETRefauto command. The unit is hertz.

The command argument is:

- **frequency -** <nrf> (range: > 0, ≤ 1E34)

Default: 1000

Related Commands: :ANLR:REF:FREQ?, :ANLR:SETRefauto

Command Syntax: :ANLR:REF:FREQ **frequency**

Example: `:ANLR:REF:FREQ 5E3`

:ANLR:REF:FREQ?

Returns the FREQ unit reference value for the analog analyzer. The response unit is hertz.

Response argument(s):

- **frequency -** <nrf>

Related Commands: :ANLR:REF:FREQ

Command Syntax: :ANLR:REF:FREQ?

Response Syntax: :ANLR:REF:FREQ **frequency**

Example: `:ANLR:REF:FREQ?`

Response: `:ANLR:REF:FREQ 5000`

:ANLR:REF:WATT

Sets the WATT unit impedance reference value for the analog analyzer. The unit is ohm.

The command argument is:

- **impedance -** <nrf> (range: > 0, ≤ 1E34)

Default: 8.0

Related Commands: :ANLR:REF:WATT?

Command Syntax: :ANLR:REF:WATT **impedance**

Example: `:ANLR:REF:WATT 2`

:ANLR:REF:WATT?

Returns the WATT unit impedance reference value for the analog analyzer. The response unit is ohms.

Response argument(s):

- **impedance -** <nrf>

Related Commands: :ANLR:REF:WATT

Command Syntax: :ANLR:REF:WATT?

Response Syntax: :ANLR:REF:WATT **impedance**

Example: `:ANLR:REF:WATT?`

Response: `:ANLR:REF:WATT 2`

:ANLR:RESPonse

The frequency specified by this command is used with the automatic reading rate selections of the :ANLR:RDGRate command (AUTO, AFASt and APRecise) to help determine the correct reading rate. The Analog Analyzer filter frequency is set to the specified frequency if the function meter is in a tuning mode (i.e., BP (bandpass), BR (bandreject), THDampl, THDRatio, or XTALk).

The argument is always in units of Hertz.

The command argument is:

- **freq -** <nrf> (range: > 0, ≤ 204775)

Default: 20.0

Related Commands: :ANLR:FILTerfreq, :ANLR:TUNingsrc?

Command Syntax: :ANLR:RESPonse **freq**

Example: `:ANLR:RESPONSE 1000`

:ANLR:RESPonse?

Returns the currently specified response frequency.

Response argument(s):

- **freq -** <nrf>

Related Commands: :ANLR:RESPonse

Command Syntax: :ANLR:RESPonse?

Response Syntax: :ANLR:RESPonse **freq**

Example: `:ANLR:RESPONSE?`

Response: `:ANLR:RESPONSE 1000`

:ANLR:SET?

Returns all analog analyzer settings. The response consists of a sequence of the analog analyzer settings that may be sent back to the instrument at a later time in order to reset all analyzer settings to the same state. Note that some commands are sent more than once and transition through several states in order to achieve a final instrument state.

Response argument(s):

- **settings -** <response message unit> [<response message unit>] ...

Related Commands: (see all other Analog Analyzer setting commands)

Command Syntax: :ANLR:SET?

Example:

```
:ANLR:SET?
```

Response:

```
:ANLR:REF:DBM 600;DBRA 0.3873V;DBRB 0.3873V;
FREQ 1000;WATT 8;:ANLR:CHANNEL A;AUTORANGE A
,OFF;RANGE A,0.04V;AUTORANGE B,OFF;RANGE A,0
.04V;FAUTORANGE ON;COUPLING A,AC;COUPLING B,
AC;IMPEDANCE A,ZHI;IMPEDANCE B,ZHI;INPUT A,X
LR;INPUT B,XLR;PRANGE AUTO;MODE AMPLITUDE;DE
TECTOR RMS;WTG NONE;TUNINGSRC FIXED;FILTERFR
EQ 1000HZ;HPFILTER F10;LPFILTER F500K;RESPON
SE 20;RDGRATE R8;WFDETECTOR NAB;WFFILTER WEI
GHTED
```

:ANLR:SETRefauto

Automatically sets measurement reference values based on real time measurements. The DBR argument sets the DBRA reference from the channel A level meter and the DBRB reference from the channel B level meter.

The FREQ argument loads the FREQUENCY reference from the frequency meter for the channel currently selected by the function meter channel setting (see :ANLR:CHANnel).

The command argument is:

- **reference -** { DBR | FREQ }

Related Commands: :ANLR:REF, :ANLR:REF?, :ANLR:CNANnel

Command Syntax: :ANLR:SETRefauto **reference**

Example:

```
:ANLR:SETREFAUTO DBR
```

:ANLR:TUNingsrc

Sets the frequency steering method of the bandpass and bandreject tuned filters. The filter is used when the reading meter mode is bandpass, bandreject, THD+N ampl, THD+N ratio, or crosstalk. The possible tuning sources are: analog generator, frequency counter, digital generator, and fixed frequency.

If a :ANLR:FILTerfreq command is received, the tuning source will be set to fixed (FIXed).

The command argument is:

- **method -** { FIXed | AGEN | CNTR | DGEN }

Default: FIXed

Related Commands: :ANLR:FILTerfreq, :ANLR:TUNingsrc?

Command Syntax: :ANLR:TUNingsrc **method**

Example: `:ANLR:TUNINGSRC AGEN`

:ANLR:TUNingsrc?

Returns the currently selected bandpass and bandreject filter steering source.

Response argument(s):

- **method -** { FIXed | AGEN | CNTR | DGEN }

Related Commands: :ANLR:TUNingsrc

Command Syntax: :ANLR:TUNingsrc?

Response Syntax: :ANLR:TUNingsrc **method**

Example: `:ANLR:TUNINGSRC?`

Response: `:ANLR:TUNINGSRC AGEN`

:ANLR:WFDetector

Sets the function meter Wow & Flutter detector.

The command argument is:

- **type -** { NAB | IEC | JIS }

Default: NAB

Related Commands: :ANLR:WFDetector?

Command Syntax: :ANLR:WFDetector type

Example: `:ANLR:WFDETECTOR NAB`

:ANLR:WFDetector?

Returns the currently selected function meter Wow & Flutter detector.

Response argument(s):

- **type -** { NAB | IEC | JIS }

Related Commands: :ANLR:WFDetector

Command Syntax: :ANLR:WFDetector?

Response Syntax: :ANLR:WFDetector **type**

Example: `:ANLR:WFDETECTOR?`

Response: `:ANLR:WFDETECTOR NAB`

:ANLR:WFFilter

Specifies the function meter Wow & Flutter filter. The available filters are: Weighted, Unweighted, Weighted-High Band, Unweighted-High Band, Wide-High Band, and Scrape-High Band.

The command argument is:

- **filter -** { WEIGhted | HIUNweighted | HIWeighted | SCRape | UNWeighted | WIDeband }

Default: WEIGhted

Related Commands: :ANLR:WFFilter?

Command Syntax: :ANLR:WFFilter **filter**

Example: `:ANLR:WFFILTER WEIGHTED`

:ANLR:WFFilter?

Returns the function meter Wow & Flutter filter setting.

Response argument(s):

- **filter -** { WEIGhted | HIUNweighted | HIWeighted | SCRape | UNWeighted | WIDeband }

Related Commands: :ANLR:WFFilter

Command Syntax: :ANLR:WFFilter?

Response Syntax: :ANLR:WFFilter **filter**

Example: `:ANLR:WFFILTER?`

Response: `:ANLR:WFFILTER WEIGHTED`

:ANLR:WTG

Selects a filter slot containing the filter to be used for all measurement functions except Bandpass, Crosstalk, and Wow & Flutter.

The error :ERRM 509,18,":ANLR:WTG, ANLR, FILTER NOT INSTALLED IN REQUESTED SLOT.” will be generated if an empty slot is selected.

Use the *OPT? query to determine which slots have filters installed. Appendix D describes Audio Precision weighting filters that may be installed into the seven available filter slots.

The command argument is:

- **filter -** { NONE | S1 | S2 | S3 | S4 | S5 | S6 | S7 }

Default: NONE

Related Commands: OPT?, :ANLR:WTG?

Command Syntax: :ANLR:WTG **filter**

Example: `:ANLR:WTG S3`

:ANLR:WTG?

Returns the currently selected function meter weighting filter used for all measurement functions except Bandpass, Crosstalk, and Wow & Flutter.

Response argument(s):

- **filter -** { NONE | S1 | S2 | S3 | S4 | S5 | S6 | S7 }

Related Commands: :ANLR:WTG

Command Syntax: :ANLR:WTG?

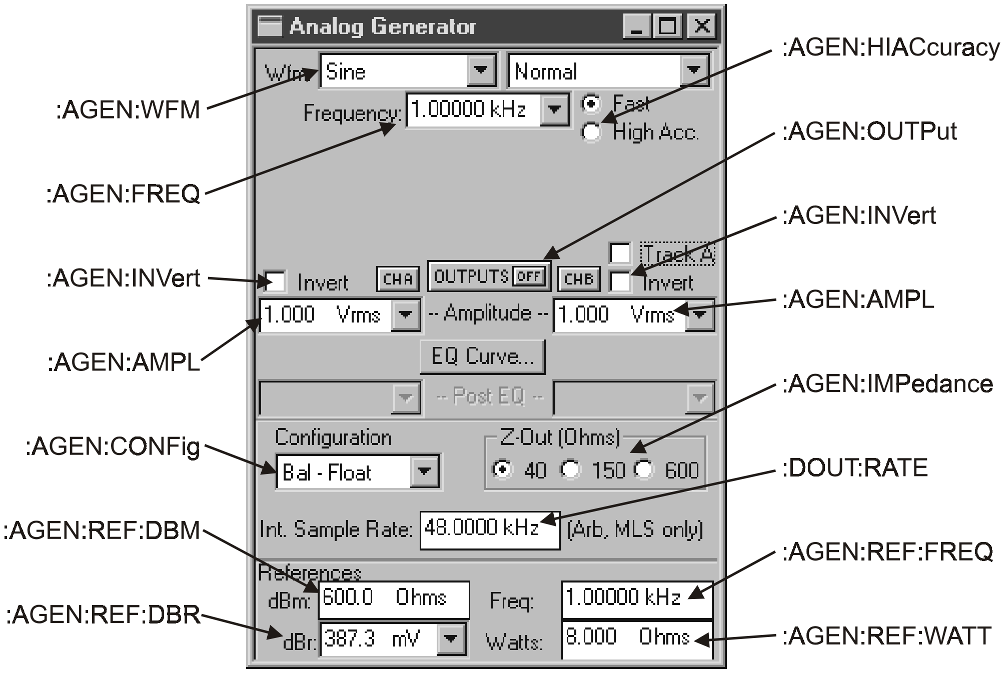
Response Syntax: :ANLR:WTG **filter**

Example: `:ANLR:WTG?`

Response: `:ANLR:WTG S3`

6. Analog Generator Commands

The header path for the analog generator is :AGEN:. These commands set parameters for the analog generator. These commands are invalid and will cause execution errors to be reported if used with a model SYS-2300 System Two which has no analog generator hardware and no D/A converter hardware.



Analog Generator

Wfm: Sine — Normal

Frequency: 1.00000 kHz ● Fast ○ High Acc.

□ Invert CHA OUTPUTS OFF CHB □ Track A □ Invert

1.000 Vrms -- Amplitude -- 1.000 Vrms

EQ Curve...

-- Post EQ --

Configuration: Bal - Float

Z-Out (Ohms): ● 40 ○ 150 ○ 600

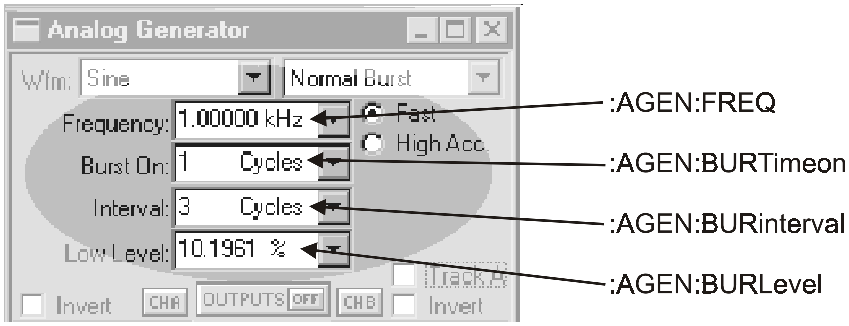
Int. Sample Rate: 48.0000 kHz (Arb, MLS only)

References

dBm: 600.0 Ohms Freq: 1.00000 kHz

dBr: 387.3 mV Watts: 8.000 Ohms

Figure 6-1. AGEN normal commands



Analog Generator

Wfm: Sine — Normal Burst

Frequency: 1.00000 kHz ● Fast ○ High Acc.

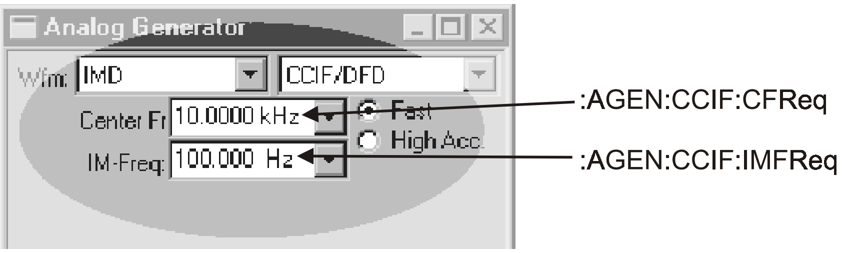
Burst On: 1 Cycles

Interval: 3 Cycles

Low Level: 10.1961 %

□ Invert CHA OUTPUTS OFF CHB □ Track A □ Invert

Figure 6-2. Normal burst mode commands



Analog Generator

Wfm: IMD CCIF/DFD

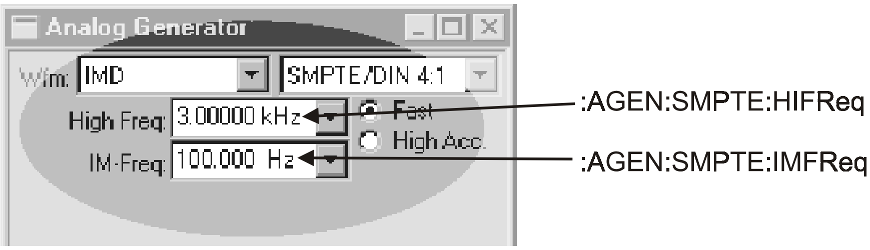
Center Fr 10.0000 kHz ● Fast ○ High Acc.

IM-Freq: 100.000 Hz

:AGEN:CCIF:CFReq

:AGEN:CCIF:IMFReq

Figure 6-3. IMD CCIF/DFD mode commands



Analog Generator

Wfm: IMD SMPTE/DIN 4:1

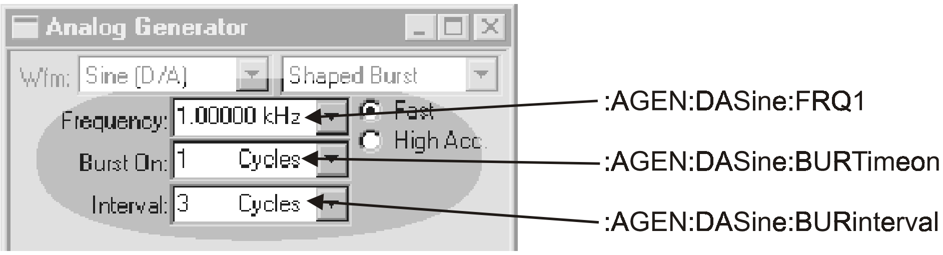
High Freq: 3.00000 kHz ● Fast ○ High Acc.

IM-Freq: 100.000 Hz

:AGEN:SMPTE:HIFReq

:AGEN:SMPTE:IMFReq

Figure 6-4. IMD SMPTE/DIN mode commands



Analog Generator

Wfm: Sine (D/A) Shaped Burst

Frequency: 1.00000 kHz ● Fast ○ High Acc.

Burst On: 1 Cycles

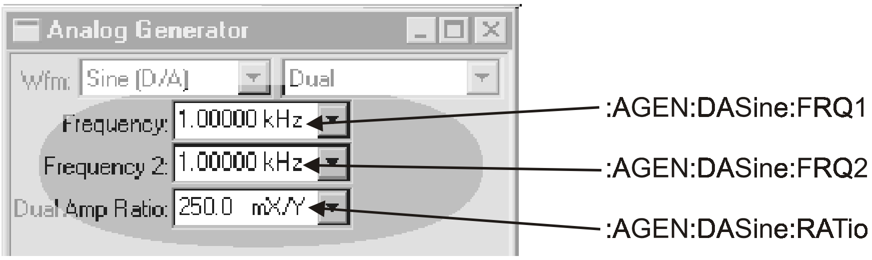
Interval: 3 Cycles

:AGEN:DASine:FRQ1

:AGEN:DASine:BURTimeon

:AGEN:DASine:BURinterval

Figure 6-5. Shaped burst commands



Analog Generator

Wfm: Sine (D/A) Dual

Frequency: 1.00000 kHz

Frequency 2: 1.00000 kHz

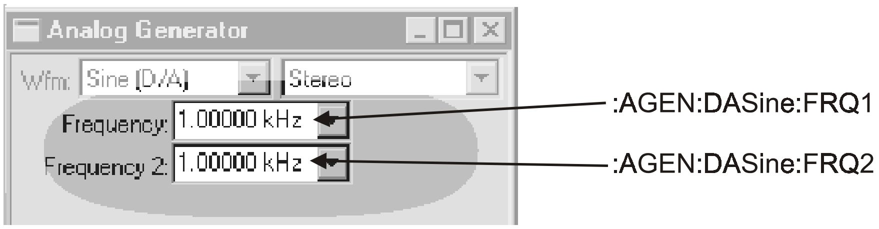
Dual Amp Ratio: 250.0 mX/Y

:AGEN:DASine:FRQ1

:AGEN:DASine:FRQ2

:AGEN:DASine:RATio

Figure 6-6. Dual commands

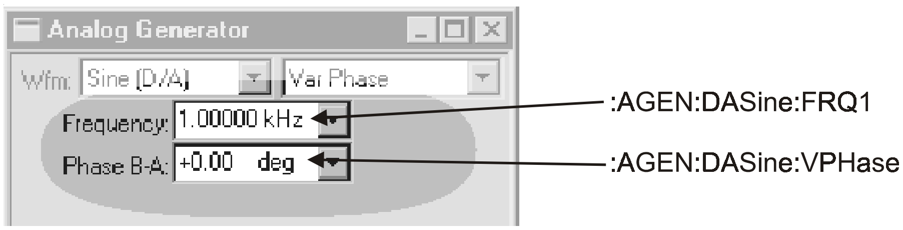


Wfm: Sine (D/A) Stereo

Frequency: 1.00000 kHz

Frequency 2: 1.00000 kHz

Figure 6-7. Stereo commands

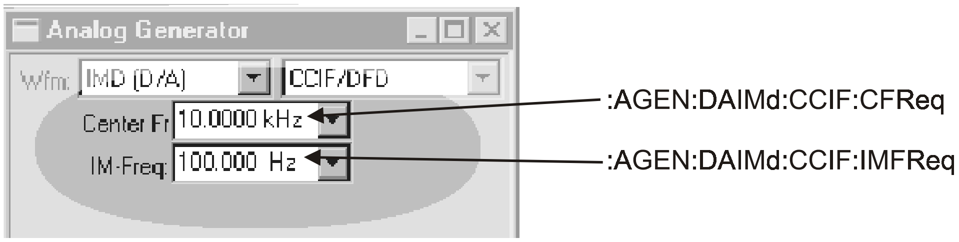


Wfm: Sine (D/A) Var Phase

Frequency: 1.00000 kHz

Phase B-A: +0.00 deg

Figure 6-8.Variable phase commands

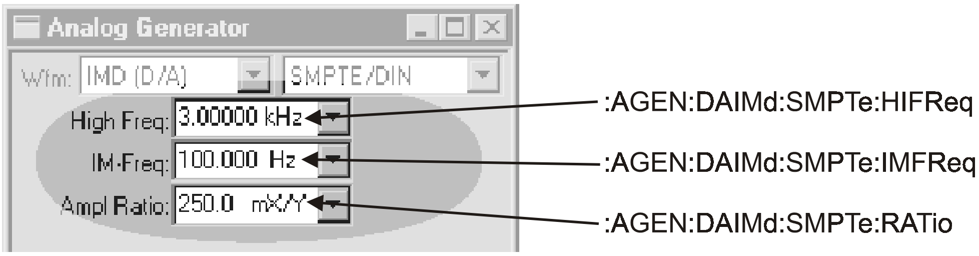


Wfm: IMD (D/A) CCIF/DFD

Center Fr 10.0000 kHz

IM-Freq: 100.000 Hz

Figure 6-9. IMD (D/A) CCIF/DFD mode commands



Wfm: IMD (D/A) SMPTE/DIN

High Freq: 3.00000 kHz

IM-Freq: 100.000 Hz

Ampl Ratio: 250.0 mX/Y

Figure 6-10. IMD (D/A) SMPTE/DIN mode commands

:AGEN:AMPL

Sets the amplitude of the analog generator output for the indicated channel.

The minimum signal amplitude setting is 0.0 V. The maximum signal amplitude is dependent upon waveform type, frequency and output configuration.

The maximum amplitude is reduced to 1/2 of normal for frequencies below 20 Hz and above 50 kHz.

The maximum amplitude for the unbalanced output configuration is 1/2 of the balanced configuration

The maximum amplitude for noise and square waveforms is 1/2 that of the other waveforms.

These relationships are illustrated in the following table(s). Note that all relationships are relative to a maximum voltage of 26.66 V:

All waveforms except Noise and Square
MAXIMUM = 26.66 V

Balanced

Freq	Ampl
<20 Hz	0.5 MAX
20 Hz - 50 kHz	MAX
>50 kHz	0.5 MAX

Unbalanced

Freq	Ampl
<20Hz	0.25 MAX
20Hz - 50kHz	0.5 MAX
>50 kHz	0.25 MAX

Square

Balanced

Freq	Ampl
<20 Hz	Not allowed
20 Hz - 20 kHz	0.5 MAX
>20 kHz	Not allowed

Unbalanced

Freq	Ampl
<20 Hz	Not allowed
20 Hz - 20 kHz	0.25 MAX
>20 kHz	Not allowed

Noise

Balanced

Freq	Ampl
<20 Hz	Not allowed
20 Hz - 100 kHz	0.5 MAX
>100 kHz	Not allowed

Unbalanced

Freq	Ampl
<20 Hz	Not allowed
20 Hz - 100 kHz	0.25 MAX
>100 kHz	Not allowed

The command arguments are:

- **channel** - { AB | A | B }
- **ampl** - <nrf> (range: see table above) { V | DBR | DBU | DBV | DBM | IDBR | VP | VPP | W }

Error(s): An out of range error will be reported if the amplitude setting is out of range for the current combination of output configuration, frequency, and waveform.

Default: AB, 1.0 V

Related Commands: :AGEN:CONFig, :AGEN:FREQ, :AGEN:WFM, :AGEN:AMPL?

Command Syntax: :AGEN:AMPL **channel**, **ampl**

Example: `:AGEN:AMPL AB,2.0V`

:AGEN:AMPL?

Returns the amplitude setting for the indicated channel (in the specified units).

The command arguments are:

- **channel** - { A | B }
- **units** - { V | DBR | DBU | DBV | DBM | IDBR | VP | VPP | W }

Response argument(s):

- **response_channel** - { A | B }
- **ampl** - <nrf> { V | DBR | DBU | DBV | DBM | IDBR | VP | VPP | W }

Related Commands: :AGEN:FREQ, :AGEN:AMPL

Command Syntax: :AGEN:AMPL? **channel**, **units**

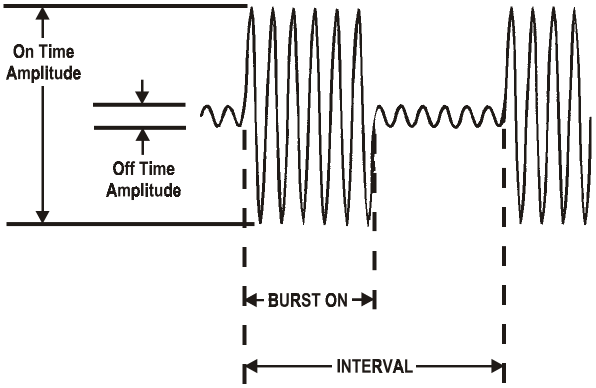
Response Syntax: :AGEN:AMPL **response_channel**, **ampl**

Example: `:AGEN:AMPL? A,DBU`

Response: `:AGEN:AMPL A,8.23871DBU`

:AGEN:BURinterval

Sets the length (in cycles) of the sine burst interval when the generator waveform is SINE,BURST or SINE,GBURST or SIN,TBURST. The burst interval is the sum of the burst on time and the burst off time. Commands affecting the burst interval and burst on time must be issued in a sequence that ensures the burst interval is always greater than the burst on time.



The command argument is:

- **cycles** - <nr1> (range: ≥ 2, ≤ 65536)

Default: 3

Related Commands: :AGEN:BURinterval?, :AGEN:BURTimeon, :AGEN:BURLevel, :AGEN:WFM

Command Syntax: :AGEN:BURinterval **cycles**

Example: `:AGEN:BURINTERVAL 1000`

:AGEN:BURinterval?

Returns the length (in cycles) of the burst interval (see diagram under :AGEN:BURINTERVAL).

Response argument(s):

- **cycles** - <nr1>

Related Commands: :AGEN:BURinterval

Command Syntax: :AGEN:BURinterval?

Response Syntax: :AGEN:BURinterval **cycles**

Example: `:AGEN:BURINTERVAL?`

Response: `:AGEN:BURINTERVAL 1000`

:AGEN:BURLevel

Sets the amplitude of the burst signal during the "burst off time" (see diagram under :AGEN:BURINTERVAL). This amplitude is expressed relative to the "burst on time" amplitude.

The command argument is:

- **level** - <nrf> (range: ≥ 0.0098, ≤ 100 PCT)
  { PCT | DB | PPM | X_Y }

Default: 10.0 PCT

Related Commands: :AGEN:AMPL, :AGEN:BURinterval, :AGEN:BURTimeon, :AGEN:BURLevel?, :AGEN:WFM

Command Syntax: :AGEN:BURLevel **level**

Example: `:AGEN:BURLEVEL 12.5PCT`

:AGEN:BURLevel?

Returns the burst off time amplitude (relative to the burst on time amplitude).

The command argument is:

- **units** - { PCT | DB | PPM | X_Y }

Response argument(s):

- **level** - <nrf> { PCT | DB | PPM | X_Y }

Related Commands: :AGEN:BURLevel

Command Syntax: :AGEN:BURLevel? **units**

Response Syntax: :AGEN:BURLevel **level**

Example: `:AGEN:BURLEVEL? PCT`

Response: `:AGEN:BURLEVEL 12.549PCT`

:AGEN:BURTimeon

Sets the length (in cycles) of the burst on time. An execution error will be generated if burst setup commands are not sent in an order that ensures the burst on time will always be less than the burst interval.

The command argument is:

- **cycles** - <nr1> (range: ≥ 1, ≤ 65535)

Default: 1

Related Commands: :AGEN:BURinterval, :AGEN:BURLevel, :AGEN:WFM, :AGEN:BURTimeon?

Command Syntax: :AGEN:BURTimeon **cycles**

Example: `:AGEN:BURTIMEON 50`

:AGEN:BURTimeon?

Returns the length (in cycles) of the burst on time.

Response argument(s):

- **cycles** - <nr1>

Related Commands: :AGEN:BURTimeon

Command Syntax: :AGEN:BURTimeon?

Response Syntax: :AGEN:BURTimeon **cycles**

Example: `:AGEN:BURTIMEON?`

Response: `:AGEN:BURTIMEON 50`

:AGEN:CCIF

These commands set the parameters for the analog CCIF intermodulation waveform provided by the IMD option. These parameters are used whenever the CCIF waveform is selected with the :AGEN:WFM IMD, CCIF command.

> *Note: These waveforms are unavailable if the IMD option is not installed. An execution error will be reported if these commands are received with no IMD option installed.*

:AGEN:CCIF:CFReq

Sets the center frequency of the two-tone CCIF intermodulation test waveform. The two tones will be equally spaced above and below this center frequency (there is no signal at the center frequency).

The command argument is:

- **frequency** - <nrf> (range: ≥ 4500, ≤ 204775 HZ) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 10000 HZ

Related Commands: :AGEN:WFM,AGEN:CCIF:IMFReq

Command Syntax: :AGEN:CCIF:CFReq **frequency**

Example: `:AGEN:CCIF:CFREQ 15000HZ`

:AGEN:CCIF:CFReq?

Returns the current setting for the center frequency of the two tone CCIF intermodulation test waveform.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :AGEN:CCIF:CFReq

Command Syntax: :AGEN:CCIF:CFReq? **units**

Response Syntax: :AGEN:CCIF:CFReq **frequency**

Example: `:AGEN:CCIF:CFREQ? HZ`

Response: `:AGEN:CCIF:CFREQ 15000HZ`

:AGEN:CCIF:IMFReq

Sets the difference frequency of the two tone CCIF intermodulation distortion test waveform in hertz units. This command accepts any <nrf> number but coerces the setting according to the table below.

Argument Range		
low limit (>=)	**high limit (<)**	**Setting**
0.0 Hz	89.4 Hz	80 Hz
89.4 Hz	109.4 Hz	100 Hz
109.4 Hz	129.6 Hz	120 Hz
129.6 Hz	153 Hz	140 Hz
153 Hz	218 Hz	200 Hz
218 Hz	354 Hz	250 Hz
354 Hz	706 Hz	500 Hz
706 Hz	--	1000 Hz

The command argument is:

- **frequency** - <nrf> (range: > 0, ≤ 1E34 HZ)

Default: 80.0

Related Commands: :AGEN:WFM, AGEN:CCIF:HIFReq

Command Syntax: :AGEN:CCIF:IMFReq **frequency**

Example: `:AGEN:CCIF:IMFREQ 400`

:AGEN:CCIF:IMFReq?

Returns the current setting for the difference frequency of the two tone CCIF intermodulation test waveform in hertz units. The response will be one of: 80, 100, 120, 140, 200, 250, 500, 1000.

Response argument(s):

- **frequency** - <nr1> (range: 80 | 100 | 120 | 140 | 200 | 250 | 500 | 1000)

Related Commands: :AGEN:CCIF:IMFReq

Command Syntax: :AGEN:CCIF:IMFReq?

Response Syntax: :AGEN:CCIF:IMFReq **frequency**

Example: `:AGEN:CCIF:IMFREQ?`

Response: `:AGEN:CCIF:IMFREQ 500`

:AGEN:CONFig

Sets the analog generator output configuration. The possible settings are balanced-float (BF), balanced-ground (BG), unbalanced-float (UF), unbalanced-ground (UG), common mode test (CMTSt).

The command argument is:

- **configuration** - { BF | BG | CMTSt | UF | UG }

Default: BF

Related Commands: :AGEN:CONFig?

Command Syntax: :AGEN:CONFig **configuration**

Example: `:AGEN:CONFIG BF`

:AGEN:CONFig?

Returns the current output configuration setting.

Response argument(s):

- **configuration** - { BF | BG | CMTSt | UF | UG }

Related Commands: :AGEN:CONFig

Command Syntax: :AGEN:CONFig?

Response Syntax: :AGEN:CONFig **configuration**

Example: `:AGEN:CONFIG?`

Response: `:AGEN:CONFIG BF`

:AGEN:DAIMd:CCIF

These commands set the IMD (D/A) parameters for the digitally generated CCIF waveform provided through the System Two D/A converters. These parameters are used whenever the IMD (D/A) waveform is selected with the :AGEN:WFM DAIMd, CCIF command.

> *Note: These waveforms are unavailable for model SYS-2022 which has no DSP. An execution error will be reported if these commands are received with no DSP option installed.*

:AGEN:DAIMd:CCIF:CFReq

Sets the center frequency of the two tone digitally generated CCIF intermodulation test waveform. The two tones will be equally spaced above and below this center frequency (there is no signal at the center frequency).

The command argument is:

- **frequency** - <nrf> (range: ≥ 4500, ≤ 25380 HZ) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 10000 HZ

Related Commands: :AGEN:WFM,AGEN:DAIMd:CCIF:IMFReq

Command Syntax: :AGEN:DAIMd:CCIF:CFReq **frequency**

Example: `:AGEN:DAIMD:CCIF:CFREQ 15000HZ`

:AGEN:DAIMd:CCIF:CFReq?

Returns the current setting for the center frequency of the two tone digitally generated CCIF intermodulation test waveform.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :AGEN:DAIMd:CCIF:CFReq

Command Syntax: :AGEN:DAIMd:CCIF:CFReq? **units**

Response Syntax: :AGEN:DAIMd:CCIF:CFReq **frequency**

Example: `:AGEN:DAIMD:CCIF:CFREQ? HZ`

Response: `:AGEN:DAIMD:CCIF:CFREQ 15000HZ`

:AGEN:DAIMd:CCIF:IMFReq

Sets the difference frequency of the two tone digitally generated CCIF intermodulation test waveform in hertz.

The command argument is:

- **frequency** - <nrf> (range: ≥ 80, ≤ 2000)

Default: 80.0

Related Commands: :AGEN:WFM, AGEN:DAIMd:CCIF:CFReq

Command Syntax: :AGEN:DAIMd:CCIF:IMFReq **frequency**

Example: `:AGEN:DAIMD:CCIF:IMFREQ 500`

:AGEN:DAIMd:CCIF:IMFReq?

Returns the current setting for the difference frequency of the two tone digitally generated CCIF intermodulation test waveform in hertz.

Response argument(s):

- **frequency** - <nrf>

Related Commands: :AGEN:DAIMd:CCIF:IMFReq

Command Syntax: :AGEN:DAIMd:CCIF:IMFReq?

Response Syntax: :AGEN:DAIMd:CCIF:IMFReq **frequency**

Example: `:AGEN:DAIMD:CCIF:IMFREQ?`

Response: `:AGEN:DAIMD:CCIF:IMFREQ 500`

:AGEN:DAIMd:SMPTe

These commands set the IMD (D/A) parameters for the digitally generated SMPTE waveform provided through the System Two D/A converters. These parameters are used whenever the IMD (D/A) waveform is selected with the :AGEN:WFM DAIMd, SMPTe command.

> *Note: These waveforms are unavailable for model SYS-2022 which has no DSP. An execution error will be reported if these commands are received with no DSP option installed.*

:AGEN:DAIMd:SMPTe:HIFReq

Sets the high frequency (upper frequency) of the two tone digitally generated SMPTE intermodulation test waveform.

The command argument is:

- **frequency** - <nrf> (range: ≥ 2000, ≤ 25380 HZ) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 3000 HZ

Related Commands: :AGEN:WFM; AGEN:DAIMd:SMPTe:IMFReq, AGEN:DAIMd:SMPTe:RATio

Command Syntax: :AGEN:DAIMd:SMPTe:HIFReq **frequency**

Example: `:AGEN:DAIMD:SMPTE:HIFREQ 7000HZ`

:AGEN:DAIMd:SMPTe:HIFReq?

Returns the current setting for the high frequency (upper) of the two tone digitally generated SMPTE intermodulation test waveform.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :AGEN:DAIMd:SMPTe:HIFReq

Command Syntax: :AGEN:DAIMd:SMPTe:HIFReq? **units**

Response Syntax: :AGEN:DAIMd:SMPTe:HIFReq **frequency**

Example: `:AGEN:DAIMD:SMPTE:HIFREQ? HZ`

Response: `:AGEN:DAIMD:SMPTE:HIFREQ 7000HZ`

:AGEN:DAIMd:SMPTe:IMFReq

Sets the low frequency of the two tone digitally generated SMPTE intermodulation test waveform is in hertz units.

The command argument is:

- **frequency** - <nrf> (range: ≥ 40, ≤ 500)

Default: 80

Related Commands: :AGEN:WFM, AGEN:DAIMd:SMPTe:HIFReq, AGEN:DAIMd:SMPTe:Ratio

Command Syntax: :AGEN:DAIMd:SMPTe:IMFReq **frequency**

Example:

```
:AGEN:DAIMD:SMPTE:IMFREQ 250
```

:AGEN:DAIMd:SMPTe:IMFReq?

Returns the current setting for the low frequency of the two tone digitally generated SMPTE intermodulation test waveform in hertz units.

Response argument(s):

- **frequency** - <nrf> (range: ≥ 40, ≤ 500)

Related Commands: :AGEN:DAIMd:SMPTe:IMFReq

Command Syntax: :AGEN:DAIMd:SMPTe:IMFReq?

Response Syntax: :AGEN:DAIMd:SMPTe:IMFReq **frequency**

Example:

```
:AGEN:DAIMD:SMPTE:IMFREQ?
```

Response:

```
:AGEN:DAIMD:SMPTE:IMFREQ 250
```

:AGEN:DAIMd:SMPTe:RATio

Sets the amplitude ratio of the low frequency sinewave to the high frequency sinewave for the digitally generated SMPTE intermodulation test waveform.

The command argument is:

- **ratio** - <nrf> (range: ≥ 1e-6, ≤ 1) { X_Y | DB | PCT | PPM }

Default: 0.250 X_Y

Related Commands: :AGEN:WFM, AGEN:DAIMd:SMPTe:HIFReq, :AGEN:DAIMd:SMPTe:IMFReq, :AGEN:DAIMd:SMPTe:RATio?

Command Syntax: :AGEN:DAIMd:SMPTe:RATio **ratio**

Example:

```
:AGEN:DAIMD:SMPTE:RATIO -10DB
```

:AGEN:DAIMd:SMPTe:RATio?

Returns the current setting for the amplitude ratio of the low frequency sinewave to the high frequency sinewave for the digitally generated SMPTE intermodulation test waveform.

The command argument is:

- **units** - { X_Y | DB | PCT | PPM }

Response argument(s):

- **ratio** - <nrf> { X_Y | DB | PCT | PPM }

Related Commands: :AGEN:DAIMd:SMPTe:RATio

Command Syntax: :AGEN:DAIMd:SMPTe:RATio? **units**

Response Syntax: :AGEN:DAIMd:SMPTe:RATio **ratio**

Example: `:AGEN:DAIMD:SMPTE:RATIO? PCT`

Response: `:AGEN:DAIMD:SMPTE:RATIO 31.6228PCT`

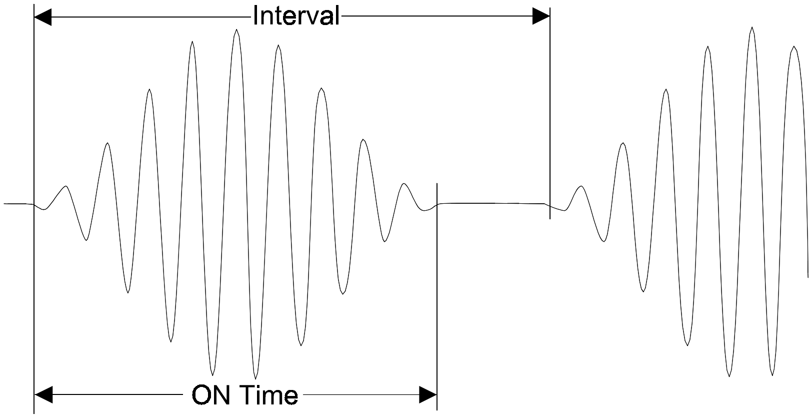
:AGEN:DASine

These commands control the parameters for the digitally generated D/A Sinewave functions provided by the System Two D/A converters via the analog generator output.

> *Note: These waveforms are unavailable for model SYS-2022 which has no DSP. An execution error will be reported if these commands are received with no DSP option installed.*

:AGEN:DASine:BURinterval

Sets the length (in cycles) of the shaped burst interval. The burst interval is the sum of the burst on time and the burst off time. Commands affecting the burst interval and burst on time must be issued in a sequence that ensures the burst interval is always greater than the burst on time.



The command argument is:

- **cycles** - <nr1> (range: ≥ 2, ≤ 65536)

Default: 3

Related Commands: :AGEN:DASine:BURinterval?, :AGEN:DASine:BURTimeon, :AGEN:WFM

Command Syntax: :AGEN:DASine:BURinterval **cycles**

Example: `:AGEN:DASINE:BURINTERVAL 1000`

:AGEN:DASine:BURinterval?

Returns the length (in cycles) of the burst interval (see diagram under :AGEN:DASINE:BURINTERVAL).

Response argument(s):

- **cycles** - <nr1>

Related Commands: :AGEN:DASine:BURinterval

Command Syntax: :AGEN:DASine:BURinterval?

Response Syntax: :AGEN:DASine:BURinterval **cycles**

Example: `:AGEN:DASINE:BURINTERVAL?`

Response: `:AGEN:DASINE:BURINTERVAL 1000`

:AGEN:DASine:BURTimeon

Sets the length (in cycles) of the burst on time. Commands must be sent in an order that ensures the burst on time will always be less than the burst interval. Otherwise, an execution error will be generated.

The command argument is:

- **cycles** - <nr1> (range: ≥ 1, ≤ 65535)

Default: 1

Related Commands: :AGEN:DASine:BURinterval, :AGEN:WFM, :AGEN:DASine:BURTimeon?

Command Syntax: :AGEN:DASine:BURTimeon **cycles**

Example: `:AGEN:DASINE:BURTIMEON 50`

:AGEN:DASine:BURTimeon?

Returns the length (in cycles) of the burst on time.

Response argument(s):

- **cycles** - <nr1>

Related Commands: :AGEN:DASine:BURTimeon

Command Syntax: :AGEN:DASine:BURTimeon?

Response Syntax: :AGEN:DASine:BURTimeon **cycles**

Example: `:AGEN:DASINE:BURTIMEON?`

Response: `:AGEN:DASINE:BURTIMEON 50`

:AGEN:DASine:FRQ1

Sets a frequency for the digitally generated sinewaves available with the :AGEN:WFM DASine functions (SINE, VPHase, STEReo, DUAL, SHAPed). The frequency of this sinewave is defined for each function in the table below.

SINE	Single sinewave frequency for both channels A & B
VPHase	Single sinewave frequency for both channels A & B with variable phase for channel B.
STEReo	Channel A sinewave frequency
DUAL	Two tone sinewave signal, controls the frequency of the higher amplitude sinewave.
SHAPed	Frequency of sinewave (duration set in the BURTimeon command, and interval between the start of consecutive bursts set by BURinterval command). The shaped burst uses a raised cosine shape rather than the rectangular shape of the normal burst.

The command argument is:

- **frequency** - <nrf> (range: ≥ 10, ≤ 25380) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 999.996 HZ

Related Commands: :AGEN:WFM, :AGEN:DASine:FRQ2, :AGEN:DASine:RATio, :AGEN:DASine:FRQ1?

Command Syntax: :AGEN:DASine:FRQ1 **frequency**

Example: `:AGEN:DASINE:FRQ1 15500HZ`

:AGEN:DASine:FRQ1?

Returns the current setting for the Frequency 1 control for digitally generated sinewaves available with the :AGEN:WFM DASine function.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :AGEN:DASine:FRQ1

Command Syntax: :AGEN:DASine:FRQ1? **units**

Response Syntax: :AGEN:DASine:FRQ1 **frequency**

Example: `:AGEN:DASINE:FRQ1? HZ`

Response: `:AGEN:DASINE:FRQ1 15500HZ`

:AGEN:DASine:FRQ2

Sets one of the two sinewave frequencies available for the STEReo and DUAL digitally generated waveforms available with the :AGEN:WFM DASine functions. The signal definition for the sinewave at this frequency is defined for each function in the table below.

STEReo	Channel B sinewave frequency
DUAL	Controls the frequency of the reduced amplitude sinewave (Reduced amplitude set with the AGEN:RATio command).

The command argument is:

- **frequency** - <nrf> (range: ≥ 10, ≤ 25380) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 999.999 HZ

Related Commands: :AGEN:WFM, :AGEN:DASine:FRQ1, :AGEN:DASine:RATio, :AGEN:DASine:FRQ2?

Command Syntax: :AGEN:DASine:FRQ2 **frequency**

Example: `:AGEN:DASINE:FRQ2 17500HZ`

:AGEN:DASine:FRQ2?

Returns the current setting for the FRQ2 control for the STEReo and DUAL digitally generated sinewaves available with the :AGEN:WFM DASine functions.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :AGEN:DASine:FRQ2

Command Syntax: :AGEN:DASine:FRQ2? **units**

Response Syntax: :AGEN:DASine:FRQ2 **frequency**

Example: `:AGEN:DASINE:FRQ2? HZ`

Response: `:AGEN:DASINE:FRQ2 17500HZ`

:AGEN:DASine:RATio

Sets the ratio of the FRQ2 sinewave amplitude relative to the amplitude of the FRQ1 sinewave for the digitally generated DUAL sinewave waveform available with the ":AGEN:WFM DASine, DUAL" function. The FRQ2 sinewave amplitude will always be ≤ the FRQ1 sinewave amplitude by this ratio. Ranges: -120 dB to 0 dB, 0.0001% to 100%, 1 to 1E+6 PPM, 1E-6 to 1.0 X/Y.

The command argument is:

- **ratio** - <nrf> (table range: ≥ 1e-6, ≤ 1 X_Y) { X_Y | DB | PCT | PPM }

Default: 0.250 X_Y

Related Commands: :AGEN:DASine:FRQ1, :AGEN:DASine:FRQ2, :AGEN:DASine:RATio?

Command Syntax: :AGEN:DASine:RATio **ratio**

Example: `:AGEN:DASINE:RATIO -50DB`

:AGEN:DASine:RATio?

Returns the current setting for the ratio of the FRQ2 sinewave amplitude relative to the amplitude of the FRQ1 sinewave for the DUAL digitally generated sinewave waveform available with the :AGEN:WFM DASine functions.

The command argument is:

- **units** - { DB | PCT | PPM | X_Y }

Response argument(s):

- **ratio** - <nrf> { DB | PCT | PPM | X_Y }

Related Commands: :AGEN:DASine:FRQ1, :AGEN:DASine:FRQ2, :AGEN:DASine:RATio

Command Syntax: :AGEN:DASine:RATio? **units**

Response Syntax: :AGEN:DASine:RATio **ratio**

Example: `:AGEN:DASINE:RATIO? X_Y`

Response: `:AGEN:DASINE:RATIO 0.00316228X_Y`

:AGEN:DASine:VPHase

Sets the phase of the channel B sinewave relative to the channel A sinewave for the digitally generated VPHase sinewave waveform provided by the :AGEN:WFM DASine, VPHase command. Units are degrees.

The command argument is:

- **phase** - <nrf> (range: ≥ -180, ≤ 179.98)

Default: 0.00

Related Commands: :AGEN:WFM

Command Syntax: :AGEN:DASine:VPHase **phase**

Example: `:AGEN:DASINE:VPHASE 70.5`

:AGEN:DASine:VPHase?

Returns the phase of the digitally generated variable phase sinewave waveforms provided by the :AGEN:WFM DASine, VPHase command. Units are degrees.

Response argument(s):

- **phase** - <nrf>

Related Commands: :AGEN:DASine:VPHase

Command Syntax: :AGEN:DASine:VPHase?

Response Syntax: :AGEN:DASine:VPHase **phase**

Example: `:AGEN:DASINE:VPHASE?`

Response: `:AGEN:DASINE:VPHASE 70.5`

:AGEN:FREQ

Sets the analog generator output frequency for analog sinewave (including BURST) and squarewave waveforms. The range of values for this command is dependent upon the currently selected waveform type. The following table details the frequency range for each relevant waveform:

Waveform Type	Specific	Min	Max
Sine (SINE)	Sine (SINE)	10	204.775 k
	Burst (BURSt)	20	100.012 k
	Gated Burst (GBURst)	20	100.012 k
	Triggered Burst (TBURst)	20	100.012 k
Square(SQUare)	---	20	20.012 k
Noise (NOISe)	Psuedo-random Pink Bandpass (PPBP)	20	100.012 k
	Random Pink Bandpass (RPBP)		100.012 k
Special D/A (DASPecial)	Polarity (POLarity)	10	11.280 k

The command argument is:

- **frequency** - <nrf> (range: ≥ 10, ≤ 204775 HZ; see table above) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 1000 HZ

Related Commands: :AGEN:FREQ?

Command Syntax: :AGEN:FREQ **frequency**

Example: `:AGEN:FREQ 1200HZ`

:AGEN:FREQ?

Returns the current analog generator sinewave output frequency in the indicated units.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :AGEN:FREQ?

Command Syntax: :AGEN:FREQ? **units**

Response Syntax: :AGEN:FREQ **frequency**

Example: `:AGEN:FREQ? HZ`

Response: `:AGEN:FREQ 1200HZ`

:AGEN:HIACcuracy

Sets the analog generator in high accuracy (ON) or fast settling (OFF) mode. High accuracy mode provides greater accuracy and resolution, but requires from 150 milliseconds (above 50 Hz) to 750 milliseconds (at 10 Hz) for complete settling each time the frequency is changed.

The command argument is:

- **state** - { OFF | ON }

Default: OFF

Related Commands: :AGEN:HIACcuracy?

Command Syntax: :AGEN:HIACcuracy **state**

Example: `:AGEN:HIACCURACY ON`

:AGEN:HIACcuracy?

Returns the analog generator high accuracy setting.

Response argument(s):

- **state** - { OFF | ON }

Related Commands: :AGEN:HIACcuracy

Command Syntax: :AGEN:HIACcuracy?

Response Syntax: :AGEN:HIACcuracy **state**

Example: `:AGEN:HIACCURACY?`

Response: `:AGEN:HIACCURACY ON`

:AGEN:IMPedance

Sets the analog generator output impedance. Sending an invalid impedance for the current output configuration will generate an execution error. The valid choices are shown in the table:

Output Configuration	Impedance(s)
Bal-Float, Bal-Ground, CMTST	40, 150, 600
Unbal-Float, Unbal-Ground	20, 600

The command argument is:

- **impedance** - { Z40 | Z20 | Z150 | Z600 }

Default: Z40

Related Commands: :AGEN:CONFig, :AGEN:IMPedance?, *OPT?

Command Syntax: :AGEN:IMPedance **impedance**

Example: `:AGEN:IMPEDANCE Z20`

:AGEN:IMPedance?

Returns the analog generator output impedance setting.

Response argument(s):

- **impedance** - { Z40 | Z20 | Z150 | Z600 }

Related Commands: :AGEN:IMPedance

Command Syntax: :AGEN:IMPedance?

Response Syntax: :AGEN:IMPedance **impedance**

Example: `:AGEN:IMPEDANCE?`

Response: `:AGEN:IMPEDANCE Z20`

:AGEN:INVert

Sets the analog generator output phase inversion of the indicated channel. If Invert is turned ON the output is shifted 180 degrees. Channels A and B are in phase when the Invert setting is the same for both (both ON or both OFF). Invert ON shifts the analog generator sinewave and squarewave waveforms 180 degrees with respect to the generator SYNC OUTPUT signal.

The command arguments are:

- **channel** - { AB | A | B }
- **state** - { OFF | ON }

Default: AB, OFF

Related Commands: :AGEN:INVert?

Command Syntax: :AGEN:INVert **channel, state**

Example: `:AGEN:INVERT A,ON`

:AGEN:INVert?

Returns the analog generator output invert setting for the indicated channel.

The command argument is:

- **channel** - { A | B }

Response argument(s):

- **response_channel** - { A | B }
- **state** - { OFF | ON }

Related Commands: :AGEN:INVert

Command Syntax: :AGEN:INVert? **channel**

Response Syntax: :AGEN:INVert **response_channel, state**

Example: `:AGEN:INVERT? A`

Response: `:AGEN:INVERT A,ON`

:AGEN:OUTPut

Sets the analog generator outputs on or off.

OFF turns both A and B outputs OFF.

AB turns both A and B outputs ON.

A turns A output ON and B output OFF.

B turns B output ON and A output OFF.

The command argument is:

- **output_channel** - { OFF | A | AB | B }

Default: OFF

Related Commands: :AGEN:OUTPut

Command Syntax: :AGEN:OUTPut **output_channel**

Example: `:AGEN:OUTPUT A`

:AGEN:OUTPut?

Returns the output state of the analog generator.

Response argument(s):

- **output_channel** - { OFF | A | AB | B }

Related Commands: :AGEN:OUTPut

Command Syntax: :AGEN:OUTPut?

Response Syntax: :AGEN:OUTPut **output_channel**

Example: `:AGEN:OUTPUT?`

Response: `:AGEN:OUTPUT A`

:AGEN:REF

These commands set the reference values for parameters for the analog generator units that require references.

:AGEN:REF:DBM

Sets the DBM unit impedance reference value for the analog generator. The argument unit is ohms.

The command argument is:

- **impedance** - <nrf> (range: > 0, ≤ 1E34)

Default: 600.0

Related Commands: :AGEN:REF:DBM?

Command Syntax: :AGEN:REF:DBM **impedance**

Example: `:AGEN:REF:DBM 600`

:AGEN:REF:DBM?

Returns the DBM unit impedance reference value for the analog generator. The response unit is ohms.

Response argument(s):

- **impedance** - <nrf>

Related Commands: :AGEN:REF:DBM

Command Syntax: :AGEN:REF:DBM?

Response Syntax: :AGEN:REF:DBM **impedance**

Example: `:AGEN:REF:DBM?`

Response: `:AGEN:REF:DBM 600`

:AGEN:REF:DBR

Sets the analog generator DBR unit reference value.

The command argument is:

- **setting** - <nrf> (range: > 0, ≤ 1E34 V) { V | DBU | DBV }

Default: 0.3873V (-6 dBu)

Related Commands: :AGEN:REF:DBR?

Command Syntax: :AGEN:REF:DBR **setting**

Example: `:AGEN:REF:DBR 1DBV`

:AGEN:REF:DBR?

Returns the current DBR unit reference setting for the analog generator in the specified units.

The command argument is:

- **units** - { V | DBU | DBV }

Response argument(s):

- **setting** - <nrf> { V | DBU | DBV }

Related Commands: :AGEN:REF:DBR

Command Syntax: :AGEN:REF:DBR? **units**

Response Syntax: :AGEN:REF:DBR **setting**

Example: `:AGEN:REF:DBR? V`

Response: `:AGEN:REF:DBR 1.12202V`

:AGEN:REF:FREQ

Sets the FREQ unit reference value for the analog generator. The argument unit is hertz.

The command argument is:

- **frequency** - <nrf> (range: > 0, ≤ 1E34)

Default: 1000

Related Commands: :AGEN:REF:FREQ?

Command Syntax: :AGEN:REF:FREQ **frequency**

Example: `:AGEN:REF:FREQ 5000`

:AGEN:REF:FREQ?

Returns the FREQ unit reference value for the analog generator. The response unit is hertz.

Response argument(s):

- **frequency** - <nrf>

Related Commands: :AGEN:REF:FREQ

Command Syntax: :AGEN:REF:FREQ?

Response Syntax: :AGEN:REF:FREQ **frequency**

Example: `:AGEN:REF:FREQ?`

Response: `:AGEN:REF:FREQ 5000`

:AGEN:REF:WATT

Sets the WATT unit impedance reference value for the analog generator. The unit is ohms.

The command argument is:

- **impedance** - <nrf> (range: > 0, ≤ 1E34)

Default: 8.0

Related Commands: :AGEN:REF:WATT?

Command Syntax: :AGEN:REF:WATT **impedance**

Example: `:AGEN:REF:WATT 2`

:AGEN:REF:WATT?

Returns the WATT unit impedance reference value for the analog generator. The response unit is ohms.

Response argument(s):

- **impedance** - <nrf>

Related Commands: :AGEN:REF:WATT

Command Syntax: :AGEN:REF:WATT?

Response Syntax: :AGEN:REF:WATT **impedance**

Example: `:AGEN:REF:WATT?`

Response: `:AGEN:REF:WATT 2`

:AGEN:SET?

Returns all Analog Generator settings. The response consists of a sequence of the Analog Generator settings that may be sent back to the instrument at a later time in order to reset all analog generator settings to the same state. Note that some commands are sent more than once and transition through several states in order to achieve a final hardware state.

Response argument(s):

- **settings** - <response message unit> [<response message unit>] ...

Related Commands: (see all other generator commands)

Command Syntax: :AGEN:SET?

Response Syntax: :AGEN:SET **settings**

Example: `:AGEN:SET?`

Response:

```
 :AGEN:REF:DBM 600;DBR 0.3873V;FREQ 1000;WAT
T 8;:AGEN:OUTPUT OFF;HIACCURACY OFF;CONFIG B
F;IMPEDANCE Z40;INVERT A,OFF;INVERT B,OFF;AM
PL A,0.999957V;AMPL B,0.999957V;FREQ 1000HZ;
WFM SINE,SINE;BURINTERVAL 3;BURLEVEL 10PCT;B
URTIMEON 1;:AGEN:CCIF:CFREQ 10000HZ;IMFREQ 8
0;:AGEN:SMPTE:HIFREQ 3000HZ;IMFREQ 60;:AGEN:
DAIMD:CCIF:CFREQ 10000HZ;IMFREQ 80;:AGEN:DAI
MD:SMPTE:HIFREQ 3000HZ;IMFREQ 80;RATIO 0.25X
_Y;:AGEN:DASINE:FRQ1 999.996HZ;FRQ2 999.996H
Z;BURINTERVAL 3;BURTIMEON 1;RATIO 0.25X_Y;VP
HASE 0
```

:AGEN:SMPTe

These commands set the IMD parameters for the analog SMPTE waveform provided by the IMD option for System Two. These parameters are used whenever the SMPTE waveform is selected with the :AGEN:WFM IMD, SMPTe command.

> *Note: These waveforms are unavailable if the IMD option is not installed. An execution error will be reported if these commands are received with no IMD option installed.*

:AGEN:SMPTe:HIFReq

Sets the high frequency (upper frequency) of the two tone SMPTE intermodulation test waveform.

The command argument is:

- **frequency** - <nrf> (range: ≥ 2000, ≤ 204775) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 3000

Related Commands: :AGEN:WFM,AGEN:SMPTe:IMFReq

Command Syntax: :AGEN:SMPTe:HIFReq **frequency**

Example: `:AGEN:SMPTE:HIFREQ 7000HZ`

:AGEN:SMPTe:HIFReq?

Returns the current setting for the high frequency (upper) of the analog two tone SMPTE intermodulation test waveform.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :AGEN:SMPTe:HIFReq

Command Syntax: :AGEN:SMPTe:HIFReq? **units**

Response Syntax: :AGEN:SMPTe:HIFReq **frequency**

Example: `:AGEN:SMPTE:HIFREQ? HZ`

Response: `:AGEN:SMPTE:HIFREQ 7000HZ`

:AGEN:SMPTe:IMFReq

Sets the low frequency of the two tone SMPTE intermodulation test waveform in hertz units. This command accepts any <nrf> number, but coerces the setting according to the table below.

Argument Range		Setting
low limit (>=)	high limit (<)	
0.0 Hz	44.7 Hz	40 Hz
44.7 Hz	54.7 Hz	50 Hz
54.7 Hz	64.8 Hz	60 Hz
64.8 Hz	77 Hz	70 Hz
77 Hz	109 Hz	100 Hz
109 Hz	177 Hz	125 Hz
177 Hz	353 Hz	250 Hz
353 Hz	---	500 Hz

The command argument (in hertz) is:

- **frequency** - <nrf> (range: > 0, < 1E34)

Default: 60.0

Related Commands: :AGEN:WFM, AGEN:SMPTe:HIFReq

Command Syntax: :AGEN:SMPTe:IMFReq **frequency**

Example: `:AGEN:SMPTE:IMFREQ 300`

:AGEN:SMPTe:IMFReq?

Returns the current setting for the low frequency of the two tone SMPTE intermodulation test waveform in hertz units. The query response will be one of: 40, 50, 60, 70, 100, 125, 250, 500.

Response argument(s):

- **frequency** - <nr1>

Related Commands: :AGEN:SMPTe:IMFReq

Command Syntax: :AGEN:SMPTe:IMFReq?

Response Syntax: :AGEN:SMPTe:IMFReq **frequency**

Example: `:AGEN:SMPTE:IMFREQ?`

Response: `:AGEN:SMPTE:IMFREQ 250`

:AGEN:WFM

Specifies a generator waveform type. Availability of some waveforms requires that specific hardware modules be present. These include IMD waveforms (IMD option), burst-triggered-gated sine, square waves, and white-pink-selective random or pseudo-random noise waveforms (BUR option), and all digital generated waveforms (DSP module). An Execution error will be reported if a waveform is selected in the absence of the required hardware.

Required Hardware Option	Waveforms
BUR	SINE (BURSt, TBURst, GBURst) NOISe (PP, PW, PPBP, RP, RW, RPBP, USASi) SQUARe
IMD	IMD (SMP1, SMP4, CCIF, D30, D100, DIMB)
DSP (Models SYS-2322 & SYS-2222)	IMDA, DAARbitrary, DAMLs, DASPecial

Some waveform types may be specified with only one argument, while others require two arguments. The first argument specifies the type of waveform. The second argument selects a specific waveform sub-type.

In the case of Square and Arbitrary waveform types, a second argument is not required, but a second parameter of NONE will be accepted. This capability is provided in order to support macro definitions with fixed place holders for argument variables.

If arbitrary waveform is specified, a generator arbitrary waveform must already be specified with the :DGEN:ARBWfm command or an error will be generated. The :AGEN:WFM DAARBITRARY command and the :DGEN:WFM ARBITRARY command both use the same arbitrary waveform loaded into the DSP digital arbitrary waveform generator buffers. See the :DGEN:ARBWfm command for more details.

Waveform Type (argument 1)	Waveform sub-Type(argument 2)
D/A Arb Wfm (DAARbitrary)	No argument required (but will accept NONE)
D/A IMD (DAIMd)	CCIF/DFD (CCIF), SMPTE/DIN (SMPTe)
D/A Sine (DASine)	Dual (DUAL), Shaped Burst (SHAPed), Normal (SINE), Stereo (STEReo), Variable Phase (VPHase)
IMD (IMD)	CCIF/DFD (CCIF), DIM B (DIMB), DIM 100 (D100), DIM 30 (D30), SMPTE/DIN 1:1 (SMP1), SMPTE/DIN 4:1 (SMP4)
D/A MLS (DAMLs)	Pink #1 (P1), Pink #2 (P2), Pink #3 (P3), Pink #4 (P4), White #1 (W1), White #2 (W2), White #3 (W3), White #4 (W4)
Noise (NOISe)	Pseudo-random Pink (PP), Pseudo-random White (PW), Pseudo-random Pink BP (PPBP), Random Pink (RP), Random White (RW), Random Pink BP (RPBP), Burst USASI (USASi)
Sine (SINE)	Normal Burst (BURSt), Gated Burst (GBURst), Equalized Sine (EQSine), Normal Sine (SINE), Trig Burst (TBURst)
Special D/A (DASPecial)	Polarity (POLarity)
Square (SQUare)	No argument required (but will accept NONE)

The command arguments are:

- **type** - { SINE | DAARbitrary | DAIMd | DAMLs | DASine | DASPecial | IMD | NOISe | SQUare }
- **subtype** - { SINE | BURSt | CCIF | D30 | D100 | DIMB | DUAL |GBURst | NONE | P1 | P2 | P3 | P4 | POLarity | PP | PPBP | PW | RP | RPBP | RW | SHAPed | SMP1 | SMP4 | SMPTe | STEReo | TBURst | USASi | VPHase | W1 | W2 | W3 | W4 }

Default: SINE, SINE

Related Commands: :DGEN:ARBWfm, :DGEN:WFM, :AGEN:WFM?

Command Syntax: :AGEN:WFM **type**[**, subtype**]

Example 1: `:AGEN:WFM SQUARE`

Example 2: `:AGEN:WFM IMD,SMP1`

:AGEN:WFM?

Returns the current type of waveform (and optional specific waveform) being generated. For SQUARE and DAARbitrary waveform types there will be only one argument. For all other types of waveforms there will be two arguments.

Response argument(s):

- **type** - { SINE | DAARbitrary | DAIMd | DAMLs | DASine | DASPecial | IMD | NOISe | SQUare }
- **subtype** - { SINE | BURSt | CCIF | D30 | D100 | DIMB | DUAL | GBURst | NONE | P1 | P2 | P3 | P4 | POLarity | PP | PPBP | PW | RP | RPBP | RW | SHAPed | SMP1 | SMP4 | SMPTe | STEReo | TBURst | USASi | VPHase | W1 | W2 | W3 | W4 }

Related Commands: :AGEN:WFM

Command Syntax: :AGEN:WFM?

Response Syntax: :AGEN:WFM **type**[, **subtype**]

Example 1: `:AGEN:WFM?`

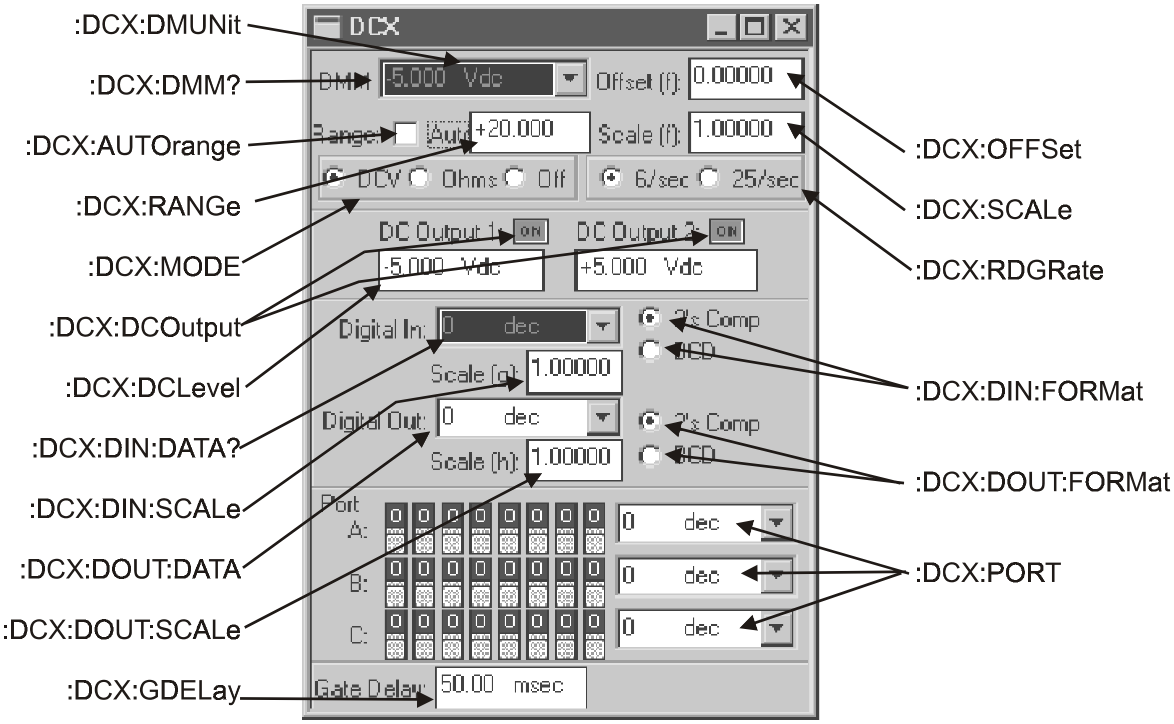
Response 1: `:AGEN:WFM SQUARE`

Example 2: `:AGEN:WFM?`

Response 2: `:AGEN:WFM IMD,SMP1`

7. DCX-127 Commands

The header path for the DCX-127 commands is :DCX:.



DMM: -5.000 Vdc
Offset (f): 0.00000
Range: □ Auto +20.000
Scale (f): 1.00000
● DCV ○ Ohms ○ Off
● 6/sec ○ 25/sec
DC Output 1: ON -5.000 Vdc
DC Output 2: ON +5.000 Vdc
Digital In: 0 dec
● 2's Comp ○ BCD
Scale (g): 1.00000
Digital Out: 0 dec
● 2's Comp ○ BCD
Scale (h): 1.00000
Port A: 0 0 0 0 0 0 0 0 0 dec
B: 0 0 0 0 0 0 0 0 0 dec
C: 0 0 0 0 0 0 0 0 0 dec
Gate Delay: 50.00 msec

Figure 7-1. DCX-127 panel, showing relationship with GPIB commands

:DCX:AUTorange

Enables or disables DMM autoranging. When autoranging is disabled, the ranging will be fixed at the current range. Changes to ranging can then be made using the :DCX:RANGE command. If a fixed range command is received while autoranging is enabled, autoranging will be disabled.

The command argument is:

- **state** - { ON | OFF }

Default: ON

Related Commands: :DCX:AUTorange?, DCX:RANGe

Command Syntax: :DCX:AUTorange **state**

Example: `:DCX:AUTORANGE ON`

:DCX:AUTorange?

Returns the current state of DMM autoranging.

Response argument(s):

- **state** - { ON | OFF }

Related Commands: :DCX:AUTorange

Command Syntax: :DCX:AUTorange?

Response Syntax: :DCX:AUTorange **state**

Example: `:DCX:AUTORANGE?`

Response: `:DCX:AUTORANGE ON`

:DCX:DCLevel

Sets the DC output level (in volts) of the indicated channel.

The command arguments are:

- **channel** - { DC1 | DC2 }
- **amplitude** - <nrf> (range: > -10.50002, < 10.50002)

Default: DC1, 0.0; DC2, 0.0

Related Commands: :DCX:DCLevel?

Command Syntax: :DCX:DCLevel **channel**, **amplitude**

Example: `:DCX:DCLEVEL DC1,5.3`

:DCX:DCLevel?

Returns the output level setting (in volts) of the indicated channel.

The command arguments are:

- **channel** - { DC1 | DC2 }

Response argument(s):

- **response_channel** - { DC1 | DC2 }
- **amplitude** - <nrf>

Related Commands: :DCX:DCLevel

Command Syntax: :DCX:DCLevel? **channel**

Response Syntax: :DCX:DCLevel **response_channel**, **amplitude**

Example: `:DCX:DCLEVEL? DC1`

Response: `:DCX:DCLEVEL DC1,5.3`

:DCX:DCOutput

Enables/disables the output of the indicated channel.

The command arguments are:

- **channel** - { DC1 | DC2 }
- **state** - { OFF | ON }

Default: DC1, OFF; DC2, OFF

Related Commands: :DCX:DCOutput?

Command Syntax: :DCX:DCOutput **channel**, **state**

Example: `:DCX:DCOUTPUT DC1,ON`

:DCX:DCOutput?

Returns the output status of the indicated channel.

The command argument is:

- **channel** - { DC1 | DC2 }

Response argument(s):

- **response_channel** - { DC1 | DC2 }
- **state** - { OFF | ON }

Related Commands: :DCX:DCOutput

Command Syntax: :DCX:DCOutput? **channel**

Response Syntax: :DCX:DCOutput **response_channel**, **state**

Example: `:DCX:DCOutput? DC1`

Response: `:DCX:DCOUTPUT DC1,ON`

:DCX:DIN

Compound command header for DCX digital input commands.

:DCX:DIN:DATA?

Returns (as a decimal number) the data at the 22-bit (21 bit plus sign) front panel digital input port. The available units are DEC (unscaled decimal) and G_X (scaled decimal, *see* :DCX:DIN:SCALe).

The first parameter in the response is the next available measurement.

The digital input port settling is enabled or disabled by the algorithm parameter of the :SETTLING:DCX command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

If input format is specified as BCD, then invalid binary inputs which cannot be decoded to valid BCD format bit patterns will cause erroneous BCD values to be read from the port. An error will not be generated. This leads to the following behavior:

Binary Input	BCD Output	BCD	Unit Command Response
0000 0000	0000 0000	00	
0000 0001	0000 0001	01	
0000 0010	0000 0010	02	
0000 0011	0000 0011	03	
...			
0000 1001	0000 1001	09	
0000 1010	0001 0000	10	<- incorrect
0000 1011	0001 0001	11	<- incorrect
0000 1100	0001 0010	12	<- incorrect
0000 1101	0001 0011	13	<- incorrect
0000 1110	0001 0100	14	<- incorrect
0000 1111	0001 0101	15	<- incorrect
0001 0000	0001 0000	10	<- correct
0001 0001	0001 0001	11	

Command argument(s):

- **data_units** - { DEC | G_X }

Response argument(s):

- **data** <nr1> (range: ≥ -2097152, ≤ 2097151 DEC) { DEC | G_X }
- **settle_timeout** - <nr1>

Related Commands: :DCX:DIN:FORMat

Command Syntax: :DCX:DIN:DATA? **data_units**

Response Syntax: :DCX:DIN:DATA **data**, **settle_timeout**

Example: `:DCX:DIN:DATA? DEC`

Response: `:DCX:DIN:DATA 254DEC,0`

:DCX:DIN:FORMat

Sets the format for the input digital data. The format can be either 2’s complement (TWOS) or BCD.

The command argument is:

- **format** - { TWOS | BCD }

Default: TWOS

Related Commands: :DCX:DIN:FORMat?

Command Syntax: :DCX:DIN:FORMat **format**

Example: `:DCX:DIN:FORMAT TWOS`

:DCX:DIN:FORMat?

Returns the current digital input format.

Response argument(s):

- **format** - { TWOS | BCD }

Related Commands: :DCX:DIN:FORMat

Command Syntax: :DCX:DIN:FORMat?

Response Syntax: :DCX:DIN:FORMat **format**

Example: `:DCX:DIN:FORMAT?`

Response: `:DCX:DIN:FORMAT TWOS`

:DCX:DIN:RATE

Note: There is no corresponding APWIN panel control for this command, but there is a comparable APBASIC command.

Sets the internal strobe rate for the Digital Input port. A parameter value of 0 will select the external strobe pin of the Digital Input connector. The available internal strobe rates are: 4, 8, 16, and 32. The selected strobe rate will be determined by rounding up to the next higher value. For example, parameter settings greater than 4 and less than or equal to 8 will be rounded up to 8. Parameters less than zero or greater than 32 will cause a parameter error.

Setting	Resulting rate
>16 (and <=32)	32
>8 (and <=16)	16
>4 (and <=8)	8
>1 (and <=4)	4
0	External source

The command argument is:

- **rate** - <nr1> (range: ≥ 0, ≤ 32)

Default: 32

Related Commands: :DCX:DIN:RATE?

Command Syntax: :DCX:DIN:RATE **rate**

Example: `:DCX:DIN:RATE 10`

:DCX:DIN:RATE?

Returns the Digital Input port internal strobe rate. The possible responses are 0 (external), 4, 8, 16, or 32.

Response argument(s):

- **rate** - <nr1>

Related Commands: :DCX:DIN:RATE

Command Syntax: :DCX:DIN:RATE?

Response Syntax: :DCX:DIN:RATE **rate**

Example: `:DCX:DIN:RATE?`

Response: `:DCX:DIN:RATE 16`

:DCX:DIN:SCALe

Sets the scaling factor to be used with the :DCX:DIN:DATA? query in g(x) units.

The command argument is:

- **factor** - <nrf> (range: ≥ -1E34, ≤ 1E34)

Default: 1.0

Related Commands: :DCX:DIN:DATA?, :DCX:DIN:SCALe?

Command Syntax: :DCX:DIN:SCALe **factor**

Example: `:DCX:DIN:SCALE 2.5`

:DCX:DIN:SCALe?

Returns the current scaling factor to be used with the :DCX:DIN:DATA? query in g(x) units.

Response argument(s):

- **factor** - <nrf>

Related Commands: :DCX:DIN:DATA?, :DCX:DIN:SCALe

Command Syntax: :DCX:DIN:SCALe?

Response Syntax: :DCX:DIN:SCALe **factor**

Example: `:DCX:DIN:SCALE?`

Response: `:DCX:DIN:SCALE 2.5`

:DCX:DIN:SET?

Returns the current setting of all the DCX digital input parameters.

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands: (see all other :DCX:DIN query commands)

Command Syntax: :DCX:DIN:SET?

Response Syntax: :DCX:DIN:SET **settings**

Example: :DCX:DIN:SET?

Response: `:DCX:DIN:FORMAT TWOS;SCALE 1;RATE 32`

:DCX:DMM?

Returns the current DCX Multi-Meter reading in the indicated units. The first parameter in the response is the next available measurement.

The DCX Multi-Meter settling is enabled or disabled by the algorithm parameter of the :SETTLING:DCX command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

If the query units are not valid for the current mode (see :DCX:MODE) an error will be generated.

Query argument(s):

- **units** - { F_O | F_V | O | V }

The units for this query are

f(O) - scaled and offset Ohms

f(V) - scaled and offset Volts

O - 0hms

V - Volts

Response argument(s):

- **rdg** - <nrf>
- **settle_timeout** - <nr1>

Related Commands: :DCX:MODE, :DCX:MODE, :DCX:OFFSET, :DCX:SCALe

Command Syntax: :DCX:DMM? **units**

Response Syntax: :DCX:DMM? **rdg**, **settle_timeout**

Example: `:DCX:DMM? V`

Response: `:DCX:DMM 6.2V,1`

:DCX:DOUT

Compound command header for DCX digital output commands.

:DCX:DOUT:DATA

Sets the data at the 22-bit (21 bits plus sign) front panel digital output port. The available units are decimal and g(x) (uses digital output scaling factor). If g(x) units (G_X) are selected the data sent by this command will be scaled by the value set in the :DCX:DOUT:SCALe command.

The command argument is:

- **setting** - <nr1> (range: ≥ -2097152, ≤ 2097151) { DEC | G_X }

Default: 0 DEC

Related Commands: :DCX:DOUT:FORMat

Command Syntax: :DCX:DOUT:DATA **setting**

Example: `:DCX:DOUT:DATA 64DEC`

:DCX:DOUT:DATA?

Returns the current data at the 22-bit (21 bits plus sign) front panel digital output port. The available units are decimal and g(x) (uses digital output scaling factor). When the units are g(x) the return value is the unscaled data. This supports resending of the response string to achieve the data at the output port.

The command argument is:

- **units** - { DEC | G_X }

Response argument(s):

- **setting** - <nr1> { DEC | G_X }

Related Commands: :DCX:DOUT:FORMat

Command Syntax: :DCX:DOUT:DATA? **units**

Response Syntax: :DCX:DOUT:DATA **setting**

Example: `:DCX:DOUT:DATA? DEC`

Response: `:DCX:DOUT:DATA 64DEC`

:DCX:DOUT:FORMat

Sets the format for the digital output data. The format can be either twos complement or BCD.

The command argument is:

- **format** - { TWOS | BCD }

Default: TWOS

Related Commands: :DCX:DOUT:FORMat?

Command Syntax: :DCX:DOUT:FORMat **format**

Example: `:DCX:DOUT:FORMAT TWOS`

:DCX:DOUT:FORMat?

Returns the current digital output format.

Response argument(s):

- **format** - { TWOS | BCD }

Related Commands: :DCX:DOUT:FORMat

Command Syntax: :DCX:DOUT:FORMat?

Response Syntax: :DCX:DOUT:FORMat **format**

Example: `:DCX:DOUT:FORMAT?`

Response: `:DCX:DOUT:FORMAT TWOS`

:DCX:DOUT:SCALe

Sets the digital output scaling factor. When g(x) units (G_X) are selected with the :DCX:DOUT:DATA command, the firmware computes the actual transmitted value from the relationship

output value = entry value ∗ Scale (g)

where entry value is the decimal value entered with the :DCX:DOUT:DATA command and Scale (g) is the value entered with the :DCX:DOUT:SCALe command.

The command argument is:

- **factor** - <nrf> (range: ≥ -1E34, ≤ 1E34)

Default: 1.0

Related Commands: :DCX:DOUT:DATA, :DCX:DOUT:SCALe?

Command Syntax: :DCX:DOUT:SCALe **factor**

Example: `:DCX:DOUT:SCALE 5.0`

:DCX:DOUT:SCALe?

Returns the current digital output scaling factor.

Response argument(s):

- **factor** - <nrf>

Related Commands: :DCX:DOUT:DATA?, :DCX:DOUT:SCALe

Command Syntax: :DCX:DOUT:SCALe?

Response Syntax: :DCX:DOUT:SCALe **factor**

Example: `:DCX:DOUT:SCALE?`

Response: `:DCX:DOUT:SCALE 5`

:DCX:DOUT:SET?

Returns the current setting of all the DCX digital output parameters.

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands: (see all other :DCX:DIN query commands)

Command Syntax: :DCX:DOUT:SET?

Response Syntax: :DCX:DOUT:SET **settings**

Example: `:DCX:DOUT:SET?`

Response: `:DCX:DOUT:DATA 0DEC;FORMAT TWOS;SCALE 1`

:DCX:GDELay

Sets the delay (in seconds) between the sweep gate (Pin 6) and the delayed sweep gate (Pin 1) of the DCX Program Control Output.

The command argument is:

- **delay** - <nrf> (range: ≥ 0.050, ≤ 12.75)

Default: 0.050

Related Commands: :DCX:GDELay?

Command Syntax: :DCX:GDELay **delay**

Example: `:DCX:GDELAY 100e-3`

:DCX:GDELay?

Returns the programmed delay (in seconds) between the sweep gate (Pin 6) and the delayed sweep gate (Pin 1) of the DCX Program Control Output.

Response argument(s):

- **delay** - <nrf>

Related Commands: :DCX:GDELay

Command Syntax: :DCX:GDELay?

Response Syntax: :DCX:GDELay **delay**

Example: `:DCX:GDELAY?`

Response: `:DCX:GDELAY 0.1`

:DCX:MODE

Sets the mode of the DCX-127 Digital Multimeter. The available modes are off, ohmmeter, and voltmeter.

The command argument is:

- **mode** - { VOLTs | OFF | OHMS }

Default: VOLTs

Related Commands: :DCX:DMM, :DCX:MODE?

Command Syntax: :DCX:MODE **mode**

Example: `:DCX:MODE VOLTS`

:DCX:MODE?

Returns the current mode of the DCX-127 Digital Multimeter.

Response argument(s):

- **mode** - { VOLTs | OFF | OHMS }

Related Commands: :DCX:DMM

Command Syntax: :DCX:MODE?

Response Syntax: :DCX:MODE **mode**

Example: `:DCX:MODE?`

Response: `:DCX:MODE VOLTS`

:DCX:OFFSet

Sets the offset for the current DMM mode. The return value to a ":DCX:DMM?" (with function units, either F_O or F_V) is computed by the formula

Return Value = scale * (Measured + offset).

For example, with a scaling factor of 3.2, an offset of 1.5 volts and a raw measurement of 2.0 volts, the response to ":DCX:DMM? F_V" would be 11.2 V (11.2 = 3.2 * (2.0 + 1.5)).

The command argument is:

- **offset** - <nrf> (range: ≥ -1E34, ≤ 1E34)

Default: 0.0

Related Commands: :DCX:OFFSet?

Command Syntax: :DCX:OFFSet **offset**

Example: `:DCX:OFFSET 1.5`

:DCX:OFFSet?

Returns the currently programmed function offset value.

Response argument(s):

- **offset** - <nrf>

Related Commands: :DCX:OFFSet

Command Syntax: :DCX:OFFSet?

Response Syntax: :DCX:OFFSet **offset**

Example: `:DCX:OFFSET?`

Response: `:DCX:OFFSET 1.5`

:DCX:PORT

Sets the output of the indicated 8-bit digital control port. The range of values is 0 through 255 (decimal). The example sets all of the control pins on port A high.

The command arguments are:

- **port** - { A | B | C }
- **setting** - <nr1> (range: ≥ 0, ≤ 255)

Default: A, 0; B, 0; C, 0

Related Commands: :DCX:PORT?

Command Syntax: :DCX:PORT **port**, **setting**

Example: `:DCX:PORT A, 255`

:DCX:PORT?

Returns the state of the indicated digital control output port (in decimal).

The command argument is:

- **port** - { A | B | C }

Response argument(s):

- **response_port** - { A | B | C }
- **setting** - <nr1>

Related Commands: :DCX:PORT

Command Syntax: :DCX:PORT? **port**

Response Syntax: :DCX:PORT **response_port**, **setting**

Example: `:DCX:PORT? A`

Response: `:DCX:PORT A,255`

:DCX:RANGe

Sets the DMM input range. The available ranges are shown in the following table (depending on current DMM mode). The range is calculated from the argument parameter (rounded up to the next higher range). If the DMM autoranging is enabled when this command is received, autoranging is disabled and the requested range will be selected.

Voltmeter	Ohmmeter
200m	200
2	2k
20	20k
200	200k
500	2M

For the ohmmeter, any value larger than 2M will result in selection of the 2M range. If the value is less than 2M, it will be rounded up to the next higher range. For example, if 350 ohms is specified, the 2k range would be selected.

For the voltmeter, any value larger than 500 volts will result in selection of the 500 volt range. If the value is less than 500 volts, it will be rounded up to the next higher range. For example, if 25 volts is specified, the 200 volt range would be selected.

Note that separate range settings are maintained for each meter, and it is not necessary to be in a particular mode to set the range for that mode.

The command argument is: (O = OHMS)

- **input** - <nrf> (range: > 0, ≤ 1E34) { V | O }

Default: 500 V

Related Commands: :DCX:RANGe?

Command Syntax: :DCX:RANGe **input**

Example: `:DCX:RANGE 200V`

:DCX:RANGe?

Returns the current range setting for the DMM. If the DMM mode is consistent with the requested unit, the actual range setting will be returned. If not, the last range set with the :DCX:RANGe command will be returned (if the command has been issued since the last mode change), or the return value will reflect the range at the time of the last mode change (if the DCX:RANGe command has not been issued since the mode change).

The command argument is:

- **unit** - { V | O }

Response argument(s):

- **input** - <nrf>

Related Commands: :DCX:RANGe

Command Syntax: :DCX:RANGe? **unit**

Response Syntax: :DCX:RANGe **input** { V | O }

Example: `:DCX:RANGE? V`

Response: `:DCX:RANGE 200V`

:DCX:RDGRate

Sets the DMM internal reading rate. The available rates are 6 readings per second and 25 readings per second.

The command argument is:

- **rate** - { R6 | R25 }

Default: R6

Related Commands: :DCX:RDGRate?

Command Syntax: :DCX:RDGRate **rate**

Example: `:DCX:RDGRATE R6`

:DCX:RDGRate?

Returns the current DMM internal reading rate.

Response argument(s):

- **rate** - { R6 | R25 }

Related Commands: :DCX:RDGRate

Command Syntax: :DCX:RDGRate?

Response Syntax: :DCX:RDGRate **rate**

Example: `:DCX:RDGRATE?`

Response: `:DCX:RDGRATE R6`

:DCX:SCALe

Sets the scaling factor for the current DMM mode. The return value for the ":DCX:DMM?" (with function units, either F_O or F_V) is computed by the formula

Return Value = scale * (Measured + offset).

For example, with a scaling factor of 3.2, an offset of 1.5 volts and a raw measurement of 2.0 volts, the response to ":DCX:DMM? F_V" would be 11.2 V (11.2 = 3.2   (2.0 + 1.5)).

The command argument is:

- **scale** - <nrf> (range: ≥ -1E34, ≤ 1E34)

Default: 1.0

Related Commands: :DCX:SCALe?, :DCX:OFFset, :DCX:MODE

Command Syntax: :DCX:SCALe **scale**

Example: `:DCX:SCALE 3.2`

:DCX:SCALe?

Returns the currently programmed function scale factor.

Response argument(s):

- **scale** - <nrf>

Related Commands: :DCX:SCALe

Command Syntax: :DCX:SCALe?

Response Syntax: :DCX:SCALe **scale**

Example: `:DCX:SCALE?`

Response: `:DCX:SCALE 3.2`

:DCX:SET?

Returns the current setting of all the DCX parameters.

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands: (see all other query commands)

Command Syntax: :DCX:SET?

Response Syntax: :DCX:SET **settings**

Example:

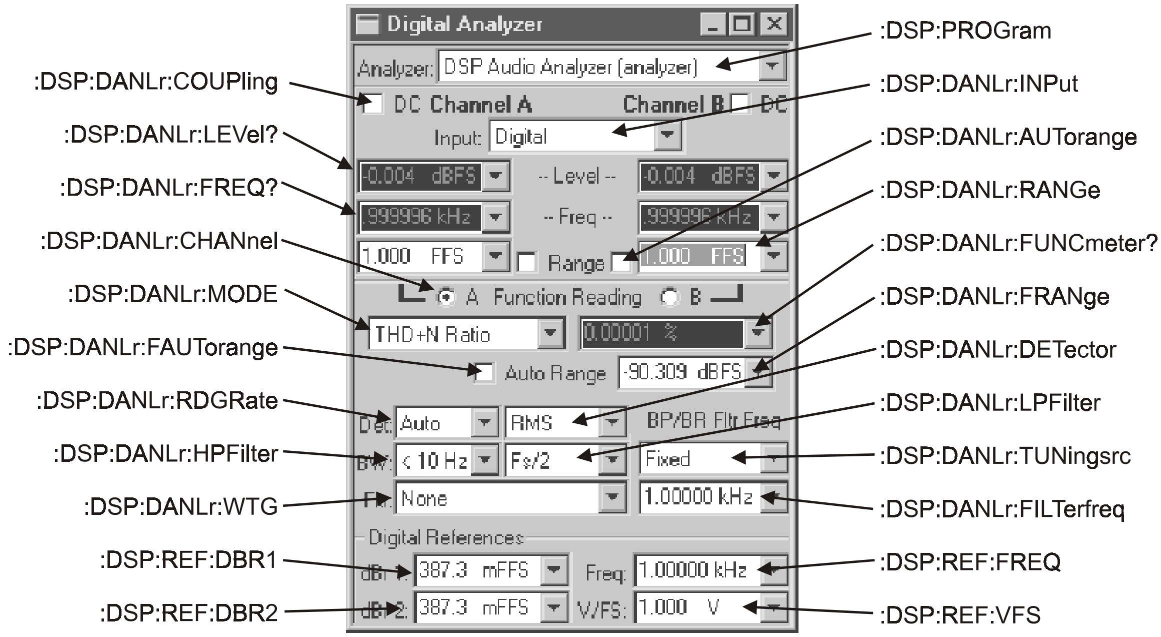
```
:DCX:SET?
```

Response:

```
:DCX:DCLEVEL DC1,0;DCLEVEL DC2,0;DCOUTPUT
DC1,OFF;DCOUTPUT DC2,OFF;MODE VOLTS;RANGE
500V;RANGE 2E+6O;AUTORANGE ON;OFFSET 0;SCALE
1;RDGRATE R6;GDELAY 0.05;PORT A,0;PORT
B,0;PORT C,0;:DCX:DIN:FORMAT TWOS;SCALE
1;RATE 32;:DCX:DOUT:DATA 0DEC;FORMAT
TWOS;SCALE 1
```

8. Digital Analyzer Commands

These commands set parameters for the digital analyzer. The header-path for the digital analyzer is :DSP:. These commands are invalid and will cause execution errors if used with a System Two model SYS-2022 (no DSP hardware) and will only work with analog input on SYS-2222.



Digital Analyzer

Analyzer: DSP Audio Analyzer (analyzer)

□ DC Channel A   Channel B □ DC

Input: Digital

-0.004 dBFS -- Level -- -0.004 dBFS

.999996 kHz -- Freq -- .999996 kHz

1.000 FFS □ Range □ 1.000 FFS

● A Function Reading ○ B

THD+N Ratio   0.00001 %

□ Auto Range   -90.309 dBFS

Det: Auto   RMS   BP/BR Fltr Freq

BW: < 10 Hz   Fs/2   Fixed

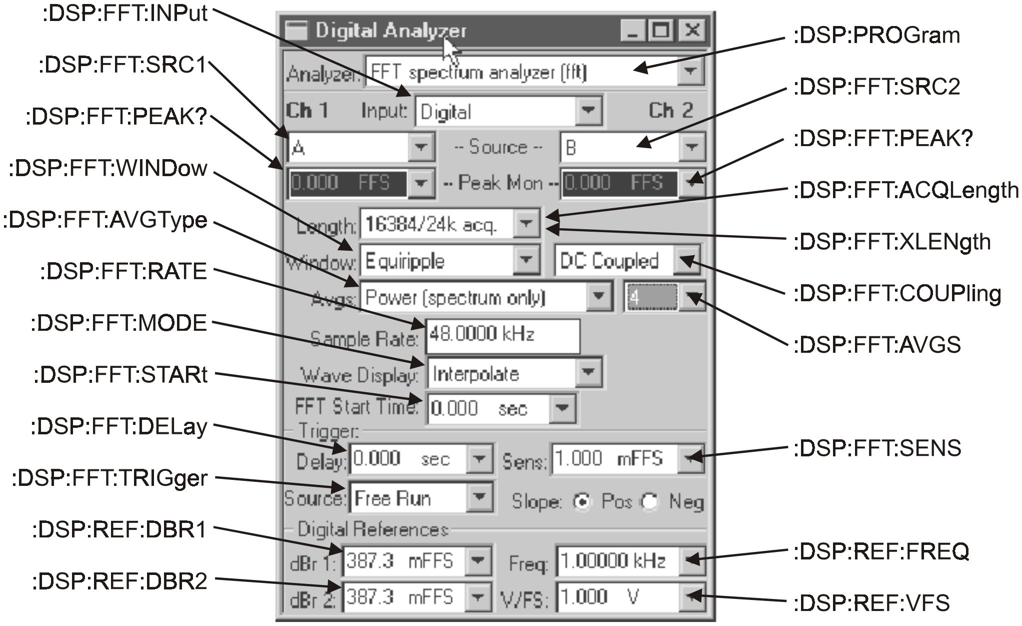
Fltr: None   1.00000 kHz

Digital References

dBr 1: 387.3 mFFS   Freq: 1.00000 kHz

dBr 2: 387.3 mFFS   V/FS: 1.000 V

Figure 8-1.Digital Analyzer control panel showing relationship with GPIB commands



Digital Analyzer

Analyzer: FFT spectrum analyzer (fft)

Ch 1   Input: Digital   Ch 2

A   -- Source --   B

0.000 FFS   -- Peak Mon --   0.000 FFS

Length: 16384/24k acq.

Window: Equiripple   DC Coupled

Avgs: Power (spectrum only)   4

Sample Rate: 48.0000 kHz

Wave Display: Interpolate

FFT Start Time: 0.000 sec

Trigger

Delay: 0.000 sec   Sens: 1.000 mFFS

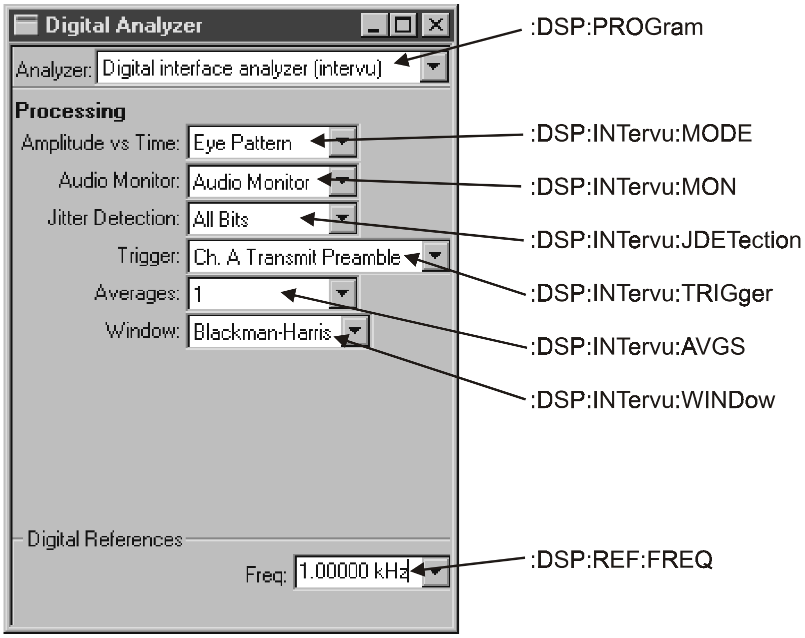
Source: Free Run   Slope: ● Pos ○ Neg

Digital References

dBr 1: 387.3 mFFS   Freq: 1.00000 kHz

dBr 2: 387.3 mFFS   V/FS: 1.000 V

Figure 8-2. Digital Analyzer panel, FFT mode



Analyzer: Digital interface analyzer (intervu)

Processing

Amplitude vs Time: Eye Pattern

Audio Monitor: Audio Monitor

Jitter Detection: All Bits

Trigger: Ch. A Transmit Preamble

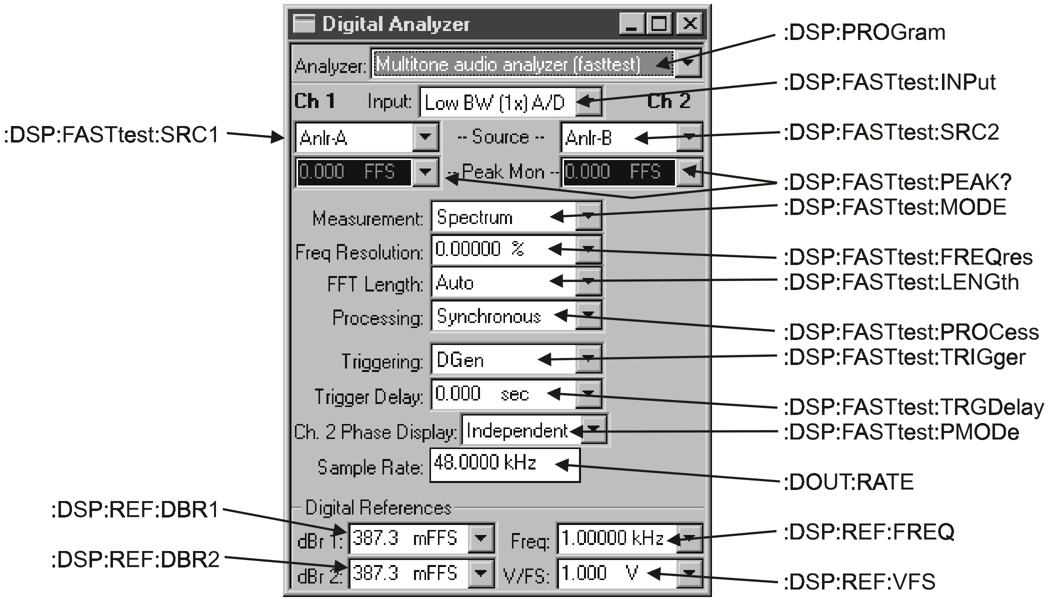
Averages: 1

Window: Blackman-Harris

Digital References

Freq: 1.00000 kHz

Figure 8-3. Digital Analyzer panel, Intervu mode



Analyzer: Multitone audio analyzer (fasttest)

Ch 1 Input: Low BW (1x) A/D Ch 2

Anlr-A -- Source -- Anlr-B

0.000 FFS -- Peak Mon -- 0.000 FFS

Measurement: Spectrum

Freq Resolution: 0.00000 %

FFT Length: Auto

Processing: Synchronous

Triggering: DGen

Trigger Delay: 0.000 sec

Ch. 2 Phase Display: Independent

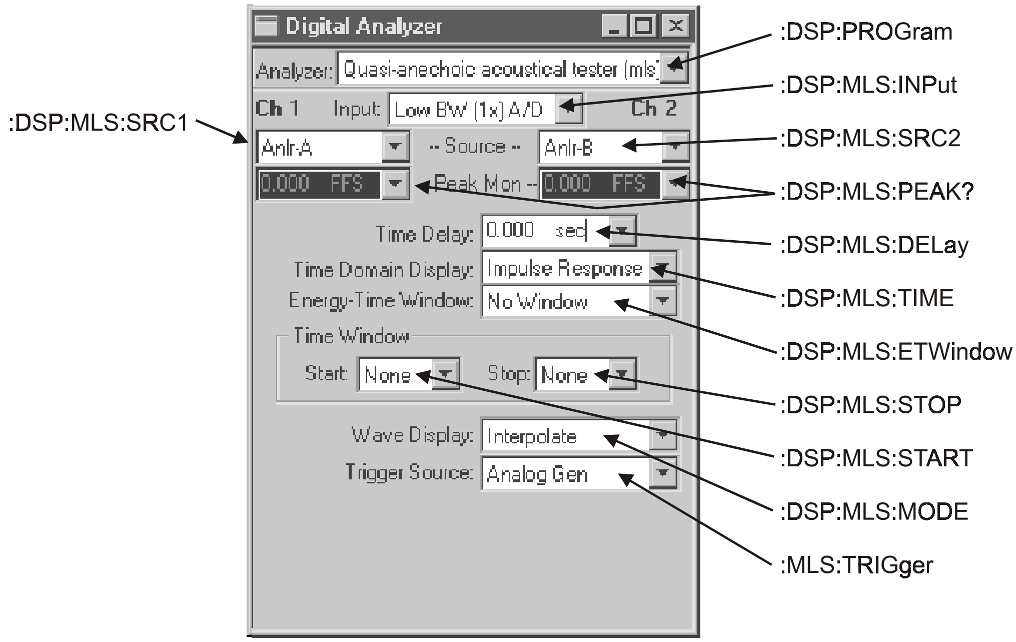
Sample Rate: 48.0000 kHz

Digital References

dBr 1: 387.3 mFFS Freq: 1.00000 kHz

dBr 2: 387.3 mFFS V/FS: 1.000 V

Figure 8-4. Digital Analyzer panel, FastTest mode



Digital Analyzer

Analyzer: Quasi-anechoic acoustical tester (mls)

Ch 1 Input: Low BW (1x) A/D Ch 2

Anlr-A -- Source -- Anlr-B

0.000 FFS -- Peak Mon -- 0.000 FFS

Time Delay: 0.000 sec

Time Domain Display: Impulse Response

Energy-Time Window: No Window

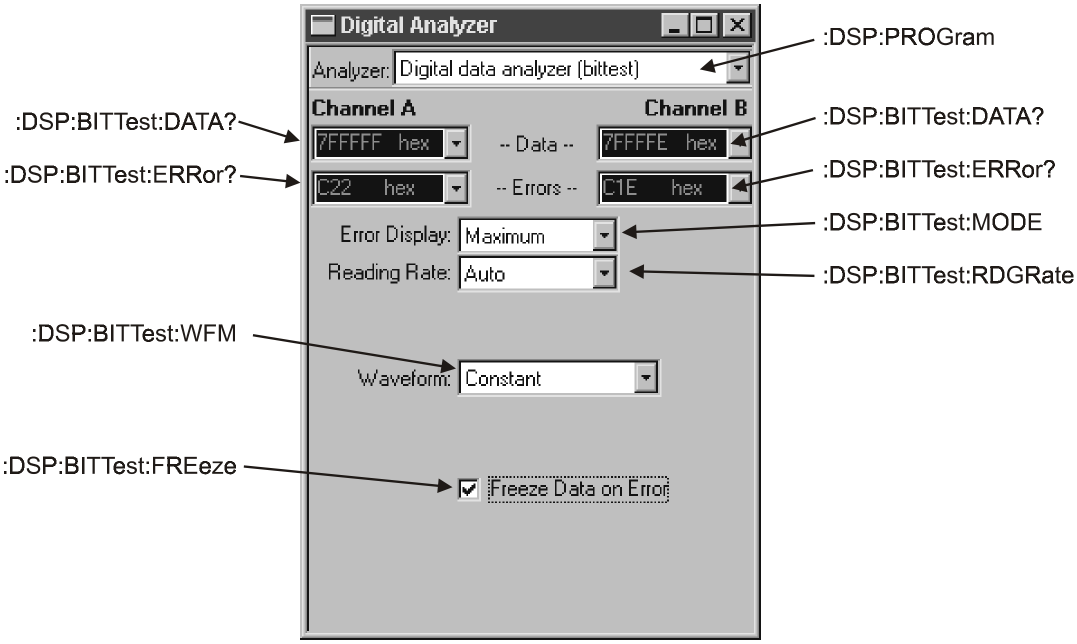
Time Window

Start: None Stop: None

Wave Display: Interpolate

Trigger Source: Analog Gen

Figure 8-5. Digital Analyzer panel, MLS mode



Digital Analyzer

Analyzer: Digital data analyzer (bittest)

Channel A Channel B

7FFFFF hex -- Data -- 7FFFFE hex

C22 hex -- Errors -- C1E hex

Error Display: Maximum

Reading Rate: Auto

Waveform: Constant

☑ Freeze Data on Error

Figure 8-6. Digital Analyzer panel, BITTEST mode

DSP Discussion

The DSP programs in System Two are Digital Audio Analyzer, FASTTEST, FFT, MLS, INTERVU, and BITTEST. The following table identifies which program is applicable for each instrument model:

Program	Model			
	2022	2222	2322	2300
FFT, FASTTEST, MLS, DANLR		X	X	X
INTERVU, BITTEST			X	X

- Digital Audio Analyzer (DANLr) - DSP version of ANLR.
- FASTTEST - Provides time or frequency domain views of the signal.
- FFT - Provides general-purpose time domain (oscilloscope) display of waveforms or frequency domain (spectrum analyzer) display of signals, including the received jitter signal on Dual Domain units.
- MLS - the Quasi-Anechoic Acoustical Tester (MLS) program uses Maximum Length Sequence (MLS) testing to characterize the linear response of acoustical and electronic devices.
- INTERVU - analyzes the AES/EBU or consumer digital interface input signal of Digital I/O-equipped System Two models via a 67 MHz sample rate A/D converter.
- BITTEST – analyzes the AES/EBU data stream for bit errors based on the type of waveform being generated by the Digital Audio Generator.

Digital Audio Analyzer (DANLR) and BITTEST are realtime analyzers.

Batch Mode DSP Program Operating States for FASTTEST, FFT, MLS, and Intervu

Batch mode DSP programs operate in two states, the SETUP state and the READING state. These two states are controlled by the :DSP:OPSTate command and are mutually exclusive. The application programmer must place the DSP program into the appropriate states in order to setup the DSP program and then acquire measurement data. The :DSP:OPSTate command is valid only when FASTTEST, FFT, MLS, or Intervu batch mode DSP programs are loaded and is invalid if received when no DSP program is loaded or when the ANALYZER DSP program is loaded. The :DSP:OPSTate command is implicitly set to the SETUP state when a batch mode DSP program is loaded by the :DSP:PROGram command.

Loading any batch-mode DSP program with the :DSP:PROG command or issuing the :DSP:OPSTate SETUP command places the DSP program in the SETUP state. The DSP program must be in this SETUP state before most DSP GPIB commands will be accepted by the instrument. Measurement acquisition or processing commands (:DSP:ACQX?, :DSP:XFRM?, :DSP:REPR?) and measurement data queries (such as :DSP:FASTtest:AMPL? or :DSP:FASTtest:PHASe?) are not valid if received while the DSP is in this state and will cause execution errors. Measurements of the DSP realtime peak meters are an exception to this rule. These peak meters can only be read in the SETUP state.

The :DSP:OPSTate READING command places the DSP program in the reading state. The DSP program must be in this READING state before batch mode measurement acquisition or processing can be initiated or measurement data can be read from the DSP. Most GPIB DSP setup commands and the realtime peak meter queries are not valid if received while the DSP is in this state and will cause execution errors (will be ignored).

The READING operating state is valid for the GPIB DSP commands listed below. Note that these commands are invalid in the SETUP operating state.

- :DSP:ACQX?
- :DSP:FASTtest:AMPL?
- :DSP:FASTtest:PHASe?
- :DSP:FFT:AMPL?
- :DSP:INTervu:AMPL?
- :DSP:INTervu:JITTer?
- :DSP:INTervu:LOWer?
- :DSP:INTervu:PROBability?
- :DSP:INTervu:UPPer?
- :DSP:MLS:AMPL?
- :DSP:MLS:PHASe?
- :DSP:OPSTate
- :DSP:REPRocess?
- :DSP:XFRM?
- :DSP:BATCH?

The application programmer must set the appropriate DSP operating state when sending commands to the System Two. The general rule is to send :DSP:OPSTate SETUP before sending commands to setup the DSP program prior to a signal acquisition. The :DSP:OPSTate READING command must then be sent to place the DSP in READING state, followed by :DSP:ACQX? and other reading queries. The :DSP:OPSTate SETUP command should then be sent again to place the DSP back into the SETUP state. It is good practice to check for execution errors when developing GPIB programs in order to detect improper DSP states when using the DSP setup, acquisition, and measurement query commands.

The following is an example of the correct command sequence for a spectrum sweep of the FFT using an internal log sweep of amplitudes of 31 frequency bins (1/3 octave intervals) from 20 Hz to 20 kHz:

```
:HEADer OFF
:DSP:PROGram FFT
REM Send additional FFT setup commands
:DSP:FFT:INPUT AD1X
:DSP:SRCParams FREQ, HZ, 20, 20000, 30, LOG
:DSP:TIMeout 10
:DSP:OPSTATE READing,AMP1,AMP2
:DSP:ACQX?
REM Read the ACQX? query response string
REM Request spectrum measurements for ch1 & ch2.
:DSP:BATCH? ASCII,AMP1,DBV,AMP2,DBV
REM Read the response string
:DSP:OPSTATE SETup
```

The following is an example of the correct command sequence for a FASTTEST DSP sweep of response and distortion with a table of frequencies for the multitone signal:

```
:HEADER OFF
:DSP:PROGRAM FASTTEST
:DSP:FASTTEST:INPUT AD1X
:DSP:SRCPARAMS FREQ,HZ,17.57,19998.04,30,ARBITRARY
:DSP:TABLE 1,ASCII,#017.57,23.43,29.29,41.01,52.73,64.45,82.03
,99.6,123.04,158.2,199.21,251.95,316.4,398.43,498.04,632.81,80
2.73,1001.91,1248.04,1599.6,1998.04,2501.95,3152.34,4001.95,49
98.04,6351.56,7998.04,10001.95,12498.04,16001.95,19998.04
:DSP:TSELECT 1
REM Setup Fasttest response and interchannel phase
measurements for ch1 & ch2.
:DSP:FASTTEST:MODE RESPONSE
:DSP:TIMEOUT 10
:DSP:OPSTATE READING,AMP1,AMP2
:DSP:ACQX?
:DSP:BATCH? ASCII,AMP1,V,AMP2,V,PHA2,DEG
REM Read the response string
REM Setup Fasttest distortion measurements for ch1 & ch2.
:DSP:OPSTATE SETUP
:DSP:FASTTEST:MODE DISTORTION
:DSP:OPSTATE READING,AMP1,AMP2
:DSP:REPROCESS?
REM Read the response string
:DSP:BATCH? ASCII,AMP1,V,AMP2,V
REM Read the response string
:DSP:OPSTATE SETUP
```

Command Interaction

The DSP goes through three distinct phases while operating. First, in the acquisition phase the acquire buffer is filled with time based data until filled to the appropriate length. Next, in the transform phase, an FFT is performed to convert the time based acquisition data into frequency based data. Finally, in the process phase any post-processing is performed on the frequency based data (see Figure 8-7).

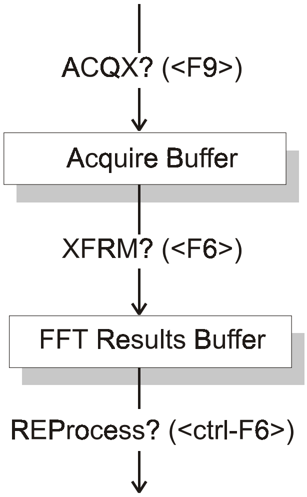


Figure 8-7. DSP flow diagram

The table on the following page shows which settings commands affect each phase (process commands). Blank entries in the table indicate that the process command (:DSP:ACQX?, :DSP:XFRM?, or :DSP:REPRocess?) is not affected by that setup command. For example, the :DSP:FASTtest:INPut command changes the input from analog to digital, but this will only affect data when a new acquisition is performed. If the input is changed and either the XFRM? or REPRocess? command is executed, the transform or reprocess will be done on the old data (already acquired). This will not automatically force a new acquisition.

	ACQX	XFRM	REPRocess
:DSP:SRCParams start, stop, steps, mode arguments	✔	✔	✔
:DSP:SRCParams instrument, parameter, units arguments	✔		
:DSP:FASTtest:FREQres	✔	✔	✔
:DSP:FASTtest:INPut	✔		
:DSP:FASTtest:LENGth	✔		
:DSP:FASTtest:MODE	✔	✔	✔
:DSP:FASTtest:PHASe	✔	✔	✔
:DSP:FASTtest:PROCess	✔	✔	
:DSP:FASTtest:SRC1	✔		
:DSP:FASTtest:SRC2	✔		
:DSP:FASTtest:TRGDelay	✔		
:DSP:FASTtest:TRIGger	✔		
:DSP:FFT:ACQLength	✔		
:DSP:FFT:AVGS	✔		
:DSP:FFT:AVGT	✔		
:DSP:FFT:COUPling	✔		
:DSP:FFT:DELay	✔		
:DSP:FFT:INPut	✔		
:DSP:FFT:MODE	✔	✔	✔
:DSP:FFT:SRC1	✔		
:DSP:FFT:SRC2	✔		
:DSP:FFT:STARt	✔	✔	
:DSP:FFT:TRIGger	✔		
:DSP:FFT:TSLope	✔		
:DSP:FFT:WINDow	✔	✔	
:DSP:FFT:XLENgth	✔	✔	
:DSP:INTervu:AVGS	✔		
:DSP:INTervu:JDETection	✔		
:DSP:INTervu:MODE	✔	✔	✔
:DSP:INTervu:MON			
:DSP:INTervu:TRIGger	✔		
:DSP:INTervu:WINDow	✔	✔	
:DSP:MLS:DELay	✔	✔	
:DSP:MLS:ETWindow	✔	✔	
:DSP:MLS:INPut	✔		
:DSP:MLS:MODE	✔	✔	✔
:DSP:MLS:SRC1	✔		
:DSP:MLS:SRC2	✔		
:DSP:MLS:STARt	✔	✔	
:DSP:MLS:STOP	✔	✔	
:DSP:MLS:TIME	✔	✔	
:DSP:MLS:TRIGger	✔		

:DSP:ACQX?

This query causes the DSP to do an acquisition and transform using the current DSP settings and to return a timeout status. The meter and channel selection is determined by the DSP program currently in memory and it's parameter settings. This command is valid only in OPSTate READing mode.

An execution error will be generated if there is no batch mode DSP program loaded (FFT, FASTTEST, MLS, or INTERVU) or the batch mode DSP program is in OPSTate SETup mode.

A query response will be generated when either the acquisition completes or the acquisition times out (see :DSP:TIMeout).

The response is the acquisition time-out status. If the response is 0 (zero), the command did NOT timeout. A 1 (one) response indicates the DSP acquisition timed out.

NOTE: Various run-time errors can be generated by the DSP during the execution of batch-mode commands. It is recommended to check the error queue (e.g., via :ERRN? command) after completion of the :DSP:ACQX?, :DSP:REPR?, and :DSP:XFRM? commands.

The response argument is:

- **status** - <nr1> (range: 0 or 1)

Related Commands: :DSP:XFRM?, :DSP:REPRocess?, :DSP:TIMeout

Command Syntax: :DSP:ACQX?

Response Syntax: :DSP:ACQX **status**

Example: `:DSP:ACQX?`

Response: `:DSP:ACQX 0`

:DSP:BATCh?

This query sweeps the batch-mode DSP program currently loaded and returns an entire batch sweep data array. The data to be read depends on the parameters of the :DSP:SRCParams command and the :DSP:OPSTate READing command. The batch-mode DSP programs are: FFT, FASTTEST, MLS, and Intervu.

If the :DSP:SRCParams "mode" parameter specifying the batch is LINear or LOG, the source data points are calculated at run time within this command. If the parameter is ARBitrary, the source data points are provided by the user in a table via the :DSP:TABLe command and the :DSP:TSELect command.

Note that the combination of channel and measurement must have been enabled with the :DSP:OPSTate READing command prior to issuing this command. One or more of the measurements specified by the :DSP:OPSTate READing parameters must be included in the arguments to the :DSP:BATCh? query.

The source domain and unit, start, stop, number of points, and increment algorithm are specified with the :DSP:SRCParams command.

The BATCH? query interacts with the :DSP:SRCParams commands when parameters are set up to perform frequency domain spectrum sweeps. Subsequent sweeps require re-use of the ACQX?, XFRM?, or REPROCESS? commands in order to assure correct handling of peak-picking with LOG or LIN sweeps.

The response is a sequence of source/data1/.../dataN tuples. The number of "data" values per tuple and their sequence is determined by the sequence of measurements specified in the command invocation. The maximum number of data values per tuple is 4.

The response data consists of either a definite length Arbitrary Block Response Data (if the requested format is BINary or RBINary), or a sequence of comma-separated floating point numbers in ASCII representation. BINary and RBINary formats specify each data value in a tuple as a four-byte single floating point number formatted the same but with different ordering of the bytes. BINary is specified by the IEEE standard 488.2 with the most significant byte sent first, commonly called “big-endian.” RBINary is a byte-reversed version of BINary with the most significant byte sent last, per standard Intel x86 floating point representation, commonly called “little-endian.”

IEEE Std 754-1985 and IEEE Std 488.2-1992 specify the BINARY format. See Appendix F for a detailed description of this specification.

The 'meas' parameter must be appropriate for the DSP program currently loaded, as shown below in the table:

Table 8-1. Measurement parameters valid for each DSP program used with :DSP:BATCH?

	FASTTEST	FFT	INTERVU	MLS
AMP1	✓	✓		✓
AMP2	✓	✓		✓
PHA1	✓			✓
PHA2	✓			✓
AMPLitude			✓	
JITTer			✓	
LOWer			✓	
UPPer			✓	
PROBability			✓	

The table below shows the valid units for each DSP program that is loaded and the parameter that is being measured.

Table 8-2. Units valid for each DSP program used with BATCH?

DSP Program	Measurement Parameter	Input Type: INPUT	Source Type: SRC1 & SRC2	Valid Units
FASTTEST & FFT	AMP1 or AMP2	AD1X	A, AMPLitude, B, GENA, GENB	DBM, DBRA, DBRB, DBU, DBV, V
	AMP1 or AMP2	AD1X	RATio	DB, PCT, X_Y
	AMP1 or AMP2	AD1X	JITTer	UI
	AMP1 or AMP2	DIGital	A, B	DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFS, V, VP, VPP
	PHA1 or PHA2	AD1X or DIGital	any	DEG
MLS	AMP1 or AMP2	AD1X	A, AMPLitude, B, GENA, GENB	DBM, DBRA, DBRB, DBU, DBV, V
	AMP1 or AMP2	DIGital	A, B	DBFS, DBU, DBV, FFS, PCTFS, V, VP, VPP
	PHA1 or PHA2	AD1X or DIGital	any	DEG
INTERVU	AMPLitude	-	-	DBV, V
	LOWer or UPPer	-	-	DBM, DBRA, DBRB, DBU, DBV, V
	JITTer	-	-	SEC, UI
	PROBability	-	-	DB, PCT, X_Y

The query arguments are:

- **format** – { ASCii | BINary | RBINary }
- **meas1** – { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **unit1** – { DBFS | DBM | DBRA | DBRB | DBR1 | DBR2 | DBU | DBV | DEG | FFS | PCTFS | V | VP | VPP | DB | PCT | X_Y | UI }
- **meas2** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }

- **unit2** - { DBFS | DBM | DBRA | DBRB | DBR1 | DBR2 | DBU | DBV | DEG | FFS | PCTFS | V | VP | VPP | DB | PCT | X_Y | UI }
- **meas3** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **unit3** - { DBFS | DBM | DBRA | DBRB | DBR1 | DBR2 | DBU | DBV | DEG | FFS | PCTFS | V | VP | VPP | DB | PCT | X_Y | UI }
- **meas4** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **unit4** - { DBFS | DBM | DBRA | DBRB | DBR1 | DBR2 | DBU | DBV | DEG | FFS | PCTFS | V | VP | VPP | DB | PCT | X_Y | UI }

The response arguments are:

- **format** – { ASCII | BINARY | RBINARY }
- **pointcount** - <nr1>
- **data** - <definite length arb block data>

Related Commands: :DSP:SRCParams, :DSP:OPSTate, :DSP:TABLe

Command Syntax: :DSP:BATCH? format, meas1, unit1 [,meas2, unit2, meas3, unit3, meas4, unit4]

Response Syntax: :BATCH format, pointcount, data

Example1:

```
:DSP:BATCH? ASCII,AMP1,DBFS
```

Response1:

```
ASCII,10,10.0021,-170.025,21.5435,-
170.387,46.4172,-166.733,99.9985,-
165.625,215.441,-166.438,464.161,-
165.308,1000,-10.2539,2154.44,-
11.0212,4641.59,-161.3,10000,-166.133
```

Example2:

```
:DSP:BATCH? ASCII,AMP1,DBFS
```

Response2:

```
ASCII,12,0.00100004,-
0.545619,0.00109995,0.437011,0.00120004,1.09
21,0.00129996,1.404,0.00140005,1.17513,0.001
49996,0.567096,0.00160006,-
0.414736,0.00169997,-1.07711,0.00180006,-
1.40101,0.00189998,-1.18798,0.00200007,-
0.588427,0.00209999,0.392337
```

Example3:

```
:DSP:BATCH? ASCII,AMP1,DBFS
```

Response3:

```
ASCII,10,10.0021,-170.025,21.5435,-
170.387,46.4172,-166.733,99.9985,-
165.625,215.441,-166.438,464.161,-
165.308,1000,-10.2539,2154.44,-
11.0212,4641.59,-161.3,10000,-166.133
```

:DSP:BITTest

Compound header for BITTEST Digital Data Analyzer DSP program commands.

:DSP:BITTest:CLEar

This command is used to clear the Bittest Totalize/Maximum mode error counters. Note that the error count in the Normal mode is not affected by this command.

This command will not clear the error counters if "Freeze Data On Error" mode is enabled and the counters are frozen. In this case, disable the "Freeze Data On Error" mode with the :DSP:BITTest:FREeze OFF command prior to using this command.

Related Commands: :DSP:BITTest:ERRor?, :DSP:BITTest:FROZen?; :DSP:BITTest:FREeze

Command Syntax: :DSP:BITTest:CLEar

Example: `:DSP:BITTEST:CLEAR`

:DSP:BITTest:DATA?

This query returns the current data word from the indicated digital audio channel.

The data may be encoded in either decimal units (DEC) or hexadecimal (HEX) units. If the data is frozen when the "Freeze Data On Error" mode is enabled, the data may be read one time. Subsequent reads with this command will return a special not-a-number value (214783648DEC or 80000000HEX) to indicate that the frozen data has been previously read.

The command arguments are:

- **channel** - { A | B }
- **unit** - { DEC | HEX }

Related Commands: :DSP:BITTest:ERRor?, :DSP:BITTest:FREeze, :DSP:BITTest:FROZen?

Command Syntax: :DSP:BITTest:DATA? **channel**, **unit**

Example: `:DSP:BITTEST:DATA? A,HEX`

Response: `:DSP:BITTEST:DATA 7FFFFEHEX`

:DSP:BITTest:ERRor?

This query returns the current error count for the selected digital audio channel.

Received data is measured to determine if it matches the data transmitted. Only the number of bits selected by the :DIN:RESolution command will be analyzed. This comparison is done with algorithms which are not sensitive to delay between the digital generator and the BITTEST analyzer. The number of errors in the received data per measurement interval are counted independently for each channel. The :DSP:BITTest:MODE command selects the type of analysis to be performed.

Error measurements are based on the waveform selected by the :DSP:BITTest:WFM command. The random, sine, and walking bit waveforms compare each successively received sample with the algorithm used to generate the signal. Two samples are required before the error analysis synchronizes with the signal, so errors may be counted during the initial measurement interval even though no errors actually exist.

The Random waveform selection on BITTEST requires either a compatible real-time random noise signal or the output of a digital reproducer playing a recording previously made from compatible signals.

The data may be encoded in either decimal units (DEC) or hexadecimal (HEX) units. If the error count is frozen when the "Freeze Data On Error" mode is enabled, the error count may be read one time. Subsequent reads with this command will return a special not-a-number error count value (214783648DEC or 80000000HEX) to indicate that the frozen error count has been previously read.

The command arguments are:

- **channel** - { A | B }
- **unit** - { DEC | HEX }

Related Commands: :DSP:BITTest:DATA?, :DSP:BITTest:FREeze, :DSP:BITTest:FROZen?, :DSP:BITTest:WFM

Command Syntax: :DSP:BITTest:ERRor? **channel**, **unit**

Example: `:DSP:BITTEST:ERROR? A,HEX`

Response: `:DSP:BITTEST:ERROR F91HEX`

:DSP:BITTest:FREeze

This command sets the state of the BITTEST “Freeze Data On Error” mode.

If ON, the response to the :DSP:BITTest:DATA? query will be the data values received when the first error occurred on either channel, and the response to the :DSP:BITTest:ERRor? query will be the frozen error counts.

If OFF, the response to the :DSP:BITTest:DATA? will continue updating at the present reading rate, independent of any errors detected.

This command must be set to OFF in order to clear the frozen error data with the :DSP:BITTest:CLEar command.

The command argument is:

- **freezestate** – { OFF | ON }

Default: OFF

Related Commands: :DSP:BITTest:FROZen?, :DSP:BITTest:ERRor?, :DSP:BITTest:DATA?

Command Syntax: :DSP:BITTest:FREeze **freezestate**

Example: `:DSP:BITTEST:FREEZE ON`

:DSP:BITTest:FREeze?

This query returns the state of the BITTEST “Freeze Data On Error” mode.

Related Commands: :DSP:BITTest:FREeze

Command Syntax: :DSP:BITTest:FREeze?

Example: `:DSP:BITTEST:FREEZE?`

Response: `:DSP:BITTEST:FREEZE ON`

:DSP:BITTest:FROZen?

This query returns the data-frozen status of BITTEST if the “Freeze Data On Error” mode is enabled. The response is 0 if error measurements are not frozen or 1 if error measurements are frozen due to error detection during the last measurement interval on either input channel.

Related Commands: :DSP:BITTest:FREeze, :DSP:BITTest:ERRor?, :DSP:BITTest:DATA?

Command Syntax: :DSP:BITTest:FROZen?

Example: `:DSP:BITTEST:FROZEN?`

Response: `:DSP:BITTEST:FROZEN 1`

:DSP:BITTest:MODE

The MODE command selects the type of error count analysis to be performed on both input channels and resets the error counter in Totalize and Maximum modes.

In the Normal mode, the number of errors detected during the last measurement interval (1/4 second, 1/8 second, 1/16 second) is counted and provided by the :DSP:BITTEST:ERRor? query.

The Maximum mode latches the largest error count that occurs during any single measurement interval and provides this count as the response to the :DSP:BITTEST:ERRor? query.

The Totalize mode provides a running total of all errors accumulated since the Totalize mode was selected or the :DSP:BITTest:CLEar command was received interval and provides this count as the response to the :DSP:BITTEST:ERRor? query.

The command arguments are:

- **mode** { NORMal | MAXimum | TOTalize }

Default: NORMal

Related Commands: :DSP:BITTest:CLEar, :DSP:BITTest:ERRor?, :DSP:BITTest:FREeze

Command Syntax: :DSP:BITTest:MODE **mode**

Example: `:DSP:BITTEST:MODE TOTALIZE`

:DSP:BITTest:MODE?

This query returns the current error count mode.

The response argument are:

- **mode** { NORMal | MAXimum | TOTalize }

Related Commands: :DSP:BITTEST:MODE

Command Syntax: :DSP:BITTEST:MODE?

Example: `:DSP:BITTEST:MODE?`

Response: `:DSP:BITTEST:MODE TOTALIZE`

:DSP:BITTest:RDGRate

This command specifies the update rate (Auto-4/s-8/s-16/s) of the BITTEST Data and Error readings until an error occurs. The AUTO setting specifies 4 readings per second, equivalent to specifying R4.

The command argument is:

- **rate** { AUTO | R4 | R8 | R16 }

Default: **AUTO**

Related Commands: none

Command Syntax: :DSP:BITTest:RDGRate **rate**

Example: `:DSP:BITTEST:RDGRATE R16`

:DSP:BITTest:RDGRate?

This command returns the update rate setting (Auto-4/s-8/s-16/s) of the BITTEST Data and Error readings.

Related Commands: :DSP:BITTest:RDGRate

Command Syntax: :DSP:BITTest:RDGRate?

Example: `:DSP:BITTEST:RDGRATE?`

Response: `:DSP:BITTEST:RDGRATE R16`

:DSP:BITTest:SET?

This query returns BITTEST analyzer settings for FREEZE, MODE, RDGRATE, and WFM.

Related Commands: all BITTEST settings

Command Syntax: :DSP:BITTEST:SET?

Example: `:DSP:BITTEST:SET?`

Response:

```
:DSP:BITTEST:FREEZE OFF;MODE NORMAL;RDGRATE
AUTO;WFM SINE
```

:DSP:BITTest:WFM

This command selects the type of waveform that BITTest expects to receive for error analysis.

Signal generation must be coordinated with BITTest analysis. BITTest can analyze five different digital-domain waveforms generated by the System Two digital generator, or previously digitally recorded from the digital generator. These waveforms are Random (:DGEN:WFM SPCial, RANDom), Constant (:DGEN:WFM SPCial,CONStant), Walking-1 (:DGEN:WFM SPCial, WLK1), Walking-0 (:DGEN:WFM SPCial, WLK0), and Sine (:DGEN:WFM SINE, SINE). When testing real-time devices, the same waveform must be selected on both the Digital Generator and the BITTest analyzer. When analyzing a previously-recorded or distantly-generated signal, the BITTest waveform must correspond to the waveform being analyzed.

Constant (:DSP:BITTest:WFM CONStant)

The Special Constant digital generator waveform (:DGEN:WFM SPCial,CONSTant) generates a continuous stream of data samples at the same fixed value. This is the digital equivalent of a DC voltage source. The data word to be output is controlled by the :DGEN:AMPLitude command. NOTE: Dither must be turned off (:DGEN:DITHertype NONE) for BITTest analysis to function with the Constant waveform. Constant mode aids in the investigation of data-dependent errors in digital systems. If a Constant waveform is digitally recorded and then reproduced, or is transmitted from a distant System Two, the local System Two Digital Generator Waveform must be set to Special Constant (:DGEN:WFM SPCial,CONSTant) and the Amplitude (:DGEN:AMPL) must be exactly the same as was recorded or is being transmitted even though the locally-generated waveform is not being used. The digital output resolution (:DOUT:RESolution) must at least equal the digital input resolution (:DIN:RESolution) for BITTest analysis to function properly.

Random (:DSP:BITTest:WFM RANDom)

On the Special Random digital generator waveform (:DGEN:WFM SPCial,RANDom) selects a unique psuedo-random noise waveform designed for use with BITTest. Sample amplitudes are uniformly distributed between plus and minus full scale. The amplitude is not adjustable. This signal is intended for error rate testing of communications links, AES/EBU interfaces, and digital recorders. It provides the most thorough error testing of any of the waveforms. Digital generator dither is automatically turned off in this waveform, so the digital generator Dither setting is not important. The digital output resolution (:DOUT:RESolution) must at least equal the digital input resolution (:DIN:RESolution) for BITTest analysis to function properly.

Walking Bit (:DSP:BITTest:WFM WLK1 and WLK0)

There are two walking bit modes. The digital generator Walking-1 waveform (:DGEN:WFM SPCial,WLK1) sets all bits to 0 except one bit that is set to 1. This single high bit is continuously incremented from lower to upper bits. When it reaches the MSB it wraps around to the LSB of the selected word width (:DOUT:RESolution). The digital generator Walking-0 waveform (:DGEN:WFM SPCial,WLK0) sets all bits to 1 except one bit that is set to 0. This single low bit is continuously incremented from lower to upper bits. When it reaches the MSB it wraps around to the LSB of the selected word width (:DOUT:RESolution). The amplitudes of the walking bit signals are not adjustable. Digital generator dither is automatically turned off in these waveforms, so the Dither setting is not important. The digital output resolution (:DOUT:RESolution) must at least equal the digital input resolution (:DIN:RESolution) for BITTEST analysis to function properly.

Sinewave (:DSP:BITTest:WFM SINE)

The digital generator Sine waveform (:DGEN:WFM SINE,SINE) allows error checking of digital audio devices with gain or equalization. It is difficult to test such devices with bit-pattern-based test signals. The sinewave retains its shape passing through any linear digital device. Thus, only the sinewave is useful for error testing when the device under test has gain or equalization. The sinewave analysis technique used by BITTest is not sensitive to the dither levels of System Two's digital generator, so Dither may be turned on or off (:DGEN:DITHertype NONE). Any high-quality digital domain sinewave is compatible with System Two's BITTest analyzer.

The command arguments are:

- **wfm** – { CONStant | RANDom | SINE | WLK0 | WLK1 }

Default: SINE

Related Commands: :DGEN:AMPL, :DGEN:WFM, :DGEN:DITHERTYPE, :DOUT:RESolution, :DIN:RESolution, :DSP:BITTest:WFM?

Command Syntax: :DSP:BITTest:WFM **wfm**

Example: `:DSP:BITTEST:WFM CONSTANT`

:DSP:BITTest:WFM?

This query returns the present BITTEST waveform setting. See the discussion of waveforms for the :DSP:BITTest:WFM command above for more information.

The response argument is:

- **wfm** – { SINE | CONStant | RANDom | WLK0 | WLK1 }

Related Commands: :DSP:BITTest:WFM

Command Syntax: :DSP:BITTest:WFM?

Example: `:DSP:BITTEST:WFM?`

Response: `:DSP:BITTEST:WFM CONSTANT`

:DSP:DANLr

Compound header for Digital Audio Analyzer DSP program commands.

:DSP:DANLr:AUTorange

This command enable or disables autoranging of the indicated channel(s) level meter. The valid channel selections are channel A (A), channel B (B) or both channels A and B (AB). The valid settings for autoranging are OFF and ON.

If the range command (:DSP:DANLr:RANGe) is received for a channel that is currently in autorange, autoranging will be turned OFF.

The command argument is:

- **channel** - { AB | A | B }
- **state** - { ON | OFF }

Default: AB, ON

Related Commands: :DSP:DANLr:AUTorange?

Command Syntax: :DSP:DANLr:AUTorange **channel, state**

Example: `:DSP:DANLR:AUTORANGE A,OFF`

:DSP:DANLr:AUTorange?

This query returns the autoranging setting for the indicated DSP Digital input. The valid channel settings are A and B. The valid autorange responses are OFF and ON.

Command argument:

- **channel** - { A | B }

Response argument(s):

- **channe**l - { A | B }
- **state** - { ON | OFF }

Related Commands: :DSP:DANLr:AUTorange

Command Syntax: :DSP:DANLr:AUTorange? **channel**

Response Syntax: :DSP:DANLr:AUTorange **channel, state**

Example: `:DSP:DANLR:AUTORANGE? A`

Response: `:DSP:DANLR:AUTORANGE A,OFF`

:DSP:DANLr:CHANnel

This command specifies the channel (A or B) to be used for the main Function meter. The input is selected by the :DSP:DANLr:INPut command.

The command argument is:

- **subframe** - { A | B }

Default: A

Related Commands: :DSP:DANLr:INPut, :DSP:DANLr:CHANnel?

Command Syntax: :DSP:DANLr:CHANnel **subframe**

Example: `:DSP:DANLR:CHANNEL A`

:DSP:DANLr:CHANnel?

This query returns the currently selected channel for the Function meter.

Response argument(s):

- **subframe** - { A | B }

Related Commands: :DSP:DANLr:CHANnel

Command Syntax: :DSP:DANLr:CHANnel?

Response Syntax: :DSP:DANLr:CHANnel **subframe**

Example: `:DSP:DANLR:CHANNEL?`

Response: `:DSP:DANLR:CHANNEL A`

:DSP:DANLr:COUPling

This command sets the coupling for the indicated channel(s). The valid settings for coupling are AC and DC.

The command arguments are:

- **channel** - { AB | A | B }
- **coupling** - { AC | DC }

Default: AB, AC

Related Commands: :DSP:DANLr:COUPling?

Command Syntax: :DSP:DANLr:COUPling **channel, coupling**

Example: `:DSP:DANLR:COUPLING A,AC`

:DSP:DANLr:COUPling?

This query returns the coupling setting for the indicated channel. The valid responses are AC and DC.

The command argument is:

- **channel** - { A | B }

Response argument(s):

- **response_channel** - { A | B }
- **coupling** - { AC | DC }

Related Commands: :DSP:DANLr:COUPling

Command Syntax: :DSP:DANLr:COUPling? **channel**

Response Syntax: :DSP:DANLr:COUPling **response_channel, coupling**

Example: `:DSP:DANLR:COUPLING? A`

Response: `:DSP:DANLR:COUPLING A,AC`

:DSP:DANLr:DETector

This command sets the detector type of the function meter for all measurement functions.

The command arguments are:

- **type** - { FRMS | QPEak | RMS }

Default: FRMS

Related Commands: :DSP:DANLr:DETector?

Command Syntax: :DSP:DANLr:DETector **type**

Example: `:DSP:DANLR:DETECTOR FRMS`

:DSP:DANLr:DETector?

This query returns the detector type of the function meter.

Response argument(s):

- **type** - { FRMS | QPEak | RMS }

Related Commands: :DSP:DANLr:DETector

Command Syntax: :DSP:DANLr:DETector?

Response Syntax: :DSP:DANLr:DETector **type**

Example: `:DSP:DANLR:DETECTOR?`

Response: `:DSP:DANLR:DETECTOR PEAK`

:DSP:DANLr:FAUTorange

This command enables or disables the function meter autoranging. If the :DSP:DANLr:FRANge command is received while a channel is in autorange, autoranging will be turned OFF for that channel.

The command argument is:

- **state** - { ON | OFF }

Default: ON

Related Commands: :DSP:DANLr:FRANge, :DSP:DANLr:FAUTorange?

Command Syntax: :DSP:DANLr:FAUTorange **state**

Example: `:DSP:DANLR:FAUTORANGE ON`

:DSP:DANLr:FAUTorange?

This query returns the state of the function meter autoranging. The possible states are ON and OFF.

Response argument(s):

- **state** - { ON | OFF }

Related Commands: :DSP:DANLr:FAUTorange

Command Syntax: :DSP:DANLr:FAUTorange?

Response Syntax: :DSP:DANLr:FAUTorange **state**

Example: `:DSP:DANLR:FAUTORANGE?`

Response: `:DSP:DANLR:FAUTORANGE ON`

:DSP:DANLr:FILTerfreq

This command specifies the bandpass/bandreject filter frequency when the tuning source is FIXed (see :DSP:DANLr:TUNingsrc) and when the current mode is either crosstalk (XTALk), THD+N ratio (THDRatio), THD+N ampl (THDAmpl), or bandpass (BP).

If the current mode is crosstalk, THD+N ratio, THD+N ampl, or bandpass and the tuning source is not fixed (FIXed), this command will force the tuning source to fixed (FIXed).

The filter is tunable across the audio spectrum from 0.04% to 42% of the sample rate, for example, 20 Hz to 20 kHz at a 48 kHz sample rate.

The command argument is:

- **frequency** - <nrf> (range: ≥ 0.04%, ≤ 42% of the sample rate) { HZ | CENT | DECS | DHZ | **DPCT** | DPPM | F_R | OCTS | PCTHz }

Default: 1000.0 HZ

Related Commands: :DSP:DANLr:TUNingsrc, :DSP:DANLr:FILTerfreq?

Command Syntax: :DSP:DANLr:FILTerfreq **frequency**

Example: `:DSP:DANLR:FILTERFREQ 3500 HZ`

:DSP:DANLr:FILTerfreq?

This query returns the bandpass/bandreject filter frequency setting in the specified units (regardless of the tuning source setting).

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R |

OCTS | PCTHz }

Related Commands: :DSP:DANLr:FILTerfreq

Command Syntax: :DSP:DANLr:FILTerfreq? **units**

Response Syntax: :DSP:DANLr:FILTerfreq **frequency**

Example: `:DSP:DANLR:FILTERFREQ? HZ`

Response: `:DSP:DANLR:FILTERFREQ 3500HZ`

:DSP:DANLr:FRANge

This command sets the function meter gain ranging. If this command is received when the function meter ranging is set to autorange, the autoranging will be turned off and the function meter will be set to the specified range.

The System Two will determine the appropriate range setting from this value. If a setting value does not match a range exactly the next higher range will be used. The ranges are:

FFS	%FS	dBFS
1.0	100	0.000
500 m	50	-6.021
250 m	25	-12.041
125 m	12.5	-18.062
62.5 m	6.25	-24.082
31.25 m	3.125	-30.103
15.63 m	1.563	-36.124
7.813 m	0.7813	-42.144
3.906 m	0.3906	-48.165
1.953 m	0.1953	-54.185
976.6 µ	0.09766	-60.206
488.3 µ	0.04883	-66.227
244.1 µ	0.02441	-72.247
122.1 µ	0.01221	-78.268
61.04 µ	0.00610	-84.288
30.52 µ	0.00305	-90.309

The command argument is:

- **range** - <nrf> (range: see table above) { FFS | DBFS | DBR1 | DBR2 | PCTFs }

Default: 1 FFS

Related Commands: :DSP:DANLr:FRANge?

Command Syntax: :DSP:DANLr:FRANge **range**

Example: `:DSP:DANLR:FRANGE -20DBFS`

:DSP:DANLr:FRANge?

This query returns the current function meter gain range setting.

The command argument is:

- **units** - { FFS | DBFS | DBR1 | DBR2 | PCTFs }

Response argument(s):

- **range** - <nrf> {FFS | DBFS | DBR1 | DBR2 | PCTFs}

Related Commands: :DSP:DANLr:FRANge

Command Syntax: :DSP:DANLr:FRANge? **units**

Response Syntax: :DSP:DANLr:FRANge **range**

Example: `:DSP:DANLR:FRANGE? DBFS`

Response: `:DSP:DANLR:FRANGE -18.0618DBFS`

:DSP:DANLr:FREQ?

This query specifies the Digital Domain Audio Analyzer Frequency Meter response units and returns the frequency measurement for the indicated channel.

The first parameter in the response is the next available measurement.

The DSP Digital Analyzer Frequency Meter is enabled or disabled by the algorithm parameter of the :SETTLING:DANLr command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

The command arguments are:

- **channel** - { A | B }
- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **freq** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :SETTling:DSP:DANLr

Command Syntax: :DSP:DANLr:FREQ? **channel, units**

Response Syntax: :DSP:DANLr:FREQ? **freq, settle_timeout**

Example: `:DSP:DANLR:FREQ? A,HZ`

Response: `:DSP:DANLR:FREQ 4814.39HZ,0`

:DSP:DANLr:FUNCmeter?

This query returns the measurement from the function meter in the specified units. If the specified units are not appropriate for the current function meter mode and type of input, then an argument error will be generated. The appropriate units for each Function meter mode and type of input are listed in Table 8-3.

Table 8-3. Valid units for :DSP:DANLR:FUNCmeter? quwry

Mode	INPUT	Units
Amplitude, Bandpass, THD+N Absolute	AD1X	DBM, DBRA, DBRB, DBU, DBV, V
	DIGital	DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFS, V, VP, VPP
2-Channel Ratio, Crosstalk, THD+N Ratio	AD1X and DIGital	DB, PCT, X_Y

The first parameter in the response is the next available measurement.

The Digital Domain Audio Analyzer Function Meter settling is enabled or disabled by the algorithm parameter of the :SETTLING:DANLr command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

Command argument:

- **units** - { DB | DBFS | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCT | PCTFs | V | VP | VPP | X_Y }

Response argument(s):

- **rdg** - <nrf>
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DSP:DANLr:MODE

Command Syntax: :DSP:DANLr:FUNCmeter? **units**

Response Syntax: :DSP:DANLr:FUNCmeter rdg, **settle_timeout**

Example: `:DSP:DANLR:FUNCMETER? FFS`

Response: `:DSP:DANLR:FUNCMETER 0.168785FFS,0`

:DSP:DANLr:HPFilter

This command specifies the frequency of the high pass filter prior to the function meter. This filter does not affect channel A and B level meters or the frequency meters. There are four filter settings: 10 Hz (F10), 22 Hz (F22), 100 Hz (F100), and 400 Hz (F400).

The command argument is:

- **frequency** - { F10 | F22 | F100 | F400 }

Default: F10

Related Commands: :DSP:DANLr:HPFilter?

Command Syntax: :DSP:DANLr:HPFilter **frequency**

Example: `:DSP:DANLR:HPFILTER F10`

:DSP:DANLr:HPFilter?

This query returns the current setting of the high pass filter.

Response argument(s):

- **frequency** - { F10 | F22 | F100 | F400 }

Related Commands: :DSP:DANLr:HPFilter

Command Syntax: :DSP:DANLr:HPFilter?

Response Syntax: :DSP:DANLr:HPFilter **frequency**

Example: `:DSP:DANLR:HPFILTER?`

Response: `:DSP:DANLR:HPFILTER F10`

:DSP:DANLr:INPut

This command sets the input source for the indicated channel(s). The potential input sources are: Digital (DIGital) and Low Bandwidth A/D (AD1X).

The command arguments are:

- **source** - { DIGital | AD1X }

Default: DIGital

Related Commands: :DSP:DANLr:INPut?

Command Syntax: :DSP:DANLr:INPut **source**

Example: `:DSP:DANLR:INPUT AD1X`

:DSP:DANLr:INPut?

This query returns the current input source of the indicated channel. The possible responses are: Digital (DIGital) and Low Bandwidth A/D (AD1X).

Response argument(s):

- **source** - { DIGital | AD1X }

Related Commands: :DSP:DANLr:INPut

Command Syntax: :DSP:DANLr:INPut?

Response Syntax: :DSP:DANLr:INPut **source**

Example: `:DSP:DANLR:INPUT?`

Response: `:DSP:DANLR:INPUT AD1X`

:DSP:DANLr:LEVel?

This query returns the level measurement for the indicated channel in the specified units. This measurement is not affected by any filter settings for this subsystem.

The first parameter in the response is the next available measurement.

The Digital Domain Audio Analyzer Level Meter settling is enabled or disabled by the algorithm parameter of the :SETTLING:DANLr command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

The valid choice of units depends upon the input type (:DSP:DANLR:INPUT AD1X or :DSP:DANLR:INPUT DIGital). This dependency is shown in Table 8-4.

Table 8-4. Valid units for each :DSP:DANLR:INPUT type

INPUT	Valid Units
AD1X	DBM, DBRA, DBRB, DBU, DBV, V
DIGital	DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFS, V, VP, VPP

The command argument is:

- **channel** - { A | B }
- **units** - { DBFS | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCTFs | V | VP | VPP }

Response argument(s):

- **level** - <nrf>
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DSP:DANLr:INPut, :DSP:DANLr:AUTorange

Command Syntax: :DSP:DANLr:LEVel? channel, **units**

Response Syntax: :DSP:DANLr:LEVel level, **settle_timeout**

Example:

```
:DSP:DANLR:LEVEL? A,PCTFS
```

Response:

```
:DSP:DANLR:LEVEL 59.2349PCTFS,0
```

:DSP:DANLr:LPFilter

This command sets the function meter low pass filter.

The command argument is:

- **frequency** - { FS_2 | F15K | F20K }

Default: FS_2

Related Commands: :DSP:DANLr:LPFilter?

Command Syntax: :DSP:DANLr:LPFilter **frequency**

Example:

```
:DSP:DANLR:LPFILTER F15K
```

:DSP:DANLr:LPFilter?

This query returns the current function meter low pass filter setting.

Response argument(s):

- **frequency** - { FS_2 | F15K | F20K }

Related Commands: :DSP:DANLr:LPFilter

Command Syntax: :DSP:DANLr:LPFilter?

Response Syntax: :DSP:DANLr:LPFilter **frequency**

Example:

```
:DSP:DANLR:LPFILTER?
```

Response:

```
:DSP:DANLR:LPFILTER F15K
```

:DSP:DANLr:MODE

This command specifies the mode of the function meter. The valid modes are Amplitude (AMPLitude), 2-Channel Ratio (RATio), Crosstalk (XTALk), THD+N Ratio (THDRatio), THD+N Amplitude (THDampl), and Bandpass (BP).

The command argument is:

- **mode** - { AMPLitude | BP | RATio | THDampl | THDRatio | XTALk }

Default: AMPLitude

Related Commands: :DSP:DANLr:MODE?

Command Syntax: :DSP:DANLr:MODE **mode**

Example: `:DSP:DANLR:MODE RATIO`

:DSP:DANLr:MODE?

This query returns the current mode of the Digital Analyzer Function meter.

Response argument(s):

- **mode** - { AMPLitude | BP | RATio | THDampl | THDRatio | XTALk }

Related Commands: :DSP:DANLr:MODE

Command Syntax: :DSP:DANLr:MODE?

Response Syntax: :DSP:DANLr:MODE **mode**

Example: `:DSP:DANLR:MODE?`

Response: `:DSP:DANLR:MODE RATIO`

:DSP:DANLr:RANGe

This command sets the input signal range for the indicated channel(s). The inputs to this command are the channel and the expected reading value in FFS units. When this command is received autoranging will be turned OFF and the specified range will be set.

The System Two will determine the appropriate range setting from this value. If a setting value does not match a range exactly the next higher range will be used. The ranges are:

FFS	%FS	dBFS
1.0	100	0.000
500 m	50	-6.021
250 m	25	-12.041
125 m	12.5	-18.062
62.5 m	6.25	-24.082
31.25 m	3.125	-30.103
15.63 m	1.563	-36.124
7.813 m	0.7813	-42.144
3.906 m	0.3906	-48.165
1.953 m	0.1953	-54.185
976.6 µ	0.09766	-60.206
488.3 µ	0.04883	-66.227
244.1 µ	0.02441	-72.247
122.1 µ	0.01221	-78.268
61.04 µ	0.00610	-84.288
30.52 µ	0.00305	-90.309

The command arguments are:

- **channel** - { A | B | AB }
- **setting** - <nrf> (range: > 1.0E-6, ≤ 1.0) { FFS }

Related Commands: :DSP:DANLr:AUTorange, :DSP:DANLr:RANGe?

Command Syntax: :DSP:DANLr:RANGe **channel**, **setting**

Example:

```
:DSP:DANLR:RANGE A,0.1FFS
```

:DSP:DANLr:RANGe?

This query returns the channel range setting in the specified units. Which reference is used when the response units is DBR depends on the channel. When channel is A, DBR refers to :DSP:REF:DBR1, and when channel is B, DBR refers to :DSP:REF:DBR2.

The command arguments are:

- **channel** - { A | B }
- **units** - { FFS | PCTFs | DBFS | DBR }

Response argument(s):

- **response_channel** - { A | B }
- **setting** - <nrf> { FFS | PCTFs | DBFS | DBR }

Related Commands: :DSP:DANLr:RANGe

Command Syntax: :DSP:DANLr:RANGe? **channel, units**

Response Syntax: :DSP:DANLr:RANGe **response_channel, setting**

Example: `:DSP:DANLR:RANGE? A,FFS`

Response: `:DSP:DANLR:RANGE A,0.125FFS`

:DSP:DANLr:RDGRate

This command specifies the reading rate used for all three meters (level, frequency and function meter). The possible settings are Auto, 4/sec, 8/sec, 16/sec, 32/sec, 64/sec, and 128/sec.

The 3 optional meter arguments are only accepted in AUTO mode, and are used in the determination of the correct reading rate for that mode. For optimum reading rate, the parameter corresponding to the type of reading to be taken must be specified.

For example, if the :DSP:DANLr:LEV? command is to be used, specify LEVel with the AUTO option. If :DSP:DANLr:FREQ? is to be used, select FREQ. For :DSP::DANLr:FUNC?, select FUNCmeter. Multiple selections are valid if more than one meter type is to be used at this reading rate. If no meter type is specified, the reading rate will be valid, but may be slower than the fastest reading rate valid for that meter.

NOTE: Failure to specify the correct meter type with an automatic type reading rate may result in readings taken at a rate that is too fast for the input frequency and meter in use.

The command argument is:

- **rate** - { R8 | R4 | R16 | R32 | R64 | R128 | AUTO }
- **meter1** - { FREQ | FUNCmeter | LEVel }
- **meter2** - { FREQ | FUNCmeter | LEVel }
- **meter3** - { FREQ | FUNCmeter | LEVel }

Default: R8

Related Commands: :DSP:DANLr:RDGRate?

Command Syntax: :DSP:DANLr:RDGRate **rate[, meter1,meter2, meter3]**

Example: `:DSP:DANLR:RDGRATE AUTO,FREQ`

:DSP:DANLr:RDGRate?

This query returns the current reading rate, and the selected meters for the Audio Analyzer DSP program.

Response argument(s):

- **rate** - { R8 | R4 | R16 | R32 | R64 | R128 | AUTO}
- **meter1** - { FREQ | FUNCmeter | LEVel}
- **meter2** - { FUNCmeter | LEVel }
- **meter3** - { LEVel }

Related Commands: :DSP:DANLr:RDGRate

Command Syntax: :DSP:DANLr:RDGRate?

Response Syntax: :DSP:DANLr:RDGRate **rate[, meter1,meter2, meter3]**

Example: `:DSP:DANLR:RDGRATE?`

Response: `:DSP:DANLR:RDGRATE AUTO,LEVEL,FUNCMETER`

:DSP:DANLr:RESPonse

The frequency specified by this command is used with the automatic reading rate selections of the :DSP:DANLr:RDGRate command (AUTO) to help determine the correct reading rate. The Digital Analyzer filter.frequency is set to the specified frequency if the function meter is in a tuning mode (BP (bandpass), THDampl, THDRatio or XTALk).

The argument is always in units of Hertz.

If the function meter is in a tuning mode (i.e., BP (bandpass), THDampl, THDRatio or XTALk), the Digital Analyzer filter frequency is set to the specified frequency.

The argument is always in Hertz.

The command argument is:

- **freq** - <nrf> (range: > 0, =47% of sample rate)

Default: 20.0

Related Commands: :DSP:DANLr:FILTerfreq,
:DSP:DANLr:TUNingsrc?

Command Syntax: :DSP:DANLr:RESPonse **freq**

Example: `:DSP:DANLr:RESPONSE 5000`

:DSP:DANLr:RESPonse?

This query returns the currently specified response frequency. The response is always in hertz.

Response argument(s):

- **freq** - <nrf>

Related Commands: :DSP:DANLr:RESPonse

Command Syntax: :DSP:DANLr:RESPonse?

Response Syntax: :DSP:DANLr:RESPonse **freq**

Example: `:DSP:DANLr:RESPONSE?`

Response: `:DSP:DANLr:RESPONSE 5000`

:DSP:DANLr:SET?

This query returns the states of all DSP Audio Analyzer settings. The response consists of a sequence of the DSP Audio Analyzer settings that may be sent back to the instrument at a later time in order to reset all settings to the same state. Note that some commands are sent more than once and transition through several states in order to achieve a final instrument state.

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands:

Command Syntax: :DSP:DANLr:SET?

Response Syntax: :DSP:DANLr:SET **settings**

Example: :DSP:DANLR:SET?

Response:

```
:DSP:DANLR:AUTORANGE A,ON;AUTORANGE B,ON;CHA
NNEL A;COUPLING A,AC;COUPLING B,AC;DETECTOR
FRMS;HPFILTER F10;INPUT DIGITAL;LPFILTER FS_
2;MODE THDRATIO;RESPONSE 20;WTG UNWT;RDGRATE
 R8;TUNINGSRC FIXED;FILTERFREQ 999.999HZ;FAU
TORANGE ON
```

:DSP:DANLr:TUNingsrc

This command specifies the signal source for setting the bandpass/bandreject filter. The valid sources are: counter, analog generator, digital generator, or fixed. This command is only valid when :DSP:DANLr:MODE is set to: crosstalk, THD+N ratio, THD+N amplitude, or bandpass. In any other mode this command will cause an execution error.

If the current mode is crosstalk, THD+N ratio, THD+N amplitude, or bandpass when the filter frequency command (:DSP:DANLr:FILTerfreq) is received, the tuning source will be forced to FIXed (if not already in this mode).

The command argument is:

- **src** - { FIXed | AGEN | CNTR | DGEN }

Default: FIXed

Related Commands: :DSP:DANLr:FILTerfreq, :DSP:DANLr:MODE, :DSP:DANLr:TUNingsrc?

Command Syntax: :DSP:DANLr:TUNingsrc **src**

Example: :DSP:DANLR:TUNINGSRC CNTR

:DSP:DANLr:TUNingsrc?

This query returns the current tuning source

Response argument(s):

- **src** - { FIXed | AGEN | CNTR | DGEN }

Related Commands: :DSP:DANLr:TUNingsrc

Command Syntax: :DSP:DANLr:TUNingsrc?

Response Syntax: :DSP:DANLr:TUNingsrc **src**

Example: `:DSP:DANLR:TUNINGSRC?`

Response: `:DSP:DANLR:TUNINGSRC 3`

:DSP:DANLr:WTG

This command specifies weighting filters to be used with the function meter. These filters are typically used in noise and THD+N measurements.

The available weighting filters, for AMPLitude, 2-Ch Ratio, THD+N AMPL and THD+N RATIO are UNWT (unweighted), AWTG (A-weighting), CCIR (CCIR Weighting), FWTG (F weighting), CCITt (CCITT Weighting), and CMSG (C-message Weighting).

For Crosstalk the available filter is FX1 (the frequency of the source signal).

The available filters for Bandpass are FX1 (source frequency), FX2 (source frequency 2nd harmonic), FX3 (source frequency 3rd harmonic), FX4 (source frequency 4th harmonic), FX5 (source frequency 5th harmonic). See Table 8-5.

Table 8-5. Available WTG filters

Ampl, 2-Ch Ratio, THD+N Ampl, THD+N Ratio	Crosstalk	Bandpass
Unweighted (UNWTD)	Narrow, Freq x1 (FX1)	Narrow, Freq x1 (FX1)
A-weighting (AWTG)		Narrow, Freq x2 (FX2)
CCIR (CCIR)		Narrow, Freq x3 (FX3)
F-weighting (FWTG)		Narrow, Freq x4 (FX4)
CCITT weighting (CCITt)		Narrow, Freq x5 (FX5)
C-message weighting (CMSG)		

The command argument is:

- **weighting** - { UNWT | AWTG | CCIR | CCITt | CMSG | FWTG | FX1 | FX2 | FX3 | FX4 | FX5 }

Default: UNWT

Related Commands: :DSP:DANLr:WTG?

Command Syntax: :DSP:DANLr:WTG **weighting**

Example: `:DSP:DANLR:WTG UNWT`

:DSP:DANLr:WTG?

This query returns the currently selected function meter weighting filter.

Response argument(s):

- **weighting** - { UNWT | AWTG | CCIR | CCITt | CMSG | FWTG | FX1 | FX2 | FX3 | FX4 | FX5 }

Related Commands: :DSP:DANLr:WTG

Command Syntax: :DSP:DANLr:WTG?

Response Syntax: :DSP:DANLr:WTG **weighting**

Example: `:DSP:DANLR:WTG?`

Response: `:DSP:DANLR:WTG UNWT`

:DSP:DATA

This command downloads data into either of the DSP acquisition buffers. This data must have been originally uploaded from a System Two acquisition buffer or transfer buffer. Data can only be uploaded when the DSP program is FASTTEST, FFT, or MLS.

The valid first argument values are: channel 1 acquisition buffer (A1), channel 2 acquisition buffer (A2), channel 1 transfer buffer (T1), channel 2 transfer buffer (T2)

The command arguments are:

- **dspbuffer** - { A1 | A2 | T1 | T2 }
- **data** - <definite length arb block data>

Related Commands: :DSP:DATA?

Command Syntax: :DSP:DATA **dspbuffer, data**

Example: `:DSP:DATA A1, #xyy...`

:DSP:DATA?

This query returns the buffer contents for specific DSP programs. Not all DSP programs will return either acquisition and/or transform buffer contents (see Table 8-6).

Table 8-6. Buffer contents for each DSP program

DSP Program	Acquisition Buffers	Transform Buffers
Audio Analyzer (DANLr)	No	No
Intervu (INTervu	No	No
FASTTEST (FASTtest)	Yes	No
FFT (FFT)	Yes	Yes
MLS (MLS)	Yes	Yes

:DSP:Data? will cause a command error if invoked while Digital Analyzer or Intervu is loaded. This command will also cause an argument error if a FASTTEST transform buffer is queried.

The response to a query of a transform buffer will be a portion of the acquisition buffer's time based data. The portion of the acquisition buffer returned depends on which DSP program is loaded and the relevent program parameters.

For the FFT DSP program, the response to a query of a transform buffer is determined by the acquisition buffer length (:DSP:FFT:ACQLength), the trigger delay (:DSP:FFT:DELay), the start time (:DSP:FFT:STARt), and the transform length (:DSP:FFT:XLENgth). (see diagram below).

The data query will return data for the region labeled Acquisition when the argument is A1 or A2 (depending on channel). The data query will return the region labeled Transform when the argument is T1 or T2 (depending on channel).

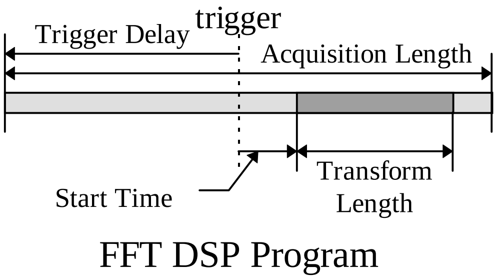


Figure 8-8. FFT Trigger time relationships

For FASTTEST, the response to a query of a transform buffer is determined by the acquisition/transform buffer length (:DSP:FASTtest:LENGth)and the trigger delay (:DSP: FASTtest:DELay). In FASTTEST the acquisition length is equal to the transform length.

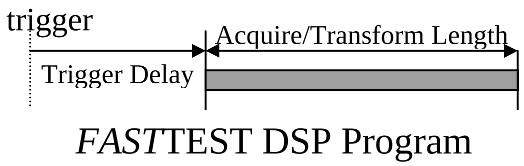


Figure 8-9. FASTtest Trigger time relationships

For MLS, the response to a query of a transform buffer is determined by the acquisition buffer length (:DSP:MLS:ACQLength), the start time, and the stop time (:SRCParams).

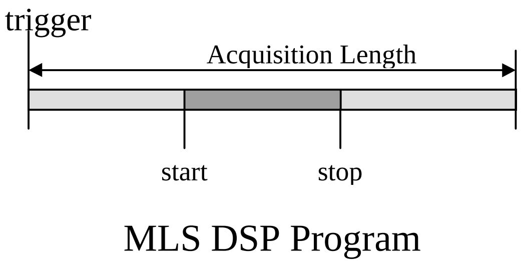


Figure 8-10. MLS Trigger time relationships

The response will NOT be the frequency based data resulting from an FFT. To get the FFT results it is necessary to interrogate the individual points by setting the :DSP:SRCParams, :DSP:OPSTate, and using the individual reading queries (e.g. :DSP:FASTtest:AMPL?)

The response data will always be definite length arbitrary block data.

The command argument is:

- **dspbuffer** - { A1 | A2 | T1 | T2 }

Response argument(s):

- **response_dspbuffer** - { A1 | A2 | T1 | T2 }
- **data** - <definite length arb block data>

Related Commands: :DSP:DATA

Command Syntax: :DSP:DATA? **dspbuffer**

Response Syntax: :DSP:DATA **response_dspbuffer, data**

Example: `:DSP:DATA? A1`

Response: `:DSP:DATA A1,#xyy...`

:DSP:FASTtest

Compound command header for Multitone Audio Analyzer commands.

:DSP:FASTtest:AMPL?

This query returns a single point amplitude measurement from FASTTEST. The first parameter specifies the DSP channel (either 1 or 2). The second parameter specifies the response units for time or frequency, and the third parameter specifies the value of the source point being interrogated. The units of the third parameter are specified by the :DSP:SRCParams command (second parameter). Note that the correct choice of units depends upon the input type (AD1X or DIGital) and the source type. Table 8-7 shows this dependency.

Table 8-7. :DSP:FASTTEST:AMPL? units dependency

Input Type: INPUT	Source Type: SRC1 & SRC2	Valid Units
AD1X	A, AMPLitude, B, GENA, GENB	DBM, DBRA, DBRB, DBU, DBV, V
AD1X	RATio	DB, PCT, X_Y
AD1X	JITTer	UI
DIGital	A, B	DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFS, V, VP, VPP

To enable this measurement, the AMP1 and/or AMP2 parameters must be specified in the :DSP:OPSTate READing command.

The query argument(s) are:

- **channel** - <nr1> (range: 1 or 2)
- **units** - { DBFS | DBM | DBRA | DBRB | DBR1 | DBR2 | DBU | DBV | FFS | PCTFS | V | VP | VPP | DB | PCT | X_Y | UI }
- **src_value** - <nrf> (range: depends on SRCParams settings)

The response argument is:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:FASTtest:AMPL? **channel, units, src_value**

Response Syntax: :DSP:FASTtest:AMPL **reading**

Example: `:DSP:FASTTEST:AMPL? 1,V,20000`

Response: `:DSP:FASTTEST:AMPL 0.5`

:DSP:FASTtest:FREQres

This command specifies the percent of frequency error correction when PROCess is set to FREQ. It also specifies the Frequency Resolution used in Response and Distortion Measurement functions.

Frequency error correction corrects for frequency error between the multitone signal at the input and the frequency of the multitone signal loaded into the digital generator arbitrary waveform buffer. The range of error correction is set by the FREQres argument value. Error correction occurs during acquisition of the signal.

In the Response measurement function, the amplitudes of all FFT bins within plus and minus the Frequency Resolution value of each source value are combined in RSS (root-sum-square) fashion and furnished as the integrated amplitude of the bins within that range. The purpose of this function is to provide accurate frequency response measurements of devices with wow and flutter. Wow and flutter spreads the energy from a single tone across a narrow spectral band.

In Distortion function, the amplitudes of all FFT bins within plus and minus the Frequency Resolution value of each source value are excluded from the RSS computation of energy falling between tones. Distortion function defines all signals other than the fundamental tones as distortion and noise. Entering a non-zero value of Frequency Resolution causes flutter sidebands to not be included in the distortion measurement.

If frequency error correction is required during acquisition but frequency resolution is not required during Response and Distortion measurements, then set FREQres to 0 after acquisition with ACQX? and then use REPROCESS? to reprocess the FFT results with frequency resolution set to 0 percent.

The command argument is (in units of %):

- **resolution** - <nrf> (range: ≥ 0, ≤ 13)

Default: 0.0

Related Commands: :DSP:FASTtest:PROCess,
:DSP:FASTtest:MODE,
:DSP:FASTtest:FREQres?

Command Syntax: :DSP:FASTtest:FREQres **resolution**

Example: `:DSP:FASTTEST:FREQRES 2`

:DSP:FASTtest:FREQres?

This query returns the frequency resolution parameter setting.

Response argument(s):

- **resolution** - <nrf>

Related Commands: :DSP:FASTtest:FREQres

Command Syntax: :DSP:FASTtest:FREQres?

Response Syntax: :DSP:FASTtest:FREQres **resolution**

Example: `:DSP:FASTTEST:FREQRES?`

Response: `:DSP:FASTTEST:FREQRES 3`

:DSP:FASTtest:INPut

This command specifies the whether the input signal is Analog or Digital domain. When the signal is from the Analog domain the input signal is routed through the A/D converters that operate at the internal sample rate (AD1X).

The command argument is:

- **input** - { DIGital | AD1X }

Default: DIGITAL

Related Commands: :DSP:FASTtest:SRC1,
:DSP:FASTtest:SRC2,
:DSP:FASTtest:INPut?

Command Syntax: :DSP:FASTtest:INPut **input**

Example: `:DSP:FASTTEST:INPUT AD1X`

:DSP:FASTtest:INPut?

This query returns the setting of the INPut command.

Response argument(s):

- **input** - { DIGital | AD1X }

Related Commands: :DSP:FASTtest:INPut

Command Syntax: :DSP:FASTtest:INPut?

Response Syntax: :DSP:FASTtest:INPut **input**

Example: `:DSP:FASTTEST:INPUT?`

Response: `:DSP:FASTTEST:INPUT AD1X`

:DSP:FASTtest:LENGth

This command specifies the FASTTEST FFT record length. Longer transform lengths produce better frequency resolution in the resulting FFT, but require longer times to acquire and to transform the signal.

The Auto selection will automatically set the DSP acquisition buffer and transform length to be exactly twice the length of the generator waveforms loaded into the Digital Generator buffer. This setup is necessary for correct operation of the Noise function of FASTTEST.

The valid lengths are Auto (AUTO), 16384 (L16K), 8192 (L8K), 4096 (L4K), 2048 (L2K), 1024 (L1K), and 512 (L512) samples.

The command argument is:

- **length** - { AUTO | L1K | L16K | L2K | L4K | L512 | L8K }

Default: AUTO

Related Commands: :DSP:FASTtest:LENGth?

Command Syntax: :DSP:FASTtest:LENGth **length**

Example: `:DSP:FASTTEST:LENGTH L16K`

:DSP:FASTtest:LENGth?

This query returns the FASTTEST FFT record length setting.

Response argument(s):

- **length** - { AUTO | L1K | L16K | L2K | L4K | L512 | L8K }

Related Commands: :DSP:FASTtest:LENGth

Command Syntax: :DSP:FASTtest:LENGth?

Response Syntax: :DSP:FASTtest:LENGth **length**

Example: `:DSP:FASTTEST:LENGTH?`

Response: `:DSP:FAST:LENGTH L16K`

:DSP:FASTtest:MODE

This command specifies the type of processing applied to the FASTTEST FFT results. The choices are spectrum, response, distortion, noise, masking, and crosstalk.

The command argument is:

- **type** - { DISTortion | MASKing | NOISe | RESPonse | SPECtrum | XTALk }

Default: SPECtrum

Related Commands: :DSP:FASTtest:MODE?

Command Syntax: :DSP:FASTtest:MODE **type**

Example: `:DSP:FASTTEST:MODE SPECTRUM`

:DSP:FASTtest:MODE?

This query returns FASTTEST Mode setting.

Response argument(s):

- **type** - { DISTortion | MASKing | NOISe | RESPonse | SPECtrum | XTALk }

Related Commands: :DSP:FASTtest:MODE

Command Syntax: :DSP:FASTtest:MODE?

Response Syntax: :DSP:FASTtest:MODE **type**

Example: `:DSP:FASTTEST:MODE?`

Response: `:DSP:FASTTEST:MODE SPECTRUM`

:DSP:FASTtest:PEAK?

This query returns an unsettled measurement from the peak monitor, for the specified channel. This reading is used to determine whether or not the input is near full scale (or clipped). These meters are only valid when the DSP is in OPSTate SETup mode.

Use these peak monitors when using the System Two A/D converters as a signal source in order to determine whether or not the converters are being driven into clipping with fixed analog input ranges.

The command arguments are:

- **channel** - <nr1> (range: 1 or 2)
- **units** - { DBFS | FFS | PCTFs }

Response argument(s):

- **peak** - <nrf> (range: depends on SRCParams settings)

Related Commands: :DSP:FASTtest:PEAK

Command Syntax: :DSP:FASTtest:PEAK? **channel**, **units**

Response Syntax: :DSP:FASTtest:PEAK? **peak**

Example: `:DSP:FASTTEST:PEAK? 1,FFS`

Response: `:DSP:FASTTEST:PEAK 0.548102FFS`

:DSP:FASTtest:PHASe?

This query returns a single point phase measurement from FASTTEST. The first parameter specifies the DSP channel (either 1 or 2), the second parameter specifies the response units, and the third parameter specifies the frequency value of the source point being interrogated. The units of the third parameter are specified by the :DSP:SRCParams command (second parameter).

To enable this reading, the PHA1 and/or PHA2 parameters must be specified in the :DSP:OPSTate READing command.

The query argument(s) are:

- **channel** - <nr1> (range: 1 or 2)
- **units** - { DEG }
- **src_value** - <nrf> (range: depends on SRCParams settings }

The response argument is:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:FASTtest:PHASe? **channel, units,src_value**

Response Syntax: :DSP:FASTtest:PHASe? **reading**

Example: `:DSP:FASTTEST:PHASE? 1,DEG,19998`

Response: `:DSP:FASTTEST:PHASE 4.1748`

:DSP:FASTtest:PKTRig

This command specifies the trigger mode of the FASTtest peak meters (real-time meters). These meters are only valid when the DSP is in OPSTate SETup mode. A zero (0) argument will set the DSP to return the next available peak meter reading. A one (1) argument will set the DSP to abort the current measurement cycle and start a new measurement cycle when the :DSP:FASTtest:PEAK? query is received.

This command will cause an execution error if received when the FASTTEST program is in OPSTate READing mode.

The command argument is:

- **mode** - <nr1> (range: 0 or 1)

Default: 0

Related Commands: :DSP:FASTtest:PEAK?

Command Syntax: :DSP:FASTtest:PKTRig **mode**

Example: `:DSP:FASTTEST:PKTRIG 1`

:DSP:FASTtest:PKTRig?

This query returns the setting of the FASTTEST PKTRig command.

The response argument is:

- **mode** - <nr1> (range: 0 or 1)

Related Commands: :DSP:FASTtest:PKTRig

Command Syntax: :DSP:FASTtest:PKTRig?

Response Syntax: :DSP:FASTtest:PKTRig **mode**

Example: `:DSP:FASTTEST:PKTRIG?`

Response: `:DSP:FASTTEST:PKTRIG 1`

:DSP:FASTtest:PMODe

This command specifies the phase measurement mode for FASTTEST channel 2, either Independent channel phase measurements, or Interchannel phase measurements. FASTTEST can measure the phase of sinewave components of either channel of the received multitone signal or the interchannel phase.

When set to Independent, the Ch. 2 Phase measurement will return the phase of the Channel 2 signal. When Interchannel is selected, the Ch. 2 Phase measurement will return the phase difference between Channel 2 and Channel 1. In either case, the Ch. 1 Phase measurement always returns the Channel 1 phase.

The command argument is:

- **mode** - { INDependent | ICHannel }

Default: INDependent

Related Commands: :DSP:FASTtest:PMODe?

Command Syntax: :DSP:FASTtest:PMODe **mode**

Example: `:DSP:FASTTEST:PMODE ICHANNEL`

:DSP:FASTtest:PMODe?

This query returns the setting of the FASTTEST PMODe command.

Response argument(s):

- **mode** - { INDependent | ICHannel }

Related Commands: :DSP:FASTtest:PMODe,
:DSP:FASTtest:PHASe

Command Syntax: :DSP:FASTtest:PMODe?

Response Syntax: :DSP:FASTtest:PMODe **mode**

Example: `:DSP:FASTTEST:PMODE?`

Response: `:DSP:FASTTEST:PMODE ICHANNEL`

:DSP:FASTtest:PROCess

This command specifies the type of processing to be performed during and acquisition an transform with FASTTEST.

Synchronous mode (SYNC) should be used when utilizing synchronous multitone waveforms that have no frequency shift. Synchronous multitone waveforms may be created by MAKEWAV2.EXE. Synchronous multitone waveforms do not require an FFT window function.

Freq Correction mode (FREQ) performs frequency error correction during acquisition. The FREQres command must be used to specify the degree of frequency error correction allowed. FASTTEST will compare the acquired waveform with the waveforms in the DGEN arbitrary waveform buffer and correct for any shift detected in the tone frequencies, within the range allowed by the setting of the FREQres command. Freq Correction mode assumes a synchronous multitone waveform and it does not use a window function during FFT processing. Use this mode when the input signal may have a small degree of frequency shift or sample rate error.

The Hann window mode is provided for use when measuring non-synchronous signals.

The command arguments are:

- **type** - { SYNC | FREQ | WINDowed }

Default: SYNC

Related Commands: :DSP:FASTtest:PROCess?

Command Syntax: :DSP:FASTtest:PROCess **type**

Example: `:DSP:FASTTEST:PROCESS SYNC`

:DSP:FASTtest:PROCess?

This query returns the setting of the FASTTEST PROCess command.

Response argument(s):

- **type** - { SYNC | FREQ | WINDowed }

Related Commands: :DSP:FASTtest:PROCess

Command Syntax: :DSP:FASTtest:PROCess?

Response Syntax: :DSP:FASTtest:PROCess **type**

Example: `:DSP:FASTTEST:PROCESS?`

Response: `:DSP:FASTTEST:PROCESS SYNC`

:DSP:FASTtest:SET?

This query returns the states of all DSP FASTTEST settings. The response consists of a sequence of the DSP FASTTEST settings that may be sent back to the instrument at a later time in order to reset all settings to the same state. Note that some commands are sent more than once and transition through several states in order to achieve a final instrument state.

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands:

Command Syntax: :DSP:FASTtest:SET?

Response Syntax: :DSP:FASTtest:SET **settings**

Example: `:DSP:FASTTEST:SET?`

Response:

```
:DSP:FASTTEST:FREQRES 0;INPUT DIGITAL;SRC1 A
;SRC2 B;INPUT AD1X;SRC1 A;SRC2 B;INPUT AD1X;
LENGTH AUTO;MODE SPECTRUM;PMODE ICHANNEL;PROC
ESS SYNC;TRGDELAY 0;TRIGGER DGEN;PKTRIG 1
```

:DSP:FASTtest:SRC1

This command specifies the signal source for the currently selected input type for channel 1. The possible sources for each input are shown in Table 8-8.

Table 8-8. Possible sources for each input for SRC1

Input	Source
Digital	A, B, None
Low BW (1x) A/D	Anlr A, Anlr B, Function Meter Amplitude, Function Meter Ratio, Gen A, Gen B, Jitter Signal, None

In the following command argument list an entry of "A" (or "B") will be interpreted in two ways, depending on whether the input is in Digital or A/D mode. In Digital mode "A" will be interpreted as subframe A. In A/D mode "A" will be interpreted as analog analyzer channel A.

The command argument is:

- **source** - { A | AMPLitude | B | GENA | GENB | JITTer | NONE | RATio }

Default: A

Related Commands: :DSP:FASTtest:INPut, :DSP:FASTtest:SRC1?

Command Syntax: :DSP:FASTtest:SRC1 **source**

Example: `:DSP:FASTTEST:SRC1 A`

:DSP:FASTtest:SRC1?

This query returns the signal source for the currently selected input for channel 1.

Response argument(s):

- **source** - { A | AMPLitude | B | GENA | GENB | JITTer | NONE | RATio }

Related Commands: :DSP:FASTtest:SRC1

Command Syntax: :DSP:FASTtest:SRC1?

Response Syntax: :DSP:FASTtest:SRC1 **source**

Example: `:DSP:FASTTEST:SRC1?`

Response: `:DSP:FASTTEST:SRC1 A`

:DSP:FASTtest:SRC2

This command specifies the signal source for the currently selected input type for channel 2. The possible sources for each input are shown in Table 8-9.

Table 8-9. Possible sources for each input for SRC2

Input	Source
Digital	A, B, None
Low BW (1x) A/D	Anlr A, Anlr B, Function Meter Amplitude, Function Meter Ratio, Gen A, Gen B, Jitter Signal, None

In the following command argument list an entry of "A" (or "B") will be interpreted in two ways, depending on whether the input is in Digital or A/D mode. In Digital mode "A" will be interpreted as subframe A. In A/D mode "A" will be interpreted as analog analyzer channel A.

The command argument is:

- **source** - { A | AMPLitude | B | GENA | GENB | JITTer | NONE | RATio }

Default: B

Related Commands: :DSP:FASTtest:INPut, :DSP:FASTtest:SRC2?

Command Syntax: :DSP:FASTtest:SRC2 **source**

Example: `:DSP:FASTTEST:SRC2 A`

:DSP:FASTtest:SRC2?

This query returns the signal source for the currently selected input for channel 2.

Response argument(s):

- **source** - { B | A | AMPLitude | GENA | GENB | JITTer | NONE | RATio }

Related Commands: :DSP:FASTtest:SRC2

Command Syntax: :DSP:FASTtest:SRC2?

Response Syntax: :DSP:FASTtest:SRC2 **source**

Example: `:DSP:FASTTEST:SRC2?`

Response: `:DSP:FASTTEST:SRC2 A`

:DSP:FASTtest:TRGDelay

This command specifies the triggering delay to be used when TRIGGER is set to tight, normal, or loose trigger modes. The argument unit is seconds.

The maximum value that may be specified is dependent on the sample rate according to the following formula:

max TRGDelay = (65536*64)/(sample rate/2)

The command argument is:

- **delay** - <nrf> (range: ≥ 0, ≤ max TRGDelay above)

Default: 0.0

Related Commands: :DSP:FASTtest:TRGDelay?

Command Syntax: :DSP:FASTtest:TRGDelay **delay**

Example: `:DSP:FASTTEST:TRGDELAY 0.100`

:DSP:FASTtest:TRGDelay?

This query returns the setting of the FASTTEST TRGDelay command.

Response argument(s):

- **delay** - <nrf>

Related Commands: :DSP:FASTtest:TRGDelay

Command Syntax: :DSP:FASTtest:TRGDelay?

Response Syntax: :DSP:FASTtest:TRGDelay **delay**

Example: `:DSP:FASTTEST:TRGDELAY?`

Response: `:DSP:FASTTEST:TRGDELAY 0.1`

:DSP:FASTtest:TRIGger

This command specifies the triggering mode to be used for signal acquisition when ACQX? is received. The available modes are: tight, normal, loose, external, digital generator, and off.

Tight, normal, and loose modes are designed for recognition of a multitone signal that may be degraded by the device under test and that is not being generated in real time by the System Two. Signal acquisition will terminate when the input signal meets the acquisition criteria specified by the tight, normal, and loose signal matching criteria. These criteria require the DSP to compare the input signal with the spectral characteristics of the signals contained in the Digital Generator arbitrary waveform buffers. Acquisition is complete when the criteria are met or the DSP Timeout value has been exceeded. The result will determine the numeric response to the :DSP:ACQX? command. If the signal meets the criteria before the timeout period, then the response to :DSP:ACQX? will be 0, otherwise it will be 1. Note that if no signal is present, the acquisition will not terminate until the timeout period has been exceeded.

External mode provides for an external signal to trigger acquisition after the ACQX? command has been received. The External triggering selection is operational only with Dual Domain units. It is the signal connected to pin 3 of the 15-pin D-sub connector on the rear of the DSP module. If pin 3 is high (or open circuit, in which case it is pulled high by an internal pull-up resistor), triggering occurs at the next digital sample. Pulling pin 3 low from an external device holds off triggering, with acquisition being triggered on the next sample after pin 3 is pulled high.

Digital generator mode is used when the signal source driving the device under test is the System Two arbitrary waveform generated by the Digital Generator arbitrary waveform or the Analog Generator arbitrary waveform.

Off mode (free run) forces immediate acquisition when ACQX? is received.

The command argument is:

- **source** - { DGEN | EXTernal | LOOSe | NORMal | OFF | TIGHt }

Default: DGEN

Related Commands: :DSP:FASTtest:TRIGger?

Command Syntax: :DSP:FASTtest:TRIGger **source**

Example: `:DSP:FASTTEST:TRIGGER EXTERNAL`

:DSP:FASTtest:TRIGger?

This query returns the setting of the FASTTEST TRIGger command.

Response argument(s):

- **source** - { DGEN | EXTernal | LOOSe | NORMal | OFF | TIGHt }

Related Commands: :DSP:FASTtest:TRIGger

Command Syntax: :DSP:FASTtest:TRIGger?

Response Syntax: :DSP:FASTtest:TRIGger **source**

Example: `:DSP:FASTTEST:TRIGGER?`

Response: `:DSP:FASTTEST:TRIG EXTERNAL`

:DSP:FFT

Compound command header for FFT Spectrum Analyzer commands.

:DSP:FFT:ACQLength

This command specifies the FFT DSP acquisition buffer length. The beginning of the DSP acquisition buffer is specified relative to the trigger point by the trigger delay command (:DSP:FFT:DELay).

For a more detailed explanation of the relationship between start time, acquisition length, trigger delay, and transform length, see the :DSP:DATA? query on page 8-28..

The argument to this command indicates that either the DSP acquisition buffer is the same length as the transform buffer or it is equal to 24k (see :DSP:FFT:XLENgth command).

The command argument is:

- **length** - { XLENgth | L24K }

Default: XLENgth

Related Commands: :DSP:FFT:DELay, DSP:FFT:STARttime, :DSP:FFT:XLENgth?

Command Syntax: :DSP:FFT:ACQLength **length**

Example: `:DSP:FFT:ACQLENGTH XLENGTH`

:DSP:FFT:ACQLength?

This query returns the FFT transform length.

Response argument(s):

- **length** - { XLENgth | L24K }

Related Commands: :DSP:FFT:ACQLength

Command Syntax: :DSP:FFT:ACQLength?

Response Syntax: :DSP:FFT:ACQLength **length**

Example: `:DSP:FFT:ACQLENGTH?`

Response: `:DSP:FFT:ACQLENGTH XLENGTH`

:DSP:FFT:AMPL?

This query returns a single point amplitude measurement from the FFT DSP program. The first parameter specifies the DSP channel (either 1 or 2). The second parameter specifies the response units for time or frequency, and the third parameter specifies the value of the source point being interrogated. The units of the third parameter are specified by the :DSP:SRCParams command (in the second parameter). Note that the correct choice of units depends upon the input type (AD1X or DIGital) and the source type. This dependency is shown in Table 8-10.

Table 8-10. :DSP:FFT:AMPL? source dependency

Input Type: INPUT	Source Type: SRC1 & SRC2	Valid Units
AD1X	A, AMPLitude, B, GENA, GENB	DBM, DBRA, DBRB, DBU, DBV, V
AD1X	RATio	DB, PCT, X_Y
AD1X	JITTer	UI
DIGital	A, B	DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFS, V, VP, VPP

To enable this reading, the AMP1 and/or AMP2 parameters must be specified in the :DSP:OPSTate READing command.

The query argument(s) are:

- **channel** - <nr1> (range1 or 2)
- **units** - { DBFS | DBM | DBRA | DBRB | DBR1 | DBR2 | DBU | DBV | FFS | PCTFS | V | VP | VPP | DB | PCT | X_Y | UI }
- **src_value** - <nrf> (range: depends on SRCParams settings)

The response argument is:

- **reading** - <nrf>

Related Commands: :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:FFT:AMPL? **channel, units, src_value**

Response Syntax: :DSP:FFT:AMPL **reading**

Example: `:DSP:FFT:AMPL? 1,DBFS,1002`

Response: `:DSP:FFT:AMPL -15.2595`

:DSP:FFT:AVGS

This command specifies the number of acquisitions to average. The FFT DSP program has the ability to average many successive acquisitions of a signal in either the frequency domain or in the time domain.

Note: the acquisition time will increase as the number of averages increases. Therefore, the DSP acquisition timeout set with the :DSP:TIMEOUT command may need to be increased in order to avoid a timeout.

The command argument will be rounded up to the next higher average of the possible averages: 1, 2, 4, 8, 16, 32, 64, 128, 256, 1024, 2048, 4096.

The command argument is:

- **average** - <nr1> (range: ≥ 1, ≤ 4096)

Default: 1

Related Commands: :DSP:FFT:AVGType, :DSP:FFT:AVGS?

Command Syntax: :DSP:FFT:AVGS **average**

Example: `:DSP:FFT:AVGS 3`

:DSP:FFT:AVGS?

This query returns the setting of the :DSP:FFT:AVGS command.

Response argument(s):

- **average** - <nr1> (range: ≥ 1, ≤ 4096)

Related Commands: :DSP:FFT:AVGS

Command Syntax: :DSP:FFT:AVGS?

Response Syntax: :DSP:FFT:AVGS **average**

Example: `:DSP:FFT:AVGS?`

Response: `:DSP:FFT:AVGS 4`

:DSP:FFT:AVGType

This command specifies the type of FFT averaging to perform. The possible settings for this command are influenced by the :DSP:FFT:WINDow command. Table 8-11 describes the setting number and the windows that are valid with that setting.

Table 8-11. Setting number and valid windows

Type	Meaning	Valid Windowing
1	Power (spectrum only)	<all>
2	Sync, re-align	Blackman-Harris, Hann, Flat-Top, Equiripple, None
3	Sync	Blackman-Harris, Hann, Flat-Top, Equiripple, None
4	Sync, re-align, move center first	"None, move to bin center"
5	Sync, re-align, average first	"None, move to bin center"
6	Sync, move center first	"None, move to bin center"
7	Sync, average first	"None, move to bin center"

An execution error will be generated if an invalid combination of averaging type and window is specified when an FFT is performed.

The command argument is:

- **type** - <nr1> (range: ≥ 1, ≤ 7)

Default: 1

Related Commands: :DSP:FFT:AVGS, :DSP:FFT:WINDow,\
:DSP:FFT:AVGType?

Command Syntax: :DSP:FFT:AVGType **type**

Example: `:DSP:FFT:AVGTYPE 3`

:DSP:FFT:AVGType?

This query returns the setting of the :DSP:FFT:AVGType command.

Response argument(s):

- **type** - <nr1> (range: ≥ 1, ≤ 7)

Related Commands: :DSP:FFT:AVGS, :DSP:FFT:AVGType

Command Syntax: :DSP:FFT:AVGType?

Response Syntax: :DSP:FFT:AVGType **type**

Example: `:DSP:FFT:AVGTYPE?`

Response: `:DSP:FFT:AVGTYPE 3`

:DSP:FFT:COUPling

This command specifies the type of coupling applied to the acquired waveform.

In DC Coupled mode, time and frequency data will show any DC offset. For example, the DC offset of a digital generator Sine+Offset waveform may be measured with the DC Coupled mode.

In Subtract Average mode, the DSP will subtract from each sample the average value of all samples in the acquisition buffer.

In Subtract 1/2 Peak to Peak, the DSP will subtract, from each sample, the average of the positive peak and the negative peak.

The command argument is:

- **type** - { DC | SUBavg | SUBHalf }

Default: DC

Related Commands: :DSP:FFT:COUPling?

Command Syntax: :DSP:FFT:COUPling **type**

Example: `:DSP:FFT:COUPLING DC`

:DSP:FFT:COUPling?

This query returns the setting of the :DSP:FFT:COUPLING command.

Response argument(s):

- **type** - { DC | SUBavg | SUBHalf }

Related Commands: :DSP:FFT:COUPling

Command Syntax: :DSP:FFT:COUPling?

Response Syntax: :DSP:FFT:COUPling **type**

Example: `:DSP:FFT:COUPLING?`

Response: `:DSP:FFT:COUPLING DC`

:DSP:FFT:DELay

This command specifies trigger delay in seconds. This time represents the (signed) offset of the start of the DSP acquisition buffer relative to the trigger point. For a more detailed explanation see Section 2 of this manual. The range is -(length of DSP acquisition buffer) to ~1.35 seconds.

The command argument (in seconds) is:

- **time** - <nrf> (range: depends on DSP acquisition buffer length and internal sample rate)

Default: 0.0

Related Commands: :DSP:FFT:DELay

Command Syntax: :DSP:FFT:DELay **time**

Example: `:DSP:FFT:DELAY -0.03`

:DSP:FFT:DELay?

This query returns the FFT trigger delay (in seconds).

Response argument(s):

- **time** - <nrf>

Related Commands: :DSP:FFT:DELay

Command Syntax: :DSP:FFT:DELay?

Response Syntax: :DSP:FFT:DELay **time**

Example: `:DSP:FFT:DELAY?`

Response: `:DSP:FFT:DELAY -0.050`

:DSP:FFT:INPut

This command specifies the source of the input, either Digital or Analog domain signals. To measure digital domain signals, select DIGital.

Three choices for A/D converters provide three different ranges of sample rate derived from the internal sample rate set by the :DOUT:RATE command.

AD1X selects the high resolution converters that operate at 1X the internal sample rate. These converters provide a measurement bandwidth of 0.47 of the internal sample rate.

AD_4 selects the high resolution converter effectively operating at ¼ the internal sample rate. The measurement bandwidth is approximately 1/8 the internal sample rate.

AD4X selects a lower resolution fast converter that operates at 4X the internal sample rate. The measurement bandwidth is approximately 2X the internal sample rate.

The command argument is:

- **input** - { DIGital | AD1X | AD4X | AD_4 }

Default: DIGital

Related Commands: :DSP:FFT:INPut?

Command Syntax: :DSP:FFT:INPut **input**

Example: `:DSP:FFT:INPUT AD_4`

:DSP:FFT:INPut?

This query returns the setting of the :DSP:FFT:INPUT command.

Response argument(s):

- **input** - { DIGital | AD1X | AD4X | AD_4 }

Related Commands: :DSP:FFT:INPut

Command Syntax: :DSP:FFT:INPut?

Response Syntax: :DSP:FFT:INPut **input**

Example: `:DSP:FFT:INPUT?`

Response: `:DSP:FFT:INPUT AD_4`

:DSP:FFT:MODE

This command specifies the type of time domain processing of a sampled signal (this does not apply to FFT result data). The valid choices are: INTRpolate (Interpolate), RAW (Display Samples), PEAK (Peak Values), and ABS (Absolute Values)

If INTRpolate is selected, the DSP will compute the requested data value (see :DSP:SRCParams) by interpolating from adjacent sample points in the signal.

If RAW is selected the DSP will return the closest actual measured value with no interpolation.

In PEAK mode the DSP will return the largest value (positive or negative) between the last requested source point and the current data point. The sign is preserved. Use this mode to measure the envelope of a time domain signal in volts units, for example the gain overshoot of a compressor circuit stimulated with a sine burst waveform.

In ABS mode the DSP uses the same process as Peak Values mode but responds with the absolute value (the sign is not preserved). Use this mode to measure the envelope of a signal in dB units.

The command argument is:

- **mode** - { INTRpolate | ABS | PEAK | RAW }

Default: INTRpolate

Related Commands: :DSP:FFT:MODE?

Command Syntax: :DSP:FFT:MODE **mode**

Example: `:DSP:FFT:MODE RAW`

:DSP:FFT:MODE?

This query returns the setting of the :DSP:FFT:MODE command.

Response argument(s):

- **mode** - { INTRpolate | ABS | PEAK | RAW }

Related Commands: :DSP:FFT:MODE

Command Syntax: :DSP:FFT:MODE?

Response Syntax: :DSP:FFT:MODE **mode**

Example: `:DSP:FFT:MODE?`

Response: `:DSP:FFT:MODE RAW`

:DSP:FFT:PEAK?

This query returns an unsettled reading from the peak monitor for the specified channel.

The command arguments are:

- **channel** - <nr1> (range: 1 or 2)
- **units** - { DBFS | FFS | PCTFs }

Response argument(s):

- **peak** - <nrf>

Related Commands: :DSP:FFT:PEAK

Command Syntax: :DSP:FFT:PEAK? **channel, units**

Response Syntax: :DSP:FFT:PEAK? **peak**

Example: `:DSP:FFT:PEAK? 1,FFS`

Response: `:DSP:FFT:PEAK 0.125632FFS`

FFT:PKTRig

This command specifies the trigger mode of the FFT peak meters (real-time meters). These meters are only valid when the DSP is in OPSTate SETup mode. A zero (0) argument will set the DSP to return the next available peak meter reading. A one (1) argument will set the DSP to abort the current reading cycle and start a new reading cycle when the :DSP:FFT:PEAK? query is received.

This command will cause an execution error if received when the FFT program is in OPSTate READing mode.

The command argument is:

- **mode** - <nr1> (range: 0 or 1)

Default: 1

Related Commands: :DSP:FFT:PEAK?

Command Syntax: :DSP:FFT:PKTRig **mode**

Example: `:DSP:FFT:PKTRIG 1`

:DSP:FFT:PKTRig?

This query returns the setting of the :DSP:FFT:PKTRig command.

The response argument is:

- **mode** - <nr1> (range: 0 or 1)

Related Commands: :DSP:FFT:PKTRig

Command Syntax: :DSP:FFT:PKTRig?

Response Syntax: :DSP:FFT:PKTRig **mode**

Example: `:DSP:FFT:PKTRIG?`

:DSP:FFT:SENS

This command specifies the trigger sensitivity for :DSP:FFT:TRIGsrc.

The command argument is:

- **sens** - <nrf> (range: ≥ 0, ≤ 1) { FFS | DBFS | PCTFs }

Default: 1E-4FFS

Related Commands: :DSP:FFT:TRIGsrc

Command Syntax: :DSP:FFT:SENS **sens**

Example:

```
:DSP:FFT:SENS -36.4782DBFS
```

:DSP:FFT:SENS?

This query returns the setting of :DSP:FFT:SENS command.

The command argument is:

- **unit** -{ FFS | DBFS | PCTFs }

Response argument:

- **sens** - <nrf> (range: ≥ 0, ≤ 1)

Related Commands: :DSP:FFT:SENS

Command Syntax: :DSP:FFT:SENS? **unit**

Response Syntax: :DSP:FFT:SENS **sens**

Example:

```
:DSP:FFT:SENS? FFS
```

Response:

```
:DSP:FFT:SENS 0.015FFS
```

:DSP:FFT:SET?

This query returns all settings for :DSP:FFT. The response consists of a sequence of the DSP FFT settings that may be sent back to the instrument at a later time in order to reset all settings to the same state. Note that some commands are sent more than once and transition through several states in order to achieve a final instrument state.

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands:

Command Syntax: :DSP:FFT:SET?

Response Syntax: :DSP:FFT:SET **settings**

Example:

```
:DSP:FFT:SET?
```

Response:

```
 :DSP:FFT:ACQLENGTH XLENGTH;AVGS 4;AVGTYPE 1
;COUPLING DC;DELAY 0;MODE INTRPOLATE;INPUT D
IGITAL;SRC1 A;SRC2 B;INPUT AD1X;SRC1 JITTER;
SRC2 JITTER;INPUT AD1X;START 0;TRIGGER FREE;
TSLOPE POS;WINDOW EQR;XLENGTH 8192;SENS 0.01
5FFS;PKTRIG 1
```

:DSP:FFT:SRC1

This command specifies the signal source for the currently selected input type for channel 1. Table 8-12 shows the possible sources for each input.

Table 8-12. Possible sources for each input for:DSP:FFT:SRC1

Input	Source
Digital	A, B, None
Low BW (1x) A/D, High BW (4x) A/D, Low BW (/4) A/D	Anlr A, Anlr B, Function Meter Amplitude, Function Meter Ratio, Gen A, Gen B, Jitter Signal, None

In the following command argument list an entry of "A" (or "B") will be interpreted in two ways, depending on whether the input is in Digital or A/D mode. In Digital mode "A" will be interpreted as subframe A. In A/D mode "A" will be interpreted as analog analyzer channel A.

The command argument is:

- **source** - { A | AMPLitude | B | GENA | GENB | JITTer | NONE | RATio }

Default: A

Related Commands: :DSP:FFT:INPut, :DSP:FFT:SRC1?

Command Syntax: :DSP:FFT:SRC1 **source**

Example: `:DSP:FFT:SRC1 A`

:DSP:FFT:SRC1?

This query returns the signal source for the currently selected input for channel 1.

Response argument(s):

- **source** - { A | AMPLitude | B | GENA | GENB | JITTer | NONE | RATio }

Related Commands: :DSP:FFT:SRC1

Command Syntax: :DSP:FFT:SRC1?

Response Syntax: :DSP:FFT:SRC1 **source**

Example: `:DSP:FFT:SRC1?`

Response: `:DSP:FFT:SRC1 A`

:DSP:FFT:SRC2

This command specifies the signal source for the currently selected input type for channel 2. Table 8-13 shows the possible sources for each input.

Table 8-13. Possible sources for each input for r:DSP:FFT:SRC2

Input	Source
Digital	A, B, None
Low BW (1x) A/D, High BW (4x) A/D, Low BW (/4) A/D	Anlr A, Anlr B, Function meter Amplitude, Function Meter Ratio, Gen A, Gen B, Jitter Signal, None

In the following command argument list an entry of "A" (or "B") will be interpreted in two ways, depending on whether the input is in Digital or A/D mode. In Digital mode "A" will be interpreted as subframe A. In A/D mode "A" will be interpreted as analog analyzer channel A.

The command argument is:

- **source** - { A | AMPLitude | B | GENA | GENB | JITTer | NONE | RATio }

Default: B

Related Commands: :DSP:FFT:INPut, :DSP:FFT:SRC2?

Command Syntax: :DSP:FFT:SRC2 **source**

Example: `:DSP:FFT:SRC2 A`

:DSP:FFT:SRC2?

This query returns the signal source for the currently selected input for channel 2.

Response argument(s):

- **source** - { A | AMPLitude | B | GENA | GENB | JITTer | NONE | RATio }

Related Commands: :DSP:FFT:SRC2

Command Syntax: :DSP:FFT:SRC2?

Response Syntax: :DSP:FFT:SRC2 **source**

Example: `:DSP:FFT:SRC2?`

Response: `:DSP:FFT:SRC2 A`

:DSP:FFT:STARt

This command specifies the start time of the transform buffer relative to the trigger point. The argument to this command must be greater than or equal to the trigger delay (:DSP:FFT:DELay). For a more detailed explanation see Section 2 of this manual.

The command argument (in seconds) is:

- **time** - <nrf> (range: ≥ -1E34, ≤ 1E34)

Default: 0.0

Related Commands: :DSP:FFT:DELay, :DSP:FFT:STARt?

Command Syntax: :DSP:FFT:STARt **time**

Example: `:DSP:FFT:START -3.5E-3`

:DSP:FFT:STARt?

This query returns the start time (in seconds) of the transform buffer relative to the trigger point.

Response argument(s):

- **time** - <nrf>

Related Commands: :DSP:FFT:STARt

Command Syntax: :DSP:FFT:STARt?

Response Syntax: :DSP:FFT:STARt **time**

Example: `:DSP:FFT:START?`

Response: `:DSP:FFT:START -0.00350004`

:DSP:FFT:TSLope

This command specifies the acquisition triggering Slope (positive or negative). Selecting Positive causes triggering to occur on a positive transition of the input signal. Negative causes triggering on a negative transition of the input signal.

The command argument is:

- **slope** - { POS | NEG }

Default: POS

Related Commands: :DSP:FFT:TRIGger, :DSP:FFT:TSLope?

Command Syntax: :DSP:FFT:TSLope **slope**

Example: `:DSP:FFT:TSLope POS`

:DSP:FFT:TSLope?

This query returns the setting of the DSP:FFT:TSLOPE command.

Response argument(s):

- **slope** - { POS | NEG }

Related Commands: :DSP:FFT:TSLope

Command Syntax: :DSP:FFT:TSLope?

Response Syntax: :DSP:FFT:TSLope **slope**

Example: `:DSP:FFT:TSLOPE?`

Response: `:DSP:FFT:TSLOPE POS`

:DSP:FFT:TRIGger

This command specifies the DSP FFT program trigger source. Acquisition of signal into the FFT acquisition buffer may commence when the :DSP:ACQX? command is received or may wait for a trigger event after ACQX?, depending upon the setting of the Trigger Source field. The selections are FREE (Free Running), C1Auto (Channel #1 Auto), C1Fixed (Channel #1 Fixed), C2Auto (Channel #2 Auto), C2Fixed (Channel #2 Fixed), DGEN (Digital Generator), ACMains (AC Mains), AGEN (Analog Generator), EXTernal (External), and JGEN (Jitter Generator).

When FREE is selected, signal acquisition begins immediately after the :DSP:ACQX? command is received, regardless of signal amplitude. This is the typical operating mode with steady-state test signals.

The four Channel #1 and Channel #2 selections are software triggers, monitoring the signal (which may come from Digital or A/D sources) on the specified channel. Channel #1 Fix and Channel #2 Fix use a fixed threshold of 1.0%FS (-40 dBFS) on the channel referred to as the triggering threshold and will trigger on the first signal excursion of the selected slope (positive or negative) above that amplitude. The Channel 1 Auto and Channel 2 Auto selections will cause triggering at one-half the peak-to-peak value if the selected channel has a signal amplitude greater than digital infinity zero.

The Digital Generator selection functions only on Dual Domain units (SYS-2300 series). If the digital generator is generating any of the waveforms selectable in the Waveform field, a Digital Generator trigger occurs at each zero crossing of the waveform, positive-going or negative-going. If the Digital Generator is generating a signal from a waveform buffer, a Digital Generator trigger occurs as the first sample is read from the waveform buffer.

The AC Mains selection is the AC line powering System Two.

The Analog Generator Sync selection is the same signal as that at the Generator Aux Signals Sync Output BNC on the front panel of System Two. This signal is a squarewave at the analog generator frequency in sinewave and squarewave waveforms, the envelope of the burst signal in all Burst waveforms, a squarewave at the lower IMD frequency in SMPTE IMD waveform, a squarewave at 1/2 the frequency spacing in CCIF IMD waveform, the squarewave IMD signal in DIM IMD waveform, and a pulse at the pseudo-random repetition rate in Pseudo noise modes. There is no sync signal in Random noise modes.

The External selection refers to pin 3 of the 15-pin D-sub connector on the rear of the DSP module. This source is operational only with the SYS-2300 series Dual Domain units. If pin 3 is high (or open circuit, in which case it is pulled high by an internal pull-up resistor), triggering occurs at the next digital sample. Pulling pin 3 low from an external device holds off triggering, with acquisition being triggered on the next sample after pin 3 is pulled high. This External selection is unaffected by the Slope.

The JGEN selection provides a synchronization trigger from the digital output jitter generator for acquisition of jitter signals from the digital input jitter detector.

The command argument is:

- **source** - { FREE | ACMains | AGEN | C1Auto | C1Fixed | C2Auto | C2Fixed | DGEN | EXTernal | JGEN }

Default: FREE

Related Commands: :DSP:FFT:TRIGger?

Command Syntax: :DSP:FFT:TRIGger **source**

Example: `:DSP:FFT:TRIGGER FREE`

:DSP:FFT:TRIGger?

This query returns the setting of the :DSP:FFT:TRIGGER command.

Response argument(s):

- **source** - { FREE | ACMains | AGEN | C1Auto | C1Fixed | C2Auto | C2Fixed | DGEN | EXTernal | JGEN }

Related Commands: :DSP:FFT:TRIGger

Command Syntax: :DSP:FFT:TRIGger?

Response Syntax: :DSP:FFT:TRIGger **source**

Example: `:DSP:FFT:TRIGGER?`

Response: `:DSP:FFT:TRIGGER FREE`

:DSP:FFT:WINDow

This command specifies the windowing function to be applied to the signal during FFT processing. Unless the signal to be analyzed was deliberately generated in such a fashion as to be synchronous (to go through an exact integer number of cycles) with the transform buffer, a window function must be applied. Each available window function has a different set of trade-offs in effective close-in selectivity (signal spillover into nearby FFT bins). An amplitude measurement error will result without a window function if the signal does not fall at an exact bin center frequency. The available windows selections for the FFT program are Blackman-Harris, Hann, Flat-top, Equiripple, None, and None, Move to BIN Center (NMTBc).

The command argument is:

- **window** - { BH | EQR | FLAT | HANN | NMTBc | NONE }

Default: BH

Related Commands: :DSP:FFT:WINDow?

Command Syntax: :DSP:FFT:WINDow **window**

Example: `:DSP:FFT:WINDOW NONE`

:DSP:FFT:WINDow?

This query returns the setting of the :DSP:FFT:WINDOW command.

Response argument(s):

- **window** - { BH | EQR | FLAT | HANN | NMTBc | NONE }

Related Commands: :DSP:FFT:WINDow

Command Syntax: :DSP:FFT:WINDow?

Response Syntax: :DSP:FFT:WINDow **window**

Example: `:DSP:FFT:WINDOW?`

Response: `:DSP:FFT:WIND NONE`

:DSP:FFT:XLENgth

This command specifies the FFT transform buffer length. This transform buffer length defines how large a region of the DSP acquisition buffer will be transformed to the frequency domain by the FFT function. The start time (:DSP:FFT:STARt) specifies (relative to the trigger point) where in the DSP acquisition buffer the region starts.

For a more detailed explanation of the relationship between start time, acquisition length, trigger delay, and transform length see Section 2 of this manual.

The argument to this command will be rounded up to the nearest legitimate setting: 256, 512, 1024, 2048, 4096, 8192, 16384.

The command argument is:

- **length** - <nr1> (range: ≥ 0, ≤ 16384; round up to the nearest of 256, 512, 1024, 2048, 4096, 8192, 16384)

Default: 1024

Related Commands: :DSP:FFT:ACQLength, :DSP:FFT:DELay, :DSP:FFT:STARt, :DSP:FFT:XLENgth?

Command Syntax: :DSP:FFT:XLENgth **length**

Example:

```
:DSP:FFT:XLENGTH 512
```

:DSP:FFT:XLENgth?

This query returns the setting of the :DSP:FFT:XLENGTH command.

Response argument(s):

- **length** - <nr1> (range: ≥ 0, ≤ 16384; round up to the nearest of 256, 512, 1024, 2048, 4096, 8192, 16384)

Related Commands: :DSP:FFT:XLENgth

Command Syntax: :DSP:FFT:XLENgth?

Response Syntax: :DSP:FFT:XLENgth **length**

Example:

```
:DSP:FFT:XLENGTH?
```

Response:

```
:DSP:FFT:XLENGTH 512
```

:DSP:INTervu

Compound command header for Digital Interface Analyzer commands.

:DSP:INTervu:AMPL?

This query returns a single point amplitude measurement from the Intervu DSP program.

The first parameter specifies the response units of either volts (V) or dBV (DBV).

The second parameter specifies the value of the source point being interrogated. The units of the second parameter are specified by the first and second parameters of the :DSP:SRCParams command.

This measurement must be enabled with the :DSP:OPSTate READING AMPL command.

The query argument(s) are:

- **units** - { V | DBV }
- **src_value** - <nrf> (range: depends on SRCParams settings)

The response argument(s) are:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:INTervu:AMPL? **units, src_value**

Response Syntax: :DSP:INTervu:AMPL **reading**

Example1:

```
 *RST;:DSP:PROGRAM INTERVU;:DGEN:OUTPUT OFF;
DITHERTYPE NONE;:DIN:FORMAT GENMON;:DOUT:FOR
MAT XLR;RATE 44100HZ;AMPL 5;:DSP:SRCPARAMS T
IME,SEC,0,0,0,ARB;INTERVU:MODE INTRPOLATE;:D
ELAY .1;:DSP:OPSTATE READING,AMPL;ACQX?;:DSP
:INTERVU:AMPL? V,100E-6
```

(Read an INTERVU waveform sample at 100 uS after the trigger point with 44.1 kHz sample rate * 32 bits per frame when all zeros are sent as data and pulse amplitude is 10 Vpp with digital output unterminated).

Response1:

```
:DSP:ACQX 0;:DSP:INTERVU:AMPL -4.88232
```

Example2:

```
 *RST;:DSP:PROGRAM INTERVU;:DGEN:OUTPUT OFF;
DITHERTYPE NONE;:DIN:FORMAT GENMON;:DOUT:FOR
MAT XLR;RATE 44100HZ;AMPL 5;:DSP:SRCPARAMS F
REQ,HZ,0,0,0,ARB;INTERVU:MODE INTRPOLATE;:DE
LAY .1;:DSP:OPSTATE READING,AMPL;ACQX?;:DSP:
INTERVU:AMPL? V,1411200
```

(Read an INTERVU spectrum bin at the 1.4112 MHz fundamental frequency for 44.1 kHz sample rate * 32 bits per frame when all zeros are sent as data and pulse amplitude is 10 Vpp with digital output unterminated).

Response2: `:DSP:INTERVU:AMPL 5.43174`

:DSP:INTervu:AVGS

This command specifies the number of acquisitions to average. The argument to this command will be rounded up to the nearest legitimate setting: 1, 2, 4, 8, 16, 32, 64, 128.

The command argument is:

- **number** - <nr1> (range: ≥ 1, ≤ 128)

Default: 1

Related Commands: :DSP:INTervu:AVGS?

Command Syntax: :DSP:INTervu:AVGS **number**

Example: `:DSP:INTERVU:AVGS 17`

:DSP:INTervu:AVGS?

This query returns the setting of the :DSP:INTERVU:AVGS command.

Response argument(s):

- **number** - <nr1> (range: ≥ 1, ≤ 128)

Related Commands: :DSP:INTervu:AVGS

Command Syntax: :DSP:INTervu:AVGS?

Response Syntax: :DSP:INTervu:AVGS **number**

Example: `:DSP:INTERVU:AVGS?`

Response: `:DSP:INTERVU:AVGS 32`

:DSP:INTervu:JDETection

This command specifies the jitter detection mode of the INTERVU DSP program (Preamble, Stable Bits, All Bits).

The Preamble selection (PREamble) uses the average rate of the trailing edge of the first three-UI-wide pulse in each preamble as the stable clock reference.

The Stable Bits selection (STABLe) derives the stable reference clock at 1/4 the actual cell (bit) rate, synchronized to the beginning transition of the preamble.

The All Bits selection (ALL) derives the stable reference clock at the actual cell (bit) rate. Since there are 64 cells per frame and the frame rate is the audio sample rate, the reference clock is at 64 times the sample rate and the effective jitter measurement bandwidth is 32 times the audio sample rate (1.536 MHz at a 48 kHz sample rate).

In addition to measuring jitter on an AES/EBU or SPDIF/EIAJ serial digital input signal, INTERVU can also measure jitter on any 28 kHz – 13 MHz squarewave connected to the BNC digital input connector. This feature permits direct measurement of clock jitter on A/D and D/A converters. The waveform of the jitter may be acquired (time domain) or a spectrum analysis of the jitter may be performed (frequency domain). Use either Squarewave Rising (RISE) or Squarewave Falling (FALL) to activate this feature. The RISE selection measures jitter on rising edges of the signal and the FALL selection measures on falling edges.

The command argument is:

- **type** - { STABle | ALL | PREamble | FALL | RISE }

Default: STABle

Related Commands: :DSP:INTervu:JDETection?

Command Syntax: :DSP:INTervu:JDETection **type**

Example: `:DSP:INTERVU:JDETECTION PREAMBLE`

:DSP:INTervu:JDETection?

This query returns the settings of the :DSP:INTERVU:JDETECTION command.

Response argument(s):

- **type** - { STABle | ALL | PREamble }

Related Commands: :DSP:INTervu:JDETection

Command Syntax: :DSP:INTervu:JDETection?

Response Syntax: :DSP:INTervu:JDETection **type**

Example: `:DSP:INTERVU:JDETECTION?`

Response: `:DSP:INTERVU:JDETECTION PREAMBLE`

:DSP:INTervu:JITTer?

This query returns a single point jitter measurement from Intervu. The first parameter specifies the response units of seconds or unit intervals (UI) and the second parameter specifies the value of the source point being interrogated. The units of the second parameter are specified by the :DSP:SRCParams command (second parameter).

To enable this reading, the JITTer parameter must be specified in the :DSP:OPSTate READing command.

The query argument(s) are:

- **units** - { SEC | UI }
- **src_value** - <nrf> (range: depends on SRCParams settings)

The response argument(s) are:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:INTervu:JITTer? **units, src_value**

Response Syntax: :DSP:INTervu:JITTer **reading**

Example1:

```
*RCL 0;:HEADER ON;:DGEN:OUTPUT AB;AMPL
AB,0DBFS;WFM SPCIAL,JTEST;:DIN:FORMAT
GENMON;SCALEFREQBY MEASURED;BANDWIDTH
50;:DOUT:FORMAT XLR;AMPL 5;CABLESIM
OFF;CMOUTPUT OFF;INVALID 0;JWFM
SINE;JAMPL 1.0UI;JFREQ 1000;PREEMPHASIS
OFF;RATE 44100HZ;RESOLUTION 24;RFENABLE
FIXED;NOUTPUT OFF;:DSP:TIMEOUT 5;PROGRAM
INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION
ALL;MODE RAW;MON JITTER;TRIGGER
JITTER;WINDOW BH;:DELAY 0.5;:DSP:OPST
SET;SRCP TIME,SEC,0,0,0,LIN;OPST
READ,JITTER;ACQX?;:DELAY
1;:DSP:INTERVU:JITT? SEC,1E-3
```

(measure jitter amplitude at 1 ms from trigger point)

Response1:

```
:DSP:ACQX 0;:DSP:INTERVU:JITTER -1.29927E-07
```

Example2:

```
*RCL 0;:HEADER ON;:DGEN:OUTPUT AB;AMPL
AB,0DBFS;WFM SPCIAL,JTEST;:DIN:FORMAT
GENMON;SCALEFREQBY MEASURED;BANDWIDTH
50;:DOUT:FORMAT XLR;AMPL 5;CABLESIM
OFF;CMOUTPUT OFF;INVALID 0;JWFM SINE;JAMPL
1UI;JFREQ 1000;PREEMPHASIS OFF;RATE
44100HZ;RESOLUTION 24;RFENABLE FIXED;NOUTPUT
OFF;:DSP:TIMEOUT 5;PROGRAM
INTERVU;INTERVU:AVGS 1;JDETECTION ALL;MODE
RAW;MON JITTER;TRIGGER JITTER;WINDOW
BH;:DELAY 0.5;:DSP:OPST SET;SRCP
FREQ,HZ,0,0,0,LOG;OPST
```

```
READ,JITTER;ACQX?;:DELAY
1;:DSP:INTERVU:JITT? HZ,1000
```

(measure jitter amplitude spectrum at 1 kHz)

Response2:

```
:DSP:ACQX 0;:DSP:INTERVU:JITTER 1.07544E-07
```

:DSP:INTervu:LOWer?

This query returns a single point lower eye pattern amplitude measurement from Intervu. The first parameter specifies the response units and the second parameter specifies the value of the source point being interrogated. The units of the second parameter are specified by the :DSP:SRCParams command (3rd parameter).

To enable this reading, the LOWer parameter must be specified in the :DSP:OPSTate READing command.

The query argument(s) are:

- **units** - { DBM | DBRA | DBRB | DBU | DBV | V }
- **src_value** - <nrf> (range: depends on SRCParams settings)

The response argument(s) are:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:INTervu:UPPer, :DSP:SRCParams,:DSP:OPSTate

Command Syntax: :DSP:INTervu:LOWer? **units, src_value**

Response Syntax: :DSP:INTervu:LOWer **reading**

Example:

```
 :HEADER ON;:MON:MODE STEREO;STEREO DSP;:DGE
N:OUTPUT AB;AMPL AB,-10DBFS;WFM SINE,SINE;:D
IN:FORMAT XLR;SCALEFREQBY MEASURED;:DOUT:FOR
MAT XLR;AMPL 5;CABLESIM OFF;CMOUTPUT OFF;INV
ALID 0;JWFM NONE;NOUTPUT OFF;PREEMPHASIS OFF
;RATE 44100HZ;RESOLUTION 24;RFENABLE FIXED;:
DSP:TIMEOUT 5;PROGRAM INTERVU;:DSP:INTERVU:A
VGS 1;JDETECTION ALL;MODE EYE;MON JITTER;TRI
GGER ARCV;WINDOW BH;:DELAY 0.5;:DSP:SRCP TIM
E,SEC,0,0,0,LIN;OPST READ,LOWER,UPPER;ACQX?;
:DELAY 0.5;:DSP:INT:LOWER? V,50E-9;UPPER? V,
50E-9
```

(measure the lower and upper eye pattern amplitude at 50 nS from the beginning of the eye pattern)

Response:

```
 :DSP:ACQX 0;:DSP:INTERVU:LOWER -1.91648;:DS
P:INTERVU:UPPER 1.91645
```

:DSP:INTervu:MODE

Four modes are available in INTERVU for processing the amplitude-versus-time relationship of a sampled digital interface signal. These modes apply to time domain measurements (amplitude versus time graphs) and histograms, but have no effect on frequency domain spectrum measurements.

The four available modes are INTRpolate (Interpolate in APWIN), RAW (Display Samples), PEAK (Peak Values), and EYE (Eye Pattern).

When INTRpolate is selected, the DSP module will perform an interpolation calculation based on the fact that the signal was band-limited by an internal 30 MHz low-pass filter before sampling.

When RAW is selected, no processing takes place in the DSP module. For each time point (see :DSP:SRCParams), the DSP returns the amplitude of the nearest-in-time sample .

When PEAK is selected, the DSP returns the largest value (positive or negative) between the last time point and the current time point. The sign of the largest value is preserved (DSP returns positive and negative values).

The Eye Pattern selection processes the signal acquisition to produce upper and lower eye pattern amplitude vs time data. Following acquisition of the digital interface signal and extraction of an average clock signal from it, the worst-case (nearest to zero volts) amplitude is determined for each time increment relative to the beginning of each data cell. These values are used when UPPER and LOWER are selected as parameters of the :DSP:OPSTate READ command.

The valid settings for the :DSP:SRCParams parameters, :DSP:OPSTate READ parameters, and :DSP:INTERVU measurement queries are dependent upon the MODE parameter. The table below shows the valid combinations of these settings for INTERVU measurements.

The MODE command must be set to EYE in order to acquire Eye Pattern measurements. All other measurements are possible if MODE is set to any other setting except EYE.

Table 8-14. Valid combinations of :DSP:INTERVU:MODE and INTERVU measurement queries

	Probability **PROB?**	**Jitter** **JITTer?**	**Amplitude** **AMPL?**	**Upper Eye** **UPPer?**	**Lower Eye** **LOWer?**
Amplitude :DSP:OPSTate READ,AMPL	**Valid**	Invalid	Invalid	Invalid	Invalid
Jitter :DSP:OPSTate READ,JITTER	**Valid**	Invalid	Invalid	Invalid	Invalid
Frequency :DSP:SRCPARAM FREQ, …	**Valid**	**Valid**	**Valid**	Invalid	Invalid
Time :DSP:SRCPARAM TIME, …	**Valid**	**Valid**	**Valid**	**Valid**	**Valid**

Note: Probability, Jitter, and Amplitude measurements are only valid when :DSP:INTervu:MODE is INTRpolate, RAW, or PEAK. Upper and Lower Eye measurements are only valid when :DSP:INTervu:MODE is EYE.

The command argument is:

- **mode** - { INTRpolate | EYE | PEAK | RAW }

Default: INTRpolate

Related Commands: :DSP:SRCParams, :DSP:INTervu:MODE?

Command Syntax: :DSP:INTervu:MODE **mode**

Example: `:DSP:INTERVU:MODE RAW`

:DSP:INTervu:MODE?

This query returns the data processing mode of the Intervu DSP program.

Response argument(s):

- **mode** - { INTRpolate | EYE | PEAK | RAW }

Related Commands: :DSP:INTervu:MODE

Command Syntax: :DSP:INTervu:MODE?

Response Syntax: :DSP:INTervu:MODE **mode**

Example: `:DSP:INTERVU:MODE?`

Response: `:DSP:INTERVU:MODE RAW`

:DSP:INTervu:MON

This command specifies whether the built-in loudspeaker (or stereo headphones plugged into System Twos front-panel jack) monitor the imbedded digital audio signal or the demodulated jitter signal. In the Audio Monitor field, select Imbedded Audio or Jitter Signal as desired. In the Speaker-Headphone (MON) subsystem, specify Stereo and the DSP Monitor selection or Mono and either DSP Monitor A or DSP Monitor B.

The command argument is:

- **src** - { AUDio | JITTer }

Default: AUDio

Related Commands: :MON:CHANnel, :MON:MODE, :MON:SRC, :DSP:INTervu:MON?

Command Syntax: :DSP:INTervu:MON **src**

Example: `:DSP:INTERVU:MON AUDIO`

:DSP:INTervu:MON?

This query returns the signal source for the built-in loudspeaker or stereo headphones.

Response argument(s):

- **src** - { AUDio | JITTer }

Related Commands: :DSP:INTervu:MON

Command Syntax: :DSP:INTervu:MON?

Response Syntax: :DSP:INTervu:MON **src**

Example: `:DSP:INTERVU:MON?`

Response: `:DSP:INTERVU:MON AUDIO`

:DSP:INTervu:PROBability?

This query returns a single point probability measurement from Intervu. The first parameter specifies the response units and the second parameter specifies the value of the source point being interrogated. The units of the second parameter are specified by the :DSP:SRCParams command (3rd parameter).

To enable this reading, the PROBability parameter must be specified in the :DSP:OPSTate READing command.

The query argument(s) are:

- **units** - { DB | PCT | X_Y }
- **src_value** - <nrf> (range: depends on SRCParams settings)

The response argument(s) are:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:INTervu:PROBability? **units,src_value**

Response Syntax: :DSP:INTervu:PROBability **reading**

Example: `:DSP:INTERVU:PROBABILITY? PCT, 20000`

Response: `:DSP:INTERVU:PROBABILITY 30`

:DSP:INTervu:SET?

This query returns the state of the Intervu DSP program settings (averages, jitter detection, monitor, trigger, and window).

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands: :DSP:INTervu:SET

Command Syntax: :DSP:INTervu:SET?

Response Syntax: :DSP:INTervu:SET **settings**

Example:

```
:DSP:INTERVU:SET?
```

Response:

```
:DSP:INTERVU:AVGS 1;JDETECTION STABLE;MODE I
NTRPOLATE;MONITOR AUDIO;TRIGGER ARCV;WINDOW
BH
```

:DSP:INTervu:TRIGger

This command specifies which portion of the serial digital interface pulse train or an interface Error triggers the acquisition of the digital interface signal into INTERVUs 256 k sample acquisition buffer.

The two Receive Preamble selections (ARCV & BRCV) cause signal to be acquired at the first Channel A (left) or Channel B (right) Preamble.

The two Transmitted Preamble selections (AXMT & BXMT) cause signal to be acquired beginning at the start of the first Channel A or Channel B Transmit Preamble.

The Receive Error selection (RCVerr) is a pre-trigger, causing the 256k samples (about 3.9 milliseconds) immediately preceding an interface Error Flag to be retained (approximately 39 microseconds of signal following the occurrence of the error will also be retained).

The Receive Block selection (RCVBlock) causes signal to be acquired beginning at the end of the first Channel Status Block Preamble received.

The Jitter Generator selection (JITTer) causes a trigger at every zero crossing of the sinewave, squarewave, or noise signal generated by the DIO jitter generator.

The External Pre-Trigger and External Post-Trigger selections (XPRetrig & XPOSttrig) operate in conjunction with pin 3 of the 15-pin D-sub connector on the rear of the DSP module. If pin 3 (which is internally pulled up) is high, triggering occurs at the next digital sample. Pulling pin 3 low from an external device holds off triggering, with acquisition being triggered on the next sample after pin 3 is pulled high. The External Pre-Trigger selection results in retaining the 256 k samples immediately preceding this sample, while the External Post-

Trigger selection retains the 256 k samples immediately following this sample.

The command argument is:

- **type** - { ARCV | AXMT | BRCV | BXMT | JITTer | RCVerr | RCVBlock | XPOSttrig | XPRetrig }

Default: ARCV

Related Commands: :DSP:INTervu:TRIGger?

Command Syntax: :DSP:INTervu:TRIGger **type**

Example: `:DSP:INTERVU:TRIGGER ARCV`

:DSP:INTervu:TRIGger?

This query returns the interface condition which triggers the acquisition of the digital interface signal. See :DSP:INTervu:TRIGger for a more detailed explanation.

Response argument(s):

- **type** - { ARCV | AXMT | BRCV | BXMT | JITTer | RCVerr | RCVBlock | XPOSttrig | XPRetrig }

Related Commands: :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:TRIGger?

Response Syntax: :DSP:INTervu:TRIGger **type**

Example: `:DSP:INTERVU:TRIGGER?`

Response: `:DSP:INTERVU:TRIGGER ARCV`

:DSP:INTervu:UPPer?

This query returns a single point upper eye pattern amplitude measurement from INTervu. The first parameter specifies the response units and the second parameter specifies the value of the source point being interrogated. The units of the second parameter are specified by the :DSP:SRCParams command (3rd parameter).

To enable this reading, the UPPer parameter must be specified in the :DSP:OPSTate READing command.

The query argument(s) are:

- **units** - { DBM | DBRA | DBRB | DBU | DBV | V }
- **src_value** - <nrf> (range: depends on SRCParams settings)

The response argument(s) are:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:INTervu:LOWer, :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:INTervu:UPPer? units, src_value

Response Syntax: :DSP:INTervu:UPPer **reading**

Example: `:DSP:INTERVU:UPPER? V, 20000`

Response: `:DSP:INTERVU:UPPER 1.5`

:DSP:INTervu:WINDow

This command specifies the windowing function to be applied prior to the FFT computation. This function is applied to the segment of interest in the acquisition buffer.

The fundamental mathematics behind an FFT makes the assumption that the portion of an acquired signal being transformed is a perfect, synchronous section of a signal which continues indefinitely. This implies that the beginning of one section must "match" exactly with the end of the previous section. If this is not the case, a window function must be applied to force this condition.

Each available window function has a different set of trade-offs in effective close-in selectivity (signal spillover into nearby FFT bins) and in amplitude measurement error if the signal does not fall at an exact bin center frequency.

The command argument is:

- **window** - { BH | EQR | FLAT | HANN | NONE }

Default: BH

Related Commands: :DSP:INTervu:WINDow?

Command Syntax: :DSP:INTervu:WINDow **window**

Example: `:DSP:INTERVU:WINDOW BH`

:DSP:INTervu:WINDow?

This query returns the windowing function to be applied prior to the FFT computation.

Response argument(s):

- **window** - { BH | EQR | FLAT | HANN | NONE }

Related Commands: :DSP:INTervu:WINDow

Command Syntax: :DSP:INTervu:WINDow?

Response Syntax: :DSP:INTervu:WINDow **window**

Example: `:DSP:INTERVU:WINDOW?`

Response: `:DSP:INTERVU:WINDOW BH`

:DSP:MLS

Compound command header for Digital Interface MLS (Maximum Length Sequence) commands. The algorithm for using the MLS DSP program involves several steps:

1. Load program and setup input, mode, and source1.
2. Setup.
3. Set the source to MLS Time.
4. Set source start time to the beginning of the impulse.
5. Set source stop time to just prior to the first reflection.
6. Enable a data buffer to MLS Ch 1 Ampl
7. Acquire the time based data.
8. Set Delay Time to location of the signal peak.
9. Change source to MLS Freq.
10. Set the start freq and stop freq.
11. Retransform the acquired data.
12. Examine the freq and phase response data points

At this point the data in the two data buffers will contain the frequency response and the phase response.

Assume the impulse response has been previously characterized, so the start time of the impulse, the time of the impulse peak, and the time of the first reflection are known. Also assume that the values of these events are:

Start time of impulse - 0.0023 s
Time of impulse peak - 0.0035 s
Time of first reflection - 0.0055 s

Then the following list of commands will accomplish the above algorithm:

```
:DSP:PROGRAM MLS
:DSP:MLS:INPUT AD1X
:DSP:MLS:MODE PEAK
:DSP:SRCPARAMS TIME, SEC, 0.0023, 0.0055, 16383, ARBITRARY
:DSP:MLS:DELAY 0.0035
:DSP:OPSTATE READING, AMP1
:DSP:ACQX?
:DSP:OPSTATE SETUP
:DSP:SRCPARAMS FREQ, HZ, 20, 20000, 16383, ARBITRARY
:DSP:RETRANSFORM
```

Below is a macro definition that implements the above sequence of commands. This macro assumes that the start time of the impulse ($1), the time of the impulse peak ($2), and the time just prior to the first reflection ($3) are passed as parameters.

```
DMC "DO_MLS",#3250
      :DSP:PROGRAM MLS;:DSP:MLS:INPUT AD1X;MODE PEAK;:DSP:SRCPARAMS \
      TIME,SEC,$1,$3,16383,ARBITRARY;:DSP:MLS:DELAY $2;\
      :DSP:OPSTATE READING,AMP1;:DSP:ACQX?;:DSP:OPSTATE SETUP;\
      DSP:SRCPARAMS FREQ,HZ,20,20000,16383,ARBITRARY;:DSP:RETRANSFORM
```

The format of the macro invocation and the macro invocation appear below:

```
(DO_MLS <start time>, <stop time>, <peak time>)
DO_MLS 0.0023, 0.0035, 0.0055
```

The response to the macro invocation would be:

```
0
```

(Since there is only one query in the macro.)

Once the data has been acquired and transformed, the :DSP:MLS:AMPL? command is used to select and retrieve individual data points.

:DSP:MLS:AMPL?

This query returns a single point amplitude measurement from MLS. The first parameter specifies the DSP channel (either 1 or 2), the second parameter specifies the response units, and the third parameter specifies the value of the source point being interrogated. The units of the third parameter are specified by the :DSP:SRCParams command (second parameter). Note that the correct choice of units depends upon the input type (AD1X or DIGital) and the source type. This dependency is shown in Table 8-15.

Table 8-15. Units dependencies,:DSP:MLS: AMPL? query

Input Type: INPUT	Source Type: SRC1 & SRC2	Valid Units
AD1X	A, AMPLitude, B, GENA, GENB	DBM, DBRA, DBRB, DBU, DBV, V
DIGital	A, B	DBFS, DBU, DBV, FFS, PCTFS, V, VP, VPP

The query argument(s) are:

- **channel** - <nr1> (range: ≥ 1, ≤ 2)
- **units** - { DBFS | DBM | DBRA | DBRB | DBU | DBV | FFS | PCTFS | V | VP | VPP }
- **src_value** - <nrf> (range: depends on SRCParams settings)

To enable this reading, the AMP1 and/or AMP2 parameters must be specified in the :DSP:OPSTate READing command.

The response argument(s) are:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:MLS:AMPL? **channel, units, src_value**

Response Syntax: :DSP:MLS:AMPL **reading**

Example: `:DSP:MLS:AMPL? 1, FFS, 20000`

Response: `:DSP:MLS:AMPL 0.6`

:DSP:MLS:DELay

This command specifies the signal time delay reference (for phase measurements). This information allows the DSP to subtract out the transit time delay from the phase readings. The possible units are: seconds, feet and meters.

The maximum value is:

3.141 Seconds (3545 Feet, 1081 Meters)

The command argument is:

- **delay** - <nrf> (range: ≥ 0, ≤ 3.141 s maximum range depends on internal sample rate)
  { SEC | FT | M }

Default: 0.0 SEC

Related Commands: :DSP:MLS:DELay?

Command Syntax: :DSP:MLS:DELay **delay**

Example: `:DSP:MLS:DELAY 0.50 SEC`

:DSP:MLS:DELay?

This query returns the speaker to microphone time delay reference (in specified units).

Query argument:

- **units** - { SEC | FT | M }

Response argument(s):

- **delay** - <nrf> { SEC | FT | M }

Related Commands: :DSP:MLS:DELay

Command Syntax: :DSP:MLS:DELay? **units**

Response Syntax: :DSP:MLS:DELay **delay**

Example: `:DSP:MLS:DELAY? SEC`

Response: `:DSP:MLS:DELAY 0.5SEC`

:DSP:MLS:ETWindow

This command specifies frequency window used in the energy-time curve computation process. The energy-time curve computation process involves transforming the impulse response to the frequency domain, doing further processing in the frequency domain, and transforming the result back to the time domain. A frequency window may be used for the conversion from frequency domain back to time domain. The choices are: No window (NONE), half Hann (HALF), Hann (HANN), <240 >8kHz (H240), and <120 >16kHz (H120).

The command argument is:

- **window** - {NONE | H120 | H240 | HALF | HANN }

Default: NONE

Related Commands: :DSP:MLS:ETWindow?

Command Syntax: :DSP:MLS:ETWindow **window**

Example: `:DSP:MLS:ETWINDOW HANN`

:DSP:MLS:ETWindow?

This query returns the frequency window used in energy-time curve computations.

Response argument(s):

- **window** - {NONE | H120 | H240 | HALF | HANN }

Related Commands: :DSP:MLS:ETWindow

Command Syntax: :DSP:MLS:ETWindow?

Response Syntax: :DSP:MLS:ETWindow **window**

Example: `:DSP:MLS:ETWINDOW?`

Response: `:DSP:MLS:ETWINDOW HANN`

:DSP:MLS:INPut

This command specifies whether the input signal is Analog or Digital (and the internal sample rate multiplier). When the signal is Analog the input signal is routed through A/Ds that operate at the internal sample rate (AD1X).

The command argument is:

- **input** - { DIGital | AD1X }

Default: DIGital

Related Commands: :DSP:MLS:INPut?

Command Syntax: :DSP:MLS:INPut **input**

Example: `:DSP:MLS:INPUT AD1X`

:DSP:MLS:INPut?

This query response indicates if the input domain is analog or digital.

Response argument(s):

- **input** - { DIGital | AD1X }

Related Commands: :DSP:MLS:INPut

Command Syntax: :DSP:MLS:INPut?

Response Syntax: :DSP:MLS:INPut **input**

Example: `:DSP:MLS:INPUT?`

Response: `:DSP:MLS:INPUT AD1X`

:DSP:MLS:MODE

Three modes are available in MLS for processing the amplitude-versus-time relationship of a sampled digital interface signal. These modes are applicable to digital storage oscilloscope operation (amplitude versus time graphs) and histograms, but have no effect on FFT spectrum analysis.

The three available modes are INTRpolate (Interpolate in APWIN), RAW (Display Samples), and PEAK (Peak Values).

When INTRpolate is selected, the DSP module will perform an interpolation calculation using the nearest amplitude points.

When RAW is selected, no processing takes place in the DSP module. At each time value plotted on the X-axis, the DSP simply sends the amplitude of the nearest-in-time acquired sample.

When PEAK is selected, the DSP searches all sample amplitudes in the acquisition buffer between each pair of horizontal axis time values and stores the largest positive or negative value in that span (preserving the plus or minus sign).

The command argument is:

- **mode** - { INTRpolate | PEAK | RAW }

Default: INTRpolate

Related Commands: :DSP:MLS:MODE?

Command Syntax: :DSP:MLS:MODE **mode**

Example: `:DSP:MLS:MODE RAW`

:DSP:MLS:MODE?

This query returns the data processing mode of the Intervu DSP program.

Response argument(s):

- **mode** - { INTRpolate | PEAK | RAW }

Related Commands: :DSP:MLS:MODE

Command Syntax: :DSP:MLS:MODE?

Response Syntax: :DSP:MLS:MODE **mode**

Example: `:DSP:MLS:MODE?`

Response: `:DSP:MLS:MODE RAW`

:DSP:MLS:PEAK?

This query returns a reading (unsettled only) from the peak monitor on the specified channel.

The command arguments are:

- **channel** - <nr1> (range: ≥ 1, ≤ 2)
- **units** - { FFS | DBFS | PCTFs }

Response argument(s):

- **peak** - <nrf> { FFS | DBFS | PCTFs }

Related Commands:

Command Syntax: :DSP:MLS:PEAK? **channel, units**

Response Syntax: :DSP:MLS:PEAK? **peak**

Example: `:DSP:MLS:PEAK? 1, FFS`

Response: `:DSP:MLS:PEAK 0.5FFS`

:DSP:MLS:PHASe?

This query returns a single point phase measurement from MLS. The first parameter specifies the DSP channel (either 1 or 2), the second parameter specifies the response units of degrees, and the third parameter specifies the value of the source point being interrogated. The units of the third parameter are specified by the :DSP:SRCParams command (second parameter).

To enable this reading, the PHA1 and/or PHA2 parameters must be specified in the :DSP:OPSTate READing command.

The query argument(s) are:

- **channel** - <nr1> (range: ≥ 1, ≤ 2)
- **units** - { DEG }
- **src_value** - <nrf> (range: depends on SRCParams settings)

The response argument(s) are:

- **reading** - <nrf>

Default: none

Related Commands: :DSP:SRCParams, :DSP:OPSTate

Command Syntax: :DSP:MLS:PHASe? **channel, units, src_value**

Response Syntax: :DSP:MLS:PHASe **reading**

Example: `:DSP:MLS:PHASE? 1, DEG, 20000`

Response: `:DSP:MLS:PHASE 145`

:DSP:MLS:PKTRig

This command specifies the trigger mode of the MLS peak meters (real-time meters). These meters are only valid when the DSP is in OPSTate SETup mode. A zero (0) argument will set the DSP to return the next available peak meter reading. While a one (1) argument will set the DSP to abort the current reading cycle and start a new reading cycle when the :DSP:MLS:PEAK? query is received.

The command argument is:

- **mode** - <nr1> (range: ≥ 0, ≤ 1)

Default: 1

Related Commands: :DSP:MLS:PEAK?

Command Syntax: :DSP:MLS:PKTRig **mode**

Example: `:DSP:MLS:PKTRIG 1`

:DSP:MLS:PKTRig?

This query returns the trigger mode of the MLS peak meters (real-time meters). These meters are only valid when the DSP is in OPSTate SETup mode. A zero (0) response argument indicates the DSP will return the next available peak meter reading. While a one (1) argument indicates the DSP will abort the current reading cycle and start a new reading cycle when the :DSP:MLS:PEAK? query is received.

The response argument is:

- **mode** - <nr1> (range: ≥ 0, ≤ 1)

Related Commands: :DSP:MLS:PKTRig

Command Syntax: :DSP:MLS:PKTRig?

Response Syntax: :DSP:MLS:PKTRig **mode**

Example:
```
:DSP:MLS:PKTRIG?
```

Response:
```
:DSP:MLS:PKTRIG 1
```

:DSP:MLS:SET?

This query returns all settings for :DSP:MLS

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands: :DSP:MLS:SET

Command Syntax: :DSP:MLS:SET?

Response Syntax: :DSP:MLS:SET **settings**

Example:
```
:DSP:MLS:SET?
```

Response:
```
:DSP:MLS:DELAY 0SEC;ETWINDOW NONE;INPUT DIGI
TAL;SRC1 A;SRC2 B;INPUT AD1X;SRC1 A;SRC2 B;I
NPUT DIGITAL;MODE INTRPOLATE;START NONE;STOP
 NONE;TIME IMPULSE;TRIGGER AGEN;PKTRIG 1
```

:DSP:MLS:SRC1

This command specifies the signal source for the currently selected input type for channel 1. The possible sources (for each input) are shown in Table 8-16.

Table 8-16. Possible sources for each input type for :DSP:MLS:SRC1 command

Input	Source
Digital	A, B, None
Low BW (1x) A/D	Anlr A, Anlr B, Function Meter Amplitude, Gen A, Gen B, None

In the following command argument list an entry of "A" (or "B") will be interpreted in two ways, depending on whether the input is in Digital or A/D mode.In Digital mode "A" will be interpreted as subframe A; while in A/D mode "A" will be interpreted as Analyzer channel A.

The command argument is:

- **source** - { A | AMPLitude | B | GENA | GENB | NONE }

Default: A

Related Commands: :DSP:MLS:INPut, :DSP:MLS:SRC1?

Command Syntax: :DSP:MLS:SRC1 **source**

Example: `:DSP:MLS:SRC1 A`

:DSP:MLS:SRC1?

This query returns the signal source for the currently selected input for channel 1.

Response argument(s):

- **source** - { A | AMPLitude | B | GENA | GENB | NONE }

Related Commands: :DSP:MLS:SRC1

Command Syntax: :DSP:MLS:SRC1?

Response Syntax: :DSP:MLS:SRC1 **source**

Example: `:DSP:MLS:SRC1?`

Response: `:DSP:MLS:SRC1 A`

:DSP:MLS:SRC2

This command specifies the signal source for the currently selected input type for channel 2. The possible sources (for each input) are shown in Table 8-17.

Table 8-17. Possible sources for each input type for :DSP:MLS:SRC2 command

Input	Source
Digital	A, B, None
Low BW (1x) A/D	Anlr A, Anlr B, Function Meter Amplitude, Gen A, Gen B, None

In the following command argument list an entry of "A" (or "B") will be interpreted in two ways, depending on whether the input is in Digital or A/D mode.In Digital mode "A" will be interpreted as subframe A; while in A/D mode "A" will be interpreted as Analyzer channel A.

The command argument is:

- **source** - { A | AMPLitude | B | GENA | GENB | NONE }

Default: B

Related Commands: :DSP:MLS:INPut, :DSP:MLS:SRC2?

Command Syntax: :DSP:MLS:SRC2 **source**

Example: `:DSP:MLS:SRC2 A`

:DSP:MLS:SRC2?

This query returns the signal source for the currently selected input for channel 2.

Response argument(s):

- **source** - { B | A | AMPLitude | GENA | GENB | NONE }

Related Commands: :DSP:MLS:SRC2

Command Syntax: :DSP:MLS:SRC2?

Response Syntax: :DSP:MLS:SRC2 **source**

Example: `:DSP:MLS:SRC2?`

Response: `:DSP:MLS:SRC2 A`

:DSP:MLS:STARt

This command specifies the value (length) of the Time Start Window.

When a section of the impulse response (direct arrival signal before reflections, for example) is isolated and transformed into the frequency domain, the impulse amplitude at the beginning and ending of that section will generally not be exactly the same and thus will not splice smoothly. The sharp edges introduced into the impulse response by splicing unequal amplitudes will produce ripples in the resulting frequency response plot. Windowing the time domain data by attenuating the amplitude at the beginning and end of the section to be transformed will reduce this rippling, but also reduces the steepness of transitions in the frequency response plots. The Time Start Window selects the window applied to the beginning of the impulse response (time domain) when transforming it to the frequency domain.

The time window is made up of two half-windows. The first half is specified by :DSP:MLS:STARt and is used to process the first portion of data, beginning at the start specified in :DSP:SRCParam. The second half-window is specified by :DSP:MLS:STOP and processes the later portion of data, ending at the stop specified in :DSP:SRCParams.

Separate selection of the Time Start and Stop half-windows permits creation of asymmetrical windows, which provide the optimum match to the asymmetrical shape of the typical impulse response. The available selections are a family of half-cycle raised cosine functions labeled NONE, LT5 (<5%), LT10 (<10%), LT20 (<20%) and LT30 (<30%). The numeric value refers to the amount of the data record (time span multiplied by sample period) taken up by the window's transition from zero to full amplitude. The Time Start Window half-window starts with an amplitude of zero at the Sweep panel Start time and climbs to an amplitude of 1.00 (no attenuation) at or before the selected percentage of the record.

The command argument is:

- **length** - { NONE | LT10 | LT20 | LT30 | LT5 }

Default: NONE

Related Commands: :DSP:MLS:STOP, :DSP:SRCParams,:DSP:MLS:STARt?

Command Syntax: :DSP:MLS:STARt **length**

Example: `:DSP:MLS:START LT5`

:DSP:MLS:STARt?

This query returns an argument indicating the length of the Time Start Window. The valid responses are: NONE, LT5 (<5%), LT10 (<10%), LT20 (<20%), and LT30 (<30%).

Response argument(s):

- **length** - { NONE | LT10 | LT20 | LT30 | LT5 }

Related Commands: :DSP:MLS:STARt

Command Syntax: :DSP:MLS:STARt?

Response Syntax: :DSP:MLS:STARt **length**

Example: `:DSP:MLS:START?`

Response: `:DSP:MLS:START LT5`

:DSP:MLS:STOP

This command specifies the value (length) of the Time Stop Window.

When a section of the impulse response (direct arrival signal before reflections, for example) is isolated and transformed into the frequency domain, the impulse amplitude at the beginning and ending of that section will generally not be exactly the same and thus will not splice smoothly. The sharp edges introduced into the impulse response by splicing unequal amplitudes will produce ripples in the resulting frequency response plot. Windowing the time domain data by attenuating the amplitude at the beginning and end of the section to be transformed will reduce this rippling, but also reduces the steepness of transitions in the frequency response plots. The Time Stop Window selects the window applied to the end of the impulse response (time domain) when transforming it to the frequency domain.

The time window is made up of two half-windows. The first half is specified by :DSP:MLS:STARt and is used to process the first portion of data, beginning at the start specified in :DSP:SRCParam. The second half-window is specified by :DSP:MLS:STOP and processes the later portion of data, ending at the stop specified in :DSP:SRCParams.

Separate selection of the Time Start and Stop half-windows permits creation of asymmetrical windows, which provide the optimum match to the asymmetrical shape of the typical impulse response. The available selections are a family of half-cycle raised cosine functions labeled NONE, LT5 (<5%), LT10 (<10%), LT20 (<20%) and LT30 (<30%). The numeric value refers to the amount of the data record (time span multiplied by sample period) taken up by the window's transition from zero to full amplitude. The Time Start Window half-window starts with an amplitude of zero at the Sweep panel Start time and climbs to an amplitude of 1.00 (no attenuation) at or before the selected percentage of the record.

The command argument is:

- **length** - { NONE | LT10 | LT20 | LT30 | LT5 }

Default: NONE

Related Commands: :DSP:MLS:STOP?

Command Syntax: :DSP:MLS:STOP **length**

Example: `:DSP:MLS:STOP LT5`

:DSP:MLS:STOP?

This query returns an argument indicating the length of the Time Stop Window. The valid responses are: NONE, LT5 (<5%), LT10 (<10%), LT20 (<20%), and LT30 (<30%).

Response argument(s):

- **length** - { NONE | LT10 | LT20 | LT30 | LT5 }

Related Commands: :DSP:MLS:STOP

Command Syntax: :DSP:MLS:STOP?

Response Syntax: :DSP:MLS:STOP **length**

Example: `:DSP:MLS:STOP?`

Response: `:DSP:MLS:STOP LT5`

:DSP:MLS:TIME

This command specifies time domain for the results of the MLS correlation.

The Impulse Response selection (IMPulse) will return the results of the MLS correlation, which is the actual impulse response of the device under test.

The Energy-Time (ET) selection will return what is commonly called an energy-time curve. The energy-time curve computation process involves transforming the impulse response to the frequency domain, doing further processing in the frequency domain, and transforming the result back to the time domain.

The command argument is:

- **length** - { IMPulse | ET }

Default: IMPulse

Related Commands: :DSP:MLS:TIME?

Command Syntax: :DSP:MLS:TIME **length**

Example: `:DSP:MLS:TIME IMPULSE`

:DSP:MLS:TIME?

This query returns the time domain for the results of the MLS correlation.

Response argument(s):

- **length** - { IMPulse | ET }

Related Commands: :DSP:MLS:TIME

Command Syntax: :DSP:MLS:TIME?

Response Syntax: :DSP:MLS:TIME **length**

Example: `:DSP:MLS:TIME?`

Response: `:DSP:MLS:TIME IMPULSE`

:DSP:MLS:TRIGger

This command specifies the DSP MLS program trigger source. Since the System Two is capable of generating one MLS signal from the analog generator and a second signal (including a different MLS signal) from the digital generator, the MLS DSP program must know to which source to cross-correlate.

Acquisition of signal into the MLS acquisition buffer commences when a trigger event occurs (either from the Digital Generator or the Analog Generator).

The command argument is:

- **source** - { AGEN | DGEN }

Default: AGEN

Related Commands: :DSP:MLS:TRIGger?

Command Syntax: :DSP:MLS:TRIGger **source**

Example: `:DSP:MLS:TRIGGER DGEN`

:DSP:MLS:TRIGger?

This query returns the DSP MLS program trigger source.

Response argument(s):

- **source** - { AGEN | DGEN }

Related Commands: :DSP:MLS:TRIGger

Command Syntax: :DSP:MLS:TRIGger?

Response Syntax: :DSP:MLS:TRIGger **source**

Example: `:DSP:MLS:TRIGGER?`

Response: `:DSP:MLS:TRIGGER DGEN`

:DSP:OPSTate

This command specifies the operation mode of the DSP for batch-mode DSP programs. There are two DSP modes available, SETup and READing. This command will cause an execution error if no DSP program is loaded.

The current DSP program must be in SETup for most DSP commands. Only during acquisition, transformation, reprocessing, and reading queries (except peak meters) must the DSP program be in READing mode.

When the DSP is in SETup mode any measurement queries (except for the peak meters) will cause an execution error. Conversely, when the DSP program is in READing mode, any setup commands (and the peak meter queries) will cause an execution error.

A secondary function of the :DSP:OPSTate command when the mode is READing is to notify the instrument what readings to expect while in READing mode. This is done with the optional parameters, which are only accepted when the mode parameter is READing. Note that the readings must be appropriate for the DSP program being used. For example, AMPLitude, JITTer, UPPer, LOWer and PROBability are only valid when running :DSP:PROGram is set to INTervu, and AMP1, AMP2, PHA1 and PHA2 are NOT valid when :DSP:PROGram is set to INTervu.

This command will also cause an execution error if the current :DSP:PROG is either NONE or DANLr. For additional discussion on the user of :DSP:OPSTate see the discussion text at the beginning of the DSP command section.

Note that the INTERVU DSP program will issue an error for eye pattern measurements if LOWER is specified but UPPER is not specified, or vice versa. The measurement will be correct but an error will be generated (for example: 520,32,"DSP (BATCH ERROR): , IF A SWEEP DATA IS SET TO EYE OPENING, OTHER SWEEP DATA MUST BE SET TO AN EYE OPENING OR NONE").

The command arguments are:

- **mode** - { READing | SETup }
- **enable1** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **enable2** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **enable3** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **enable4** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }

Default: SETup

Related Commands: :DSP:OPSTate?

Command Syntax: :DSP:OPSTate **mode[, enable1[, enable2[, enable3[, enable4]]]]**

Example: `:DSP:OPSTATE READING`

:DSP:OPSTate?

This query returns the current DSP OPSTate mode. The optional response parameters indicate what readings are expected. Note that not all DSP programs are able to produce all of the options of the enable parameters.

The response argument is:

- **mode** - { READing | SETup }
- **enable1** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **enable2** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **enable3** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **enable4** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }

Default: SETup

Related Commands:

Command Syntax: :DSP:OPSTate?

Response Syntax: :DSP:OPSTate **mode[, enable1[, enable2[, enable3 [, enable4]]]]**

Example: `:DSP:OPSTATE?`

Response: `:DSP:OPSTATE READING`

:DSP:PROGram

This command specifies the DSP program to be loaded.

There are 6 DSP programs available for System Two at this time: Bittest Digital Data Analyzer (Bittest), Digital Domain Audio Analyzer (DANLr), FFT Spectrum Analyzer (FFT), Digital Interface Analyzer (INTervu), Multitone Audio Analyzer (FASTtest), and Quasi-Anechoic Acoustical Tester (MLS).

The command argument is:

- **program** - { NONE | BITTest | DANLr | FASTtest | FFT | INTervu | MLS }

Default: NONE

Related Commands: :DSP:PROGram

Command Syntax: :DSP:PROGram **program**

Example: `:DSP:PROGRAM FFT`

:DSP:PROGram?

This query returns the name of the currently load DSP program.

Response argument(s):

- **program** - { NONE | BITTEST | DANLR | FASTTEST | FFT | INTERVU | MLS }

Related Commands: DSP:PROGram

Command Syntax: :DSP:PROGram?

Response Syntax: :DSP:PROGram **program**

Example: `:DSP:PROGRAM?`

Response: `:DSP:PROGRAM FFT`

:DSP:REF

These commands set the parameters for the DSP analyzer units which require reference values.

:DSP:REF:DBR1

This command sets the DBR1 unit reference value for the DSP analyzer. The unit is volts. This command is only active for DANLR, FFT and FASTTEST.

The command argument is:

- **setting** - <nrf> (range: ≥ -1E34, ≤ 1E34) { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Default: 0.100 FFS

Related Commands: :DSP:REF:DBR1?

Command Syntax: :DSP:REF:DBR1 **setting**

Example: `:DSP:REF:DBR1 0.80 FFS`

:DSP:REF:DBR1?

This query returns the current DBR1 unit reference setting for the DSP analyzer in the specified units. This command is only active for DANLR, FFT and FASTTEST.

The command arguments are:

- **units** - { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Response argument(s):

- **setting** - <nrf> { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Related Commands: :DSP:REF:DBR1

Command Syntax: :DSP:REF:DBR1? **units**

Response Syntax: :DSP:REF:DBR1 **setting**

Example: `:DSP:REF:DBR1? FFS`

Response: `:DSP:REF:DBR1 0.8FFS`

:DSP:REF:DBR2

This command sets the DBR2 unit reference value for the DSP analyzer. The unit is volts. This command is only active for DANLR, FFT and FASTTEST.

The command argument is:

- **setting** - <nrf> (range: ≥ -1E34, ≤ 1E34) { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Default: 0.100 FFS

Related Commands: :DSP:REF:DBR2?

Command Syntax: :DSP:REF:DBR2 **setting**

Example: `:DSP:REF:DBR2 0.80 FFS`

:DSP:REF:DBR2?

This query returns the current DBR2 unit reference setting for the DSP analyzer in the specified units. This command is only active for DANLR, FFT and FASTTEST.

The command arguments are:

- **units** - { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Response argument(s):

- **setting** - <nrf> { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Related Commands: :DSP:REF:DBR2

Command Syntax: :DSP:REF:DBR2? **units**

Response Syntax: :DSP:REF:DBR2 **setting**

Example: `:DSP:REF:DBR2? FFS`

Response: `:DSP:REF:DBR2 0.8FFS`

:DSP:REF:FREQ

This command sets the FREQ unit reference value for the DSP analyzer. The unit is hertz. This command is only active for DANLR, FFT, FASTTEST and INTERVU (i.e. all except MLS).

The command argument is:

- **frequency** - <nrf> (range: > 0, ≤ 1E34)

Default: 1000.0

Related Commands: :DSP:REF:FREQ?

Command Syntax: :DSP:REF:FREQ **frequency**

Example: `:DSP:REF:FREQ 5000`

:DSP:REF:FREQ?

This query returns the FREQ unit reference value for the DSP analyzer. The response unit is hertz.

Response argument(s):

- **frequency** - <nrf>

Related Commands: :DSP:REF:FREQ

Command Syntax: :DSP:REF:FREQ?

Response Syntax: :DSP:REF:FREQ **frequency**

Example: `:DSP:REF:FREQ?`

Response: `:DSP:REF:FREQ 5000`

:DSP:REF:VFS

This command sets the V/FS unit reference value for the DSP analyzer. The unit is volts.

The command argument is:

- **volts** - <nrf> (range: ≥ -1E34, ≤ 1E34)

Default: 1.0

Related Commands: :DSP:REF:VFS?

Command Syntax: :DSP:REF:VFS **volts**

Example: `:DSP:REF:VFS 10.5`

:DSP:REF:VFS?

This query returns the V/FS unit reference value for the DSP analyzer. The unit is volts.

Response argument(s):

- **volts** - <nrf>

Related Commands: :DSP:REF:VFS

Command Syntax: :DSP:REF:VFS?

Response Syntax: :DSP:REF:VFS **volts**

Example: `:DSP:REF:VFS?`

Response: `:DSP:REF:VFS 10.5`

:DSP:REPRocess?

This query causes the DSP to do post-processing of FFT results using the current DSP settings and to return a timeout status. This query performs the last step of the three phases performed when a DSP:ACQX? query is done, first data is acquired, second a FFT transform is performed and third a post-processed version is generated of the FFT results. The meter and channel selection is determined by the DSP program currently in memory and it's parameter settings. This command is valid only in OPSTate READing mode.

An execution error will be generated if there is no batch mode DSP program loaded or the batch mode DSP program is in OPSTate SETup mode.

A query response will be generated when either the acquisition completes or the acquisition times out (see :DSP:TIMeout).

The response argument (status) indicates acquisition timed out status (whether the query completed in the specified time). If the response is 0 (zero), the command did NOT timeout. A 1 (one) response indicates the DSP acquisition timed out.

NOTE: Various run-time errors can be generated by the DSP during the execution of batch-mode commands. It is recommended to check the error queue (e.g., via :ERRN? command) after completion of the :DSP:ACQX?, :DSP:REPR?, and :DSP:XFRM? commands.

The response argument is:

- **status** - <nr1> (range: ≥ 0, ≤ 1)

Related Commands: :DSP:ACQX?, :DSP:XFRM?, :DSP:OPSTate

Command Syntax: :DSP:REPRocess?

Response Syntax: :DSP:REPRocess **status**

Example: `:DSP:REPROCESS?`

Response: `:DSP:REPROCESS 0`

:DSP:SET?

This query returns all instrument DSP settings (this level and lower).

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands:

Command Syntax: :DSP:SET?

Response Syntax: :DSP:SET **settings**

Example: `:DSP:SET?`

Response 1: `:DSP:TIMEOUT 10.000000;PROGRAM NONE`

Response 2:

```
:DSP:REF:DBR1 0.1FFS;DBR2 0.1FFS;FREQ 1000;V
FS 1;:DSP:TIMEOUT 10.000000;PROGRAM DANLR;:D
SP:DANLR:AUTORANGE A,ON;AUTORANGE B,ON;CHANN
EL A;COUPLING A,AC;COUPLING B,AC;DETECTOR FR
MS;HPFILTER F10;INPUT DIGITAL;LPFILTER FS_2;
MODE AMPLITUDE;WTG UNWT;RDGRATE R8;TUNINGSRC
 FIXED;FILTERFREQ 1000HZ;FAUTORANGE ON
```

:DSP:SRCParams

This command sets up the currently loaded System Two DSP program to acquire and process a signal. The first parameter specifies the DSP program source parameter. Not all sources are valid for each DSP program. The second parameter specifies that source parameter units.

Table 8-18 shows the valid source parameters available for each DSP program and the valid units for each source parameter.

Table 8-18. Valid source parameters for each DSP program

DSP Program	Source Parameter	Valid Units
FASTTEST	FREQ	CENT, DECS, DHZ, DPCT, DPPM, F_R, HZ, OCTS, PCTHZ
	TIME	SEC
FFT	FREQ	CENT, DECS, DHZ, DPCT, DPPM, F_R, HZ, OCTS, PCTHZ
	TIME	SEC
INTERVU	AMPL	DBV, V
	FREQ	CENT, DECS, DHZ, DPCT, DPPM, F_R, HZ, OCTS, PCTHZ
	JITTER	SEC
	TIME	SEC
MLS	FREQ	CENT, DECS, DHZ, DPCT, DPPM, F_R, HZ, OCTS, PCTHZ
	TIME	SEC, FEET, METERS

The start parameter specifies the first source point to be measured during a sweep of multiple source points.

The stop parameter specifies the last source point to be measured during a sweep of multiple source points.

The steps parameter specifies the number of intervals between source points during a sweep of multiple source points (number of measurement points minus 1).

The mode parameter specifies the type of intervals between measurement points during a sweep of multiple source points. The ARB parameter turns off the spectrum bin peak-pick mode when used with the FREQ source parameter during spectrum measurements (FFT source parameter FREQ, INTERVU source parameter FREQ, and FASTTEST source parameter FREQ with processing mode SPECTRUM). For MLS, spectrum bin peak-pick is always on and this mode parameter is ignored.

This command will cause an execution error if there is no DSP program loaded or if the DSP program is in READing mode.

The command argument(s) are:

- **parameter** - { AMPLitude | FREQ | JITTer | TIME }
- **units** - { table: see table above }
- **start** - <nrf> (range: ≥ 0, ≤ 47% of sample rate)
- **stop** - <nrf> (range: ≥ 0, ≤ 47% of sample rate)
- **steps** - <nr1> (range: ≥ 1, ≤ 16383 for batch mode DSP sources)
- **mode** - { LINear | LOG | ARBitrary }

Default: FREQ, HZ, 20, 20000, 30, LOG

Related Commands: :DSP:OPSTate, <all other batch mode DSP measurements>, :DSP:SRCParams?, :DSP:BATch?

Command Syntax: :DSP:SRCParams **parameter, units, start, stop, steps, mode**

Example: `:DSP:SRCPARAMS AMPLITUDE, V, 0.2, 1.5, 30, LINEAR`

:DSP:SRCParams?

This query returns the source instrument parameters for the current DSP program source and parameter.

The response argument(s) are:

- **parameter** - { AMPLitude | FREQ | JITTer | TIME }
- **units** - { table: see Appendix C; Table 7 }
- **start** - <nrf> (range: ≥ 0, ≤ 47% of sample rate)
- **stop** - <nrf> (range: ≥ 0, ≤ 47% of sample rate)
- **steps** - <nr1> (range: ≥ 1, ≤ 16383 for batch mode DSP sources)
- **mode** - { LINear | LOG | ARBitrary }

Default: NONE, NONE, 20, 20000, 30, LOG (if no DSP program loaded)

Related Commands: :DSP:SRCParams

Command Syntax: :DSP:SRCParams?

Response Syntax: :DSP:SRCParams **parameter, units, start, stop, steps, mode**

Example: `:DSP:SRCPARAMS?`

Response: `:DSP:SRCPARAMS AMPLITUDE,V,0.2,1.5,30,LINEAR`

:DSP:TABLe

This command loads a sweep source table into one of two table registers for eventual use by the :DSP:BATCh? query. Use the :DSP:TSELect command to specify which table is to be used. The :DSP:BATCH? query will use the table data for sweep source values if the preceding :DSP:OPSTate READ command uses the ARBITRARY parameter. The table will not be used if the :DSP:OPSTate READ parameter is set to LOG or LIN.

The command argument(s) are:

- **table** - <nr1> (range: 1 or 2)
- **format** – { ASCii | BINary | RBINary }
- **numpoints** – <nr1> (range: ≥ 1, ≤ 1000)
- **arbblockdata** <definite length arb block data> or <indefinite length arb block data>

The **table** argument specifies either table register 1 or table register 2. The previous contents of the table register will be erase and replace with the new table data.

The **format** argument specifies the formatting of the data within the **arbblockdata** argument, either ASCII, BINARY, or RBINARY.

The numpoints argument specifies the number of table elements contained within the arbblockdata argument. This value is required and must be correct, otherwise an error will be generated and the table data will be discarded.

The **arbblockdata** argument contains the table data. The data consists of either a definite length Arbitrary Block Response Data block (if the requested format is BINary or RBINary), or a sequence of comma-separated floating point numbers in ASCII representation. BINary and RBINary formats specify each data value as a four-byte, single-precision, floating point number. BINary is specified by the IEEE standard 488.2 with the most significant byte sent first, commonly called "big-endian". RBINary is a byte-reversed version of BINary with the most significant byte sent last, compatible with the standard Intel x86 floating point representation, commonly called "little-endian".

Refer to Appendix F for more information about the bit coding of a BINARY format single floating point number.

Note that the **arbblockdata** argument may be either <definite length arb block data> or <indefinite length arb block data>. If it is formatted as <indefinite length arb block data>, then it must be terminated with a GPIB END message (NL EOI).

Default: none

Related Commands: :DSP:SRCParams, :DSP:BATCh?, :DSP:TSELect

Command Syntax: :DSP:TABLe **table**, **format**, **numpoints**, **arbblockdata**

Example:

```
 :DSP:TABLE 2,ASCII,31,#017.57,23.43,29.29,4
1.01,52.73,64.45,82.03,99.6,123.04,158.2,199
.21,251.95,316.4,398.43,498.04,632.81,802.73
,1001.95,1248.04,1599.6,1998.04,2501.95,3152
.34,4001.95,4998.04,6351.56,7998.04,10001.95
,12498.04,16001.95,19998.04
```

:DSP:TABLe?

This query returns a sweep source table from one of two sweep table registers. This must be the last query in a terminated program message because the response will be terminated with a GPIB END message (NL EOI). The response arguments are the command arguments described for the :DSP:TABLE command.

The query argument(s) are:

- **table** - <nr1> (range: 1 or 2)
- **format** – { ASCii | BINary | RBINary }

The **table** argument specifies either table register 1 or table register 2.

The **format** argument specifies the formatting of the data within the **arbblockdata** response argument, either ASCii, BINary, or RBINary.

The response argument(s) are:

- **table** - <nr1> (range: 1 or 2)
- **format** – { ASCii | BINary | RBINary }
- **numpoints** – <nr1> (range: ≥ 1, ≤ 1000)
- **arbblockdata** <indefinite length arb block data>

Default: none

Related Commands: :DSP:SRCParams, :DSP:BATCh

Command Syntax: :DSP:TABLe **table**, **format**

Response Syntax: :DSP:TABLe **table**, **format**, **numpoints**, **arbblockdata**

Example: `:DSP:TABLE? 2,ASCII`

Response:

```
 :DSP:TABLE 2,ASCII,31,#017.57,23.43,29.29,4
1.01,52.73,64.45,82.03,99.6,123.04,158.2,199
.21,251.95,316.4,398.43,498.04,632.81,802.73
,1001.95,1248.04,1599.6,1998.04,2501.95,3152
.34,4001.95,4998.04,6351.56,7998.04,10001.95
,12498.04,16001.95,19998.04
```

:DSP:TIMeout

This command sets up the DSP acquisition timeout parameter (in seconds). If the acquisition has not completed in the specified time (i.e. acquisition timed out), the :DSP:ACQX? query will return a 1 (one). Otherwise, if the acquisition does not timeout the :DSP:ACQX? query will return a 0 (zero).

The command argument(s) are:

- **timeout** - <nrf> (range: ≥ 0.000, ≤ 2147483.000) in units of seconds (approximately 24.855 days)

Default: 10.0 Secs

Related Commands: :DSP:ACQX?, :DSP:TIMeout?

Command Syntax: :DSP:TIMeout **timeout**

Example: `:DSP:TIMEOUT 0.03`

:DSP:TIMeout?

This query returns the current DSP acquisition timeout value.

The response argument(s) are:

- **timeout** - <nrf>

Default: none

Related Commands: :DSP:TIMeout

Command Syntax: :DSP:TIMeout?

Response Syntax: :DSP:TIMeout **timeout**

Example: `:DSP:TIMEOUT?`

Response: `:DSP:TIMEOUT 0.03`

:DSP:TSELect

This command selects one of the two sweep source tables for use by the :DSP:BATCH? query if the last :DSP:OPSTate READ parameter is set to ARBITRARY. If LIN or LOG is specified instead of ARBITRARY, the sweep table will not be used.

Note that the DSP peak-picking algorithm will be disabled for spectrum frequency sweeps if a sweep table is used.

A sweep table may be selected before it is defined with the :DSP:TABLe command. An error will be reported when the :DSP:BATCH? query is received if the selected sweep table is not already defined.

The command argument is:

- **table** - <nr1> (range: 1 or 2)

Default: none

Related Commands: :DSP:TABLe, :DSP:BATCh?

Command Syntax: :DSP:TSELect **table**

Example: `:DSP:TSELECT 2`

:DSP:XFRM?

This query causes the DSP to do a transform of data without acquire. A FFT transform is performed on data in the acquisition buffer, data that have been acquired using an earlier DSP:ACQX? query. The transform will be performed using the current DSP settings and return a timeout status. The meter and channel selection is determined by the DSP program currently in memory and its parameter settings. This command is valid only in OPSTate READing mode.

An execution error will be generated if there is no batch mode DSP program loaded or the batch mode DSP program is in OPSTate SETup mode.

A query response will be generated when either the acquisition completes or the acquisition times out (see :DSP:TIMeout).

The response argument (status) indicates acquisition timed out status (whether the query completed in the specified time). If the response is 0 (zero), the command did NOT timeout. A 1 (one) response indicates the DSP acquisition timed out.

NOTE: Various run-time errors can be generated by the DSP during the execution of batch-mode commands. It is recommended to check the error queue (e.g., via :ERRN? command) after completion of the :DSP:ACQX?, :DSP:REPR?, and :DSP:XFRM? commands.

The response argument is:

- **status** - <nr1> (range: ≥ 0, ≤ 1)

Default: <None>

Related Commands: :DSP:ACQX?, :DSP:SRCParams, <list all commands in table that affect XFRM?>

Command Syntax: :DSP:XFRM?

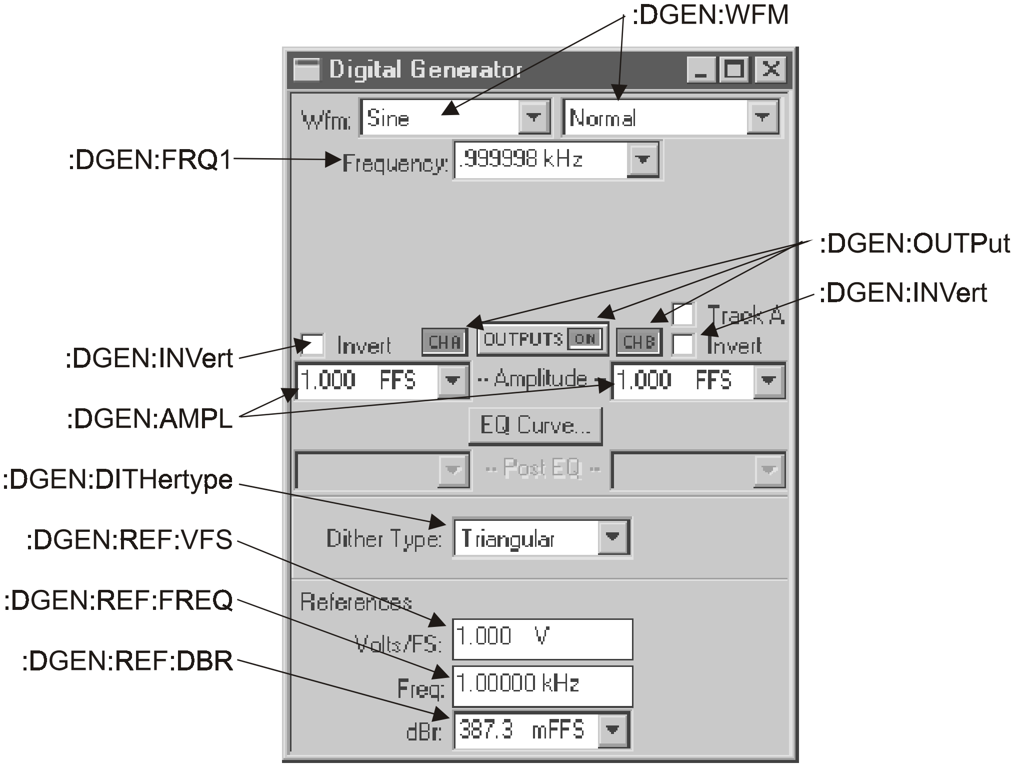
Response Syntax: :DSP:XFRM? **status**

Example: `:DSP:XFRM?`

Response: `:DSP:XFRM`

9. Digital Generator Commands

These commands set parameters for the digital generator. The header-path for the digital generator is :DGEN:. These commands are invalid and will cause execution errors if used with a System Two model SYS-2022 or SYS-2222 (no digital generator hardware).



Wfm: Sine Normal

Frequency: .999998 kHz

□ Track A

☑ Invert □ Invert

-- Amplitude -- 1.000 FFS 1.000 FFS

Dither Type: Triangular

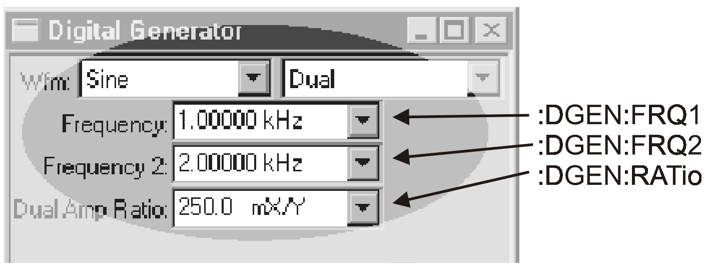
References

Volts/FS: 1.000 V

Freq: 1.00000 kHz

dBr: 387.3 mFFS

Figure 9-1. Digital Generator control panel showing relationship with GPIB commands. Also see following figures.



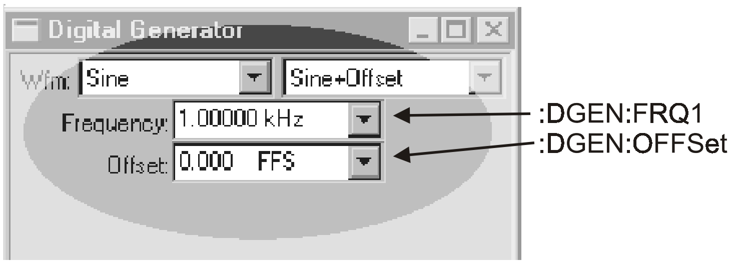
Wfm: Sine Dual

Frequency: 1.00000 kHz

Frequency 2: 2.00000 kHz

Dual Amp Ratio: 250.0 mX/Y

Figure 9-2. Part of Digital Generator panel, Sine Dual mode

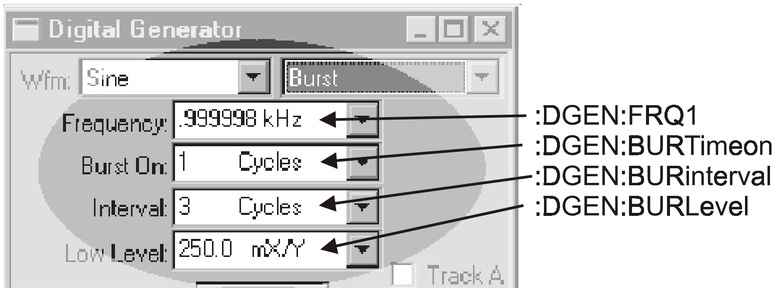


Wfm: Sine Sine+Offset

Frequency: 1.00000 kHz

Offset: 0.000 FFS

Figure 9-3. Part of Digital Generator panel, Sine Offset mode



Wfm: Sine Burst

Frequency: .999998 kHz

Burst On: 1 Cycles

Interval: 3 Cycles

Low Level: 250.0 mX/Y

□ Track A

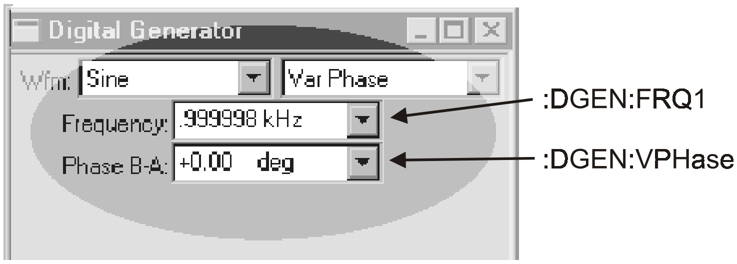
:DGEN:FRQ1

:DGEN:BURTimeon

:DGEN:BURinterval

:DGEN:BURLevel

Figure 9-4. Part of Digital Generator panel, Sine Burst mode



Wfm: Sine Var Phase

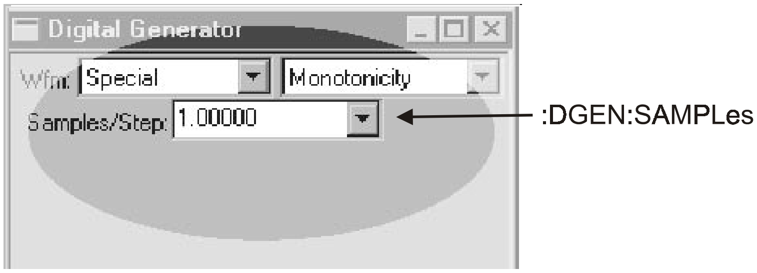
Frequency: .999998 kHz

Phase B-A: +0.00 deg

:DGEN:FRQ1

:DGEN:VPHase

Figure 9-5. Part of Digital Generator panel, Sine Variable Phase mode

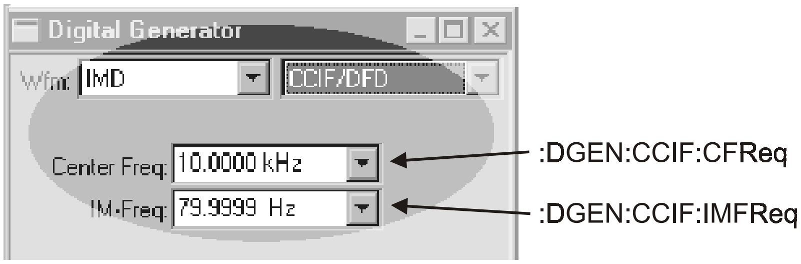


Wfm: Special Monotonicity

Samples/Step: 1.00000

:DGEN:SAMPLes

Figure 9-6. Part of Digital Generator panel, Special Monotonicity mode



Wfm: IMD CCIF/DFD

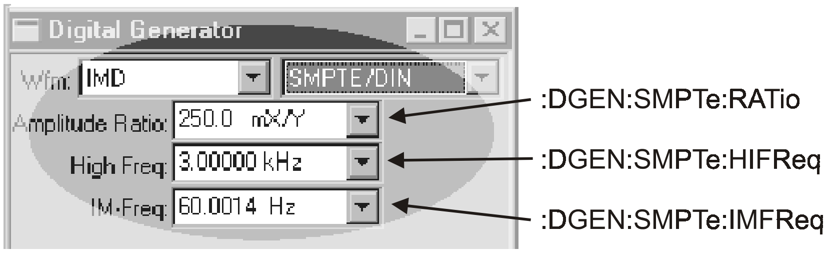
Center Freq: 10.0000 kHz

IM-Freq: 79.9999 Hz

:DGEN:CCIF:CFReq

:DGEN:CCIF:IMFReq

Figure 9-7. Part of Digital Generator panel, IMD CCIF/DFD mode



Wfm: IMD SMPTE/DIN

Amplitude Ratio: 250.0 mX/Y

High Freq: 3.00000 kHz

IM-Freq: 60.0014 Hz

:DGEN:SMPTe:RATio

:DGEN:SMPTe:HIFReq

:DGEN:SMPTe:IMFReq

Figure 9-8. Part of Digital Generator panel, IMD SMPTE/DIN mode

:DGEN:AMPL

Sets the amplitude of the specified digital generator channel.

The command arguments are:

- **channel** - { A | AB | B }
- **amplitude** - <nrf> (range: ≥ 0, ≤ 1.0 FFS) { FFS | DBFS | DBR | DBU | DBV | IDBR | PCTFs | V | VP | VPP }

Default: 1.0 FFS

Related Commands: :DGEN:AMPL?

Command Syntax: :DGEN:AMPL **channel, amplitude**

Example: `:DGEN:AMPL A,1.00FFS`

:DGEN:AMPL?

Returns the output amplitude of the specified digital generator channel in the specified units.

The command arguments are:

- **channel** - { A | B }
- **units** - { FFS | DBFS | DBR | DBU | DBV | IDBR | PCTFs | V | VP | VPP }

Response argument(s):

- **response_channel** - { A | B }
- **amplitude** - <nrf> (range: ≥ 0, ≤ 1.0 FFS) { FFS | DBFS | DBR | DBU | DBV | IDBR | PCTFs | V | VP | VPP }

Related Commands: :DGEN:AMPRatio, :DGEN:IMFReq, :DGEN:AMPL

Command Syntax: :DGEN:AMPL? **channel, units**

Response Syntax: :DGEN:AMPL **response_channel, amplitude**

Example: `:DGEN:AMPL? A, VP`

Response: `:DGEN:AMPL A,1.414VP`

:DGEN:ARBCount?

Returns the number of generator arbitrary waveform registers that can be defined in the GPIB memory for the specified size (in sample points).

If a number is passed that is not one of the valid sizes it will be rounded up to the next higher size. For any command parameter value greater than 4096, the response will be based on 8192 points.

The query argument is:

- **points** - <nr1>
  (range: 256 | 512 | 1024 | 2048 | 4096 | 8192)

The response argument is:

- **number** - <nr1>

Related Commands: :DGEN:ARBSize, :DGEN:ARBLoad, :DGEN:ARBWfm

Command Syntax: :DGEN:ARBCount? **points**

Response Syntax: :DGEN:ARBCount **number**

Example: `:DGEN:ARBCOUNT? 1024`

Response: `:DGEN:ARBCOUNT 119`

:DGEN:ARBLoad

Loads arbitrary waveform data into a generator arbitrary waveform register. The waveform must come from a file that was generated using MAKEWAV2.EXE. The first argument specifies the waveform buffer to be loaded. The range of values for the register argument is determined by the currently defined register size (see :DGEN:ARBSize). An error will be generated if the register is not valid or if the waveform is larger than the waveform register size. No size error will be generated if data block size is less than or equal to the waveform register size.

The command arguments are:

- **register** - <nr1> (range: ≥ 1, ≤ n, see :DGEN:ARBSize)
- **data** - <definite length arb block data>

Related Commands: :DGEN:ARBSize, :DGEN:ARBWFM, :DGEN:WFM, :DOUT:SRATe, :DGEN:ARBLoad?

Command Syntax: :DGEN:ARBLoad **register, data**

Example: `:DGEN:ARBLOAD 1,#43328bb...`

:DGEN:ARBLoad?

Returns the contents of the generator arbitrary waveform register in definite length arbitrary block format.

The command argument is:

- **register** - <nr1> (range: ≥ 1, ≤ n; see :DGEN:ARBSize)

The response arguments are:

- **register** - <nr1> (range: ≥ 1, ≤ n; see :DGEN:ARBSize)
- **data** - <definite length arb block data>

Related Commands: :DGEN:ARBLoad

Command Syntax: :DGEN:ARBLoad? **register**

Response Syntax: :DGEN:ARBLoad **register, data**

Example: `:DGEN:ARBLOAD? 1`

Response: `:DGEN:ARBLOAD 1,#43328bb...`

:DGEN:ARBSize

Specifies the current size of the digital generator arbitrary waveform registers. The current register size determines how many registers are available. The following table shows the relationship between the register size and the number of available registers:

Size	Number of registers	Wfm File Size
8192	16	24832
4096	31	12544
2048	62	6400
1024	119	3328
512	221	1794
256	388	1024

Changing the digital generator arbitrary waveform register size will clear the waveform registers, but not DSP memory. If the requested size is the same as the current size, the waveform registers will not be cleared.

An error will be reported if DSP hardware is not installed.

The command argument is:

- **points** - <nr1> (range: 256 | 512 | 1024 | 2048 | 4096 | 8192)

Default: 8192

Related Commands: :DGEN:ARBLoad, :DGEN:ARBWFM, :DGEN:WFM, DGEN:ARBSize?

Command Syntax: :DGEN:ARBSize **points**

Example: `:DGEN:ARBSIZE 1024`

:DGEN:ARBSize?

Returns the size (in points) of the generator arbitrary waveform registers.

The response argument is:

- **points** - <nr1> (range: 256 | 512 | 1024 | 2048 | 4096 | 8192)

Related Commands: :DGEN:ARBSize

Command Syntax: :DGEN:ARBSize?

Response Syntax: :DGEN:ARBSize **points**

Example: `:DGEN:ARBSIZE?`

Response: `:DGEN:ARBSIZE 1024`

:DGEN:ARBWfm

Specifies which arbitrary waveform register to load (transfer) into the DSP-based memory (digital generator waveform buffer). The contents of a waveform register (see :DGEN:ARBLoad) can be changed without affecting the output signal if it has already been loaded into the DSP memory (digital generator waveform buffer).

The digital generator has two waveform buffers (channel 1 and channel 2). The command always requires specification of both channels. The first parameter specifies what waveform register to load into channel 1 and the second parameter specifies what waveform register to load into channel 2 with the exceptions mentioned below, using ’0’ as a parameter.

If only one parameter is zero, both channels will be loaded with the same waveform register. For example, :DGEN:ARBW 0,3 loads both channels with waveform register 3. If both parameters are zero, no action is taken.

System Two has only one stereo pair of arbitrary waveform buffers to serve both digital and analog (D/A) output needs. An arbitrary waveform may be sent via D/A converters through the analog output stages while any other waveform is generated by the digital generator, or an arbitrary waveform may be transmitted in the digital domain while any other analog or DSP-generated waveform is sent from the analog outputs. But, if an arbitrary waveform is desired at both analog and digital outputs simultaneously, they will be from the same waveform register and at the same sample rate.

Up to 388 single-channel (mono) generator arbitrary waveforms may be stored in waveform registers using the :DGEN:WFMLOAD command (depending on the size of each waveform register, see :DGEN:ARBSize).

Errors will be reported for the following conditions:

- If DSP hardware is not installed, a command error is reported.
- If a register has not been loaded (using the ARBLoad command) it is undefined. Specifying an undefined register in the ARBWfm command causes an execution error.

The normal sequence of commands is to size the registers (:DGEN:ARBSize, if register size different than the default number of points is required), load the registers (:DGEN:ARBLoad), select which registers will be loaded into the DSP generator memory (:DGEN:ARBWfm), and output the digital generator arbitrary signal (:DGEN:WFM ARBITRARY or :AGEN:WFM DAARbitrary).

The command arguments for the two channels are:

- **wfmregch1** - <nr1> (range: ≥ 0, ≤ n; see DGEN:ARBSize)
- **wfmregch2** - <nr1> (range: ≥ 0, ≤ n; see DGEN:ARBSize)

Related Commands: :DGEN:ARBLoad, :DGEN:ARBSize, :DGEN:WFM

Command Syntax: :DGEN:ARBWfm **wfmregch1, wfmregch2**

Example: `:DGEN:ARBWFM 3,2`

:DGEN:ARBWfm?

Returns the generator arbitrary waveform registers last loaded into the DSP memory for each output channel. If the DSP memory for a channel has not been loaded the response will be 0 (zero). If after a waveform has been transferred, the waveform registers are resized, all registers will be cleared, and the response to the ARBWfm? will be 0. The contents of the DSP waveform buffer will not be changed.

An error will be reported if DSP hardware is not installed.

Response argument(s):

- **wfmregch1**- <nr1> (range: ≥ 0, ≤ n; see DGEN:ARBSize)
- **wfmregch2**- <nr1> (range: ≥ 0, ≤ n; see DGEN:ARBSize)

Related Commands: :DGEN:WFM

Command Syntax: :DGEN:ARBWfm?

Response Syntax: :DGEN:ARBWfm **wfmregch1, wfmregch2**

Example: `:DGEN:ARBWFM?`

Response: `:DGEN:ARBWFM 3,2`

:DGEN:BURinterval

Sets the length (in cycles) of the burst interval for the Sine Burst and Sine Shaped digital generator waveforms. The burst interval is the sum of the burst on time and the burst off time.

Commands affecting the burst interval and burst on time must be issued in a sequence that ensures the burst on time is always less than the burst interval.

The command argument is:

- **cycles** - <nr1> (range: ≥ 2, ≤ 65536)

Default: 3

Related Commands: :DGEN:BURinterval?, :DGEN:BURTimeon,
:DGEN:BURLevel, :DGEN:WFM

Command Syntax: :DGEN:BURinterval **cycles**

Example: `:DGEN:BURINTERVAL 1000`

:DGEN:BURinterval?

Returns the length (in cycles) of the burst interval (see diagram under :DGEN:BURinterval).

Response argument(s):

- **cycles** - <nr1> (range: ≥ 2, ≤ 65536)

Related Commands: :DGEN:BURinterval

Command Syntax: :DGEN:BURinterval?

Response Syntax: :DGEN:BURinterval **cycles**

Example: `:DGEN:BURINTERVAL?`

Response: `:DGEN:BURINTERVAL 1000`

:DGEN:BURLevel

Sets the amplitude of the Sine Burst waveform signal during the "burst off time" (see diagram under :DGEN:BURINTERVAL). This amplitude is expressed as a percentage of the "burst on time" amplitude. Does not apply to Sine Shaped Burst waveform.

The command argument is:

- **level** - <nrf> (range: ≥ 1e-6, ≤ 1) { X_Y | DB | PCT | PPM }

Default: 0.250 X_Y

Related Commands: :DGEN:AMPL, :DGEN:BURinterval, :DGEN:BURTimeon,
:DGEN:BURLevel?, :DGEN:WFM

Command Syntax: :DGEN:BURLevel **level**

Example: `:DGEN:BURLEVEL 12.5PCT`

:DGEN:BURLevel?

Returns the Sine Burst waveform burst off time amplitude (as a percentage of the burst on time amplitude).

The command argument is:

- **units** - { X_Y | DB | PCT | PPM }

Response argument(s):

- **level** - <nrf> { X_Y | DB | PCT | PPM }

Related Commands: :DGEN:BURLevel

Command Syntax: :DGEN:BURLevel? **units**

Response Syntax: :DGEN:BURLevel **level**

Example: `:DGEN:BURLEVEL? PCT`

Response: `:DGEN:BURLEVEL 12.5PCT`

:DGEN:BURTimeon

Sets the length (in cycles) of the Sine Burst and Sine Shaped Burst waveforms burst on time. Commands must be sent in an order that ensures this setting will always be less than the burst interval. If a command is sent that would result in the burst on time being greater than or equal to the burst interval, an execution error will be reported.

The command argument is:

- **cycles** - <nr1> (range: ≥ 1, ≤ 65535)

Default: 1

Related Commands: :DGEN:BURinterval, :DGEN:BURLevel, :DGEN:WFM, :DGEN:BURTimeon?

Command Syntax: :DGEN:BURTimeon **cycles**

Example: `:DGEN:BURTIMEON 50`

:DGEN:BURTimeon?

Returns the length (in cycles) of the burst on time.

Response argument(s):

- **cycles** - <nr1> (range: ≥ 1, ≤ 65535)

Related Commands: :DGEN:BURTimeon

Command Syntax: :DGEN:BURTimeon?

Response Syntax: :DGEN:BURTimeon **cycles**

Example: `:DGEN:BURTIMEON?`

Response: `:DGEN:BURTIMEON 50`

:DGEN:CCIF

These commands set the IMD parameters for the digitally generated CCIF waveform. These parameters are used whenever the CCIF waveform is selected with the :DGEN:WFM IMD, CCIF command.

:DGEN:CCIF:CFReq

Sets the center frequency of the two tone digitally generated CCIF intermodulation test waveform. The two tones will be equally spaced above and below this center frequency (there is no signal at the center frequency).

The maximum CCIF center frequency is dependent on the internal sample rate set by the :DOUT:RATE command. The maximum frequency may be calculated with the formula:

Max CFREQ ≤ (Internal sample rate * 0.47) – (IMFREQ * 0.5)

Thus the maximum frequency at 44100 Hz sample rate with IMFREQ set to 80 Hz would be 20.687 KHz.

The command argument is:

- **frequency** - <nrf> (range: ≥ 3000, ≤ Max CFREQ above) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 10000 HZ

Related Commands: :DGEN:WFM,DGEN:CCIF:IMFReq

Command Syntax: :DGEN:CCIF:CFReq **frequency**

Example: `:DGEN:CCIF:CFREQ 15000HZ`

:DGEN:CCIF:CFReq?

Returns the current setting for the center frequency of the two tone digitally generated CCIF intermodulation test waveform.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :DGEN:CCIF:CFReq

Command Syntax: :DGEN:CCIF:CFReq? units

Response Syntax: :DGEN:CCIF:CFReq frequency

Example: `:DGEN:CCIF:CFREQ? HZ`

Response: `:DGEN:CCIF:CFREQ 1500HZ`

:DGEN:CCIF:IMFReq

Sets the difference frequency of the two tone digitally generated CCIF intermodulation test waveform in hertz units. The maximum difference frequency is dependent on the internal sample rate and the present setting of the CCIF center frequency set by the :DGEN:CCIF:CFREQ command but may not be greater than 2000 Hz. The maximum setting may be calculated with the formula:

Max IMFREQ = ((0.47 * Internal Sample Rate) – CFREQ) * 2

Thus the maximum IMFREQ for an internal sample rate of 44.1 kHz with a CFREQ of 20 kHZ would be 1.454 kHz.

The command argument is:

- **frequency** - <nrf> (range: ≥ 79.9, ≤ 2000)

Default: 80

Related Commands: :DGEN:WFM, DGEN:CCIF:CFReq

Command Syntax: :DGEN:CCIF:IMFReq **frequency**

Example: `:DGEN:CCIF:IMFREQ 500`

:DGEN:CCIF:IMFReq?

Returns the current setting for the difference frequency of the two tone digitally generated CCIF intermodulation test waveform in hertz units.

Response argument(s):

- **frequency** - <nrf> (range: ≥ 80, ≤ 2000)

Related Commands: :DGEN:CCIF:IMFReq

Command Syntax: :DGEN:CCIF:IMFReq?

Response Syntax: :DGEN:CCIF:IMFReq **frequency**

Example: `:DGEN:CCIF:IMFREQ?`

Response: `:DGEN:CCIF:IMFREQ 500`

:DGEN:DITHertype

Disables digital generator dither (NONE) or selects from one of three choices of probability distribution function and frequency spectrum. Dither is random noise of one-half LSB (rectangular) or one LSB (triangular) in amplitude, added to the digital output to improve linearity, reduce distortion, and extend the dynamic range below the theoretical undithered value. The digital resolution at which dither is added is determined by the :DOUT:RESolution value.

Triangular (TRI) probability function dither has no noise modulation effect but produces a slightly worse output signal to noise ratio since its maximum amplitude is one LSB. This is normally the preferred choice.

Rectangular (RECT) probability function dither provides the best signal to noise due to its one-half LSB amplitude, but suffers from modulation noise effects.

Shaped (SHAPed) dither is triangular probability distribution noise with a rising 6 dB/octave slope with zero dB effect at 1/2 the sample rate, thus placing most of the dither power at higher frequencies where some falls out of band of most devices and where the human hearing system is less sensitive.

The command argument is:

- **type** - { TRI | NONE | RECT | SHAPed }

Default: TRI

Related Commands: :DGEN:DITHertype?

Command Syntax: :DGEN:DITHertype **type**

Example: `:DGEN:DITHERTYPE RECT`

:DGEN:DITHertype?

Returns the currently selected type of dither (or NONE if disabled).

Response argument(s):

- **type** - { TRI | NONE | RECT | SHAPed }

Related Commands: :DGEN:DITHertype

Command Syntax: :DGEN:DITHertype?

Response Syntax: :DGEN:DITHertype **type**

Example: `:DGEN:DITHERTYPE?`

Response: `:DGEN:DITHERTYPE RECT`

:DGEN:FRQ1

Sets a frequency for the digitally generated sinewaves available with the :DGEN:WFM functions. The frequency of this sinewave is defined for each waveform function in the table below. The maximum frequency is dependent on the internal sample rate set by the :DOUT:RATE command.

SINE, BURSt SINE, DCSine SPECial POLarity	Single sinewave frequency for both channels A & B
SQUare	Squarewave frequency subject to limited high frequency resolution.
SINE, VPHase	Single sinewave frequency for both channels A & B with variable phase for channel B.
SINE, STEReo	Channel A sinewave frequency
SINE, DUAL	Two tone sinewave signal, controls the frequency of the sinewave which may be set at reduced amplitudes relative to Frequency 2.

The command argument is:

- **frequency** - <nrf>
  (range: ≥ 10, ≤47% of :DOUT:RATE in Hz)
  { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }
  e.g., 25380 Hz at 54000 Hz sample rate

Default: 997.001 HZ

Related Commands: :DGEN:WFM, :DGEN:FRQ2

Command Syntax: :DGEN:FRQ1 **frequency**

Example: `:DGEN:FRQ1 15500HZ`

:DGEN:FRQ1?

Returns the current setting for the Frequency 1 control for digitally generated sinewaves available with the :DGEN:WFM function.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :DGEN:FRQ1

Command Syntax: :DGEN:FRQ1? **units**

Response Syntax: :DGEN:FRQ1 **frequency**

Example: `:DGEN:FRQ1? HZ`

Response: `:DGEN:FRQ1 1550HZ`

:DGEN:FRQ2

Sets one of the two sinewave frequencies available for the STEReo and DUAL digitally generated waveforms available with the :DGEN:WFM functions. The signal definition for the sinewave at this frequency is defined for each function in the table below.

SINE, STEReo	Channel B sinewave frequency.
SINE, DUAL	Controls the frequency of the sinewave whose amplitude is determined by the :DGEN:RATio command.

The command argument is:

- **frequency** - <nrf>
  (range: ≥ 10, ≤47% of :DOUT:RATE in Hz)
  { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }
  e.g., 25380 Hz at 54000 Hz sample rate

Default: 999.999 HZ

Related Commands: :DGEN:WFM, :DGEN:FRQ1

Command Syntax: :DGEN:FRQ2 **frequency**

Example: `:DGEN:FRQ2 17500HZ`

:DGEN:FRQ2?

Returns the current setting for the Frequency 2 control for the STEReo and DUAL digitally generated sinewaves available with the :DGEN:WFM functions.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :DGEN:FRQ2

Command Syntax: :DGEN:FRQ2? **units**

Response Syntax: :DGEN:FRQ2 **frequency**

Example: `:DGEN:FRQ2? HZ`

Response: `:DGEN:FRQ2 17500HZ`

:DGEN:INVert

Enables or disables phase inversion (180 degrees) of the signal on the specified output channel.

The command arguments are:

- **channel** - { AB | A | B }
- **state** - { OFF | ON }

Default: AB,OFF

Related Commands: :DGEN:INVert?

Command Syntax: :DGEN:INVert **channel, state**

Example: `:DGEN:INVERT A,ON`

:DGEN:INVert?

Returns the state of the channel phase inversion setting. The response will be either ON (invert 180 degrees) or OFF (no phase inversion).

The command argument is:

- **channel** - { A | B }

Response argument(s):

- **response_channel** - { A | B }
- **state** - { OFF | ON }

Related Commands: :DGEN:INVert

Command Syntax: :DGEN:INVert? **channel**

Response Syntax: :DGEN:INVert **response_channel, state**

Example: `:DGEN:INVERT? A`

Response: `:DGEN:INVERT A,ON`

:DGEN:OFFSet

Specifies the "DC offset" for the DCSine waveform. The specified value is in sine equivalent RMS when one of the digital units is selected (FFS, PCTFs, or DBFS). Therefore the actual DC offset will be 1.414 times the value entered. This is because the sine amplitude is also expressed in sine equivalent Vrms.

The offset value may be set over the range from -1.0 to +1.0 FFS. The combined peak amplitude of the sinewave plus offset value cannot exceed +1.000 FS or -1.000 FFS and an error will be reported if an attempt is made to change either parameter to a value which would cause the sum to exceed that range.

The command argument is:

- **offset** - <nrf> (range: ≥ -1.0, ≤ 1.0 depends on peak sinewave amplitude of :DGEN:AMPL) { FFS | DBFS | DBR | DBU | DBV | IDBR | PCTFs | V | VP | VPP }

Default: 0.0 FFS

Related Commands: :DGEN:OFFSet?, :DGEN:AMPL

Command Syntax: :DGEN:OFFSet **offset**

Example:
```
:DGEN:OFFSET 0.25FFS
```

:DGEN:OFFSet?

Returns the current offset setting for the DCSine waveform.

The command argument is:

- **units** - { FFS | DBFS | DBR | DBU | DBV | IDBR | PCTFs | V | VP | VPP }

Response argument(s):

- **offset** - <nrf> { FFS | DBFS | DBR | DBU | DBV | IDBR | PCTFs | V | VP | VPP }

Related Commands: :DGEN:OFFSet

Command Syntax: :DGEN:OFFSet? **units**

Response Syntax: :DGEN:OFFSet **offset**

Example:
```
:DGEN:OFFSET? FFS
```

Response:
```
:DGEN:OFFSET .25FFS
```

:DGEN:OUTPut

Sets the digital generator outputs on or off. When OFF, the output is digital zero signal. OFF does not disable output dither, therefore a small output signal will exist if dither is ON but :DGEN:OUTPUT is OFF.

- OFF turns both A and B outputs OFF.
- AB turns both A and B outputs ON.
- A turns A output ON and B output OFF.
- B turns B output ON and A output OFF.

The command argument is:

- **channel** - { OFF | A | AB | B }

Default: OFF

Related Commands: :DGEN:OUTPut?, :DGEN:OUTon

Command Syntax: :DGEN:OUTPut **channel**

Example: `:DGEN:OUTPUT A`

:DGEN:OUTPut?

Returns the output state of the digital generator.

Response argument(s):

- **channel** - { OFF | A | AB | B }

Related Commands: :DGEN:OUTPut

Command Syntax: :DGEN:OUTPut?

Response Syntax: :DGEN:OUTPut **channel**

Example: `:DGEN:OUTPUT?`

Response: `:DGEN:OUTPUT A`

:DGEN:RATio

Sets the ratio of the FRQ2 sinewave amplitude relative to the amplitude of the FRQ1 sinewave for the digitally generated DUAL sinewave waveform available with the :DGEN:WFM SINE, DUAL function. The FRQ2 sinewave amplitude will always be <= the FRQ1 sinewave amplitude by this ratio.

The command argument is:

- **ratio** - <nrf> (range: ≥ 1e-6, ≤ 1) { X_Y | DB | PCT | PPM }

Default: 0.250 X_Y

Related Commands: :DGEN:FRQ1, :DGEN:FRQ2

Command Syntax: :DGEN:RATio **ratio**

Example: `:DGEN:RATIO -50DB`

:DGEN:RATio?

Returns the current setting for the ratio of the Frequency 2 sinewave amplitude relative to the amplitude of the Frequency 1 sinewave for the DUAL digitally generated sinewave waveform available with the :DGEN:WFM SINE, DUAL function.

The command argument is:

- **units** - { X_Y | DB | PCT | PPM }

Response argument(s):

- **ratio** - <nrf> { X_Y | DB | PCT | PPM }

Related Commands: :DGEN:FRQ1, :DGEN:FRQ2

Command Syntax: :DGEN:RATio? **units**

Response Syntax: :DGEN:RATio **ratio**

Example: `:DGEN:RATio? DB`

Response: `:DGEN:RATIO -0.5DB`

:DGEN:REF

These commands set the parameters for the digital generator units that require reference values.

:DGEN:REF:DBR

Sets the digital generator DBR unit reference value. Both analog and digital reference units may be used when specifying this DBR reference.

Analog units (V, Vp, Vpp, dBu, dBV) are provided as an aid for DAC testing. Thus if the digital generator output unit of DBR is used, then the output may be set in DBR relative to the full scale output of a DAC provided that the :DGEN:REF:VFS reference has been set to the full scale output voltage of the DAC. This convenience permits specifying the DAC digital input (connected to the System Two Digital Output) in terms of DAC analog output units. For example, the digital generator VFS reference should be set to 10 V for a DAC with a full scale output of 10 V. The digital generator output may be to set drive the DAC to its 10 V full scale output by setting the digital generator DBR reference value to 10 V and specifying a digital generator output of 0.0 DBR.

The command argument is:

- **setting** - <nrf> (range: ≥ -1E34, ≤ 1E34)
  { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Default: 0.3873 FFS

Related Commands: :DGEN:REF:DBR?

Command Syntax: :DGEN:REF:DBR **setting**

Example: `:DGEN:REF:DBR 0.80 FFS`

:DGEN:REF:DBR?

Returns the current DBR unit reference setting for the digital generator in the specified units.

NOTE: The DBR digital reference settings (units of FFS, %FS, dBFS) are set relative to the current VFS reference setting (Volt/FS). A DBR reference query with analog units (units of V, Vp, Vpp, dBu, dBV) will return a value that is proportional to the DBR digital reference settings times the VFS reference setting. Using basic units (V and FFS) the relationship in equation form would be:

$$\mathrm{REF}_{\mathrm{DBR}}\ (\mathrm{V}) = \mathrm{REF}_{\mathrm{VFS}}\ (\mathrm{V}) * \mathrm{REF}_{\mathrm{DBR}}\ (\mathrm{FFS})$$

For example, if the following reference settings are received by the instrument: ":DGEN:REF:DBR 1FFS;VFS 10" the response to ":DGEN:REF:DBR? V" would be ":DGEN:REF:DBR 10V". If the VFS reference is changed to 5 V (:DGEN:REF:VFS 5), the response to ":DGEN:REF:DBR? V" would be ":DGEN:REF:DBR 5V".

The command arguments are:

- **units** - { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Response argument(s):

- **setting** - <nrf> { FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Related Commands: :DGEN:REF:DBR

Command Syntax: :DGEN:REF:DBR? **units**

Response Syntax: :DGEN:REF:DBR **setting**

Example: `:DGEN:REF:DBR? FFS`

Response: `:DGEN:REF:DBR 0.8FFS`

:DGEN:REF:FREQ

Sets the FREQ unit reference value for the digital generator. The unit is hertz.

The command argument is:

- **frequency** - <nrf> (range: > 0, ≤ 1E34)

Default: 997.00

Related Commands: :DGEN:REF:FREQ?

Command Syntax: :DGEN:REF:FREQ **frequency**

Example: `:DGEN:REF:FREQ 5000`

:DGEN:REF:FREQ?

Returns the FREQ unit reference value for the digital generator. The response unit is hertz.

Response argument(s):

- **frequency** - <nrf>

Related Commands: :DGEN:REF:FREQ

Command Syntax: :DGEN:REF:FREQ?

Response Syntax: :DGEN:REF:FREQ **frequency**

Example: `:DGEN:REF:FREQ?`

Response: `:DGEN:REF:FREQ 5000`

:DGEN:REF:VFS

Sets the V/FS unit reference value for the digital generator. The unit is volts.

The command argument is:

- **volts** - <nrf> (range: ≥ -1E34, ≤ 1E34)

Default: 1.0

Related Commands: :DGEN:REF:VFS?

Command Syntax: :DGEN:REF:VFS **volts**

Example: `:DGEN:REF:VFS 10.5`

:DGEN:REF:VFS?

Returns the V/FS unit reference value for the digital generator. The unit is volts.

Response argument(s):

- **volts** - <nrf>

Related Commands: :DGEN:REF:VFS

Command Syntax: :DGEN:REF:VFS?

Response Syntax: :DGEN:REF:VFS **volts**

Example: `:DGEN:REF:VFS?`

Response: `:DGEN:REF:VFS? 10.5`

:DGEN:SAMPles

Sets the samples per step parameter for the monotonicity, walking ones, and walking zeros special waveforms.

For monotonicity, this sets the number of samples duration of each half-cycle of the squarewave. The monotonicity squarewave runs for five complete cycles at each of the ten lowest states of the digital output signal (at the output resolution [word width] setting of the :DOUT:RESolution command).

For walking ones and walking zeros, this sets the number of digital samples at each output state. Thus, increasing the value in this field will stretch the cycle out in time.

The command argument is:

- **samplesperstep** - <nr1> (range: ≥ 1, ≤ 65535)

Default: 1

Related Commands: :DGEN:SAMPles?

Command Syntax: :DGEN:SAMPles **samplesperstep**

Example: `:DGEN:SAMPLES 100`

:DGEN:SAMPles?

Returns the current samples per step setting for the monotonicity, walking ones, and walking zeros special waveforms.

Response argument(s):

- **samplesperstep** - <nr1>

Related Commands: :DGEN:SAMPles

Command Syntax: :DGEN:SAMPles?

Response Syntax: :DGEN:SAMPles **samplesperstep**

Example:

```
:DGEN:SAMPLES?
```

Response:

```
:DGEN:SAMPLES 100
```

:DGEN:SET?

Returns a string that contains all of the Digital Generator settings.

Response argument(s):

- **settings** -<response message unit> [<response message unit>] ...

Related Commands: All other Digital Generator queries.

Command Syntax: :DGEN:SET?

Response Syntax: :DGEN:SET **settings**

Example:

```
:DGEN:SET?
```

Response:

```
:DGEN:REF:FREQ 997;DBR 0.3873FFS;VFS 1;:DGEN
:OUTPUT AB;INVERT A,OFF;INVERT B,OFF;AMPL A,
1FFS;AMPL B,1FFS;BURTIMEON 1;BURLEVEL 0.25X_
Y;BURINTERVAL 3;RATIO 0.25X_Y;DITHER TRI;ARB
SIZE 8192;:DGEN:SMPTE:RATIO 0.25X_Y;HIFREQ 3
000HZ;IMFREQ 79.9999HZ;:DGEN:CCIF:CFR 10000H
Z;IMFREQ 80HZ;:DGEN:FRQ1 997.001HZ;FRQ2 999.
999HZ;VPHASE 0;OFFSET 0FFS;WFM SINE,SINE;SAM
PLES 1
```

:DGEN:SMPTe

These commands set the parameters for the digitally generated SMPTE waveform. These parameters are used whenever the IMD waveform is selected with the :DGEN:WFM SMPTe command.

:DGEN:SMPTe:HIFReq

Sets the high frequency (upper frequency) of the two tone digitally generated SMPTE intermodulation test waveform.

The command argument is:

- **frequency** - <nrf>
  (range: ≥ 2000, ≤ 25380 47% of internal sample rate)
  { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 3000.0 HZ

Related Commands: :DGEN:WFM, DGEN:SMPTe:IMFReq, DGEN:SMPTe:RATio

Command Syntax: :DGEN:SMPTe:HIFReq **frequency**

Example: `:DGEN:SMPTE:HIFREQ 7000`

:DGEN:SMPTe:HIFReq?

Returns the current setting for the high frequency (upper) of the two tone digitally generated SMPTE intermodulation test waveform.

The command argument is:

- **units** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **frequency** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Related Commands: :DGEN:SMPTe:HIFReq

Command Syntax: :DGEN:SMPTe:HIFReq? **units**

Response Syntax: :DGEN:SMPTe:HIFReq **frequency**

Example: `:DGEN:SMPTE:HIFREQ? HZ`

Response: `:DGEN:SMPTE:HIFREQ 7000`

:DGEN:SMPTe:IMFReq

Sets the low frequency of the two tone digitally generated SMPTE intermodulation test waveform in hertz units.

The command argument is:

- **frequency** - <nrf> (range: ≥ 40, ≤ 500)

Default: 79.9999

Related Commands: :DGEN:WFM, DGEN:SMPTe:HIFReq, DGEN:RATio

Command Syntax: :DGEN:SMPTe:IMFReq **frequency**

Example: `:DGEN:SMPTE:IMFREQ 250`

:DGEN:SMPTe:IMFReq?

Returns the current setting for the low frequency of the two tone digitally generated SMPTE intermodulation test waveform in hertz units.

Response argument(s):

- **frequency** - <nrf>

Related Commands: :DGEN:SMPTe:IMFReq

Command Syntax: :DGEN:SMPTe:IMFReq?

Response Syntax: :DGEN:SMPTe:IMFReq **frequency**

Example: `:DGEN:SMPTE:IMFREQ?`

Response: `:DGEN:SMPTE:IMFREQ 250`

:DGEN:SMPTe:RATio

Sets the amplitude ratio of the HIFReq sinewave to the IMFReq sinewave for the digitally generated SMPTE intermodulation test waveform. Ranges: -120 dB to 0 dB, 0.0001% to 100%, 1 PPM to 1E+6 PPM, 1E-6 to 1.0 X/Y. The HIFReq sinewave amplitude will always be less than the IMFReq sinewave amplitude.

The command argument is:

- **ratio** - <nrf> (range: ≥1e-6, ≤ 1) { X_Y | DB | PCT | PPM }

Default: 0.250 X_Y

Related Commands: :DGEN:WFM, :DGEN:SMPTe:HIFReq, :DGEN:SMPTE:IMFReq, :DGEN:SMPTe:RATio?

Command Syntax: :DGEN:SMPTe:RATio **ratio**

Example: `:DGEN:SMPTE:RATIO -10 DB`

:DGEN:SMPTe:RATio?

Returns the current setting for the amplitude ratio of the HIFReq sinewave to the IMFReq sinewave for the digitally generated SMPTE intermodulation test waveform.

The command argument is:

- **units** - { X_Y | DB | PCT | PPM }

Response argument(s):

- **ratio** - <nrf> { X_Y | DB | PCT | PPM }

Related Commands: :DGEN:SMPTe:RATio

Command Syntax: :DGEN:SMPTe:RATio? **units**

Response Syntax: :DGEN:SMPTe:RATio **ratio**

Example: `:DGEN:SMPTE:RATIO? DB`

Response: `:DGEN:SMPTE:RATIO -0.1DB`

:DGEN:VPHase

Sets the phase of the channel B sinewave relative to the channel A sinewave for the digitally generated VPHase sinewave waveform provided by the :DGEN:WFM VPHase command. Units are degrees.

The command argument is:

- **phase** - <nrf> (range: ≥-180, ≤ 180)

Default: 0.0

Related Commands: :DGEN:WFM

Command Syntax: :DGEN:VPHase **phase**

Example: `:DGEN:VPHase 70.5`

:DGEN:VPHase?

Returns the phase of the digitally generated variable phase sinewave waveforms provided by the :DGEN:WFM VPHase command. Units are degrees.

Response argument(s):

- **phase** - <nrf>

Related Commands: :DGEN:VPHase

Command Syntax: :DGEN:VPHase?

Response Syntax: :DGEN:VPHase **phase**

Example: `:DGEN:VPHASE?`

Response: `:DGEN:VPHASE 70.5`

:DGEN:WFM

Specifies the waveform to be generated by the Digital Generator. Many commands for the Digital Generator are active only when the appropriate type of waveform is selected. For example the :DGEN:CCIF:CFREQ command can set the CCIF center frequency, but this setting will not be used until the waveform type is set to CCIF.

Some waveform types can be specified with only one argument, but most require both arguments. The first argument specifies the type of waveform. The second argument selects the sub-type of that waveform.

In the case of Square and ARBitrary waveform types, a second argument is not required. However, a second parameter of NONE will be accepted (will not change the behavior). This mechanism is being provided primarily to support macro definitions, where the command arguments might be passed as macro parameters.

Waveform Type (argument 1)	Waveform sub-Type(argument 2)
Arbitrary Wfm (ARBitrary)	no argument required (but will accept NONE)
Intermodulation Distortion (IMD)	CCIF (CCIF), SMPTE (SMPTe), DIM (DIM)
Maximum Length Sequence MLS (MLS)	Pink #1 (P1), Pink #2 (P2), Pink #3 (P3), Pink #4 (P4), White #1 (W1), White #2 (W2), White #3 (W3), White #4 (W4)
Sine (SINE)	Burst Sine (BURSt), Sine with DC offset (DCSine), Dual Sine (DUAL), Normal Sine (SINE), Shaped Burst (SHAPed), Stereo Sine (STEReo), Variable Phase Sine (VPHase),
Noise (NOISe)	Pink (PINK), White (WHITe), Burst USASI (USASi)
Special (SPCial)	Constant Value (CONStant), JTEST Signal (JTESt), Monotonicity (MONOtonic), Polarity (POLarity), Psuedo-Random Noise (RANDom), Walking Zeros (WLK0), Walking Ones (WLK1)
Square (SQUare)	no argument required (but will accept NONE)

The command argument is:

- **type** - { SINE | ARBitrary | IMD | MLS | NOISe | SPCial | SQUare }
- **subtype** - { SINE | BURSt | CCIF | CONStant | DCSine | DIM | DUAL | EQSine | JTESt | MONotonic | NONE | OFFSet | P1 | P2 | P3 | P4 | PINK | POLarity | RANDom | SHAPed | SMPTe | STEReo | USASi | VPHase | W1 | W2 | W3 | W4 | WHITe | WLK0 | WLK1 }

Default: SINE, SINE

Related Commands: :DGEN:WFM?

Command Syntax: :DGEN:WFM **type**[, **subtype**]

Example 1: `:DGEN:WFM SINE,DUAL`

Example 2: `:DGEN:WFM SQUARE`

:DGEN:WFM?

Returns the current digital generator waveform being generated. The response will consist of two arguments depending on the type of waveform. See the :DGEN:WFM command above.

Response argument(s):

- **type** - { SINE | ARBitrary | IMD | MLS | NOISe | SPCial | SQUare }
- **subtype** - { SINE | BURSt | CCIF | CONStant | DCSine | DIM | DUAL | EQSine | JTESt | MONotonic | NONE | OFFSet | P1 | P2 | P3 | P4 | PINK | POLarity | RANDom | SHAPed | SMPTe | STEReo | USASi | VPHase | W1 | W2 | W3 | W4 | WHITe | WLK0 | WLK1 }

Related Commands: :DGEN:WFM

Command Syntax: :DGEN:WFM?

Response Syntax: :DGEN:WFM **type**[, **subtype**]

Example: `:DGEN:WFM?`

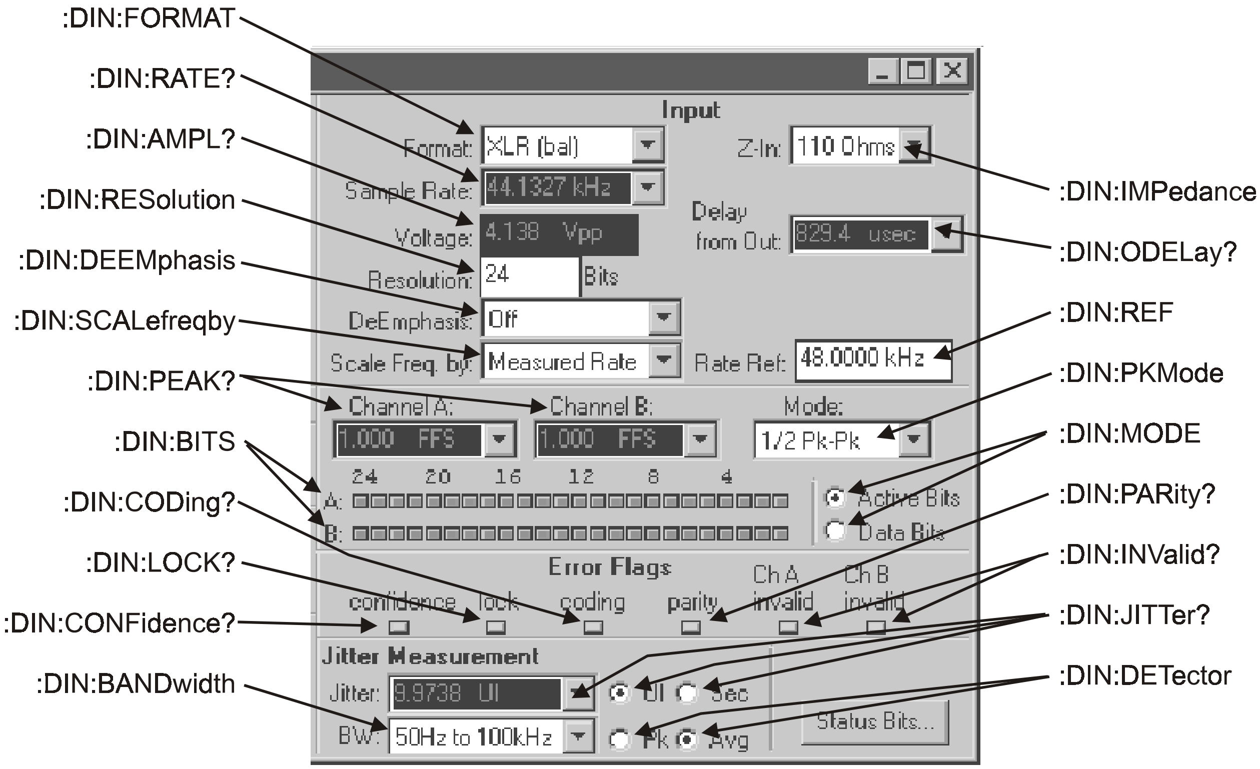
Response1: `:DGEN:WFM SINE,DUAL`

Response2: `:DGEN:WFM SQUARE`

10. Digital Input, Output, and Status Bit Commands

Digital Input Commands

The header-path for Digital Input commands is :DIN.



Input

Format: XLR (bal)

Sample Rate: 44.1327 kHz

Voltage: 4.138 Vpp

Resolution: 24 Bits

DeEmphasis: Off

Scale Freq. by: Measured Rate

Z-In: 110 Ohms

Delay from Out: 829.4 usec

Rate Ref: 48.0000 kHz

Channel A: 1.000 FFS

Channel B: 1.000 FFS

Mode: 1/2 Pk-Pk

● Active Bits ○ Data Bits

Jitter Measurement

Jitter: 8.9738 UI ● UI ○ Sec

BW: 50Hz to 100kHz ○ Pk ● Avg

Figure 10-1. DIO Panel, Input section, showing relationship with GPIB commands

:DIN:AMPL?

Returns the peak-to-peak digital interface signal amplitude in volts (Vpp)at the front panel XLR (XLR and XLRQ) or BNC (BNC and BNCQ) connector specified in the :DIN:FORMat command . This command is not useful when optical (OPTical), the rear panel serial (SERial) or parallel (PARallel) connectors are selected. If XLR Common is selected by the FORMat command, the response shows common mode noise or signal amplitude.

The first parameter in the response is the next available measurement.

The Digital Input Voltmeter settling is enabled or disabled by the algorithm parameter of the :SETTling:DIN command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

Response argument(s):

- **ampl** - <nrf>
- **settle_timeout** - <nr1>

Related Commands: :DIN:SETTling

Command Syntax: :DIN:AMPL?

Response Syntax: :DIN:AMPL? **ampl, settle_timeout**

Example: `:DIN:AMPL?`

Response: `:DIN:AMPL 5.4,0`

:DIN:BANDwidth

Specifies the jitter measurement bandwidth. Jitter is often dominated by low-frequency noise, so the value of the measured jitter is likely to increase as the lower bandwidth limit is reduced.

The argument specifies the lower limit of the jitter bandwidth. The upper limit is always 100 kHz.

The command argument (in hertz) is:

- **range** - <nr1> (range: 50 | 120 | 700 | 1200)

Default: 700

Related Commands: :DIN:BANDwidth?

Command Syntax: :DIN:BANDwidth range

Example: `:DIN:BANDWIDTH 120`

:DIN:BANDwidth?

Returns the current bandwidth selection for the Digital Input jitter measurement.

Response argument(s):

- **range** - <nr1> (range: 50 | 120 | 700 | 1200)

Related Commands: :DIN:BANDwidth

Command Syntax: :DIN:BANDwidth?

Response Syntax: :DIN:BANDwidth **range**

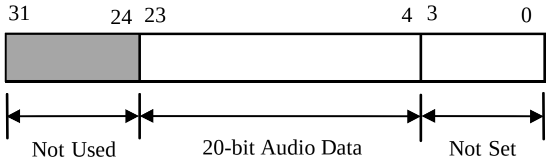
Example: `:DIN:BANDWIDTH?`

Response: `:DIN:BANDWIDTH 120`

:DIN:BITS?

Returns a 32-bit decimal integer (only the low 24 bits are available, the top 8 bits are not used) indicating the state of the digital input signal. In Active Bits mode, each bit position in the response will be set if the corresponding bit has changed logical state since the last ":DIN:MODE" command. In Data Bits mode, each bit position in the response reflects the state of the corresponding bit in the digital input signal.

The left-most available bit (bit 23) corresponds to the Most Significant Bit (MSB) of the digital input signal. The professional and consumer standards allow for up to 24-bit wide signals. When less than 24 bits are transmitted, the standards call for the digital audio data to be MSB-justified. Thus, a 20-bit input signal (see diagram below) will set the 20 left-most bits (23 through 4), but bits 3 through 0 will not be set since they do not contain changing data.



Example format for 20-bit audio data

The query argument is:

- **channel** - { A | B }

Response argument(s):

- **bits** - <nr1>

Related Commands: :DIN:MODE

Command Syntax: :DIN:BITS? channel

Response Syntax: :DIN:BITS **channel, bits**

Example: `:DIN:BITS? A`

Response: `:DIN:BITS 32`

:DIN:CODing?

Returns the state of the coding error flag of the digital input signal. The coding flag indicates a deviation from proper biphase coding in the input serial stream (ignoring preambles).

A response of 1 indicates a coding error, a response of 0 indicates no coding error.

Response argument(s):

- **coding** - <nr1>

Related Commands:

Command Syntax: :DIN:CODing?

Response Syntax: :DIN:CODing **coding**

Example: `:DIN:CODING?`

Response: `:DIN:CODING 1`

:DIN:CONFidence?

Returns the state of the confidence error flag of the digital input signal. The confidence flag is set when the ratio between the amplitude of the three UI long pulse and the following one UI-long pulse in a preamble becomes large enough to cause an increasing probability of errors when slicing the received signal into logic high and low values.

A response of 1 indicates that confidence is set (increased probability of errors), a response of 0 indicates not set (no confidence problem).

Response argument(s):

- **conf** - <nr1>

Related Commands:

Command Syntax: :DIN:CONFidence?

Response Syntax: :DIN:CONFidence **conf**

Example: `:DIN:CONFIDENCE?`

Response: `:DIN:CONFIDENCE 1`

:DIN:DEEMphasis

Specifies any de-emphasis that may be necessary to compensate for pre-emphasis applied to the audio signal to produce an overall flat audio response.

The command argument is:

- **type** - { OFF | CD | CDGain | J17 | J17Gain }

Default: OFF

Related Commands: :DIN:DEEMphasis?

Command Syntax: :DIN:DEEMphasis **type**

Example: `:DIN:DEEMPHASIS CD`

:DIN:DEEMphasis?

Returns the currently specified type of de-emphasis.

Response argument(s):

- **type** - { OFF | CD | CDGain | J17 | J17Gain }

Related Commands: :DIN:DEEMphasis

Command Syntax: :DIN:DEEMphasis?

Response Syntax: :DIN:DEEMphasis **type**

Example: `:DIN:DEEMPHASIS?`

Response: `:DIN:DEEMPHASIS CD`

:DIN:DETector

Specifies the jitter measurement detector response. This command works in conjunction with the bandwidth command (":DIN:BANDwidth") to support optimal jitter measurements.

The command argument is:

- **response** - { AVG | PEAK }

Default: AVG

Related Commands: :DIN:BANDwidth, :DIN:DETector?

Command Syntax: :DIN:DETector **response**

Example: `:DIN:DETECTOR AVG`

:DIN:DETector?

Returns the current jitter measurement detector response. The valid responses are average (AVG) and peak (PEAK).

Response argument(s):

- **response** - { AVG | PEAK }

Command Syntax: :DIN:DETector?

Response Syntax: :DIN:DETector **response**

Example: `:DIN:DETECTOR?`

Response: `:DIN:DETECTOR AVG`

:DIN:ERRFlags?

Returns the state of the error flags for certain properties of the digital input signal. These error flags are confidence, lock, coding, parity, and validity.

The **Confidence** flag is set (1) when the ratio between the amplitude of the three UI long pulse and the following one UI-long pulse in a preamble becomes large enough to cause an increasing probability of errors when slicing the received signal into logic high and low values. A set condition (1) indicates a confidence problem.

The **Lock** flag (1) indicates when the digital input phase-locked loop is unable to lock to the incoming signal. A set condition (1) indicates the digital input is not locked.

The **Coding** flag (1) indicates a deviation from proper biphase coding in the input serial stream (ignoring preambles). A set condition (1) indicates a coding problem.

The **Parity** flag (1) indicates a parity error in a subframe.

The **Invalid** flags are driven directly by the V (Validity) bit defined in the Professional and Consumer standards. One Validity bit is sent in each subframe. A set condition (1) indicates invalid.

The following table shows the bit encoding for the decimal return value.

	Confidence	Lock	Coding	Parity	A Valid	B Valid
Bit	6	5	4	3	2	1
Value	32	16	8	4	2	1

For example, a response of 24 would indicate that Lock flag and the Coding flag were set. This is interpreted to mean that phase-lock failed and improper biphase coding was detected.

A response of 0 indicates no errors of any kind.

Response argument(s):

- **din_err** - <nr1>

Related Commands: :DOUT:INVALID

Command Syntax: :DIN:ERRFlags?

Response Syntax: :DIN:ERRFlags **din_err**

Example: `:DIN:ERRFLAGS?`

Response: `:DIN:ERRFLAGS 20`

:DIN:FORMat

Specifies the source for the digital input signal. The valid sources are:

Source	Meaning
XLR (bal)	Front panel XLR digital input connector, balanced
BNC (unbal)	Front panel BNC digital input connector, unbalanced
Optical	Front panel Toslink optical input connector
Gen Mon	Digital generator XLR or BNC output connector
XLR w/EQ	Front panel XLR with EQ for 100 meter cable roll-off
BNC w/EQ	Front panel BNC with EQ for 100 meter cable roll-off
XLR Common	Center tap of digital input transformer vs. ground
Serial	Rear-panel general-purpose serial input connector
Parallel	Rear-panel parallel input connector

The command argument is:

- **source** - { BNC | BNCQ | GENMon | OPTical | PARallel | SERial | XLR | XLRCommon | XLRQ }

Default: XLR

Related Commands: :DIN:FORMat?

Command Syntax: :DIN:FORMat **source**

Example: `:DIN:FORMAT XLR`

:DIN:FORMat?

Returns the current digital input signal source. The valid responses are: XLR bal (XLR), BNC unbal (BNC), Optical (OPTical), Gen Mon (GENMon), XLR w/EQ (XLRQ), BNC w/EQ (BNCQ), XLR Common (XLRCommon), Serial (SERial), and Parallel (PARallel).

Response argument(s):

- **source** - { BNC | BNCQ | GENMon | OPTical | PARallel | SERial | XLR | XLRCommon | XLRQ }

Related Commands: :DIN:FORMat

Command Syntax: :DIN:FORMat?

Response Syntax: :DIN:FORMat **source**

Example: `:DIN:FORMAT?`

Response: `:DIN:FORMAT XLR`

:DIN:IMPedance

Specifies the input impedance for one of the XLR or BNC settings of the digital input source (see :DIN:FORMat). The three XLR setting (XLR, XLRQ, XLRC) share the same input termination (either 110 ohm or high impedance), and the two BNC settings share the same input termination (either 75 ohm or high impedance).

Connector	Input Termination(s)
XLR	110 ohms, high impedance
BNC	75 ohms, high impedance

The command argument is:

- **input** - { XLR | BNC }
- **term** - { Z110 | Z75 | ZHI }

Default: XLR, Z110; BNC, Z75

Related Commands: :DIN:IMPedance?

Command Syntax: :DIN:IMPedance **input,term**

Example: `:DIN:IMPEDANCE XLR, ZHI`

:DIN:IMPedance?

Returns the input impedance for the specified AES/EBU digital input connector (see :DIN:FORMat). The valid responses are high impedance (ZHI) and 75 Ohms (Z75) for BNC input, and high impedance (ZHI) and 110 Ohms (Z110) for XLR input.

The query argument is:

- **input** - { XLR | BNC }

Response argument(s):

- **response_input** - { XLR | BNC }
- **term** - { Z110 | Z75 | ZHI }

Related Commands: :DIN:IMPedance

Command Syntax: :DIN:IMPedance? **input**

Response Syntax: :DIN:IMPedance **response_input, term**

Example: `:DIN:IMPEDANCE? XLR`

Response: `:DIN:IMPEDANCE XLR,ZHI`

:DIN:INValid?

Returns the state of the validity bits of the digital input signal for each subframe (A and B). The invalid flags are driven directly by the V (Validity) bit defined in the Professional and Consumer standards. One Validity bit is sent in each subframe.

An response of 1 indicates that the audio data is not a valid signal.

An response of 0 indicates that that the audio data is a valid signal.

Command argument(s):

- **subframe** - { A | B)

Response argument(s):

- **subframe** - { A | B)
- **invalid** - <nr1>

Related Commands:

Command Syntax: :DIN:INValid? **subframe**

Response Syntax: :DIN:INValid **subframe, invalid**

Example: `:DIN:INVALID? A`

Response: `:DIN:INVALID A,1`

:DIN:JITTer?

Returns the jitter of the currently selected front panel XLR, BNC, or optical input connector signal (see :DIN:FORMat). This measurement is sensitive to jitter of the total signal, including transitions in the preambles and data sections of the frames.

The first parameter in the response is the next available measurement.

The Digital Input Jitter Meter settling is enabled or disabled by the algorithm parameter of the :SETTling:DIN command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

If the currently selected input (see :DIN:FORMat) is either Serial or Parallel the Jitter query will generate a command error.

The query argument is:

- **unit** - { SEC | UI }

Response argument(s):

- **jitter** - <nrf>
- **settle_timeout** - <nr1>

Related Commands: :DIN:SETTling

Command Syntax: :DIN:JITTer? **unit**

Response Syntax: :DIN:JITTer **jitter, settle_timeout**

Example: `:DIN:JITTER? UI`

Response: `:DIN:JITTER 3.5,1`

:DIN:LOCK?

Returns the state of the locked error flag of the digital input signal. The lock flag indicates when the digital input phase-locked loop is unable to lock to the incoming signal.

A response of 1 indicates the input is unlocked. A response of 0 indicates the input is locked.

Response argument(s):

- **unlock** - <nr1>

Related Commands:

Command Syntax: :DIN:LOCK?

Response Syntax: :DIN:LOCK **unlock**

Example: `:DIN:LOCK?`

Response: `:DIN:LOCK 1`

:DIN:MODE

Sets the mode of the digital input for the :DIN:BITS? measurement.

If the Active Bits mode is selected, each bit position in the ":DIN:BITS?" response will be set if the corresponding bit has changing between logical one and zero since the last ":DIN:MODE" command. If the Data Bits mode is selected, each bit position in the response reflects the state of the corresponding bit in the digital input signal.

Thus, the Active Bits mode indicates that normal data is being transmitted and any bit that is not set (zero) indicates either a stuck bit or that no signal is being transmitted in that bit. If a stuck bit is indicated in the Active Bits mode, the Data Bits mode may be used to determine whether the bit is stuck high or low.

The command argument is:

- **type** - { ACTive | DATA }

Default: ACTive

Related Commands: :DIN:BITS?, :DIN:MODE?

Command Syntax: :DIN:MODE **type**

Example: `:DIN:MODE ACTIVE`

:DIN:MODE?

Returns the mode of the :DIN:BITS? measurement. The valid responses are ACTIVE or DATA.

Response argument(s):

- **type** - { ACTive | DATA }

Related Commands: :DIN:MODE

Command Syntax: :DIN:MODE?

Response Syntax: :DIN:MODE **type**

Example: `:DIN:MODE?`

Response: `:DIN:MODE ACTIVE`

:DIN:ODELay?

Returns the time (phase) delay of the currently selected front panel XLR, BNC, or optical connector with reference to the internal Digital Generator output (labeled 'Delay from Out' on the APWIN DIO panel).

The first parameter in the response is the next available measurement.

The Digital Input Input-Delay-From-Generator-Output settling is enabled or disabled by the algorithm parameter of the :SETTling:DIN command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

If the currently selected input (see :DIN:FORMat) is either SERial or PARallel the delay from output query will generate an execution error and the query response will be 9.91E+37 (the NAN number) and 0 for the settle_timeout parameter.

Response argument(s):

- **delay** - <nrf>
- **settle_timeout** - <nr1>

Related Commands: :DIN:SETTling

Command Syntax: :DIN:ODELay?

Response Syntax: :DIN:ODELay **delay, settle_timeout**

Example: `:DIN:ODELAY?`

Response: `:DIN:ODELAY 2.5E-5,0`

:DIN:PARity?

Returns the state of the parity error flag of the digital input signal. The parity flag indicates a parity error in a subframe.

A response of 1 indicates a parity error. A response of 0 indicates no parity error.

Response argument(s):

- **parity** - <nr1>

Related Commands:

Command Syntax: :DIN:PARity?

Response Syntax: :DIN:PARity **parity**

Example: `:DIN:PARITY?`

Response: `:DIN:PARITY 1`

:DIN:PEAK?

Returns unsettled readings of the Channel A and Channel B imbedded audio signal peak level. This measurement may be used to determine if the input is near full scale (or clipped).

The query arguments are:

- **channel** - { A | B }
- **units** - { FFS | PCTFs | DBFS }

Response argument(s):

- **peak** - <nrf>

Related Commands:

Command Syntax: :DIN:PEAK? **channel, units**

Response Syntax: :DIN:PEAK **peak**

Example: `:DIN:PEAK? A,FFS`

Response: `:DIN:PEAK A,0.944061FFS`

:DIN:PKMode

Specifies the mode of the Channel A and Channel B imbedded audio signal peak level monitors.

The POS mode returns the most positive value during each measurement interval, which is approximately 1/4 second.

The NEG mode returns the most negative value during each measurement interval. When in the NEG mode, peak meter readings taken in dBFS units will be reported by taking the absolute value of the meter reading before converting to dBFS.

The ABS mode returns the largest positive-going or negative-going value during each measurement interval.

The HALF peak mode returns a value which is 1/2 the peak-to-peak range measured during the measurement interval.

The command argument is:

- **mode** - { HALF | ABS | NEG | POS }

Default: HALF

Related Commands: :DIN:PEAK?, :DIN:PKMode?

Command Syntax: :DIN:PKMode **mode**

Example: `:DIN:PKMODE NEG`

:DIN:PKMode?

Returns the current peak meter mode. The valid responses are: largest positive peak (POS), largest negative peak (NEG), largest absolute peak (ABS), and 1/2 of the peak-to-peak (HALF) reading.

Response argument(s):

- **mode** - { HALF | ABS | NEG | POS }

Related Commands: :DIN:PKMode

Command Syntax: :DIN:PKMode?

Response Syntax: :DIN:PKMode **mode**

Example: `:DIN:PKMODE?`

Response: `:DIN:PKMODE NEG`

:DIN:RATE?

Returns the current digital input sample rate. For all units except hertz, the ":DIN:REF" setting is used as the reference.

The first parameter in the response is the next available measurement.

The Digital Input Sample Rate settling is enabled or disabled by the algorithm parameter of the :SETTling:DIN command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

Command argument:

- **unit** - { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz }

Response argument(s):

- **rate** - <nrf>
- **settle_timeout** - <nr1>

Related Commands: :DIN:REF, :DIN:RUNIT, :DIN:SCALefreqby, :SETTling:DIN

Command Syntax: :DIN:RATE? **unit**

Response Syntax: :DIN:RATE **rate, settle_timeout**

Example: `:DIN:RATE? F_R`

Response: `:DIN:RATE 1.00,0`

:DIN:REF

Specifies the reference frequency used in reporting frequency measurements when the :DIN:SCALefreqby setting is REF. It is also used as the reference for the input and output sample rates. The setting units are always hertz.

The command argument is:

- **freq** - <nrf> (range: ≥ 8000, ≤ 54000)

Default: 48000

Related Commands: :DIN:RATE, :DIN:SCALefreqby, :DOUT:RATE, :DIN:REF?

Command Syntax: :DIN:REF **freq**

Example: `:DIN:REF 8000`

:DIN:REF?

Returns the current frequency reference value. This frequency is used when :DIN:SCALefreqby is set to REF. It is also used as the reference for the input and output sample rates. The response units are always hertz.

Response argument(s):

- f**r**eq - <nrf> (range: ≥ 8000, ≤ 54000)

Related Commands: :DIN:REF

Command Syntax: :DIN:REF?

Response Syntax: :DIN:REF **freq**

Example: `:DIN:REF?`

Response: `:DIN:REF 8000`

:DIN:RESolution

Specifies digital input data resolution in bits.

The digital input signal can be truncated at the LSB (least significant bit) of any desired word width (resolution) from 8 to 24 bits before being routed to Digital Analyzer programs for analysis of the imbedded audio. The Active Bits/Data Bits readings returned by the :DIN:BITS? query monitor the digital input signal before truncation by the value determined by :DIN:RESolution, so they will indicate the full word width of the external input signal independent of the :DIN:RESolution value. The value of quantization noise and distortion of the imbedded audio of digital input signals measured by the Analyzer, FFT, or FASTTEST programs will be affected by the Input Resolution setting.

The command argument (in bits) is:

- **bits** - <nr1> (range: ≥ 8, ≤ 24)

Default: 24

Related Commands: :DIN:BITS?, :DIN:RESolution?

Command Syntax: :DIN:RESolution **bits**

Example: `:DIN:RESOLUTION 20`

:DIN:RESolution?

Returns the current Input Resolution setting. The default units is bits.

Response argument(s):

- **bits** - <nr1> (range: ≥ 8, ≤ 24)

Related Commands: :DIN:RESolution

Command Syntax: :DIN:RESolution?

Response Syntax: :DIN:RESolution **bits**

Example: `:DIN:RESOLUTION?`

Response: `:DIN:RESOLUTION 20`

:DIN:SCALefreqby

Specifies the scaling source for the imbedded digital audio signals. Depending on the application, there are several sources of the digital sample rate that may be appropriate to use in the normalization. The SCALefreqby command permits selection of Output Rate (see :DOUT:RATE), Measured Rate (see :DIN:RATE), Status Bits A (the value of sample frequency encoded into the received channel A status bits), or a specified reference frequency (see :DIN:REF) as a scaling source.

The majority of applications use Measured Rate as the scaling source so that imbedded audio signal frequency measurements follows any changes in sample rate of the input source. The Output Rate selection can be used to measure the frequencyshifting effects (Vari-Speed) of digital processors and sample rate converters. The Status Bits selection refers frequency measurements to the nominal, standard sample rate (if encoded into the status bits) and will be independent of any moment-to-moment variations in the actual received sample rate.

The command argument is:

- **source** - { MEASured | OUTPut | REF | STATus }

Default: MEASured

Related Commands: :DIN:SCALefreqby?, :DIN:RATe?, :DIN:REF

Command Syntax: :DIN:SCALefreqby **source**

Example: `:DIN:SCALEFREQBY OUTPUT`

:DIN:SCALefreqby?

Returns the source of the frequency used to scale the imbedded digital audio signal. The valid responses are: output sample rate (OUTPut), input measured sample rate (INPut), channel A encoded sample rate (STATus), and user specified reference (REF).

Response argument(s):

- **source** - { MEASured | OUTPut | REF | STATus }

Related Commands: :DIN:SCALefreqby

Command Syntax: :DIN:SCALefreqby?

Response Syntax: :DIN:SCALefreqby **source**

Example: `:DIN:SCALEFREQBY?`

Response: `:DIN:SCALEFREQBY OUTPUT`

:DIN:SET?

Queries all Digital Input command settings. The response consists of a sequence of the settings that may be sent back to the instrument at a later time in order to reset all settings to the same state.

Response argument(s):

- **settings** - <response message unit> [<response message unit>] ...

Related Commands: <see all other :DIN settings queries>

Command Syntax: :DIN:SET?

Response Syntax: :DIN:SET **settings**

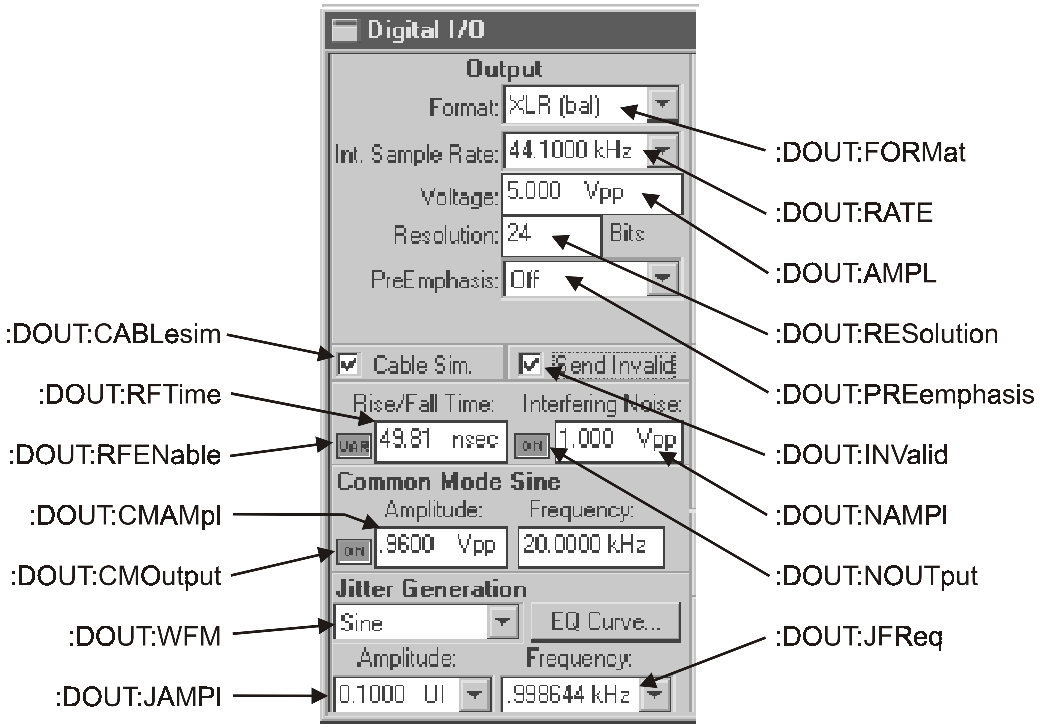
Example: `:DIN:SET?`

Response:

```
:DIN:REF 48000;BANDWIDTH 700;RESOLUTION 24;D
ETECTOR AVG;MODE ACTIVE;PKMODE HALF;SCALEFRE
QBY MEASURED;DEEMPHASIS OFF;FORMAT XLR;IMPED
ANCE BNC,Z75;IMPEDANCE XLR,Z110
```

Digital Output Commands

The header-path for Digital Output commands is :DOUT.



Digital I/O

Output

Format: XLR (bal)

Int. Sample Rate: 44.1000 kHz

Voltage: 5.000 Vpp

Resolution: 24 Bits

PreEmphasis: Off

☑ Cable Sim. ☑ Send Invalid

Rise/Fall Time: VAR 49.81 nsec

Interfering Noise: ON 1.000 Vpp

Common Mode Sine

Amplitude: ON .9600 Vpp

Frequency: 20.0000 kHz

Jitter Generation

Sine

EQ Curve...

Amplitude: 0.1000 UI

Frequency: .998644 kHz

Figure 10-2. DIO Panel, Output section, showing relationship with GPIB commands

:DOUT:AMPL

Specifies the peak-to-peak voltage amplitude of the serial pulse train at the digital output selected by the :DOUT:FORMAT command if the output is XLR, BNC, or optical. The output amplitude will not be calibrated for the unselected outputs or if the selected output is not properly terminated (110 Ohms for the XLR, 75 Ohms for the BNC). This amplitude may be varied to simulate cable attenuation. This command does not affect the general-purpose serial or parallel outputs.

The amplitude ratio of the XLR connector to the BNC connector is always 5:1. Thus, with the XLR connector selected by :DOUT:FORMat, 10 Volts selected by :DOUT:AMPL, and the XLR and BNC connectors both properly terminated, the XLR connector will have a 10 Volt pulse amplitude and the BNC will have a 2 Volt pulse amplitude.

If the BNC connector is selected with :DOUT:FORMat, :DOUT:AMPL is set to 1 Volt, and both connectors are properly terminated, then the XLR will have 5 Volts of pulse amplitude. Voltage calibration at the XLR and BNC connectors assumes a matched load, and the actual voltage will rise to approximately double the programmed value if no load is connected.

The Gen Mon connection of the DIO panel does not provide a termination, so Gen Mon measurements will show about twice the programmed value if the digital generator is unloaded.

The command argument is:

- **setting** - <nrf> (range: ≥ 0, ≤ 10.2 depends on digital output port currently enabled)

Default: 5.0

Related Commands: :DOUT:AMPL?

Command Syntax: :DOUT:AMPL **setting**

Example: `:DOUT:AMPL 1.2`

:DOUT:AMPL?

Returns the amplitude of the serial pulse train at the XLR, BNC, and optical outputs. The response has units of Vpp.

Response argument(s):

- **setting** - <nrf> (range: ≥ 0, ≤ 10.2 depends on digital output port currently enabled)

Related Commands: :DOUT:

Command Syntax: :DOUT:AMPL?

Response Syntax: :DOUT:AMPL **setting**

Example: `:DOUT:AMPL?`

Response: `:DOUT:AMPL 1.2`

:DOUT:CABLesim

Enables or disables (into the XLR and BNC paths) a fixed hardware filter that simulates a typical 1000 meter cable.

The command argument is:

- **state** - { OFF | ON }

Default: OFF

Related Commands: :DOUT:CABLesim?

Command Syntax: :DOUT:CABLesim **state**

Example: `:DOUT:CABLESIM ON`

:DOUT:CABLesim?

Returns the state (enabled or disabled) of the cable simulation filter.

Response argument(s):

- **state** - { OFF | ON }

Related Commands: :DOUT:CABLesim

Command Syntax: :DOUT:CABLesim?

Response Syntax: :DOUT:CABLesim **state**

Example: `:DOUT:CABLESIM?`

Response: `:DOUT:CABLESIM ON`

:DOUT:CMAMpl

Specifies the amplitude of the sine wave added to the XLR output as a common mode interference signal (when :DOUT:CMOutput is enabled). This signal is not added to any other outputs. The setting will be adjusted to the nearest voltage in increments of 0.08 Vpp.

The command argument (in Vpp) is:

- **setting** - <nrf> (range: ≥ 0, ≤ 20.480001)

Default: 0.960

Related Commands: :DOUT:CMFReq, :DOUT:CMOutput, :DOUT:CMAMpl?

Command Syntax: :DOUT:CMAMpl **setting**

Example: `:DOUT:CMAMPL 1.5`

:DOUT:CMAMpl?

Returns the actual amplitude setting of the common mode interference signal. The response is in units of Vpp.

Response argument(s):

- **setting** - <nrf> (range: ≥ 0, ≤ 20.4)

Related Commands: :DOUT:CMAMpl

Command Syntax: :DOUT:CMAMpl?

Response Syntax: :DOUT:CMAMpl **setting**

Example: `:DOUT:CMAMPL?`

Response: `:DOUT:CMAMPL 1.5`

:DOUT:CMFReq

Specifies the frequency of the sine wave added to the XLR output as a common mode interference signal (when :DOUT:CMOutput is enabled). This signal is not added to any other outputs. The actual input frequency setting will be selected according to the following table:

Requested Frequency			Actual Frequency
>=0.0	–	<30.0	20.0
>=30.0	–	<60.0	40.0
>=60.0	–	<120.0	80.0
>=120.0	–	<240.0	160.0
>=240.0	–	<480.0	315.0
>=480.0	–	<960.0	630.0
>=960.0	–	<1920.0	1250.0
>=1920.0	–	<3840.0	2500.0
>=3840.0	–	<7680.0	5000.0
>=7680.0	–	<15360.0	10000.0
>=15360.0	–	<30720.0	20000.0
>=30720.0	–	<40000.0	40000.0

The command argument (in hertz) is:

- **frequency** - <nr1> (range: ≥ 0, ≤ 40000)

Default: 20000.0

Related Commands: :DOUT:CMAMpl, :DOUT:CMOutput, :DOUT:CMFReq?

Command Syntax: :DOUT:CMFReq **frequency**

Example: `:DOUT:CMFREQ 2000`

:DOUT:CMFReq?

Returns the actual frequency setting of the common mode interference signal. See :DOUT:CMFReq command for list of possible return values.

Response argument(s):

- **frequency** - <nr1> (range: ≥ 0, ≤ 40000)

Related Commands: :DOUT:CMFReq

Command Syntax: :DOUT:CMFReq?

Response Syntax: :DOUT:CMFReq **frequency**

Example: `:DOUT:CMFREQ?`

Response: `:DOUT:CMFREQ 2500`

:DOUT:CMOutput

Enables or disables the addition of the common mode interference signal to the XLR output.

The command argument is:

- **state** - { OFF | ON }

Default: OFF

Related Commands: :DOUT:CMAMpl, :DOUT:CMFReq, :DOUT:CMOutput?

Command Syntax: :DOUT:CMOutput **state**

Example: `:DOUT:CMOUTPUT ON`

:DOUT:CMOutput?

Returns the state of the common mode output enable flag.

Response argument(s):

- **state** - { OFF | ON }

Related Commands: :DOUT:CMAMpl, :DOUT:CMFReq, :DOUT:CMOutput

Command Syntax: :DOUT:CMOutput?

Response Syntax: :DOUT:CMOutput **state**

Example: `:DOUT:CMOUTPUT?`

Response: `:DOUT:CMOUTPUT ON`

:DOUT:FORMat

Specifies the output for the digital signal. The valid outputs are:

Output	Meaning
XLR (bal)	Front panel XLR digital output connector, balanced
BNC (unbal)	Front panel BNC digital output connector, unbalanced
Optical	Front panel Toslink optical output connector
Parallel	Rear-panel parallel connector
Serial	Rear-panel general-purpose serial connector

The command argument is:

- **source** - { XLR | BNC | OPTical | PARallel | SERial }

Default: XLR

Related Commands: :DOUT:FORMat?

Command Syntax: :DOUT:FORMat **source**

Example: `:DOUT:FORMAT SERIAL`

:DOUT:FORMat?

Returns the currently selected digital output.

Response argument(s):

- **source** - { XLR | BNC | OPTical | PARallel | SERial }

Related Commands: :DOUT:FORMat

Command Syntax: :DOUT:FORMat?

Response Syntax: :DOUT:FORMat **source**

Example: `:DOUT:FORMAT?`

Response: `:DOUT:FORMAT SERIAL`

:DOUT:INValid

Specifies the state of the AES/EBU validity flag (bit 28 of the subframe format) for both subframes A and B. It is not possible to set the individual subframe separately.

An argument of 1 sets the validity flag to indicate that the audio data is not a valid signal.

An argument of 0 sets the validity flag to indicate that the audio data is a valid signal.

The command argument is:

- **state** - <nr1> (range: ≥ 0, ≤ 1)

Default: OFF

Related Commands: :DOUT:INValid?

Command Syntax: :DOUT:INValid **state**

Example: `:DOUT:INVALID 1`

:DOUT:INValid?

Returns the subframe validity bit setting in the digital output signal.

Response argument(s):

- **state** - <nr1> (range: ≥ 0, ≤ 1)

Related Commands: :DOUT:INValid

Command Syntax: :DOUT:INValid?

Response Syntax: :DOUT:INValid **state**

Example: `:DOUT:INVALID?`

Response: `:DOUT:INVALID 1`

:DOUT:JAMPl

Specifies the injected jitter signal amplitude when the jitter type is sine, low pass random, square, or wideband random.

The command argument is:

- **amplitude** - <nrf> (range: ≥ 0, ≤ 25.5 depends on internal sample rate for seconds units) { UI | SEC }

Default: 0.0 UI

Related Commands: :DOUT:JAMPl?

Command Syntax: :DOUT:JAMPl **amplitude**

Example: `:DOUT:JAMPL 25.5 UI`

:DOUT:JAMPl?

Returns the injected jitter amplitude setting (in the specified units).

The command argument is:

- **units** - { UI | SEC }

Response argument(s):

- amplitude - <nrf> { UI | SEC }

Related Commands: :DOUT:JAMPl

Command Syntax: :DOUT:JAMPl? **units**

Response Syntax: :DOUT:JAMPl amplitude

Example: `:DOUT:JAMPL? UI`

Response: `:DOUT:JAMPL 25.5UI`

:DOUT:JFReq

Specifies the frequency of the injected jitter signal. This frequency is only applicable when jitter type is set to sine, low pass random, or square. This is not applicable when jitter type is wideband random.

The command argument (in hertz) is:

- **frequency** - <nrf> (range: ≥ 9.14286, ≤ 99864.4)

Default: 998.644

Related Commands: :DOUT:JFReq?

Command Syntax: :DOUT:JFReq **frequency**

Example: `:DOUT:JFREQ 1500`

:DOUT:JFReq?

Returns current jitter interference signal frequency. The response is in hertz.

Response argument(s):

- **frequency** - <nrf>

Related Commands: :DOUT:JFReq

Command Syntax: :DOUT:JFReq?

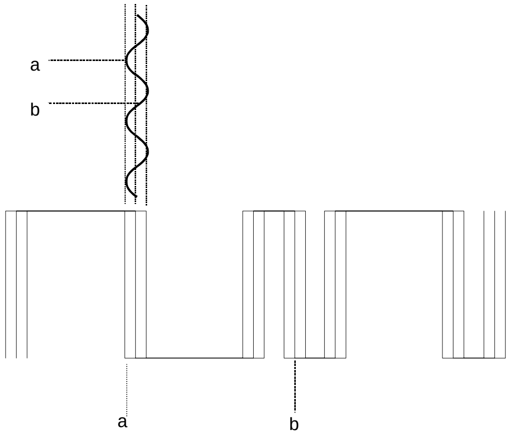
Response Syntax: :DOUT:JFReq **frequency**

Example: `:DOUT:JFREQ?`

Response: `:DOUT:JFREQ 1500`

:DOUT:JWFM

Disables the interface jitter signal or specifies the type of jitter waveform. The valid jitter waveform types are: sine, low pass random, square, or wideband random.



The diagram above illustrates a digital audio (AES/EBU) signal that has sine jitter. The sinewave represents the injected jitter signal. For a given point ***a*** on the digital audio signal, the amount of jitter is defined by the corresponding point ***a*** on the jitter signal (sinewave, in this case). The amplitude of the potential jitter around point ***a*** on the digital audio signal is defined by the amplitude of the jitter signal.

The command argument is:

- **type** - { NONE | LPRandom | SINE | SQUare | WIDeband }

Default: NONE

Related Commands: :DOUT:JWFM?

Command Syntax: :DOUT:JWFM **type**

Example: `:DOUT:JWFM SQUARE`

:DOUT:JWFM?

Returns the current injected jitter waveform type setting.

Response argument(s):

- **type** - { NONE | LPRandom | SINE | SQUare | WIDeband }

Related Commands: :DOUT:JWFM

Command Syntax: :DOUT:JWFM?

Response Syntax: :DOUT:JTYPe **type**

Example: `:DOUT:JWFM?`

Response: `:DOUT:JWFM SQUARE`

:DOUT:NAMPl

Specifies the amplitude of the interfering random white noise that may be added to the serial pulse train at the BNC connector and added as a Normal Mode signal (between pins 2 and 3) at the XLR connector. This capability is not available at the optical, general purpose serial, or parallel outputs. The maximum amplitude available is 25% of the maximum pulse amplitude at either connector. Thus, the maximum noise amplitude is approximately 2.55 Vpp at the XLR and approximately 0.6375 Vpp at the BNC connector.

If the amplitude is too high for the current format a command error will be generated.

The command argument (in Vpp) is:

- **amplitude** - <nrf> (range: ≥ 0, ≤ 2.55 Vpp XLR or ≤ 0.6375 Vpp BNC, depends on which connecter output has been selected)

Default: 0.0

Related Commands: :DOUT:NAMPl?

Command Syntax: :DOUT:NAMPl **amplitude**

Example: `:DOUT:NAMPL 1.0`

:DOUT:NAMPl?

Returns amplitude of the interfering random white noise.

Response argument(s):

- **amplitude** - <nrf> (range: ≥ 0, ≤ 2.55 Vpp XLR or ≤ 0.6375 Vpp BNC, depends on which connecter output has been selected)

Related Commands: :DOUT:NAMPl

Command Syntax: :DOUT:NAMPl?

Response Syntax: :DOUT:NAMPl **amplitude**

Example: `:DOUT:NAMPL?`

Response: `:DOUT:NAMPL 1.0`

:DOUT:NOUTput

Enables/disables interfering noise on the digital output

The command argument is:

- **state** - { OFF | ON }

Default: OFF

Related Commands: :DOUT:NOUTput?

Command Syntax: :DOUT:NOUTput **state**

Example: `:DOUT:NOUTPUT OFF`

:DOUT:NOUTput?

Returns the state of the interfering noise output (enabled or disabled).

Response argument(s):

- **state** - { OFF | ON }

Related Commands: :DOUT:NOUTput

Command Syntax: :DOUT:NOUTput?

Response Syntax: :DOUT:NOUTput **state**

Example: `:DOUT:NOUTPUT?`

Response: `:DOUT:NOUTPUT OFF`

:DOUT:PREemphasis

Disables pre-emphasis or specifies the pre-emphasis function of the imbedded digital audio output signal. Note that only 50/15 microsecond (CD) pre-emphasis (or no pre-emphasis) are defined conditions under the consumer standard, but the AES/EBU standard also defines CCITT J17 pre-emphasis.

Either pre-emphasis function may be selected at normal gain or with a headroom allowance. When program material is put through a pre-emphasis function, the natural high-frequency roll-off of most music and voice signals and typical practices of headroom allowance for peaks are sufficient to assure that high-frequency signals will not clip (exceed digital full scale). However, full-scale test signals such as sine wave sweeps or multitone signals with equal amplitude at all frequencies will clip at high frequencies when pre-emphasis is applied. To prevent this clipping due to the high-frequency boost, two additional selections (CD+10db and J17+20db) are available which automatically attenuate the signal level sufficiently to provide headroom at the highest frequencies.

The command argument is:

- **type** - { OFF | CD | CDGain | J17 | J17Gain }

Default: OFF

Related Commands: :DOUT:PREemphasis?

Command Syntax: :DOUT:PREemphasis **type**

Example: `:DOUT:PREEMPHASIS CDGAIN`

:DOUT:PREemphasis?

Returns the currently selected pre-emphasis function.

Response argument(s):

- **type** - { OFF | CD | CDGain | J17 | J17Gain }

Related Commands: :DOUT:PREemphasis

Command Syntax: :DOUT:PREemphasis?

Response Syntax: :DOUT:PREemphasis **type**

Example: `:DOUT:PREEMPHASIS?`

Response: `:DOUT:PREEMPHASIS J17`

:DOUT:RATE

Specifies the Digital Output sample rate.

The command argument is:

- **rate** - <nrf> (range: ≥ -1E34, ≤ 1E34) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 48000.0 HZ

Related Commands: :DOUT:?

Command Syntax: :DOUT:RATE **rate**

Example: `:DOUT:RATE 32000 HZ`

:DOUT:RATE?

Returns the Digital Output sample rate.

The command argument is:

- **unit** - { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Response argument(s):

- **rate** - <nrf> { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default:

Related Commands: :DOUT:RATE

Command Syntax: :DOUT:RATE? **unit**

Response Syntax: :DOUT:RATE **rate**

Example: `:DOUT:RATE?`

Response: `:DOUT:RATE 32000HZ`

:DOUT:RESolution

Specifies the digital output word width and dither amplitude of the imbedded digital audio signal. The implied unit is bits.

The width or resolution of the embedded digital audio signal may be set to any value from 8 to 24 bits. Internally, the embedded digital audio signal is always generated at 24 bits.

When any smaller value is selected the 24-bit word is rounded (not truncated) to the specified value and dither is added (unless disabled with the Digital Generator :DGEN:DITHERTYPE command) at the proper amplitude for the value entered. Bits below the value specified by :DOUT:RESOLUTION are set to zero. The output resolution is independent from the input resolution.

The command argument is:

- **bits** - <nr1> (range: ≥ 8, ≤ 24)

Default: 24

Related Commands: :DOUT:RESolution?

Command Syntax: :DOUT:RESolution **bits**

Example: `:DOUT:RESOLUTION 20`

:DOUT:RESolution?

Returns the resolution of the imbedded digital audio signal.

Response argument(s):

- **bits** - <nr1> (range: ≥ 8, ≤ 24)

Related Commands: :DOUT:RESolution

Command Syntax: :DOUT:RESolution?

Response Syntax: :DOUT:RESolution **bits**

Example: `:DOUT:RESOLUTION?`

Response: `:DOUT:RESOLUTION 20`

:DOUT:RFENable

Enables or disables variable Rise and Fall Time of the pulse train at the XLR and BNC outputs. This capability is not available at the optical, general-purpose serial, or parallel outputs.

When disabled (FIXed) the Rise and Fall Times are at their fastest, approximately 16 nanoseconds. When enabled (VARiable), new Rise and Fall Time values are set using the :DOUT:RFTime command.

The command argument is:

- **state** - { FIXed | VARiable }

Default: FIXed

Related Commands: :DOUT:RFENable?

Command Syntax: :DOUT:RFENable **state**

Example: `:DOUT:RFENABLE VARIABLE`

:DOUT:RFENable?

Returns the setting of the variable Rise and Fall time enable command (RFENable).

Response argument(s):

- **state** - { FIXed | VARiable }

Related Commands: :DOUT:RFENable

Command Syntax: :DOUT:RFENable?

Response Syntax: :DOUT:RFENable **state**

Example: `:DOUT:RFENABLE?`

Response: `:DOUT:RFENABLE VARIABLE`

:DOUT:RFTime

Sets the Rise and Fall Time of the pulse stream at the XLR and BNC outputs if :DOUT:RFENable is enabled. If :DOUT:RFENable is disabled this command will set the Rise and Fall Time to be used when this feature is enabled later, but the Rise and Fall time will stay at the fixed value, approximately 16 nanoseconds. The range is 16 ns to 400 ns.

This may be used to simulate the bandwidth reduction typical with increasing cable length. This capability is not available at the optical, general purpose serial, or parallel outputs.

The command argument (in seconds) is:

- **time** - <nrf> (range: ≥ 15.706E-9, ≤ 403.33E-9)

Default: 15.96E-9

Related Commands: :DOUT:RFENable, :DOUT:RFTime?

Command Syntax: :DOUT:RFTime **time**

Example: `:DOUT:RFTIME 200E-9`

:DOUT:RFTime?

Returns the current Rise/Fall Time of the pulse train. If RFENable is disabled this query will return the value of the Rise/Fall Time, not the actual Rise/Fall Time. The response is in seconds.

Response argument(s):

- **time** - <nrf> (range: ≥ 15.706E-9, ≤ 403.33E-9)

Related Commands: :DOUT:RFTime

Command Syntax: :DOUT:RFTime?

Response Syntax: :DOUT:RFTime **time**

Example: `:DOUT:RFTIME?`

Response: `:DOUT:RFTIME 2.0115E-07`

:DOUT:SET?

Returns the current state of the Digital Output settings.

The response consists of a sequence of the Digital Output settings that may be sent back to the instrument at a later time in order to reset these settings to the same state. Note that some commands are sent more than once and transition through several states in order to achieve a final instrument state.

Response argument(s):

- **settings** - <response message unit> [<response message unit>] ...

Related Commands: see other :DOUT setting commands

Command Syntax: :DOUT:SET?

Response Syntax: :DOUT:SET **settings**

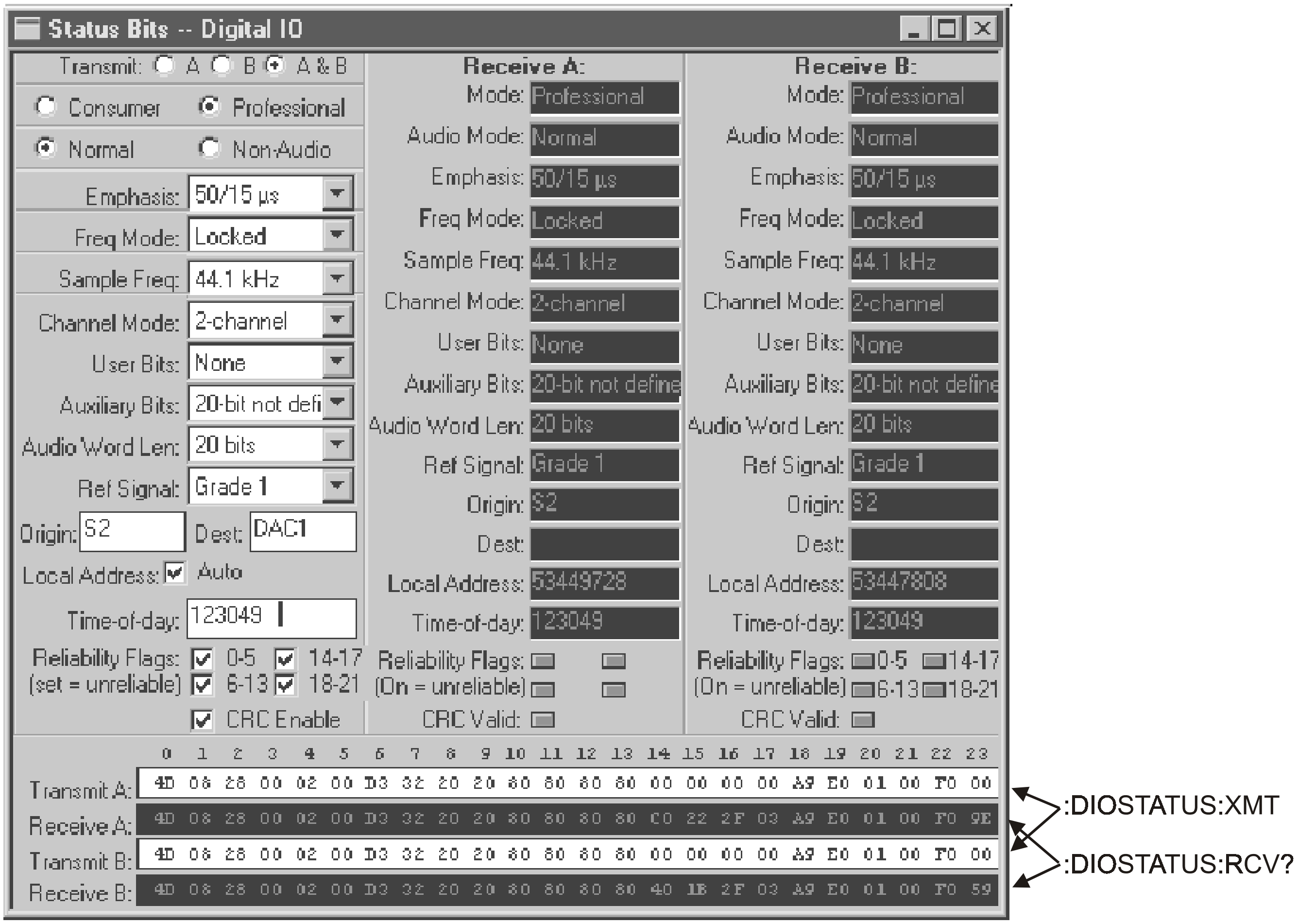
Example: `:DOUT:SET?`

Response:

```
:DOUT:FORMAT XLR;AMPL 5;CABLESIM OFF;CMOUTPU
T ON;CMAMPL 0.96;CMFREQ 20000;CMOUTPUT OFF;I
NVALID 0;JWFM SINE;JAMPL 0UI;JFREQ 998.644;J
WFM NONE;NOUTPUT ON;NAMPL 0;NOUTPUT OFF;PREE
MPHASIS OFF;RATE 48000HZ;RESOLUTION 24;RFENA
BLE VAR;RFTIME 1.59612E-08;RFENABLE FIXED;FO
RMAT XLR
```

Digital I/O Status Bits Commands

Digital I/O Status Bit commands apply to AES/EBU status bits. The header-path for Digital I/O Status Bit commands is :DIOStatus:.



Status Bits -- Digital IO

Transmit: ○ A ○ B ● A & B

○ Consumer ● Professional

● Normal ○ Non-Audio

Emphasis: 50/15 µs
Freq Mode: Locked
Sample Freq: 44.1 kHz
Channel Mode: 2-channel
User Bits: None
Auxiliary Bits: 20-bit not defi
Audio Word Len: 20 bits
Ref Signal: Grade 1
Origin: S2 Dest: DAC1
Local Address: ☑ Auto
Time-of-day: 123049
Reliability Flags: (set = unreliable) ☑ 0-5 ☑ 14-17 ☑ 6-13 ☑ 18-21
☑ CRC Enable

	Receive A:	Receive B:
Mode:	Professional	Professional
Audio Mode:	Normal	Normal
Emphasis:	50/15 µs	50/15 µs
Freq Mode:	Locked	Locked
Sample Freq:	44.1 kHz	44.1 kHz
Channel Mode:	2-channel	2-channel
User Bits:	None	None
Auxiliary Bits:	20-bit not define	20-bit not define
Audio Word Len:	20 bits	20 bits
Ref Signal:	Grade 1	Grade 1
Origin:	S2	S2
Dest:		
Local Address:	53449728	53447808
Time-of-day:	123049	123049
Reliability Flags: (On = unreliable)	▭ ▭ ▭ ▭	▭ 0-5 ▭ 14-17 ▭ 6-13 ▭ 18-21
CRC Valid:	▭	▭

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Transmit A:	4D	08	28	00	02	00	D3	32	20	20	80	80	80	80	00	00	00	00	A9	E0	01	00	F0	00
Receive A:	4D	08	28	00	02	00	D3	32	20	20	80	80	80	80	C0	22	2F	03	A9	E0	01	00	F0	9E
Transmit B:	4D	08	28	00	02	00	D3	32	20	20	80	80	80	80	00	00	00	00	A9	E0	01	00	F0	00
Receive B:	4D	08	28	00	02	00	D3	32	20	20	80	80	80	80	40	1B	2F	03	A9	E0	01	00	F0	59

:DIOSTATUS:XMT

:DIOSTATUS:RCV?

Figure 10-3. DIO Panel, Status Bits section, showing relationship with GPIB commands

:DIOStatus:RCV?

This query returns the 192 bits of data for either Channel A or B AESEBU receive subframe. The data is formatted as a quoted string of 24 comma-separated fields, each field is two hex characters. The first comma separated field (left) is byte 0.

The command argument is:

- **channel** - { A | B }

Response argument(s):

- **status** - < string data >

Related Commands: <None>

Command Syntax: :DIOStatus:RCV? **channel**

Response Syntax: :DIOStatus:RCV **status**

Example: `:DIOSTATUS:RCV? A`

Response:

```
:DIOSTATUS:RCV "8D,82,2C,00,02,00,41,50,53,32,41,
50,53,32,80,F6,8B,00,D2,04,00,00,00,7E"
```

:DIOStatus:XMT

This command sets the 192 AESEBU transmit channel status bits. The command arguments are the channel (A or B AES/EBU subframe) and the status bit data. The data is formatted as a quoted string of 24 comma-separated fields, each field is two hex characters. The first comma separated field (left) is byte 0.

Note that the string contents may be cut directly from the APWIN Transmit section of the System Two Status Bits panel after setting the bits with the high level controls. This technique simplifies the process of encoding the hex bytes by using APWIN to do it for you. Select the channel A or B output status bits field with the mouse cursor and then select Edit-Copy to copy the selection to the windows clip board. Paste the string directly into the argument field of the command string you are editing. The resultant string must have commas inserted as shown below.

The command arguments are:

- **channel** - { A | B }
- **data** - <string data>

Default:

A,"04,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
,00,00,00";B,"04,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00"

Related Commands: :DIOStatus:XMT?

Command Syntax: :DIOStatus:XMT channel, data

Example:

```
:DIOSTATUS:XMT A,"8D,82,2C,00,02,00,41,50,53,32,4
1,50,53,32,80,F6,8B,00,D2,04,00,00,00,7E"
```

:DIOStatus:XMT?

This query returns the current setting of the AESEBU transmit channel status bits. The argument to this query is the channel (A or B AES/EBU subframe).

The response consists of the command header, the channel, and a quoted string of 24 comma-separated fields, each field is two hex characters. The first comma separated field (left) is byte 0.

The command argument is:

- **channel** - { A | B }

Response argument(s):

- **response_channel** - { A | B }
- **data** - <string data>

Related Commands: :DIOStatus:XMT

Command Syntax: :DIOStatus:XMT? **channel**

Response Syntax: :DIOStatus:XMT **response_channel, data**

Example:

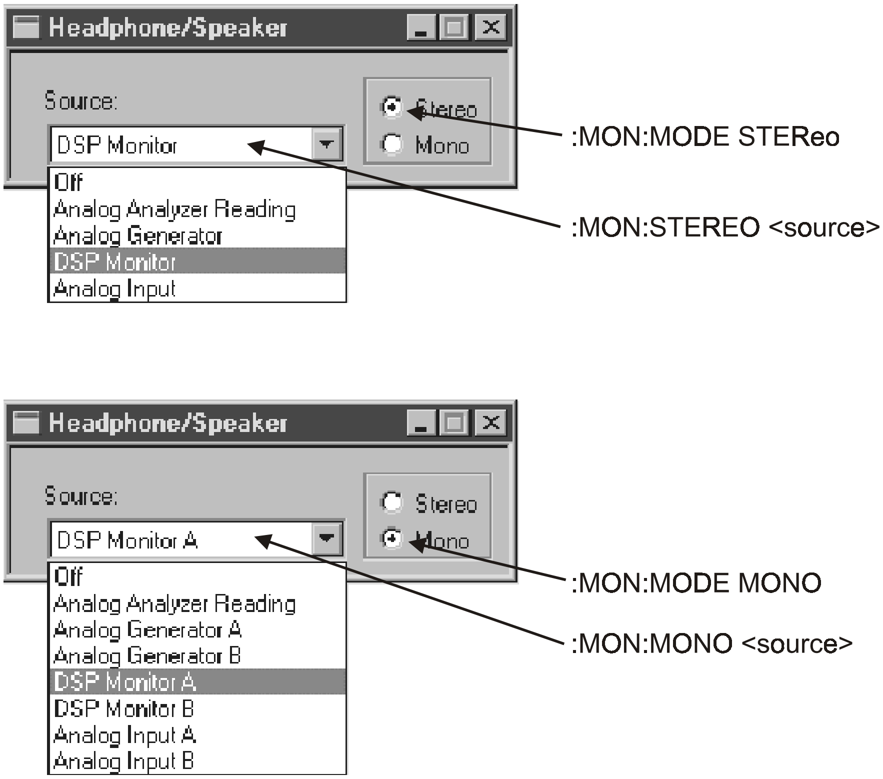
```
:DIOSTATUS:XMT? A
```

Response:

```
:DIOSTATUS:XMT A,"8D,82,2C,00,02,00,41,50,53,32,4
1,50,53,32,80,F6,8B,00,D2,04,00,00,00,7E"
```

11. Monitor Headphone/Speaker Commands

Monitor commands apply to the internal speaker and to the monitor jack on the front of the System Two. The header-path for the speaker/headphone commands is :MON:



:MON:MODE STEReo

:MON:STEREO <source>

:MON:MODE MONO

:MON:MONO <source>

Figure 11-1. Monitor panels, showing relationship with GPIB commands

:MON:MODE

Sets the instrument headphone/speaker monitor to monophonic or stereo mode.

The command argument is:

- **mode** - { STEReo | MONO }

Default: STEReo

Related Commands: :MON:MODE?

Command Syntax: :MON:MODE **mode**

Example: `:MON:MODE STEREO`

:MON:MODE?

Returns the mode of the headphone/speaker monitor, either monophonic or stereo.

Response argument(s):

- **mode** - { STEReo | MONO }

Related Commands: :MON:MODE

Command Syntax: :MON:MODE?

Response Syntax: :MON:MODE **mode**

Example: `:MON:MODE?`

Response: `:MON:MODE STEREO`

:MON:MONO

Specifies the instrument channel input for the headphone/speaker monitor when in MONO mode. If this command is received by the instrument while in STEREO mode, the setting will be stored and used when the monitor is switched to MONO mode.

The signal sources are: none (OFF), Analog Analyzer function meter (FUNCmeter), Analog Generator A monitor (GENA), Analog Generator B monitor (GENB), DSP channel A monitor (DSPA), DSP channel B monitor (DSPB), Analog Analyzer A input (INPA), and Analog Analyzer B input (INPB).

The command argument is:

- **source** - { OFF | DSPA | DSPB | FUNCmeter | GENA | GENB | INPA | INPB }

Default: OFF

Related Commands: :MON:MONO?, :MON:MODE

Command Syntax: :MON:MONO **source**

Example: `:MON:MONO INPA`

:MON:MONO?

Returns the instrument channel input for the headphone/speaker monitor when in MONO mode.

The signal sources are: none (OFF), Analog Analyzer function meter (FUNCmeter), Analog Generator A monitor (GENA), Analog Generator B monitor (GENB), DSP channel A monitor (DSPA), DSP channel B monitor (DSPB), Analog Analyzer A input (INPA), and Analog Analyzer B input (INPB).

Response argument(s):

- **source** - { OFF | DSPA | DSPB | FUNCmeter | GENA | GENB | INPA | INPB }

Related Commands: :MON:MONO, :MON:MODE

Command Syntax: :MON:MONO?

Response Syntax: :MON:MONO **source**

Example: `:MON:MONO INPA`

:MON:SET?

Returns settings for all Monitor commands. The response consists of a sequence of the Monitor settings that may be sent back to the instrument at a later time in order to reset Monitor settings to the same state.

Response argument(s):

- **settings** - <response message unit> [<response message unit>] ...

Related Commands: :MON:CHANnel?, :MON:MODE?, :MON:SRC?

Command Syntax: :MON:SET?

Response Syntax: :MON:SET **settings**

Example: `:MON:SET?`

Response: `:MON:MONO OFF;STEREO OFF;MODE STEREO`

:MON:STEReo

Selects the stereo monitor signal source.

The signal sources are: none (OFF), Analog Analyzer function meter (FUNCmeter), Analog Generator monitor (GENMon), DSP monitor (DSP), and Analog Analyzer input (INPut).

If the high bandwidth A/D selection is in use then the DSP Monitor selection does not provide a monitor signal (it is assumed that the signals being measured are above the range of human hearing).

The command argument is:

- **source** - { OFF | DSP | FUNCmeter | GENMon | INPut }

Default: OFF

Related Commands: :MON:STEReo?

Command Syntax: :MON:STEReo **source**

Example: `:MON:STEREO DSP`

:MON:STEReo?

Returns the currently selected stereo monitor source. The signal sources are: no input (OFF), Analog Analyzer function meter (FUNCmeter), Analog Generator monitor (GENMon), DSP monitor (DSP), and Analog Analyzer input (INPut).

Response argument(s):

- **source** - { OFF | DSP | FUNCmeter | GENMon | INPut }

Related Commands: :MON:STEReo

Command Syntax: :MON:STEReo?

Response Syntax: :MON:STEReo **source**

Example: `:MON:STEREO?`

Response: `:MON:STEREO DSP`

12. Settling Commands

Most measurements provided by System Two can be "settled", a process that reduces testing time by determining when a measurement value has stopped changing sufficiently to be considered stable. The settling subsystem allows selection of settling criteria for most measurements provided by System Two. These measurements have default settling parameters. Settling is enabled by default on all measurements that provide a settling capability.

The global timeout value (determined via the :SETTling:TIMeout command) is used to determine how long to wait for a measurement to settle before returning an unsettled measurement. This prevents the system from waiting indefinitely for a measurement that may never settle, such as a noisy test signal or live broadcast program material. The global timeout value may be overridden on a measurement-by-measurement basis if a non-zero settling timeout value is specified for the desired measurement.

Measurement queries can occur at any point in the measurement cycle (measurements are being constantly taken), the next measurement may be available almost immediately or it could take almost a complete measurement cycle to become available. The **trigger** argument in the settling command specifies whether measurement queries will use the next available measurement or abort the current measurement and trigger a new measurement cycle. Use triggered measurements when it is necessary to synchronize a new measurement with changes in the device under test or changes in the stimulus signal in order to avoid acquiring a “stale” measurement that may have been affected by the previous state of the test system. This is usually not a problem if settled measurements are being acquired but may be a problem with unsettled measurements.

If the **trigger** argument is disabled (trigger = 0), the next available measurement will be returned from the current measurement cycle (see diagram below). If the **trigger** argument is enabled (trigger = 1), a measurement query will abort the current measurement and trigger a new measurement cycle. In Figure 12-1 below, <meter?> means any meter query, for example :ANLR:LEVel?

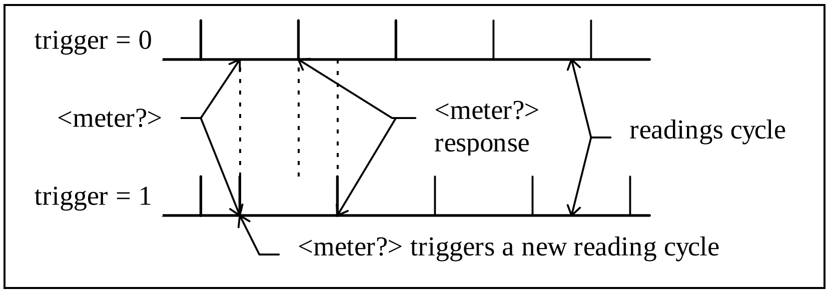
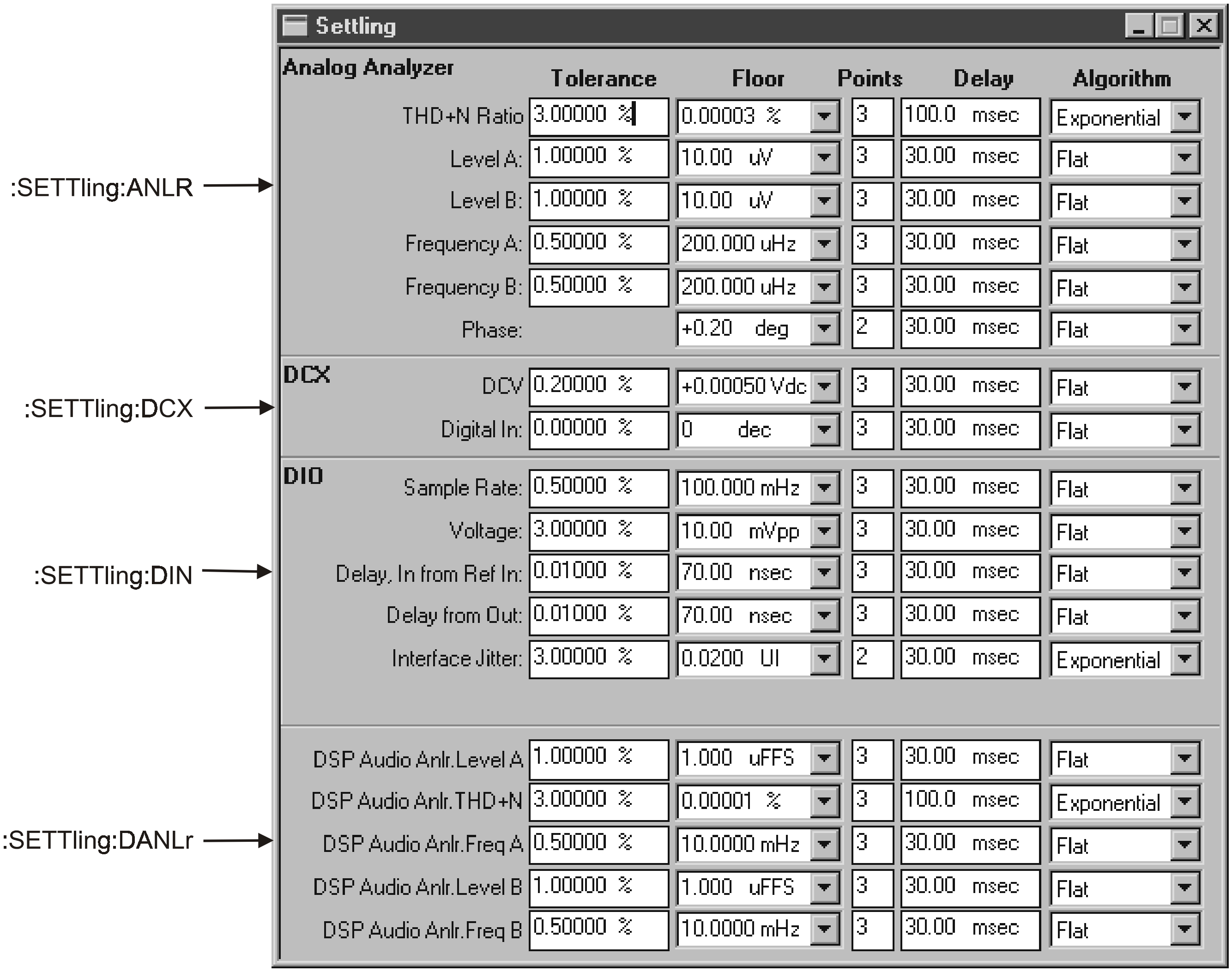


Figure 12-1. Settling illustration

See Section 2 of this manual for a more complete discussion of settling.



		Tolerance	Floor	Points	Delay	Algorithm
Analog Analyzer	THD+N Ratio	3.00000 %	0.00003 %	3	100.0 msec	Exponential
	Level A:	1.00000 %	10.00 uV	3	30.00 msec	Flat
	Level B:	1.00000 %	10.00 uV	3	30.00 msec	Flat
	Frequency A:	0.50000 %	200.000 uHz	3	30.00 msec	Flat
	Frequency B:	0.50000 %	200.000 uHz	3	30.00 msec	Flat
	Phase:		+0.20 deg	2	30.00 msec	Flat
DCX	DCV	0.20000 %	+0.00050 Vdc	3	30.00 msec	Flat
	Digital In:	0.00000 %	0 dec	3	30.00 msec	Flat
DIO	Sample Rate:	0.50000 %	100.000 mHz	3	30.00 msec	Flat
	Voltage:	3.00000 %	10.00 mVpp	3	30.00 msec	Flat
	Delay, In from Ref In:	0.01000 %	70.00 nsec	3	30.00 msec	Flat
	Delay from Out:	0.01000 %	70.00 nsec	3	30.00 msec	Flat
	Interface Jitter:	3.00000 %	0.0200 UI	2	30.00 msec	Exponential
	DSP Audio Anlr.Level A	1.00000 %	1.000 uFFS	3	30.00 msec	Flat
	DSP Audio Anlr.THD+N	3.00000 %	0.00001 %	3	100.0 msec	Exponential
	DSP Audio Anlr.Freq A	0.50000 %	10.0000 mHz	3	30.00 msec	Flat
	DSP Audio Anlr.Level B	1.00000 %	1.000 uFFS	3	30.00 msec	Flat
	DSP Audio Anlr.Freq B	0.50000 %	10.0000 mHz	3	30.00 msec	Flat

Figure 12-2. Settling Panel, showing relationship with GPIB commands

:SETTling:ANLR

Specifies the settling parameters for the indicated Analog Analyzer meters (and meter modes). There are six meters in the Analog Analyzer. The meters are: channel A Level, channel B level, channel A Frequency, channel B Frequency, Phase, and the Function meter.

The Phase meter is unique in that it is the only meter that does not use the tolerance parameter. However when a Phase meter settling command is issued, a tolerance parameter (2nd parameter) place holder must still be specified, even though the value supplied will be ignored.

The function meter has a separate set of settling parameters for each mode: Amplitude (AMPLitude), Bandpass (BP), Bandreject (BR), THD+N Amplitude (THDampl), THD+N Ratio (THDRatio), SMPTE (SMPTe), CCIF (CCIF), DIM (DIM), Wow & Flutter (WF), crosstalk (XTALk), DFD (DFD) and 2 Channel Ratio (RATio).

The following table details the valid units for the floor parameter for each meter/mode.

Meter/Mode	Units
Ch A Freq (AFReq), Ch B Freq (BFReq)	Hz (HZ), F/R (F_R), dHz (DHZ), %Hz (PCTHz), cent (CENT), octs (OCTS), decs (DECS), d% (DPCT), dPPM (DPPM)
Phase (PHASe)	degrees (DEG)
Ch A Level (ALEVel), Ch B Level (BLEVel), Amplitude (AMPL), Bandpass (BP), Bandreject (BR), THD+N Ampl (THDampl)	V (V), dBu (DBU), dBV (DBV), dBr A (DBRA), dBr B (DBRB), dBg A (DBGA), dBg B (DBGB), dBm (DBM), W (W)
THD+N Ratio (THDRatio), SMPTE/DIN (SMPTe), CCIF (CCIF), DFD (DFD), DIM (DIM), W & F (WF), 2-Ch Ratio (RATio), Crosstalk (XTALk)	% (PCT), dB (DB), PPM (PPM), X/Y (X_Y)

The command arguments are:

- **meter** - { AFReq | ALEVel | AMPLitude | BFReq | BLEVel | BP | BR |CCIF | DFD | DIM | PHASe | RATio | SMPTe | THDampl |THDRatio | WF | XTALk }
- **tolerance** - <nrf> (range: ≥ -1, ≤ 1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument parameter and unit) { CENT | DB | DBGA | DBGB | DBM | DBRA | DBRB | DBU |DBV | DECS | DEG | DHZ | DPCT | DPPM | F_R | HZ | OCTS |PCT | PCTHz | PPM | V | W | X_Y }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0.000 , ≤ 2147483.647) in units of seconds (max is approximately 24.855 days)
- **trigger** - <nr1> (range: 0 or 1)

Default: AMPLitude,3,100e-9V,3,0.03,FLAT,0,1
AFReq,0.5,250e-6HZ,2,0.02,FLAT,0,1
ALEVel,1,10e-6V,3,0.03,FLAT,0,1
BFReq,0.5,250e-6HZ,2,0.02,FLAT,0,1
BLEVel,1,10e-6V,3,0.03,FLAT,0,1
BP,5,50e-9V,3,0.03,EXP,0,1
BR,3,100e-9V,3,0.03,EXP,0,1
CCIF,5,30e-6PCT,3,0.1,EXP,0,1
DFD,5,20e-6PCT,3,0.1,EXP,0,1
DIM,3,100e-6PCT,3,0.1,EXP,0,1
PHASe,-1,0.20DEG,2,0.03,FLAT,0,1
SMPTe,3,100e-6PCT,3,0.15,EXP,0,1
THDampl,3,100e-9V,3,0.1,EXP,0,1
THDRatio,3,50e-6PCT,3,0.1,EXP,0,1
RATio,1,30e-6PCT,3,0.03,FLAT,0,1
WF,5,200e-6PCT,3,0.5,EXP,0,1
XTALk,5,30e-6PCT,3,0.03,EXP,0,1

Related Commands: :SETTling:ANLR?

Command Syntax: :SETTling:ANLR **meter, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
:SETTLING:ANLR AMPLITUDE, 3,50e-9 V,3,50e-
3,EXP,0.2,0
```

:SETTling:ANLR?

Returns the settling parameter settings for the indicated Analog Analyzer meters (and meter modes). The units argument specifies the return units for the floor argument. If the specified units are invalid for the specified mode a command error will be generated (*see* :SETTling:ANLR).

The following table details the valid units for the floor parameter for each meter/mode.

Meter/Mode	Units
Ch A Freq (AFReq), Ch B Freq (BFReq)	Hz (HZ), F/R (F_R), dHz (DHZ), %Hz (PCTHz), cent (CENT), octs (OCTS), decs (DECS), d% (DPCT), dPPM (DPPM)
Phase (PHASe)	degrees (DEG)
Ch A Level (ALEVel), Ch B Level (BLEVel), Amplitude (AMPL), Bandpass (BP), Bandreject (BR), THD+N Ampl (THDampl)	V (V), dBu (DBU), dBV (DBV), dBr A (DBRA), dBr B (DBRB), dBg A (DBGA), dBg B (DBGB), dBm (DBM), W (W)
THD+N Ratio (THDRatio), SMPTE/DIN (SMPTe), CCIF (CCIF), DFD (DFD), DIM (DIM), W & F (WF), 2-Ch Ratio (RATio), Crosstalk (XTALk)	% (PCT), dB (DB), PPM (PPM), X/Y (X_Y)

The command arguments are:

- **meter** - { AFReq | BFReq | ALEVel | AMPLitude | BLEVel | BP | BR | CCIF | DFD | DIM | PHASe | RATio | SMPTe | THDampl | THDRatio | WF | XTALk }
- **units** - { CENT | DB | DBGA | DBGB | DBM | DBRA | DBRB | DBU | DBV | DECS | DEG | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCT | PCTHz | PPM | V | W | X_Y }

Response argument(s):

- **response_meter** - { AFReq | ALEVel | AMPLitude | BFReq | BLEVel | BP | BR | CCIF | DFD | DIM | PHASe | SMPTe | THDampl | THDRatio | WF | RATio | XTALk }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument parameter and unit) { CENT | DB | DBGA | DBGB | DBM | DBRA | DBRB | DBU | DBV | DECS | DEG | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCT | PCTHz | PPM | V | W | X_Y }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds
- **trigger** - <nr1> (range: 0 or 1)

Related Commands: :SETTling:ANLR

Command Syntax: :SETTling:ANLR? **meter, units**

Response Syntax: :SETTling:ANLR **response_meter, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
:SETTLING:ANLR? AMPLITUDE, DBRA
```

Response:

```
:SETTLING:ANLR AMPLITUDE,1,0.05
DBRA,3,0.05,EXP,0.2,0
```

:SETTling:DANLr

Compound header for DSP Digital Audio Analyzer settling commands. For most of the System Two subsystems there is only one settling command (Analog Analyzer, DCX, and Digital I/O Input). The exception to this is the Digital Analyzer, which has 3 commands (and corresponding queries). The Digital Analyzer has separate settling commands (and queries) for the frequency meter, the level meter and the function meter.

CAUTION: The :SETTling:DANLr:FUNC, :SETTling:DANLr:FUNC?, :SETTling:DANLr:LEVel and :SETTling:DANLr:LEV? have an extra parameter inserted after the first parameter (compared to all other settling commands and queries). This second parameter specifies whether the meter detector is in FastRMS mode (FRMS) or in a "normal" (NORMal) mode (i.e. Q-Peak or RMS).

The following table shows the command keywords for the first parameter and second parameter (where applicable) of the Digital Analyzer settling commands. For more detail about the other parameters for these settling commands, see the specific command descriptions later in this section.

Command	Param 1 (Name)	Meter/Mode	Channel	Input	Param 2 (Detector)
:SETTling:DANLr:FREQ	CHA	*Freq*	*A*		
	CHB		*B*		
:SETTling:DANLr:FUNC	AMPA	*Ampl*		*Analog*	NORMal FRMS
	AMPD			*Digital*	NORMal FRMS
	THDA	*THD+N Ampl*		*Analog*	NORMal FRMS
	THDD			*Digital*	NORMal FRMS
	BPA	*Bandpass*		*Analog*	NORMal FRMS
	BPD			*Digital*	NORMal FRMS
	THDRatio	*THD+N Ratio*			NORMal FRMS
	RATio	*2-Chan Ratio*			NORMal FRMS
	XTALk	*Crosstalk*			NORMal FRMS
:SETTling:DANLr:LEVel	CHAA	*Level*	*A*	*Analog*	NORMal FRMS
	CHAD			*Digital*	NORMal FRMS
	CHBA		*B*	*Analog*	NORMal FRMS
	CHBD			*Digital*	NORMal FRMS

:SETTling:DANLr:FREQ

Specifies the settling parameters for the DSP Audio Analyzer frequency meter channel (A or B). The frequency meter on each channel has individual settling parameters.

Command	Param 1 (Name)	Meter	Channel
:SETTling:DANLr:FREQ	CHA	Freq	A
	CHB		B

The command arguments are:

- **channel** - { A | B }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument parameter and unit) { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds (max. is approximately 24.855 days)
- **trigger** - <nr1> (range: 0 or 1)

Default: A,0.5,0.01HZ,1,0.002,FLAT,0,1
B,0.5,0.01HZ,1,0.002,FLAT,0,1

Related Commands: :SETTling:DANLr:FREQ?, :DSP:DANLr:FREQ?

Command Syntax: :SETTling:DANLr:FREQ **channel, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
:SETTLING:DANLR:FREQ A, 1, 0.1 HZ, 3, .01,
FLAT, 0.2, 0
```

:SETTling:DANLr:FREQ?

Returns the settling parameter settings for a DSP Digital Domain Audio Analyzer frequency meter channel.

The command arguments are:

- **channel** - { A | B }
- **floorunits** - { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz }

Response argument(s):

- **channel** - { A | B }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument parameter and unit) { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds
- **trigger** - <nr1> (range: 0 or 1)

Related Commands: :SETTling:DANLr:FREQ

Command Syntax: :SETTling:DANLr:FREQ? **channel, floorunits**

Response Syntax: :SETTling:DANLr:FREQ **channel, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example: `:SETTLING:DANLR:FREQ? A,HZ`

Response: `:SETTLING:DANLR:FREQA,1,0.1HZ,3,0.01,FLAT,0.2,0`

:SETTling:DANLr:FUNC

Specifies the settling parameters for the DSP Audio Analyzer Program function meter. Settling parameter settings can be stored based on specific combinations of function meter mode, input domain, and detector type (see table below). Separate settling parameters are stored (and used) for each row in the table below.

The choices for the first parameter require valid combinations of function meter mode and input domain. These choices are:

- AMPA (amplitude mode with analog domain input),
- AMPD (amplitude mode with digital domain input),
- BPA (bandpass mode with analog input),
- BPD (bandpass mode with digital input),
- THDA (THD+N amplitude mode with analog input),
- THDD (THD+N amplitude mode with digital input),

- THDRatio (THD+N Ratio mode, same settings are used for analog and digital input),
- RATio (2-channel Ratio mode, same settings are used for analog and digital input),
- and XTALk (Crosstalk mode, same settings are used for analog and digital input).

The choices for the second parameter indicate the detector type. These choices are NORMal (when the detector is either Q-Peak or RMS) and FRMS (when the detector is in FastRMS mode). Different settling parameter settings are stored (and used) based on all combinations of the first and second parameters.

For example, one set of settling parameters could be specified for amplitude mode with analog domain input using the FastRMS detector mode. A different settling parameter setting could be specified (and stored) for amplitude mode with analog domain input using a normal detector mode. Yet, another set of settling parameters could be specified for amplitude mode with digital input using the FastRMS detector.

Param 1 Name	***Meter Mode***	***Input***	**Param 2 (Detector)**
AMPA	*Ampl*	*Analog*	NORMal
			FRMS
AMPD		*Digital*	NORMal
			FRMS
BPA	*Bandpass*	*Analog*	NORMal
			FRMS
BPD		*Digital*	NORMal
			FRMS
THDA	*THD+N Ampl*	*Analog*	NORMal
			FRMS
THDD		*Digital*	NORMal
			FRMS
THDRatio	*THD+N Ratio*		NORMal
			FRMS
RATio	*2-Chan Ratio*		NORMal
			FRMS
XTALk	*Crosstalk*		NORMal
			FRMS

The command arguments are:

- **meter** - { AMPA | AMPD | BPA | BPD | RATio | THDA | THDD | THDRatio | XTALk }
- **detector** - { FRMS | NORMal }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument parameter and unit) { DB | DBFS | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCT | PCTFs | V | VP | VPP | X_Y }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds (max. is approximately 24.855 days)
- **trigger** - <nr1> (range: 0 or 1)

Default:
AMPA,FRMS,1,1e-6 V,1,0.001,FLAT,0,1
AMPA,NORM,1,1e-6V,3,0.03,FLAT,0,1
AMPD,FRMS,1,1e-7FFS,1,0.001,FLAT,0,1
AMPD,NORM,1,1e-6FFS,3,0.03,FLAT,0,1
BPA,FRMS,3,1e-8V,2,0.02,EXP,0,1
BPA,NORM,3,1e-8V,3,0.1,EXP,0,1
BPD,FRMS,3,1e-8FFS,2,0.02,EXP,0,1
BPD,NORM,3,1e-8FFS,3,0.1,EXP,0,1
THDA,FRMS,3,1e-7V,2,0.02,EXP,0,1
THDA,NORM,3,1e-7V,3,0.1,EXP,0,1
THDD,FRMS,3,1e-7FFS,2,0.02,FLAT,0,1
THDD,NORM,3,1e-7FFS,3,0.1,EXP,0,1
THDRatio,NORM,3,10e-6PCT,3,0.1,EXP,0,1
THDRatio,FRMS,3,10e-6PCT,2,0.02,FLAT,0,1
RATio,NORM,1,100e-6PCT,3,0.03,FLAT,0,1
RATio,FRMS,1,100e-6PCT,1,0.001,FLAT,0,1
XTALk,NORM,3,10e-6PCT,3,0.1,EXP,0,1
XTALk,FRMS,3,10e-6PCT,2,0.02,EXP,0,1

Related Commands: :SETTling:DANLr:FUNC?

Command Syntax: :SETTling:DANLr:FUNC **meter, detector, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
:SETTLING:DANLR:FUNC BPD,NORMAL,3,10e-
9FFS,3,0.1,EXP,0,1
```

:SETTling:DANLr:FUNC?

Returns the settling parameter settings for the individual DSP Digital Domain Audio Analyzer meters and the Reading Meter modes. This query has three (3) parameters: meter mode or input domain, detector mode, and floor parameter units.

The following table details the valid units for the floor parameter for each meter/mode.

Meter/Mode	Units
Analog Input: Amplitude (AMPA) , Bandpass (BPA), THD+N Amplitude (THDA)	V (V), dBV (DBV), dBu (DBU), dBr A (DBRA), dBr B (DBRB), dBm (DBM)
Digital Input: Amplitude (AMPD) , Bandpass (BPD), THD+N Amplitude (THDD)	FFS (FFS), %FS (PCTFs), dBFS (DBFS), V (V), Vp (VP), Vpp (VPP), dBu (DBU), dBV (DBV), dBr 1 (DBR1), dBr 2 (DBR2)
2-Ch Ratio (RATio), Crosstalk (XTALk), THD+N Ratio (THDRatio)	dB (DB), % (PCT), X/Y (X_Y)

The command arguments are:

- **meter** - { AMPA | AMPD | BPA | BPD | RATio | THDA | THDD | THDRatio | XTALk }
- **detector** - { FRMS | NORMal }
- **units** - { DB | DBFS | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCT | PCTFs | V | VP | VPP | X_Y }

Response argument(s):

- **response_meter** - { AMPA | AMPD | BPA | BPD | RATio | THDA | THDD | THDRatio | XTALk }
- **detector** - { FRMS | NORMal }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument parameter and unit) { DB | DBFS | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCT | PCTFs | V | VP | VPP | X_Y }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds
- **trigger** - <nr1> (range: 0 or 1)

Related Commands: :SETTling:DANLr:FUNC

Command Syntax: :SETTling:DANLr:FUNC? **meter, detector, units**

Response Syntax: :SETTling:DANLr:FUNC **response_meter, detector, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example: `:SETTLING:DANLR:FUNC? BPD,NORMAL,FFS`

Response : `:SETTLING:DANLR:FUNC BPD,NORMAL,5,10e-9FFS,3,0.1,EXP,0,1`

:SETTling:DANLr:LEVel

Specifies the settling parameters for the DSP Audio Analyzer Program level meter. There is one level meter for each channel (A and B). Each level meter has individual settling parameters based on selection of channel and detector. This settling command has an extra parameter. This parameter (second in list), specifies the type of detector, either NORMal (Q-Peak or RMS) or FRMS (FastRMS).

Param 1 (Name)	*Channel*	*Input*	Param 2 (Detector)
CHAA	*A*	*Analog*	NORMal
			FRMS
CHAD		*Digital*	NORMal
			FRMS
CHBA	*B*	*Analog*	NORMal
			FRMS
CHBD		*Digital*	NORMal
			FRMS

The following table details the valid units for the floor parameter for each level meter and input combination (the detector does not affect the floor parameter units).

Meter/Channel	Units
Analog Input: Ch A Level (CHAA), Ch B Level (CHBA)	V (V), dBV (DBV), dBu (DBU), dBr A (DBRA), dBr B (DBRB), dBm (DBM)
Digital Input: Ch A Level (CHAD), Ch B Level (CHBD)	FFS (FFS), %FS (PCTFs), dBFS (DBFS), V (V), Vp (VP), Vpp (VPP), dBu (DBU), dBV (DBV), dBr 1 (DBR1), dBr 2 (DBR2)

The command arguments are:

- **channel** - { CHAA | CHAD | CHBA | CHBD }
- **detector** - { FRMS | NORMal }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument parameter and unit) { DB | DBFS | DBM | DBRA | DBRB | DBR1 | DBR2 | DBU | DBV | FFS | PCT | PCTFs | V | VP | VPP | X_Y }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { NONE | EXP | FLAT | AVG }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds (max. is approximately 24.855 days)
- **trigger** - <nr1> (range: 0 or 1)

Default: CHAA,FAST,1,1e-6V,1,0.001,FLAT,0,1
CHAA,NORM,1,1e-6V,3,0.03,FLAT,0,1
CHAD,FAST,1,1e-7FFS,1,0.001,FLAT,0,1
CHAD,NORM,1,1e-6FFS,3,0.03,FLAT,0,1
CHBA,FAST,1,1e-6V,1,0.001,FLAT,0,1
CHBA,NORM,1,1e-6V,3,0.03,FLAT,0,1
CHBD,FAST1,1e-7FFS,1,0.001,FLAT,0,1
CHBD,NORM1,1e-6FFS,3,0.03,FLAT,0,1

Related Commands: :SETTling:DANLr:LEVel?, :DSP:DANLr...

Command Syntax: :SETTling:DANLr:LEVel **channel, detector, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
 :SETTLING:DANLR:LEVEL CHBD,NORM,1,0.1V,3,.01,FL
AT,0.2,1
```

:SETTling:DANLr:LEVel?

Returns the settling parameter settings for the DSP Digital Domain Audio Analyzer channel level meters.

The following table details the valid units for the floor parameter for each meter/mode.

Meter/Channel	Units
Analog Input: Ch A Level (CHAA), Ch B Level (CHBA)	V (V), dBV (DBV), dBu (DBU), dBr A (DBRA), dBr B (DBRB), dBm (DBM)
Digital Input: Ch A Level (CHAD), Ch B Level (CHBD)	FFS (FFS), %FS (PCTFs), dBFS (DBFS), V (V), Vp (VP), Vpp (VPP), dBu (DBU), dBV (DBV), dBr 1 (DBR1), dBr 2 (DBR2)

The command arguments are:

- **channel** - { CHAA | CHAD | CHBA | CHBD }
- **detector** - { FRMS | NORMal }
- **units** - { DB | DBFS | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCT | PCTFs | V | VP | VPP | X_Y }

Response argument(s):

- **channel** - { CHAA | CHAD | CHBA | CHBD }
- **detector** - { FRMS | NORMal }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument parameter and unit) { DB | DBFS | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCT | PCTFs | V | VP | VPP | X_Y }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds
- **trigger** - <nr1> (range: 0 or 1)

Related Commands: :SETTling:DANLr:LEVel

Command Syntax: :SETTling:DANLr:LEVel? **channel, detector, units**

Response Syntax: :SETTling:DANLr:LEVel **channel, detector, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
:SETTLING:DANLR:LEVEL? CHBD,NORM,V
```

Response:

```
:SETTLING:DANLR:LEVEL CHBD,NORM,1,0.1V,3,0.01,
FLAT,0.2,1
```

:SETTling:DCX

Specifies the settling parameter settings for the DCX meters.

The following table details the valid units for the floor parameter:

Meter/Mode	Units
Digital Input	Dec (**DEC**), Scaled Dec (**G_X**)
Ohms Meter	Ohms (**O**), Scaled Ohms (**F_O**)
Volt Meter	Vdc (**VDC**), Scaled Vdc (**F_V**)

The command arguments are:

- **meter** - { DIN | OHMS | VOLTs }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: = 0 , ≤ 1E34) { DEC | F_O | F_V | G_X | O | V }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds
- **trigger** - <nr1> (range: 0 or 1)

Default: VOLTs,0.2,5*e*-4V,3,0.03,FLAT,0,1
OHMS,0.5,0.1O,3,0.03,FLAT,0,1

DIN,0,0DEC,3,0.03,FLAT,0,1

Related Commands: :SETTling:DCX?

Command Syntax: :SETTling:DCX **meter, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
:SETTLING:DCX DIN,5,0DEC,3,0.1,EXP,0.2,0
```

:SETTling:DCX?

Returns the settling parameter settings for the DCX digital input measurement.

The following table details the valid units for the floor parameter:

Meter/Mode	Units
Digital Input (DIN)	Dec (**DEC**), Scaled Dec (**G_X**)
Ohms Meter (OHMS)	Ohms (**O**), Scaled Ohms (**F_O**)
Volt Meter (VOLTs)	Vdc (**VDC**), Scaled Vdc (**F_V**)

The command arguments are:

- **meter** - { DIN | OHMS | VOLTs }
- **units** - { DEC | F_O | F_V | G_X | O | V }

Response argument(s):

- **response_meter** - { DIN | OHMS | VOLTs }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in units of percent
- **floor** - <nrf> (range: = 0 , ≤ 1E34) { DEC | F_O | F_V | G_X | O | V }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in units of seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0,. = 2147483.647) in units of seconds
- **trigger** - <nr1> (range: 0 or 1)

Related Commands: :SETTling:DCX

Command Syntax: :SETTling:DCX? **meter, units**

Response Syntax: :SETTling:DCX **response_meter, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example: `:SETTLING:DCX? DIN,DEC`

Response: `:SETTLING:DCX DIN,5,0DEC,3,0.1,EXP,0.2,0`

:SETTling:DIN

Specifies the settling parameter settings for the Digital Input meters.

The amplitude measurement is a peak-to-peak signal amplitude measurement at the front panel XLR or BNC connector. These settling parameters do not apply when optical, rear panel serial or rear panel parallel connectors are selected, because the amplitude cannot be measured at those inputs.

The following units apply to the available meters:

Meter	Units
Amplitude (AMPL)	Vpp
Interface Jitter (JITTer)	Seconds
Delay from Output (OUTDelay)	Seconds
Delay, In From Ref In (INDelay)	Seconds
Sample Rate (RATE)	Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM

The command arguments are:

- **meter** - { AMPLitude | IDELay | JITTer | OUTDelay | RATE }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in percent
- **floor** - <nrf> (range: ≥ 0 , ≤ 1E34) { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz | SEC | UI | VPP }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds (max is approximately 24.855 days)
- **trigger** - <nr1> (range: 0 or 1)

Default: AMPLitude,3,0.01VPP,3,0.03,FLAT,0,1
IDELay,0.01,70e-9SEC,3,0.03,FLAT,0,1
JITTer,3,3e-9SEC,3,0.1,EXP,0,1
OUTDelay,0.01,70e-9SEC,3,0.03,FLAT,0,1
RATE,0.5,0.1HZ,3,0.03,FLAT.0,1

Related Commands: :DIN:AMPL?, :SETTling:ANLR?

Command Syntax: :SETTling:DIN **meter, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
:SETTLING:DIN AMPLITUDE,5,.001VPP,3,.01,EXP,0.2,0
```

:SETTling:DIN?

Returns the settling parameter settings for the Digital I/O amplitude (Voltage) measurement.

The following floor units apply to the available meters:

Meter	Units
Amplitude (AMPL)	Vpp
Interface Jitter (JITTer)	Seconds
Delay from Output (OUTDelay)	Seconds
Delay, In From Ref In (INDelay)	Seconds
Sample Rate (RATE)	Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM

The command arguments are:

- **meter** - { AMPLitude | IDELay | JITTer | OUTDelay | RATE }
- **floorunits** - { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz | SEC | VPP }

Response argument(s):

- **response_meter** - { AMPLitude | IDELay | JITTer | OUTDelay | RATE }
- **tolerance** - <nrf> (range: ≥ 0 , ≤ 1E34) in percent
- **floor** - <nrf> (range: ≥ 0 , ≤ 1E34) { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz | SEC | VPP }
- **points** - <nr1> (range: ≥ 1 , ≤ 32)
- **delay** - <nrf> (range: ≥ 0 , ≤ 15) in seconds
- **algorithm** - { AVG | EXP | FLAT | NONE }
- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds
- **trigger** - <nr1> (range: 0 or 1)

Related Commands:

Command Syntax: :SETTling:DIN? **meter,floorunits**

Response Syntax: :SETTling:DIN **response_meter, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:

```
:SETTLING:DIN? AMPLITUDE
```

Response:

```
 :SETTLING:DIN AMPLITUDE,5,0.001VPP,3,0.01,EXP,0
.2,0
```

:SETTling:SET?

Returns all settling parameter settings. The response consists of a sequence of the settling command settings that may be sent back to the instrument at a later time in order to reset all settling command settings to the same state. Note that command settings will not be returned for hardware that is not installed.

Response argument(s):

- **settings** - <response message unit> [<response message unit>] ...

Related Commands: <see all other settling queries>

Command Syntax: :SETTling:SET?

Response Syntax: :SETTling:SET **settings**

Example: `:SETTLING:SET?`

Response:

```
:SETTLING:ANLR AFREQ,0.5,0.00025HZ,2,0.02,FLAT,
0,1;ANLR BFREQ,0.5,0.00025HZ,2,0.02,FLAT,0,1;AN
LR PHASE,-1,0.2DEG,2,0.03,FLAT,0,1;ANLR ALEVEL,
1,1E-05V,3,0.03,FLAT,0,1;ANLR BLEVEL,1,1E-05V,3
,0.03,FLAT,0,1;ANLR AMPLITUDE,3,1E-07V,3,0.03,F
LAT,0,1;ANLR BP,5,5E-08V,3,0.03,EXP,0,1;ANLR BR
,3,1E-07V,3,0.03,EXP,0,1;ANLR THDAMPL,3,1E-07V,
3,0.1,EXP,0,1;ANLR THDRATIO,3,5E-05PCT,3,0.1,EX
P,0,1;ANLR SMPTE,3,0.0001PCT,3,0.15,EXP,0,1;ANL
R CCIF,5,3E-05PCT,3,0.1,EXP,0,1;ANLR DFD,5,2E-0
5PCT,3,0.1,EXP,0,1;ANLR DIM,3,0.0001PCT,3,0.1,E
XP,0,1;ANLR WF,5,0.0002PCT,3,0.5,EXP,0,1;ANLR R
ATIO,1,3E-05PCT,3,0.03,FLAT,0,1;ANLR XTALK,5,3E
-05PCT,3,0.03,EXP,0,1;DCX DIN,0,0DEC,3,0.03,FLA
T,0,1;DCX OHMS,0.5,0.1O,3,0.03,FLAT,0,1;DCX VOL
TS,0.2,0.0005V,3,0.03,FLAT,0,1;DIN AMPLITUDE,3,
0.01VPP,3,0.03,FLAT,0,1;DIN IDELAY,0.01,7E-08SE
C,3,0.03,FLAT,0,1;DIN JITTER,3,3E-09SEC,3,0.1,E
XP,0,1;DIN OUTDELAY,0.01,7E-08SEC,3,0.03,FLAT,0
,1;DIN RATE,0.5,0.1HZ,3,0.03,FLAT,0,1;DANLR:FRE
Q A,0.5,0.01HZ,1,0.002,FLAT,0,1;FREQ B,0.5,0.01
HZ,1,0.002,FLAT,0,1;FUNC AMPD,NORMAL,1,1E-06FFS
,3,0.03,FLAT,0,1;FUNC AMPA,NORMAL,1,1E-06V,3,0.
03,FLAT,0,1;FUNC AMPD,FRMS,1,1E-07FFS,1,0.001,F
LAT,0,1;FUNC AMPA,FRMS,1,1E-06V,1,0.001,FLAT,0,
1;FUNC BPD,NORMAL,3,1E-08FFS,3,0.1,EXP,0,1;FUNC
 BPA,NORMAL,3,1E-08V,3,0.1,EXP,0,1;FUNC BPD,FRM
S,3,1E-08FFS,2,0.02,EXP,0,1;FUNC BPA,FRMS,3,1E-
08V,2,0.02,EXP,0,1;FUNC RATIO,NORMAL,1,0.0001PC
T,3,0.03,FLAT,0,1;FUNC RATIO,FRMS,1,0.0001PCT,1
,0.001,FLAT,0,1;FUNC XTALK,NORMAL,3,1E-05PCT,3,
0.1,EXP,0,1;FUNC XTALK,FRMS,3,1E-05PCT,2,0.02,E
XP,0,1;FUNC THDD,NORMAL,3,1E-07FFS,3,0.1,EXP,0,
1;FUNC THDA,NORMAL,3,1E-07V,3,0.1,EXP,0,1;FUNC
THDD,FRMS,3,1E-07FFS,2,0.02,FLAT,0,1;FUNC THDA,
FRMS,3,1E-07V,2,0.02,EXP,0,1;FUNC THDRATIO,NORM
AL,3,0.0001PCT,3,0.1,EXP,0,1;FUNC THDRATIO,FRMS
,3,1E-05PCT,2,0.02,FLAT,0,1;LEVEL CHAD,NORMAL,1
```

```
,1E-06FFS,3,0.03,FLAT,0,1;LEVEL CHAA,NORMAL,1,1
E-06V,3,0.03,FLAT,0,1;LEVEL CHAD,FRMS,1,1E-07FF
S,1,0.001,FLAT,0,1;LEVEL CHAA,FRMS,1,1E-06V,1,0
.001,FLAT,0,1;LEVEL CHBD,NORMAL,1,1E-06FFS,3,0.
03,FLAT,0,1;LEVEL CHBA,NORMAL,1,1E-06V,3,0.03,F
LAT,0,1;LEVEL CHBD,FRMS,1,1E-07FFS,1,0.001,FLAT
,0,1;LEVEL CHBA,FRMS,1,1E-06V,1,0.001,FLAT,0,1;
:SETTLING:TIMEOUT 4
```

:SETTling:TIMeout

Specifies the global settling timeout. The global settling timeout is used for any measurements whose individual settling timeout is set to zero (0) seconds.

The command arguments are:

- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds

Default: 4 seconds

Related Commands: :SETTling:TIMeout?, all measurements with settling parameters

Command Syntax: :SETTling:TIMeout **timeout**

Example: `:SETTLING:TIMEOUT 0`

:SETTling:TIMeout?

Returns the global settling timeout. The global settling timeout is used for any measurements whose individual timeout is set to zero (0) seconds.

Response argument(s):

- **timeout** - <nrf> (range: ≥ 0 , ≤ 2147483.647) in units of seconds

Related Commands: :SETTling:TIMeout

Command Syntax: :SETTling:TIMeout?

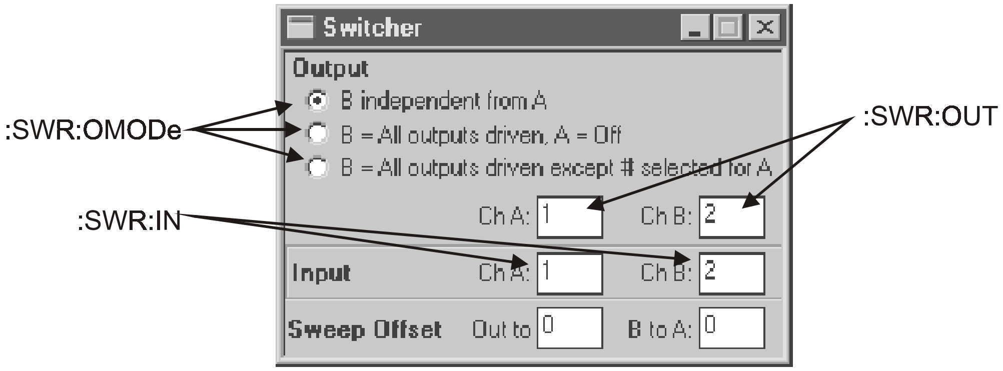
Response Syntax: :SETTling:TIMeout **timeout**

Example: `:SETTLING:TIMEOUT?`

Response: `:SETTLING:TIMEOUT 0`

13. SWR-2122 Commands

The header-path for SWR-2122 commands is :SWR:.



Output

● B independent from A

○ B = All outputs driven, A = Off

○ B = All outputs driven except # selected for A

Ch A: 1 Ch B: 2

Input Ch A: 1 Ch B: 2

Sweep Offset Out to 0 B to A: 0

Figure 13-1. Switcher panel, showing relationship with GPIB commands

:SWR:IN

This command specifies the input switch (SWR-2122F) to close for the indicated channel (A or B). The range of values is 0 to 192 (for a maximum of 16 switchers). Zero (0) indicates that all switches for that channel are open (no connection). Specifying a nonexistent switch will NOT cause an error.

The command arguments are:

- **channel** - { A | B }
- **switch** - <nr1> (range: ≥ 0 , ≤ 192)

Default: A, 0; B, 0

Related Commands: :SWR:IN?

Command Syntax: :SWR:IN **channel, switch**

Example: `:SWR:IN A,5`

:SWR:IN?

This query returns the closed input switch for the indicated channel.

The command arguments are:

- **channel** - { A | B }

Response argument(s):

- **response_channel** - { A | B }
- **switch** - <nr1> (range: ≥ 0, ≤ 192)

Related Commands: :SWR:IN

Command Syntax: :SWR:IN? **channel**

Response Syntax: :SWR:IN **channel, switch**

Example: `:SWR:IN? A`

Response: `:SWR:IN A,5`

:SWR:OMODe

The command specifies the mode of the output switcher(s). In INDependent mode, channel A and B switches are set regardless of the state of the other channel. In COMPLement mode, all switches are connect to channel B except the one specified to be connected to channel A. In ALLB mode, all output switches are connected to channel B.

The command argument is:

- **mode** - { INDependent | ALLB | COMPlement }

Default: INDependent

Related Commands: :SWR:OMODe?

Command Syntax: :SWR:OMODe **mode**

Example: `:SWR:OMODE INDEPENDENT`

:SWR:OMODe?

The query returns the current mode of the output switcher(s).

Response argument(s):

- **mode** - { INDependent | ALLB | COMPlement }

Related Commands: :SWR:OMODe

Command Syntax: :SWR:OMODe?

Response Syntax: :SWR:OMODe **mode**

Example: `:SWR:OMODE?`

Response: `:SWR:OMODE INDEPENDENT`

:SWR:OUT

This command specifies which output switch (SWR-2122M) to close. The range of values is 0 to 192 (with a maximum of 16 switchers). Zero indicates that all switches are open (no connection) for that channel. Specifying a nonexistent switch will NOT cause an error. Offset settings do not apply to this command.

If the output mode is ALLB this command will cause an error. If the output mode is COMPLement, only the setting of the B channel will cause an error.

The command arguments are:

- **channel** - { A | B }
- **switch** - <nr1> (range: ≥ 0, ≤ 192)

Default: A, 0; B, 0

Related Commands: :SWR:OUT?

Command Syntax: :SWR:OUT **channel, switch**

Example: `:SWR:OUT 0,8`

:SWR:OUT?

This query returns the output switch closed for the specified channel. If the output mode is ALLB this query will cause an error. If the output mode is COMPLement only the query of the B channel will cause an error.

The command arguments are:

- **channel** - { A | B }

Response argument(s):

- **response_channel** - { A | B }
- **switch** - <nr1> (range: ≥ 0, ≤ 192)

Related Commands: :SWR:OUT

Command Syntax: :SWR:OUT? **channel**

Response Syntax: :SWR:OUT **response_channel, switch**

Example: `:SWR:OUT? A`

Response: `:SWR:OUT A,8`

:SWR:SET?

This query returns the current state of all switcher settings if one or more switcher modules is connected to the System Two. An error will be generated if no switchers are connected.

If the output mode is ALLB there will be no OUT A or OUT B setting in the response.

If the output mode is COMPlement there will be no OUT B setting in the response.

Response argument(s):

- **settings** - <response message unit> [<response message unit>] ...

Related Commands:

Command Syntax: :SWR:SET?

Response Syntax: :SWR:SET **settings**

Example:

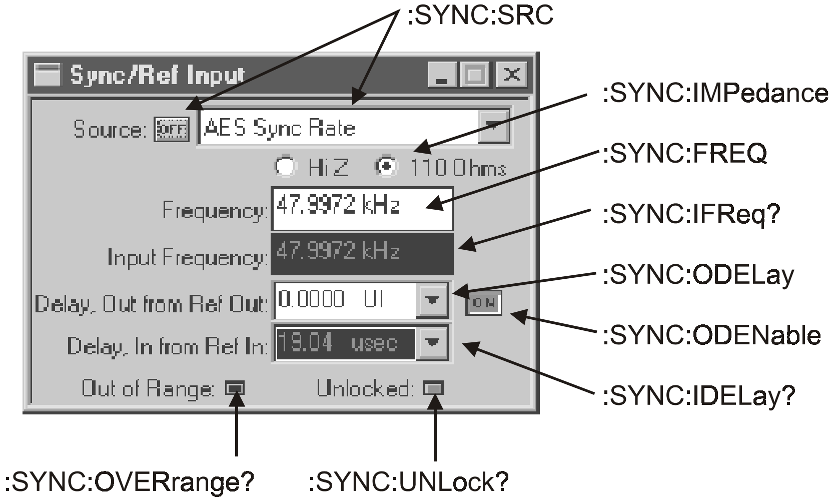
```
:SWR:SET?
```

Response:

```
:SWR:OMODE INDEPENDENT;IN A,0;IN B,0;
OFFSET CHB,0;OFFSET OUT,0;OUT A,0;OUT B,0
```

14. Sync/Ref Commands

The header-path for the Sync/Reference commands is :SYNC:.



Sync/Ref Input

Source: OFF AES Sync Rate

○ Hi Z ● 110 Ohms

Frequency: 47.9972 kHz

Input Frequency: 47.9972 kHz

Delay, Out from Ref Out: 0.0000 UI ON

Delay, In from Ref In: 19.04 usec

Out of Range: □ Unlocked: □

Figure 14-1. Sync/Ref Input panel showing relationship with GPIB commands

:SYNC:ENABle

This command enables or disables the synchronization of the internal sync clock with the external input selected by the ":SYNC:SRC" command.

The command argument is:

- **state** - { OFF | ON }

Default: OFF

Related Commands: :SYNC:ENABle?

Command Syntax: :SYNC:ENABle **state**

Example: `:SYNC:ENABLE ON`

:SYNC:ENABle?

This query returns the state of the sync enable command.

Response argument(s):

- **state** - { OFF | ON }

Related Commands: :SYNC:ENABle

Command Syntax: :SYNC:ENABle?

Response Syntax: :SYNC:ENABle **state**

Example: `:SYNC:ENABLE?`

Response: `:SYNC:ENABLE ON`

:SYNC:FREQ

This command specifies the exact expected frequency of the external Sync Input signal. This frequency sets the internal sample rate clock phase-lock loop frequency to the expected external sync signal frequency in order to optimize a lock to that signal. The System Two internal sample rate clock will not lock to the external sync input signal if the frequency specified differs by more than 15 ppm from the actual input frequency.

The command argument is:

- **frequency** - <nrf> (range: ≥ 8000, ≤ 1.5E+7 depends on :SYNC:SRC selection)

Default: 48000.0

Related Commands: :SYNC:FREQ?

Command Syntax: :SYNC:FREQ **frequency**

Example: `:SYNC:FREQ 48000`

:SYNC:FREQ?

This query returns the actual Reference Frequency.

Response argument(s):

- **frequency** - <nrf> (range: ≥ 8000, ≤ 1.5E+7; depends on :SYNC:SRC selection)

Related Commands: :SYNC:FREQ

Command Syntax: :SYNC:FREQ?

Response Syntax: :SYNC:FREQ **frequency**

Example: `:SYNC:FREQ?`

Response: `:SYNC:FREQ 48000`

:SYNC:IDELay?

This query returns the time delay (phase) of the selected front panel XLR, BNC, or optical connector (:DIN:FORMat) with respect to the selected rear panel AES/EBU Reference (sync) input signal (:SYNC:SRC). The response units is seconds.

The first parameter in the response is the next available measurement.

The Sync/Ref Delay-In-From-Ref-In measurement settling is enabled or disabled by the algorithm parameter of the :SETTling:DIN command. Settling is enabled by default (power-on, *RST, or *RCL 0).

If settling is disabled then the last parameter will be a 0.

If settling is enabled then the last parameter in the response indicates the settling status. A zero response (0) indicates that the measurement settled and did not time out. In this case, the first parameter in the response will contain the most recent measurement in the settling buffer.

A one (1) in the last parameter indicates a settling timeout. In this case the first parameter in the response will be the average of the readings in the settling buffer. The number of readings in the settling buffer is indicated by the number of points specified in the settling command for this measurement function.

Response argument(s):

- **delay** - <nrf>
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: <None>

Command Syntax: :SYNC:IDELay?

Response Syntax: :SYNC:IDELay delay, settle_timeout

Example: `:SYNC:IDELAY?`

Response: `:SYNC:IDELAY 0.5,0`

:SYNC:IFReq?

This query returns the measured frequency (Hz) of the signal selected in the source (:SYNC:SRC) field is OFF. If source is not OFF this query will return 9.91E+37, the Not-A-Number number. The response unit is hertz. This measurement does not use the settling algorithm.

Response argument(s):

- **freq** - <nrf>

Related Commands: <None>

Command Syntax: :SYNC:IFReq?

Response Syntax:: SYNC:IFReq? **freq**

Example: `:SYNC:IFREQ?`

Response: `:SYNC:IFREQ 15534.75`

:SYNC:IMPedance

This command sets the input impedance of the rear panel sync/ref inputs (XLR and BNC connectors). The ZLO argument has two potential impedance values depending on the source setting (:SYNC:SRC) shown in the table below.

Source	ZLO Impedance
AES Sync Rate	110 Ohms
Square Wave, NTSC Video, PAL/SECAM Video	75 Ohms

The command argument is:

- **impedance** - { ZLO | ZHI }

Default: ZLO

Related Commands: :SYNC:IMPedance?

Command Syntax: :SYNC:IMPedance **impedance**

Example: `:SYNC:IMPEDANCE ZHI`

:SYNC:IMPedance?

This query returns the current impedance setting of the rear panel sync/ref inputs.

Response argument(s):

- **impedance** - { ZLO | ZHI }

Related Commands: :SYNC:IMPedance

Command Syntax: :SYNC:IMPedance?

Response Syntax: :SYNC:IMPedance **impedance**

Example: `:SYNC:IMPEDANCE?`

Response: `:SYNC:IMPEDANCE ZHI`

:SYNC:ODELay

This command sets the time delay of the digital generator (front panel) output relative to the rear panel AES/EBU REF OUT XLR connector. This parameter is used when:SYNC:ODENable is ON but is ignored when :SYNC:ODENable is OFF.

The command argument is:

- **time** - <nrf> (range: ≥ -64, ≤ 63.5) { UI | SEC }

Default: 0 UI

Related Commands: :SYNC:ODENable, :SYNC:ODELay?

Command Syntax: :SYNC:ODELay **time**

Example: `:SYNC:ODELAY 3.5UI`

:SYNC:ODELay?

This query returns the current output delay setting.

Response argument(s):

- **time** - <nrf> (range: ≥ -64, ≤ 63.5) { UI | SEC }

Related Commands: :SYNC:ODELay

Command Syntax: :SYNC:ODELay?

Response Syntax: :SYNC:ODELay **time**

Example: `:SYNC:ODELAY?`

Response: `:SYNC:ODELAY 3.5UI`

:SYNC:ODENable

This command enables or disables the Out From Ref Out Delay. When enabled the delay used will be the most recent specified by ":SYNC:ODELay" or the default (0.000 UI).

The command argument is:

- **state** - { OFF | ON }

Default: OFF

Related Commands: :SYNC:ODENable?

Command Syntax: :SYNC:ODENable **state**

Example: `:SYNC:ODENABLE ON`

:SYNC:ODENable?

This query returns the current setting of the Out From Ref Out Delay command, :SYNC:ODENable.

Response argument(s):

- **state** - { OFF | ON }

Related Commands: :SYNC:ODENable

Command Syntax: :SYNC:ODENable?

Response Syntax: :SYNC:ODENable **state**

Example: `:SYNC:ODENABLE?`

Response: `:SYNC:ODENABLE ON`

:SYNC:OVERrange?

This query returns the state of the SYNC frequency over-range flag. The flag is set (1=ON) if the actual Sync Input signal frequency deviates more than +/-15 ppm from the Sync Input Frequency setting of the phase lock loop circuit (see :SYNC:IFReq), or outside the amplitude range required for reliable operation. The possible responses are 1 (ON) and 0 (OFF).

Response argument(s):

- **state** - <nr1> { range: 0 or 1 }

Related Commands: <None>

Command Syntax: :SYNC:OVERrange?

Response Syntax: :SYNC:OVERrange **state**

Example: `:SYNC:OVERRANGE?`

Response: `:SYNC:OVERRANGE 1`

:SYNC:SET?

This query returns the SYNC settings. The response consists of a sequence of the SYNC command settings that may be sent back to the instrument at a later time in order to reset all settings to the same state.

Response argument(s):

- **settings** - <response message unit> [<response message unit>] ...

Related Commands: (see other :SYNC queries)

Command Syntax: :SYNC:SET?

Response Syntax: :SYNC:SET **settings**

Example: `:SYNC:SET?`

Response:

```
:SYNC:ENABLE OFF;SRC AESEBU;ODENABLE OFF;IMP
EDANCE ZLO;FREQ 48000;ODELAY 0UI
```

:SYNC:SRC

This command specifies the input for synchronizing the internal crystal oscillator to an external reference signal. The input choices are shown in the table below. One of two input connectors may be selected, XLR AES/EBU Ref In connector, or the Video/TTL Ref In BNC. The BNC input connector signal may be used with three different types of signals shown in the table. Refer to the APWIN documentation for more information.

Signal	Connector Type: Label
AES-EBU Sync Rate	XLR: AESEBU Ref In
Square wave	BNC: Video/TTL Ref In
NTSC Video Sync Horiz Rate	BNC: Video/TTL Ref In
PAL/SECAM Video Sync Horiz Rate	BNC: Video/TTL Ref In

The command argument is:

- **signal** - { AESebu | NTSC | PAL | SQUare }

Default: AESebu

Related Commands: :SYNC:SRC?, :SYNC:ENABle

Command Syntax: :SYNC:SRC **signal**

Example: `:SYNC:SRC AESEBU`

:SYNC:SRC?

This query returns the current setting for the sync input signal source.

Response argument(s):

- **signal** - { AESebu | NTSC | PAL | SQUare }

Related Commands: :SYNC:SRC, :SYNC:ENABle

Command Syntax: :SYNC:SRC?

Response Syntax: :SYNC:SRC **signal**

Example: `:SYNC:SRC?`

Response: `:SYNC:SRC AESEBU`

:SYNC:UNLock?

This query returns the Unlocked status for the external sync input. The response is 1 (1=ON) if the internal phase-locked loop is not locked to the external sync input signal. When an AES/EBU signal is used as the sync input, the Unlocked status will also be 1 if a confidence, parity, or coding error is sensed on the reference signal. This response may require several seconds to indicate an unlocked condition after the Sync source signal changes for any reason. The possible responses are 1 (ON) and 0 (OFF).

Response argument(s):

- **state** - <nr1> (range: 0 or 1)

Related Commands: <None>

Command Syntax: :SYNC:UNLock?

Response Syntax: :SYNC:UNLock **state**

Example: `:SYNC:UNLOCK?`

Response: `:SYNC:UNLOCK 1`

Appendix A: ASCII Code Chart and IEEE-488 Codes

HEX	0	1	2	3	4	5	6	7
0	0 NUL 0 0	20 DLE 10 16	40 0 SP 20 32	60 16 0 30 48	100 0 @ 40 64	120 16 P 50 80	140 0 ‘ 60 96	160 16 p 70 112
1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 1 ! 21 33	61 17 1 31 49	101 1 A 41 65	121 17 Q 51 81	141 1 a 61 97	161 17 q 71 113
2	2 STX 2 2	22 DC2 12 18	42 2 ” 22 34	62 18 2 32 50	102 2 B 42 66	122 18 R 52 82	142 2 b 62 98	162 18 r 72 114
3	3 ETX 3 3	23 DC3 13 19	43 3 # 23 35	63 19 3 33 51	103 3 C 43 67	123 19 S 53 83	143 3 c 63 99	163 19 s 73 115
4	4 SDC EOT 4 4	24 DCL DC4 14 20	44 4 $ 24 36	64 20 4 34 52	104 4 D 44 68	124 20 T 54 84	144 4 d 64 100	164 20 t 74 115
5	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 5 % 25 37	65 21 5 35 53	105 5 E 55 69	125 21 U 55 85	145 5 e 65 101	165 21 u 75 118
6	6 ACK 6 6	26 SYN 16 22	46 6 & 26 38	66 22 6 36 54	106 6 F 46 70	126 22 V 56 86	146 6 f 66 102	166 22 v 76 118
7	7 BEL 7 7	27 ETB 17 23	47 7 ’ 27 39	67 23 7 37 55	107 7 G 47 71	127 23 W 57 87	147 7 g 67 103	167 23 w 77 119
8	10 GET BS 8 8	30 SPE CAN 18 24	50 8 (28 40	70 24 8 38 56	110 8 H 48 72	130 24 X 58 88	150 8 h 68 104	170 24 x 78 120
9	11 TCT HT 9 9	31 SPD EM 19 25	51 9) 29 41	71 25 9 39 57	111 9 I 49 73	131 25 Y 59 89	151 9 i 69 105	171 25 y 79 121
A	12 LF A 10	32 SUB 1A 26	52 10 * 2A 42	72 26 : 3A 58	112 10 J 4A 74	132 26 Z 5A 90	152 10 j 6A 106	172 26 z 7A 122
B	13 VT B 11	33 ESC 1B 27	53 11 + 2B 43	73 27 ; 3B 59	113 11 K 4B 75	133 27 [5B 91	153 11 k 6B 107	173 27 { 7B 123
C	14 FF C 12	34 FS 1C 28	54 12 , 2C 44	74 28 < 3C 60	114 12 L 4C 76	134 28 \ 5C 92	154 12 l 6C 108	174 28 | 7C 124
D	15 CR D 13	35 GS 1D 29	55 13 - 2D 45	75 29 = 3D 61	115 13 M 4D 77	135 29] 5D 93	155 13 m 6D 109	175 29 } 7D 125
E	16 SO E 14	36 RS 1E 30	56 14 . 2E 46	76 30 > 3E 62	116 14 N 4E 78	136 30 ^ 5E 94	156 14 n 6E 110	176 30 ~ 7E 126
F	17 SI F 15	37 US 1F 31	57 15 / 2F 47	77 UNL ? 3F 63	117 15 O 4F 79	137 UNT _ 5F 95	157 15 o 6F 111	177 DEL 7F 127
	Addressed Commands	Universal Commands	Listen Addresses		Talk Addresses		Secondary Addresses	

KEY

octal	10 GET	GPIB code
	BS	**ASCII character**
hex	8 8	decimal

Appendix B: S2G Default Settings

Introduction

The default settings may be restored in one of five ways:

1. Power up initialization.
2. Changing GPIB address.
3. Switching from APIB to GPIB mode.
4. *RST common command.
5. *RCL 0 common command.

Except for methods 1 through 3, which are functionally identical, each of the reset methods have slightly different implications.

At the minimal level, *RCL 0 will invoke all of the settings listed in Table B-1. Also:

- Waveform registers are cleared.
- No macros are disturbed.
- The state of macro enable is unchanged.
- The state of the output queue is unchanged.
- The event enable registers (SESE and ASESE) are unchanged.
- The event registers (SESR and ASESR) are unchanged.
- The Service request enable register (SRER) is unchanged.
- The *SAV/*RCL registers are unchanged.
- The Filter slot configuration is not scanned

The next level is *RST. When the *RST command is received, it performs all of the actions of the *RCL 0, except that:

- A hardware reset of the internal processors (except the 486 control CPU) is performed prior to issuing settings.
- The macro enable flag is cleared.

The Power-on, GPIB address change and APIB->GPIB transition resets are most significant. They perform all of the actions of *RST, plus:

- Macros are erased.
- Input and output queues are cleared.
- The SESE and ASESE are cleared.
- The SESR and ASESR are cleared.
- The SRER is cleared.
- The *SAV/*RCL registers are cleared.
- The Filter slot configuration is scanned.

Since the power-up/reset default is for no DSP programs to be loaded, the default settings of the DSP programs are not shown in Table B-1. However, each time a DSP program is loaded, the settings of those programs have their own default values to which they are initialized. Table B-2 through Table B-6 below identify the default conditions of each of those programs.

Table B-1. System Two GPIB Default Settings

```
:AGEN:
      AMPL A,0.999957V
      AMPL B,0.999957V
      BURINTERVAL 3
      BURLEVEL 10PCT
      BURTIMEON 1
      CCIF:
            CFREQ 10000HZ
            IMFREQ 80
      CONFIG BF
      DAIMD:
            CCIF:
                  CFREQ 10000HZ
                  IMFREQ 80
            SMPTE:
                  HIFREQ 3000HZ
                  IMFREQ 80
                  RATIO 0.25X_Y
      DASINE:
            FRQ1 999.996HZ
            FRQ2 999.996HZ
            BURINTERVAL 3
            BURTIMEON 1
            RATIO 0.25X_Y
            VPHASE 0
      FREQ 1000HZ
      HIACCURACY OFF
      IMPEDANCE Z40
      INVERT A,OFF
      INVERT B,OFF
      OUTPUT OFF
      REF:
            DBM 600
            DBR 0.3873V
            FREQ 1000
            WATT 8
      SMPTE:
            HIFREQ 3000HZ
            IMFREQ 60
      WFM SINE,SINE (or WFM DASINE,SINE if no analog generator)

:ANLR:
      AUTORANGE A,ON
      AUTORANGE B,ON
      CHANNEL A
      COUPLING A,AC
      COUPLING B,AC
      DETECTOR RMS
      FAUTORANGE ON
      FILTERFREQ 1000HZ
      HPFILTER F10
      IMPEDANCE A,ZHI
      IMPEDANCE B,ZHI
      INPUT A,XLR
      INPUT B,XLR
      LPFILTER F500K
      MODE AMPLITUDE
      PRANGE AUTO
      RDGRATE R8
      REF:
            DBM 600
            DBRA 0.3873V
            DBRB 0.3873V
            FREQ 1000
            WATT 8
      RESPONSE 20
      TUNINGSRC FIXED
      WFDETECTOR NAB
      WFFILTER WEIGHTED
      WTG NONE

:DCX:
      AUTORANGE ON
      DCLEVEL DC1,0
      DCLEVEL DC2,0
      DCOUTPUT DC1,OFF
      DCOUTPUT DC2,OFF
      DIN:
            FORMAT TWOS
            SCALE 1
            RATE 32
      DOUT:
            DATA 0DEC
            FORMAT TWOS
            SCALE 1
      GDELAY 0.05
      MODE VOLTS
      OFFSET 0
      PORT A,0
      PORT B,0
      PORT C,0
      RANGE 2E+06O
      RANGE 500V
      RDGRATE R6
      SCALE 1

:DGEN:
      REF:
            FREQ 997
            DBR 0.3873FFS
            VFS 1
      AMPL A,1FFS
      AMPL B,1FFS
      ARBSIZE 8192
      BURINTERVAL 3
      BURLEVEL 0.25X_Y
      BURTIMEON 1
      CCIF:
            CFR 10000HZ
            IMFREQ 80HZ
      DITHER TRI
      FRQ1 997.001HZ
      FRQ2 999.999HZ
      INVERT A,OFF
      INVERT B,OFF
      OFFSET 0FFS
      OUTPUT OFF
      RATIO 0.25X_Y
      SAMPLES 1
      SMPTE:
            RATIO 0.25X_Y
            HIFREQ 3000HZ
            IMFREQ 79.9999HZ
      VPHASE 0
      WFM SINE,SINE (All arbitrary waveform registers are cleared)
:DIN:
      BANDWIDTH 700
      DEEMPHASIS OFF
      DETECTOR AVG
      FORMAT XLR
      IMPEDANCE BNC,Z75
      IMPEDANCE XLR,Z110
      MODE ACTIVE
      PKMODE HALF
      REF 48000
      RESOLUTION 24
      SCALEFREQBY MEASURED

:DIOSTATUS:
      XMT A,"04,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
,00,00,00"
      XMT B,"04,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
,00,00,00"

:DOUT:
      AMPL 5
      CABLESIM OFF
      CMAMPL 0.96
      CMFREQ 20000
      CMOUTPUT OFF
      FORMAT XLR
      INVALID 0
      JAMPL 0UI
      JFREQ 998.644
      JWFM NONE
      NAMPL 0
      NOUTPUT OFF
      PREEMPHASIS OFF
      RATE 48000HZ
      RESOLUTION 24
      RFENABLE FIXED
      RFTIME 1.59612E-08

:DSP:
      PROGRAM NONE
      REF:
            DBR1 0.1FFS
            DBR2 0.1FFS
            FREQ 1000
            VFS 1
      SRCPARAMS FREQ,HZ,20,20000,30,LOG
      TIMEOUT 10.000000
      TSELECT 1

:MON:
      MONO OFF
      MODE STEREO
      STEREO OFF

:SETTLING:
      ANLR AFREQ,0.5,0.00025HZ,2,0.02,FLAT,0,1
      ANLR ALEVEL,1,1E-05V,3,0.03,FLAT,0,1
      ANLR AMPLITUDE,3,1E-07V,3,0.03,FLAT,0,1
      ANLR BFREQ,0.5,0.00025HZ,2,0.02,FLAT,0,1
      ANLR BLEVEL,1,1E-05V,3,0.03,FLAT,0,1
      ANLR BP,5,5E-08V,3,0.03,EXP,0,1
      ANLR BR,3,1E-07V,3,0.03,EXP,0,1
      ANLR CCIF,5,3E-05PCT,3,0.1,EXP,0,1
      ANLR DFD,5,2E-05PCT,3,0.1,EXP,0,1
      ANLR DIM,3,0.0001PCT,3,0.1,EXP,0,1
      ANLR PHASE,-1,0.2DEG,2,0.03,FLAT,0,1
      ANLR RATIO,1,3E-05PCT,3,0.03,FLAT,0,1
      ANLR SMPTE,3,0.0001PCT,3,0.15,EXP,0,1
      ANLR THDAMPL,3,1E-07V,3,0.1,EXP,0,1
      ANLR THDRATIO,3,5E-05PCT,3,0.1,EXP,0,1
      ANLR WF,5,0.0002PCT,3,0.5,EXP,0,1
      ANLR XTALK,5,3E-05PCT,3,0.03,EXP,0,1

      DANLR:
            FREQ A,0.5,0.01HZ,1,0.002,FLAT,0,1
            FREQ B,0.5,0.01HZ,1,0.002,FLAT,0,1
            FUNC AMPA,FRMS,1,1E-06V,1,0.001,FLAT,0,1
            FUNC AMPA,NORMAL,1,1E-06V,3,0.03,FLAT,0,1
            FUNC AMPD,FRMS,1,1E-07FFS,1,0.001,FLAT,0,1
            FUNC AMPD,NORMAL,1,1E-06FFS,3,0.03,FLAT,0,1
            FUNC BPA,FRMS,3,1E-08V,2,0.02,EXP,0,1
            FUNC BPA,NORMAL,3,1E-08V,3,0.1,EXP,0,1
            FUNC BPD,FRMS,3,1E-08FFS,2,0.02,EXP,0,1
            FUNC BPD,NORMAL,3,1E-08FFS,3,0.1,EXP,0,1
            FUNC RATIO,FRMS,1,0.0001PCT,1,0.001,FLAT,0,1
            FUNC RATIO,NORMAL,1,0.0001PCT,3,0.03,FLAT,0,1
            FUNC THDA,FRMS,3,1E-07V,2,0.02,EXP,0,1
            FUNC THDA,NORMAL,3,1E-07V,3,0.1,EXP,0,1
            FUNC THDD,FRMS,3,1E-07FFS,2,0.02,FLAT,0,1
            FUNC THDD,NORMAL,3,1E-07FFS,3,0.1,EXP,0,1
            FUNC THDRATIO,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1
            FUNC THDRATIO,NORMAL,3,0.0001PCT,3,0.1,EXP,0,1
            FUNC XTALK,FRMS,3,1E-05PCT,2,0.02,EXP,0,1
            FUNC XTALK,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1
            LEVEL CHAA,FRMS,1,1E-06V,1,0.001,FLAT,0,1
            LEVEL CHAA,NORMAL,1,1E-06V,3,0.03,FLAT,0,1
            LEVEL CHAD,FRMS,1,1E-07FFS,1,0.001,FLAT,0,1
            LEVEL CHAD,NORMAL,1,1E-06FFS,3,0.03,FLAT,0,1
            LEVEL CHBA,FRMS,1,1E-06V,1,0.001,FLAT,0,1
            LEVEL CHBA,NORMAL,1,1E-06V,3,0.03,FLAT,0,1
            LEVEL CHBD,FRMS,1,1E-07FFS,1,0.001,FLAT,0,1
            LEVEL CHBD,NORMAL,1,1E-06FFS,3,0.03,FLAT,0,1

      DCX DIN,0,0DEC,3,0.03,FLAT,0,1
      DCX OHMS,0.5,0.1O,3,0.03,FLAT,0,1
      DCX VOLTS,0.2,0.0005V,3,0.03,FLAT,0,1

      DIN AMPLITUDE,3,0.01VPP,3,0.03,FLAT,0,1
      DIN IDELAY,0.01,7E-08SEC,3,0.03,FLAT,0,1
      DIN JITTER,3,3E-09SEC,3,0.1,EXP,0,1
      DIN OUTDELAY,0.01,7E-08SEC,3,0.03,FLAT,0,1
      DIN RATE,0.5,0.1HZ,3,0.03,FLAT,0,1

      TIMEOUT 4

:SWR:
      IN A,0
      IN B,0
      OFFSET CHB,0
      OFFSET OUT,0
      OMODE INDEPENDENT
      OUT A,0
      OUT B,0

:SYNC:
      ENABLE OFF
      FREQ 48000
      IMPEDANCE ZLO
      ODELAY 0UI
      ODENABLE OFF
      SRC AESEBU
```

Table B-2. DSP Audio Analyzer (DANLr) Default Settings

```
:DSP:
      DANLR:
            AUTORANGE A,ON
            AUTORANGE B,ON
            CHANNEL A
            COUPLING A,AC
            COUPLING B,AC
            DETECTOR FRMS
            FAUTORANGE ON
            FILTERFREQ 1000HZ
            HPFILTER F10
            INPUT DIGITAL (AD1X if no DIO installed)
            LPFILTER FS_2
            MODE AMPLITUDE
            RDGRATE R8
            RESPONSE 20
            TUNINGSRC FIXED
            WTG UNWT (Default for XTALK and BP is FX1)
```

Table B-3. FFT Spectrum Analzyer (FFT) Default Settings

```
:DSP:
      FFT:
            ACQLENGTH XLENGTH
            AVGS 1
            AVGTYPE 1
            COUPLING DC
            DELAY 0
            INPUT DIGITAL (AD1X if DIO is not installed)
            MODE INTRPOLATE
            PKTRIG 1
            SENSITIVITY 0.00100005FFS
            SRC1 A (Both analog and digital inputs)
            SRC2 B (Both analog and digital inputs)
            START 0
            TRIGGER FREE
            TSLOPE POS
            WINDOW BH
            XLENGTH 1024
```

Table B-4. Digital Interface Analyzer (INTervu) Default Settings

```
:DSP:
      INTERVU:
            AVGS 1
            JDETECTION STABLE
            MODE INTRPOLATE
            MONITOR AUDIO
            TRIGGER ARCV
            WINDOW BH
```

Table B-5. Multitone Audio Analyzer (FASTtest) Default Settings

```
:DSP:
      FASTTEST:
            FREQRES 0
            INPUT DIGITAL (AD1X if DIO is not installed)
            LENGTH AUTO
            MODE SPECTRUM
            PKTRIG 1
            PMODE INDEPENDENT
            PROCESS SYNC
            SRC1 A (Both analog and digital inputs)
            SRC2 B (Both analog and digital inputs)
            TRGDELAY 0
            TRIGGER DGEN
```

Table B-6. Quasi-anechoic Acoustical Tester (MLS) Default Settings

```
:DSP:
      MLS:
            DELAY 0SEC
            ETWINDOW NONE
            INPUT DIGITAL (AD1X if DIO is not installed)
            MODE INTRPOLATE
            PKTRIG 1
            SRC1 A (Both analog and digital inputs)
            SRC2 B (Both analog and digital inputs)
            START NONE
            STOP NONE
            TIME IMPULSE
            TRIGGER AGEN
```

Table B-7. Digital Data Analyzer (BITTEST) Default Settings

```
:DSP:
      BITTEST:
            FREEZE OFF
            MODE NORMAL
            RDGRATE AUTO
            WFM SINE
```

Appendix C: System Two GPIB Error messages

Group 501: GPIB driver errors (Execution)

5	I/O ERROR
12	NOT ENOUGH MEMORY
19	NO SUCH DEVICE
22	INVALID ARGUMENT
35	UNSUPPORTED VALUE
65	NO LISTENERS ON THE BUS
66	INVALID ARGUMENT TO FUNCTION
67	I/O OPERATION ABORTED (TIMEOUT)
68	ASYNCHRONOUS OPERATION IN PROGRESS
69	INPUT QUEUE ERROR
70	OUTPUT QUEUE ERROR
71	QUEUE ACTIVE ERROR
73	LOST EVENT ERROR
96	UNCLASSIFIED ERROR
97	DEVICE CLEAR RECEIVED
98	EOI WAS SEEN ON LAST TRANSFER
99	ERROR QUEUE FULL

Group 502: Parser errors (Command)

1	ERROR QUEUE FULL; NO MORE USED
2	COMMAND NOT FOUND
3	INCOMPLETE COMMAND HEADER
4	TOO MANY COMMAND HEADER, MAX IS 10
5	TOO MANY PARAMETERS
6	TOO FEW PARAMETERS
7	ILLEGAL PARAMETER TYPE
8	MISSING COMMA OR SEMICOLON OR EOI
9	MISSING (OR INVALID) SUFFIX PROGRAM DATA
10	ILLEGAL COMMAND HEADER
11	BAD ARBITRARY BLOCK DATA
12	ABD PARAM TYPE NOT TERMINATED WITH EOI
13	SYNTAX ERROR
14	ILLEGAL STRING PROGRAM DATA
15	UNKNOWN PARAMETER
16	MACRO TOO BIG TO FIT IN MACRO BUFFER
17	MACRO NOT FOUND
18	MACRO TOO SMALL
19	MACRO HAS ALREADY BEEN DEFINED
20	FAILED TO LOAD MACRO
21	MACRO PARAMETER SUBSTITUTION FAILED
22	ILLEGAL COMMAND IN MACRO *DMC, *GMC ETC
23	WHEN EXECUTING MACRO FAILED TO ALLOCATE MEMORY FOR ABPD
24	DDT MACRO IS GREATER THAN 1024 BYTES
25	DDT MACRO IS LESS THAN ZERO
26	MISSING COMMA
27	ILLEGAL MACRO NAME
28	PARAMETER OUT OF RANGE
29	ILLEGAL GROUP EXECUTE TRIGGER
30	ILLEGAL INDEFINITE LENGTH ARBITRARY BLOCK DATA
31	MACRO EXECUTION AND MACRO PARAM SUBSTITUTION DOES NOT ALLOW INDEFINITE ARBS

Group 503: Macro Verification errors (Command)

Same as group 502, except error occurred at macro verification time.

Group 504: Macro Execution Errors (Execution)

Same as group 502, except error occurred at macro execution time.

Group 505: AGEN module errors (Execution)

1	ILLEGAL IMPEDANCE
2	ILLEGAL OUTPUT CONFIGURATION
3	TOO FEW PARAMETERS
4	ILLEGAL PARAMETER TO WFM COMMAND
5	COMMAND WFM FAILED
6	BURST TIME ON GREATER OR EQUAL BURST INTERVAL
7	DSP OPTION NOT INSTALLED
8	BURST OPTION NOT INSTALLED
9	IMD OPTION NOT INSTALLED
10	NOT IMPLEMENTED
11	BELOW MINIMUM AMPLITUDE
12	ABOVE MAXIMUM AMPLITUDE
13	BELOW MINIMUM FREQUENCY
14	ABOVE MAXIMUM FREQUENCY
15	W&F OPTION NOT INSTALLED
16	AGEN OPTION NOT INSTALLED
17	REQUESTED IMFREQ OUT OF RANGE
18	REQUESTED CFREQ OUT OF RANGE
19	REQUESTED HIFREQ OUT OF RANGE
20	ILLEGAL BURST LEVEL REQUESTED
21	REQUESTED IMRATIO OUT OF RANGE

Group 506: Switcher module errors (Execution)

1	INVALID CHANNEL B QUERY FOR CURRENT MODE
2	INVALID CHANNEL A QUERY FOR ALLB MODE
3	SWR MODULE NOT FOUND

Group 507: DGEN module errors (Execution)

1	TOO FEW PARAMETERS
2	ILLEGAL PARAMETER TO WFM COMMAND
3	COMMAND WFM FAILED
4	BURST TIME ON GREATER OR EQUAL BURST INTERVAL
5	ILLEGAL ARB SIZE
6	ILLEGAL OFFSET
7	DSP DOES NOT CONTAIN ARB
8	DSP OPTION NOT AVAILABLE
9	DSP NOT IDLE

10	WAVEFORM TOO BIG
11	REGISTER DOES NOT EXIST
12	BELOW MINIMUM AMPLITUDE
13	ABOVE MAXIMUM AMPLITUDE
14	WAVEFORMS FOR CH1 AND CH2 NOT SAME LENGTH
15	REQUESTED IMFREQ OUT OF RANGE
16	REQUESTED CFREQ OUT OF RANGE
17	REQUESTED FREQ OUT OF RANGE
18	REQUESTED HIFREQ OUT OF RANGE
19	REQUESTED BURST LEVEL OUT OF RANGE
20	REQUESTED RATIO OUT OF RANGE
21	INVALID REFERENCE LEVEL
22	INVALID VFS
23	INVALID DC OFFSET FOR DCSINE
24	ATTEMPT TO LOAD INVALID ARB

Group 508: DCX module errors (Execution)

1	DCX MODULE NOT FOUND
2	DOUT DATA OUT OF RANGE
3	DIN RATE OUT OF RANGE
4	DIN SCALE OUT OF RANGE
5	DOUT SCALE OUT OF RANGE
6	OFFSET OUT OF RANGE
7	SCALE OUT OF RANGE
8	INCORRECT MODE FOR REQUESTED UNIT

Group 509: ANLR module errors (Execution)

1	VALUE OUT OF RANGE
2	CHANNEL A SET TO AUTORANGE
3	CHANNEL B SET TO AUTORANGE
4	FUNCMETER SET TO AUTORANGE
5	INCOMPATIBLE MODE AND UNIT
6	INCOMPATIBLE MODE AND FILTER
7	INCOMPATIBLE MODE AND DETECTOR
8	INCOMPATIBLE MODE AND STEERING
9	INCOMPATIBLE STEERING MODE AND FILTER FREQUENCY
10	DSP OPTION NOT INSTALLED
11	IMD OPTION NOT INSTALLED
12	ILLEGAL TUNING SOURCE
13	ANLR OPTION NOT INSTALLED
14	W&F OPTION NOT INSTALLED
15	TOO MANY PARAMETERS FOR FIXED READING RATE
16	INVALID METER TYPE SELECTION
17	ANALOG GENERATOR OPTION NOT INSTALLED
18	FILTER NOT INSTALLED IN REQUESTED SLOT

Group 510: DSP module errors (Execution)

1	INCOMPATIBLE MODE AND REFERENCE
2	CAN NOT LOAD DSP PROGRAM
3	DSP PROGRAM NOT LOADED
4	SRCPARAM INCOMPATIBLE WITH DSP PROGRAM
5	INVALID SOURCE PARAMETER FOR PROGRAM
6	INVALID INPUT DOMAIN FOR SOURCE PARAMETER
7	INVALID SOURCE 1 INPUT FOR REQUESTED BATCH
8	INVALID SOURCE 2 INPUT FOR REQUESTED BATCH

9	INVALID INTERVU MODE FOR SELECTED SOURCE PARAMETER
10	INVALID READING FOR ACQUIRED DATA
11	TIMEOUT ON ACQX?, XFRM? OR REPR?
12	DSP OPTION NOT INSTALLED
13	DSP BUSY
14	ATTEMPT TO LOAD ACQUISITION OR TRANSFORM BUFFER FAILED
15	INVALID SELECTION FOR BATCH READING ENABLE
16	NO BATCH READINGS ENABLED
17	TOO MANY PARAMETERS
18	NOT ENOUGH PARAMETERS
19	REFERENCE VALUE OUT OF RANGE
20	BATCH TABLE TOO BIG
21	TABLE LOAD ABORTED WITH DCL, SDC OR EARLY EOI
22	NO BATCH TABLE LOADED OR SRCPARAMS START/STOP/STEPS CONFLICT

Group 511: Digital Analyzer DSP Program errors (Execution)

1	ILLEGAL MODE
2	DSP IS NOT SET TO DANLR MODE
3	CHANNEL A SET TO AUTORANGE
4	CHANNEL B SET TO AUTORANGE
5	FUNCMETER SET TO AUTORANGE
6	ILLEGAL UNIT
7	ILLEGAL FREQ
8	INCOMPATIBLE MODE AND FILTER
9	ILLEGAL TUNING SOURCE
10	TOO MANY PARAMETERS FOR FIXED READING RATE
11	INVALID METER TYPE SELECTION
12	AGEN OPTION NOT INSTALLED
13	DIO OPTION NOT INSTALLED
14	ANALOG OPTIONS NOT INSTALLED
15	ILLEGAL RANGE

Group 512: FFT DSP Program errors (Execution)

1	ILLEGAL MODE
2	DSP IS NOT SET TO DANLR MODE
3	CHANNEL A SET TO AUTORANGE
4	CHANNEL B SET TO AUTORANGE
5	FUNCMETER SET TO AUTORANGE
6	ILLEGAL UNIT
7	ILLEGAL FREQ
8	INCOMPATIBLE MODE AND FILTER
9	ILLEGAL TUNING SOURCE
10	TOO MANY PARAMETERS FOR FIXED READING RATE
11	INVALID METER TYPE SELECTION
12	AGEN OPTION NOT INSTALLED
13	DIO OPTION NOT INSTALLED
14	ANALOG OPTIONS NOT INSTALLED
15	ILLEGAL RANGE

Group 513: INTERVU DSP Program errors (Execution)

1	ILLEGAL MODE
2	INTERVU PROGRAM NOT LOADED
3	ATTEMPT TO SET AVERAGES FAILED
4	ATTEMPT TO SET JITTER DETECTOR FAILED
5	ATTEMPT TO SET MONITOR FAILED
6	ATTEMPT TO SET MODE FAILED
7	ATTEMPT TO SET TRIGGER SOURCE FAILED
8	ATTEMPT TO SET WINDOW SHAPE FAILED
9	DSP OPTION NOT INSTALLED
10	ATTEMPT TO INVOKE DISABLED READING
11	DSP IN OPSTATE READ
12	DSP IN OPSTATE SETUP

Group 514: FASTTEST DSP Program errors (Execution)

1	ILLEGAL SOURCE
2	FASTTEST PROGRAM NOT LOADED
3	DSP DISABLED
4	INVALID SAMPLE RATE REQUESTED
5	INVALID INPUT REQUESTED
6	INVALID LENGTH REQUESTED
7	INVALID MODE REQUESTED
8	INVALID WINDOW REQUESTED
9	INVALID DELAY REQUESTED
10	INVALID TRIGGER SOURCE REQUESTED
11	DSP IN OPSTATE READ
12	DATA REQUESTED FROM DISABLED CHANNEL
13	ATTEMPT TO INVOKE DISABLED READING
14	DSP IN OPSTATE SETUP
15	ANALOG ANALYZER OPTION NOT INSTALLED

Group 515: MLS DSP Program errors (Execution)

1	ILLEGAL SOURCE
2	MLS PROGRAM NOT LOADED
3	ILLEGAL DELAY
4	DSP IN OPSTATE READ
5	DATA REQUESTED FROM DISABLED CHANNEL
6	DSP OPTION NOT INSTALLED
7	INVALID INPUT REQUESTED
8	ATTEMPT TO INVOKE DISABLED READING
9	DSP IN OPSTATE SETUP
10	ANALOG ANALYZER OPTION NOT INSTALLED
11	ILLEGAL MODE

Group 516: Digital I/O module errors (Execution)

1	ILLEGAL IMPEDANCE
2	COMMON MODE OFF
3	RFENABLE IS SET TO FIXED
4	NOUTPUT IS SET TO OFF
5	ILLEGAL JITTER WAVEFORM
6	ILLEGAL MODE
7	ILLEGAL LENGTH
8	DIO OPTION NOT INSTALLED
9	BELOW MINIMUM AMPLITUDE
10	ABOVE MAXIMUM AMPLITUDE
11	BELOW MINIMUM FREQUENCY
12	ABOVE MAXIMUM FREQUENCY
13	ILLEGAL DITHER TYPE
14	BELOW MINIMUM TIME
15	ABOVE MAXIMUM TIME
16	ILLEGAL JITTER AMPLITUDE

Group 517: SYNC/REF module errors (Execution)

1	DELAY SETTING DISABLE
2	ILLEGAL DELAY VALUE
3	DSP OPTION NOT INSTALLED
4	ILLEGAL SYNC FREQ

Group 518: Settling module errors (Execution)

1	METER OR DETECTOR NOT FOUND
2	ILLEGAL UNIT
3	ILLEGAL ALGORITHM
4	ILLEGAL DELAY
5	ILLEGAL POINTS
6	ILLEGAL TOLERANCE
7	ILLEGAL TIME OUT
8	ILLEGAL FLOOR
9	ILLEGAL TRIGGER
10	ANLR OPTION NOT INSTALLED
11	DSP OPTION NOT INSTALLED
12	DCX NOT AVAILABLE
13	DIO OPTION NOT INSTALLED

Group 519: DSP Warnings (Execution)

1	WAVEFORM LOAD OVERRUN -- FILE IS LONGER THAN SELECTED BUFFER
2	WAVEFORM LOAD UNDERRUN -- FILE IS SHORTER THAN SELECTED BUFFER
3	CHANNEL 1 GENERATOR WAVEFORM SHOULD BE LOADED BEFORE CHANNEL 2
4	CHANNEL 1 TIME FRAME NOT SET -- MUST DO A TIME SWEEP BEFORE FREQUENCY SWEEP
5	CHANNEL 2 TIME FRAME NOT SET -- MUST DO A TIME SWEEP BEFORE FREQUENCY SWEEP
6	CH1 & CH2 TIME FRAMES NOT SET - MUST DO A TIME SWEEP BEFORE FREQUENCY SWEEP
7	NOT ENOUGH TONES IN WAVEFORM FOR RELIABLE TRIGGERING
8	LOBE WIDTH EVEN, 0 OR 1
9	FREQUENCY CORRECTION OUT OF RANGE
10	MAIN FILTER OVER-RANGED
11	SAMPLE RATE MAY BE INSUFFICIENT
12	FILTERED LEVEL RANGED
13	RMS FILTER 1 OVERLOAD OCCURRED
14	RMS FILTER 2 OVERLOAD OCCURRED
15	TRIGGER CHECK FAILED WHEN TRYING TO FREQUENCY CORRECT DOWNLOADED WAVEFORM - CORRECTION ABORTED
16	NOT ENOUGH SAMPLES DOWNLOADED FOR FREQUENCY ERROR CORRECTION
17	FREQUENCY CORRECTION SKIPPED - WAVEFORM HAS ALREADY BEEN CORRECTED
18	TRIGGER CHECK FAILED WHILE TRYING TO FREQUENCY CORRECT ACQUIRED WAVEFORM - CORRECTION ABORTED
19	CROSSTALK MODE REQUIRES AT LEAST ONE GENERATOR TONE DIFFERENT IN EACH OF THE TWO CHANNELS
20	CURRENT GENERATOR WAVEFORM HAS NO CROSSTALK TONES FOR CHANNEL 1
21	CURRENT GENERATOR WAVEFORM HAS NO CROSSTALK TONES FOR CHANNEL 2

Group 520: DSP Errors (Execution)

2 DSP PROGRAM REQUIRES DIO OR MEM OPTION

3 MAIN DSP PROCESSOR'S STACK OVERFLOW EXCEPTION WAS RAISED

4 DIO OPTION NOT PRESENT -- A/D OR DGEN ARE THE ONLY VALID INPUT SETTINGS

5 DIO OPTION NOT PRESENT -- D/A IS THE ONLY VALID OUTPUT SETTING

6 AT LEAST ONE INPUT CHANNEL MUST BE ENABLED IN ORDER TO ACQUIRE

8 DSP PROGRAM DOES NOT SUPPORT EXTERNAL SWEEPS EXCEPT FOR TIME

9 TRIGGER AND FREQUENCY CORRECTION MODES REQUIRE CHANNEL 1 AND CHANNEL 2 GENERATOR WAVEFORMS

10 TRANSFORM SIZE SETTING OUT OF BOUNDS

11 WAVEFORM FILE IS NOT AN MLS IMPULSE RESPONSE

12 WAVEFORM FILE IS NOT THE PROPER TYPE FOR SELECTED BUFFER

13 FREQUENCY RESOLUTION MAY ONLY BE A SWEEP SOURCE-2 SELECTION

14 GENERATOR AMPLITUDE MAY ONLY BE A SWEEP SOURCE-2 SELECTION

15 GENERATOR FREQUENCY MAY ONLY BE A SWEEP SOURCE-2 SELECTION

16 FFT START TIME MAY ONLY BE A SWEEP SOURCE-2 SELECTION

17 FFT PRETRIGGER MAY ONLY BE A SWEEP SOURCE-2 SELECTION

18 REFERENCE TIME MAY ONLY BE A SWEEP SOURCE-2 SELECTION

20 CHANNEL 1 DE-EMPHASIS OVERLOAD DETECTED

21 CHANNEL 2 DE-EMPHASIS OVERLOAD DETECTED

22 EXCESSIVE TONES IN WAVEFORM FOR PROPER OPERATION

23 WAVEFORM LOAD NOT OF VALID LENGTH

24 CHANNEL 1 & CHANNEL 2 GENERATOR WAVEFORMS NOT OF EQUAL LENGTH

25 GENERATOR WAVEFORM FREQUENCIES TOO CLOSE FOR TRIGGERING OR FREQUENCY CORRECTION

26 FREQUENCY CORRECTION DATA OVERRUN

27 FREQUENCY RESOLUTION SETTING CONFLICTS WITH REQUESTED FREQUENCY

28 MAXIMUM BP/BR FILTER FREQUENCY EXCEEDED

29 NARROW BANDPASS FILTER ONLY AVAILABLE AT 48 KHZ SAMPLE RATE

30 SWEEP DATA INCOMPATIBLE WITH SWEEP SOURCE, SELECT DATA = PROBABILITY OR CHANGE SWEEP SOURCE

31 SWEEP DATA = EYE INCOMPATIBLE WITH SWEEP SOURCE = FREQUENCY

32 IF A SWEEP DATA IS SET TO EYE OPENING, OTHER SWEEP DATA MUST BE SET TO AN EYE OPENING OR NONE

33 FFT TRIGGER DELAY TIME EXCEEDS THE ACQUIRE BUFFER SIZE. CONSIDER USING A SHORTER TRIGGER DELAY TIME OR A LARGER ACQUIRE SIZE

34 FFT START TIME IS SPECIFIED BEYOND THE END OF ACQUIRED DATA. CONSIDER USING A SHORTER FFT START TIME OR A LARGER ACQUIRE SIZE

35 THE SUM OF FFT START TIME AND TRANSFORM LENGTH WILL EXCEED THE ACQUIRED DATA SIZE

36 FFT LENGTH IS LARGER THAN ACQUIRED DATA SIZE. SELECT A SMALLER FFT LENGTH OR RE-ACQUIRE DATA WITH A LARGER ACQUISITION SIZE

37 CROSSTALK INFORMATION CANNOT BE DISPLAYED. THE GENERATOR WAVEFORM DOES NOT CONTAIN CROSSTALK TONES FOR CHANNEL 1.

38 CROSSTALK INFORMATION CANNOT BE DISPLAYED. THE GENERATOR WAVEFORM DOES NOT CONTAIN CROSSTALK TONES FOR CHANNEL 2.

39 FREQUENCY TO BE CORRECTED IS TOO LARGE.

40 FREQUENCY TO BE CORRECTED IS TOO SMALL.

42 FFT START TIME MUST BE EQUAL TO OR GREATER THAN TRIGGER DELAY TIME.

43 MAIN DSP PROCESSOR'S RESET EXCEPTION WAS RAISED

44 MAIN DSP PROCESSOR'S ILLEGAL INSTRUCTION EXCEPTION WAS RAISED

45 EITHER CHANNEL A OR B IS OFF.

46 CANNOT RETRANSFORM IF AVERAGING IN THE FREQUENCY DOMAIN, BECAUSE THE ACQUISITION BUFFER HAS ONLY THE LATEST ACQUISITION.

50 TOO MANY FILTERS TURNED ON FOR FUNCTION METER.

51 TRANSFORM SIZE MUST BE 8192 OR LESS WHEN USING SYNCHRONOUS AVERAGING.

52 MEMORY LIMITATIONS PREVENT SYNCHRONOUS AVERAGING BEFORE FREQUENCY CORRECTION WHEN TRANSFORM LENGTH IS 8192. CONSIDER A SMALLER TRANSFORM SIZE OR FREQUENCY CORRECT BEFORE SYNCHRONOUS AVERAGING.

53 THE DC COMPONENT OF THE CH 1 SIGNAL IS GREATER IN MAGNITUDE THAN ANY AC COMPONENT AND SO THE SYNC TO SINE PROCESS WAS HALTED. CHOOSING THE SUBTRACT 1/2 PK-PK OR SUBTRACT AVG OPTIONS ON THE PANEL MAY ALLOW YOU TO USE THE SYNC TO SINE FEATURE.

54 THE DC COMPONENT OF THE CH 2 SIGNAL IS GREATER IN MAGNITUDE THAN ANY AC COMPONENT AND SO THE SYNC TO SINE PROCESS WAS HALTED. CHOOSING THE SUBTRACT 1/2 PK-PK OR SUBTRACT AVG OPTIONS ON THE PANEL MAY ALLOW YOU TO USE THE SYNC TO SINE FEATURE.

55 CHANNEL 1 AND CHANNEL 2 GENERATOR WAVEFORMS HAVE NOT BEEN LOADED

56 THE GREATEST MAGNITUDE TONE FOR CH 1 IS TOO LOW IN FREQUENCY TO USE SYNC TO SINE. TO CALCULATE THE MINIMUM FREQUENCY FOR WHICH SYNC TO SINE WILL WORK USE THE FOLLOWING FORMULA: SAMPLE FREQUENCY IN HZ / FFT LENGTH = SIZE OF EACH FFT BIN IN HZ. 6 * SIZE OF FFT BIN IN HZ = MINIMUM FREQUENCY. TO USE SYNC TO SINE, THE GREATEST MAGNITUDE TONE MUST HAVE A FREQUENCY GREATER THAN THIS MINIMUM FREQUENCY.

57 THE GREATEST MAGNITUDE TONE FOR CH 2 IS TOO LOW IN FREQUENCY TO USE SYNC TO SINE. TO CALCULATE THE MINIMUM FREQUENCY FOR WHICH SYNC TO SINE WILL WORK USE THE FOLLOWING FORMULA: SAMPLE FREQUENCY IN HZ / FFT LENGTH = SIZE OF EACH FFT BIN IN HZ. 6 * SIZE OF FFT BIN IN HZ = MINIMUM FREQUENCY. TO USE SYNC TO SINE, THE GREATEST MAGNITUDE TONE MUST HAVE A FREQUENCY GREATER THAN THIS MINIMUM FREQUENCY.

58 RESAMPLING PROCESS ABORTED. NO ZERO CROSSINGS FOUND. CHECK LEVEL METERS TO VERIFY SIGNAL IS PRESENT.

Group 521: DSP Host Vector errors (Execution)

1 HOST VECTOR ERROR

Group 522: SAV/RCL errors (Execution)

1 ATTEMPT TO RCL FROM EMPTY REGISTER

Group 523: BITTEST errors (Execution)

1 DSP IS NOT SET TO BITTEST MODE
2 INITIALIZATION ERROR
3 INVALID DETECTOR MODE SELECTED
4 INVALID WAVEFORM SELECTION
5 INVALID READING RATE SELECTION
6 INVALID FREEZE STATE SELECTION

Appendix D: System Two Optional Filter Identification Codes

The following table provides a description for each filter ID returned by *OPT?.

ID	Name
0	Empty Slot
15	External
17	A-Weighting
18	50 us de-emph
19	200-15kHz + 15.6 K notch
21	300 Hz lowpass
22	10kHz lowpass
23	30Hz highpass
24	100Hz bandpass
25	1kHz bandpass
26	10kHz bandpass
27	100Hz hi-Q bandpass
29	20kHz Apogee LP
33	CCIR 468-3
34	50 us de-emph + 15.6k notch
35	200-15kHz + 19k notch
37	400 Hz lowpass
38	12.7kHz lowpass
40	120 Hz bandpass
41	1.2kHz bandpass
42	12.5kHz bandpass
43	500 Hz hi-Q bandpass
49	CCITT P.53
50	50 us de-emph + 19k notch
53	500 Hz lowpass
54	15kHz lowpass
56	180 Hz bandpass
57	1.5kHz bandpass
58	15kHz bandpass
59	1kHz hi-Q bandpass
65	C-Message
70	18kHz lowpass
72	250 Hz bandpass
73	2kHz bandpass
74	16kHz bandpass
75	3kHz hi-Q bandpass
81	C-Weighting
82	75us de-emph
86	19kHz lowpass
87	70 Hz highpass
88	300 Hz bandpass
89	2.04kHz bandpass
90	20kHz bandpass
91	12.5kHz hi-Q bandpass
98	75us de-emph + 15.7k notch
101	1kHz lowpass
102	20kHz lowpass
104	400 Hz bandpass
105	3kHz bandpass
106	22kHz bandpass
114	75us de-emph + 19k notch
118	22kHz lowpass
120	500 Hz bandpass
121	3.15kHz bandpass
122	24kHz bandpass
135	400 Hz bandpass
136	600 Hz bandpass
137	3.4kHz bandpass
138	25kHz bandpass
146	75us de-emph + A-wtg
149	3kHz lowpass
152	666 Hz bandpass
153	4kHz bandpass
154	30kHz bandpass
165	4kHz lowpass
167	15kHz LP + 15.6k notch
168	800Hz bandpass
169	4.5kHz bandpass
181	8kHz lowpass
182	15kHz + 19k notch
183	2kHz highpass
184	945Hz bandpass
185	5kHz bandpass
197	5kHz Low Pass
200	315Hz bandpass
201	6kHz bandpass
216	333Hz bandpass
217	8kHz bandpass
229	50kHz lowpass
288	CCIR 2k

Appendix E: System Two GPIB Firmware Information

New Features in GPIB Firmware Version 2.002

The features of APWIN listed below are now supported in System Two GPIB firmware version 2.002.

- Digital Data Analyzer (BITTEST) DSP program.
- Digital Generator – Special BITTEST Random Waveform.
- Intervu measurements of Jitter on any squarewave connected to the BNC input connector. The selections for jitter detection of squarewave are implemented: Jitter Detection Squarewave Rising, and Squarewave Falling.
- Internal sweeps for batch mode DSP programs: FASTTEST, FFT, MLS, and INTERVU. New command :DSP:BATCH?
- Two user definable internal sweep tables for batch mode DSP sweeps. New commands :DSP:TABLE, :DSP:TABLE?, and :DSP:TSELECT.

Changes in GPIB Firmware Version 2.002

The lists below summarize the changes to version 1.001 that are implemented in version 2.002.

New Commands

1. :DSP:BATCH?
2. :DSP:TABLE and :DSP:TABLE?
3. :DSP:TSELECT
4. :DSP:BITTEST:CLEAR
5. :DSP:BITTEST DATA?
6. :DSP:BITTEST:ERROR?
7. :DSP:BITTEST:FREEZE and :DSP:BITTEST:FREEZE?
8. :DSP:BITTEST:FROZEN?
9. :DSP:BITTEST:MODE and :DSP:BITTEST:MODE?
10. :DSP:BITTEST:RDGRATE and :DSP:BITTEST:RDGRATE?
11. :DSP:BITTEST:SET?
12. :DSP:BITTEST:WFM and :DSP:BITTEST:WFM?

Commands with Additional Arguments

The list below summarizes the commands that existed in version 1.001 that have had additional command argument choices added.

1. :DGEN:WFM SPCIAL,RANDOM
2. :INTERVU:JDETECTION RISE and :INTERVU:JDETECTION FALL
3. :DSP:PROGRAM BITTEST

GPIB Firmware v2.002 – Known Bugs

The following list describes all known problems in the System Two GPIB firmware version 2.002 as of the release date (July 1999):

1. The :DSP:INTervu:MON JITTer command does not connect the Jitter signal to the monitor channel as specified. Instead, the Analog Analyzer Function meter signal is applied.

   Workaround: Issue the following command sequence prior to loading the INTervu program:

   ```
   :DSP:PROGram FFT;FFT:INPut AD1X;SRC1 JITT;SRC2 JITT
   ```

2. The analog inputs to FFT, FASTTEST and MLS exhibit a 0.21 dB error when the number of selected channels changes between the time of the acquisition and the reading of the data. If either channel is changed from NONE, or to NONE, the data will exhibit the error. For example, if the source settings are set prior to the acquisition to the following

   ```
   :DSP:FFT:SRC1 A;SRC2 B
   ```

   and changed after the acquisition to

   ```
   :DSP:FFT:SRC1 A;SRC2 NONE
   ```

   data from channel 1 will read ~0.21 dB too high.

   Workaround: Don't change the SRC1 and SRC2 settings between acquisition and the reading of the data.

3. The BATCH? query requests DSP sweep data after an acquisition. If the user's program does not read the entire response string (which can be very large), and attempts to clear the I/O queue with a GPIB Select Device Clear or Device Clear, the instrument does not clear its output queue and an I/O deadlock occurs.

   Workaround: Avoid the problem by reading the entire response and discarding the unneeded data, or send a *RST to reset the instrument to clear the I/O (store the current settings first and then recall them after the reset).

4. For the BATCH? query response, response formats of BINARY or RBINARY or ASCII are garbled for data elements beyond 13650 sweep points when two or more measurements are specified by a preceding :DSP:OPSTATE READING,AMP1,AMP2 (or other valid param2 and param3 enums). Normal behavior expects the instrument to send as many bytes as the sweep requires, up to 256K sweep points in the case of Intervu timebase sweep or jitter time sweeps. The user may request many more points than this if the sweep step size yields many interpolated measurement points per acquisition point. The consequence is that sweep data is useless beyond 13650 sweep points. The maximum number of valid sweep points is probably less than 13650 points if the sweep specifies more than two measurements (e.g. MLS channel 1 and 2 amplitude data and channel 1 and 2 phase data at the same time).
   If more than 13650 sweep points is requested in BINARY or RBINARY response formats, the four-byte IEEE formatted numbers for sweep points 13651 and later will be incorrectly coded, resulting in invalid numbers. If more than 13650 sweep points is requested in ASCII format the numbers will be incorrect. Note that if only one measurement is specified by the :DSP:OPSTATE READING,AMP1 command, then the number of elements that may be read is much larger than 13650 points (probably 3x but the maximum number is not determined yet).

   Work around: Limit the number of sweep steps to less that 13650 for two or more measurement channels per sweep with any batch mode DSP program. Use a second sweep in Reprocess mode if additional data must be read for the same acquisition (e.g. second half of a 16384 bin FFT or additional measurement channels from the same acquisition).

APWIN v2.1 Features Not Implemented in GPIB Firmware Version 2.002

Please note the following features available in the APWIN v2.1 software that are not supported in the System Two GPIB firmware version 2.002:

1) Analog Generator EQ Sine.
2) Analog Generator D/A EQ Sine.
3) Digital Generator EQ Sine.
4) Digital Interface Jitter Generator EQ Sine.
5) Stimulus/response sweeps controlled by the sweep panel.
6) Compute functions for sweep data.
7) Limit testing of sweep data.
8) Regulation Mode.
9) Units of dBGA, dBGB and Watts are not supported for DSP programs with inputs set to use the analog to digital converter(s).

10) FFT Spectrum Analyzer DSP

    a) Command :DSP:FFT:TRIGger, APWIN trigger source selections of “Ch. 1 Fixed Lev” and “Ch. 2 Fixed Lev” are not supported.

    b) Commands :DSP:FFT:SRC1:JITTer and :DSP:FFT:SRC2:JITTer currently support APWIN Jitter Signal (UI) units but do not support Jitter Signal (sec) units source selection.

    c) Commands :DSP:FFT:ACQLength and :DSP:FFT:XLENgth together support 24K acquisition size with variable FFT length, or acquisition size the same as FFT length. Only Acquire sizes of Track FFT (:DSP:FFT:ACQLength:XLENgth) and 24K (:DSP:FFT:ACQLength:L24K) are supported. Acquire sizes of 800, 1.5k, 2.5k, 5k, 10k, and 19k are not supported.

11) FASTTEST Multitone Audio Analyzer DSP

    a) Command :DSP:FASTTEST:TRIG for triggering selection does not support AGEN.

    b) Commands :DSP:FASTTEST:SRC1:JITTer and :DSP:FASTTEST:SRC2:JITTer currently support APWIN Jitter Signal (UI) units but do not support Jitter Signal (sec) units source selection.

12) MLS Quasi-anechoic Acoustical Tester DSP commands :DSP:MLS:SRC1:JITTer and :DSP:MLS:SRC2:JITTer currently support APWIN Jitter Signal (UI) units but do not support Jitter Signal (sec) units for source selection.

13) DSP Audio Analyzer

    a) Command :DSP:DANLR:RANGe does not support dBr, dBFS, and PCTFs units.

    b) Command :DSP:DANLr:WTG does not support THD measurements of harmonics only. In APWIN 2.1 a new selection in the Weighting Filter menu allows the THD+N measurement function of harmonics only.

14) DGEN Digital Generator

    a) Command :DGEN:AMPL does not support output amplitude units of DEC or HEX. In APWIN 2.1 additional output amplitude units selections have been added allowing the digital output to be specified in HEX or decimal values.

    b) Command DGEN:ARBLOAD does not automatically set the sample rate when an arbitrary waveform is loaded into the digital generator. With APWIN 2.1, sample rates are now automatically set when generator waveforms are loaded using information contained in the waveform file itself. The Configuration panel allows this feature to be enabled.

15) APWIN 2.1 default values for batch-mode DSP programs may be different from the GPIB factory default settings. In APWIN 2.1 many initial default values for batch-mode DSP programs have been changed to allow easier first-time operation. Additionally, this database of default values will learn user preferences as the time/frequency domain switch is used to facilitate easier return to previous setups. This also facilitates initial setup of the several DSP programs such as FFT, INTERVU, *FAST*TEST, etc.

Appendix F: Binary Format for Floating Point Numbers

The floating point representation included here is a subset of IEEE Std 754-1985 [3] that is specified in IEEE Std 488.2-1992. This standard describes both single- and double-precision numbers. The System Two uses only single-precision numbers.

Floating Point Code Fields

Floating point numbers shall be represented by three fields. The fields are:

1. Sign field
2. Exponent field
3. Fraction field.

The size of the fields for single-precision numbers is:

Sign field width	1 bit	E (max)	+127
Exponent field width	8 bits	E (min)	-126
Fraction field width	23 bits	Exponent bias	+127
Total width	32 bits		

Basic Formats

Numbers in the single and double formats are composed of the following three fields:

(1) 1-bit sign s (s = 0 = positive; s = 1 = negative)

(2) Biased exponent e = E + bias

(3) Fraction f = .b(1)b(2)…b(p-1)

where

p = the number of significant bits

b(n) = 0 or 1

The range of the unbiased exponent, E, includes every integer between two values E(min) and E(max), inclusive, and also two other reserved values: E(min) -1 to encode ±0 and denormalized numbers, and E(max) +1 to encode ±∞ and NaNs (not a number symbol). The foregoing parameters are given above. Each non-zero numerical value has just one encoding. The fields are interpreted as follows.

A 32-bit single-format number, X, is divided as shown above. The value, v, of X is inferred from its constituent fields thus:

(1) if $e = 255$ and $f \neq 0$, then v is NaNs regardless of s

(2) if $e = 255$ and $f = 0$, then $v = (-1)^{s} \infty$

(3) if $0 < e < 255$, then $v = (-1)^{s} 2^{e-127} (1.f)$

(4) if $e = 0$ and $f \neq 0$ (denormalized numbers), then $v = (-1)^{s} 2^{-126} (0.f)$

(5) if $e = 0$ and $f = 0$, then $v = (-1)^{s} 0$ (zero)

Order of transmission

Single-precision numbers shall be transmitted in four bytes. The transmission shall be structured according to the following relationships between DIO signal lines and the fields.

DIO–	8	7	6	5	4	3	2	1	
	S	E ↑ MSBE	E	E	E	E	E	E	*First byte sent*
	E ↑ LSBE	F ↑ MSBF	F	F	F	F	F	F	*Second byte sent*
	F	F	F	F	F	F	F	F	*Third byte sent*
	F	F	F	F	F	F	F	F ↑ LSBF	*Fourth byte sent*

where

MSBE is the most significant bit of the exponent
LSBE is the least significant bit of the exponent
MSBF is the most significant bit of the fraction
LSBF is the least significant bit of the fraction
S is the sign bit
E is an exponent bit
F is a fraction bit

Example of a Single-Precision Number

A single-precision number could be encoded using these four bytes:

```
01000010   11100000   00000000   00000000
se______   ef______   ________   _______f
```

where

	binary	decimal
s	= 0	= 0
e	= 10000101	= 133
f	= .110	= .75

The number then evaluates to:

$$
\begin{aligned}
v &= (-1)^{s}\,2^{e-127}\,(1.f) \\
&= (-1)^{0}\,2^{6}\,(1.75) \\
&= (64)\,(1.75) \\
&= 112
\end{aligned}
$$

Index

Audio Precision, Inc.
PO Box 2209
Beaverton, Oregon 97075-2209
Tel: (503) 627-0832 Fax: (503) 641-8906
U.S. Toll Free: 1-800-231-7350
email: techsupport@audioprecision.com
Web: www.audioprecision.com