Audio precision®

ATS-2 GPIB
Programmer's Reference Manual

ATS-2
GPIB Programmer's Reference Guide

For GPIB firmware version 1.001

October 2002

Document part number 8211.0152 revision 0.

Audio Precision
5750 SW Arctic Drive
Beaverton, Oregon 97005
Tel: 503-627-0832 Fax: 503-641-8906
US Toll Free: 1-800-231-7350
email: info@audioprecision.com
Web: audioprecision.com

Printed in the United States of America.

II1106104109

Contents

Chapter 1
General Information

Scope of This Manual

This Programmer's Reference Manual describes how to control ATS-2 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 *ATS-2 User's Manual* or other documentation listed below for detailed information about operation of the instrument.

The GPIB firmware in your instrument implements features of ATS control software Version 1.001. Appendix E, "Firmware Information," describes features of the ATS control software that are not available in the GPIB interface firmware at this time. This appendix also provides additional information about specific versions of the firmware and known bugs.

Related Documentation

ATS-2 User's Manual 8211.0135
Contains a comprehensive description of the full capabilities of the ATS control software for ATS-2.

AP BASIC (procedure language) User's Manual and Programmer's Reference
AP Basic Language Reference 8211.0089
AP Basic Language Extensions for ATS-2 . . . . . 8211.0137
These volumes provide detailed descriptions and syntax for all ATS-2 commands.

Getting Started with ATS-2. . . . . 8211.0136
Guides hardware and software installation for ATS-2 for both APIB and GPIB interfaces.

ATS-2 Service Manual 8211.0139
Contains detailed ATS-2 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 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.

ATS-2 Overview

ATS-2 is a comprehensive audio test set that provides analog and digital audio generator and analyzer measurement functions. ATS-2 may be controlled with ATS control software running in a Windows 95/98/ME/NT/2000/XP operating system or with a GPIB controller and application software. The control method is determined by the setting of a rear panel switch.

The ATS-2 GPIB Software Development Process

A practical working knowledge of ATS-2 is required in order to develop effective GPIB software for it. Figure 1-1 illustrates a typical software development scenario in which Audio Precision ATS control software is used to develop expertise with the ATS-2 before attempting to integrate it into a larger system with your GPIB interface software programs.

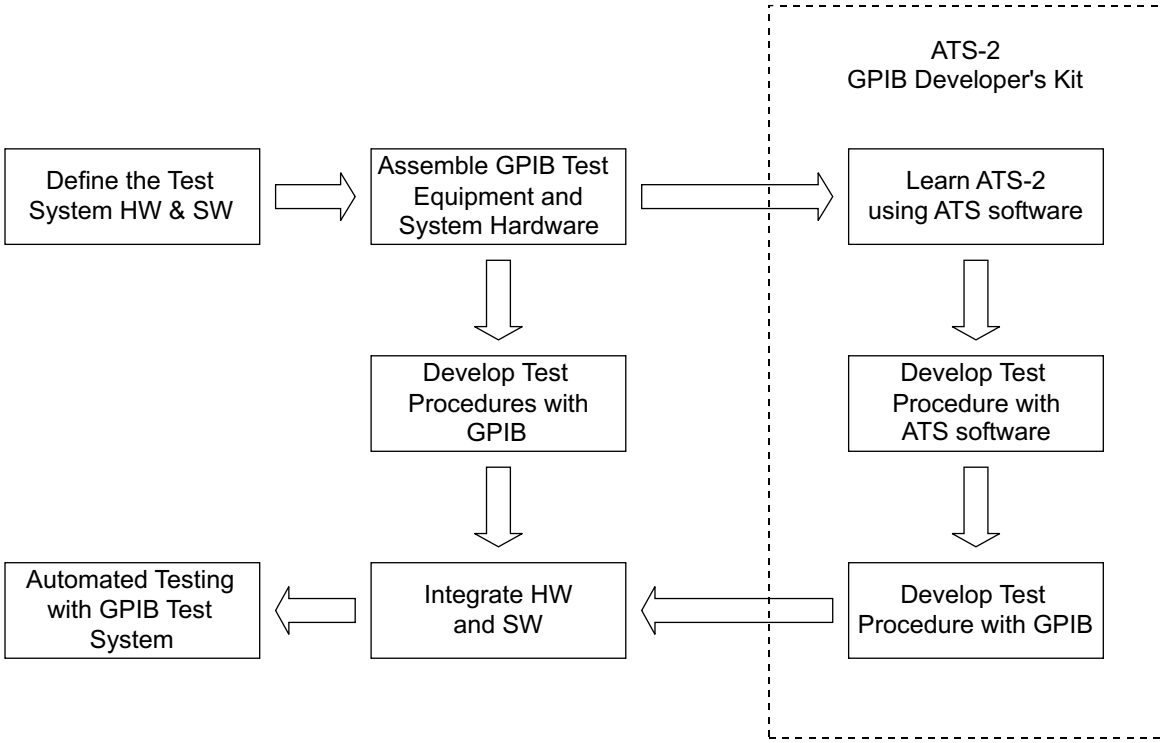


Figure 1-1. Recommended GPIB software development process using ATS control software.

The fastest development scenario for GPIB software involves using the ATS control software with ATS-2. Use AP Basic and the Learn Mode button on the tool bar to develop AP Basic macros that perform the tests you wish to develop for GPIB. If you use the ATS control software to develop your audio tests you will quickly learn how to take the best advantage of

ATS-2. With the interactive environment of ATS 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 ATS 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 ATS setup to GPIB commands. Figure 1-2 illustrates this for the Analog Generator panel in Section 5.

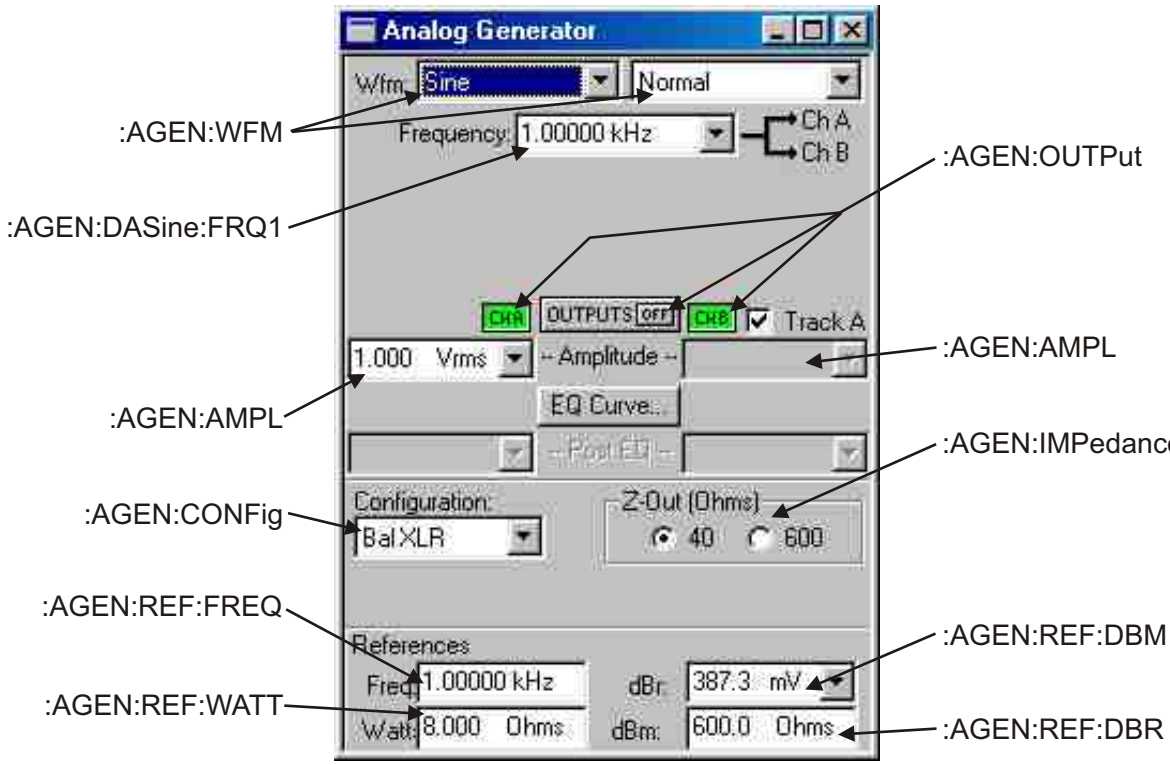


Figure 1-2. ATS-2 Analog Generator Panel with GPIB Command Call-outs.

GPIB and APIB Control Modes for Software Development

Your ATS-2 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 the ATS control software.

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 1-4 illustrates how to connect these products to the ATS-2 and how to connect an APIB controller in a computer to this system when the GPIB port is disabled.

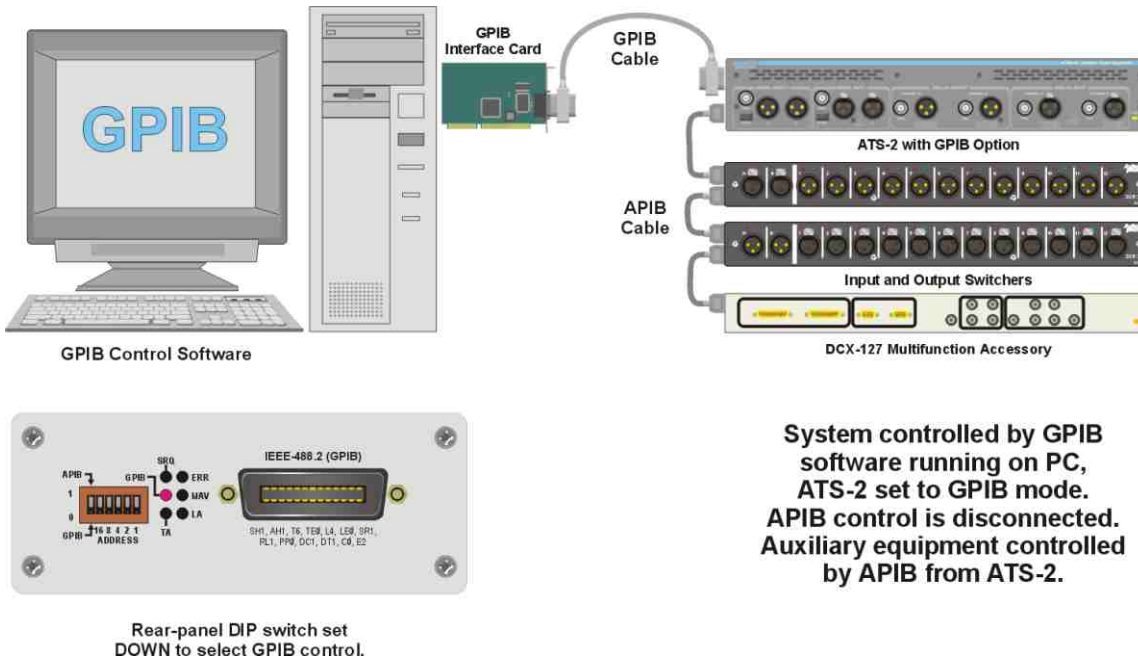


Figure 1-3. APIB connections to ATS-2 with GPIB option in GPIB control mode. SWR-2122 Switcher and DCX-127 connected to the APIB connector of the ATS-2. Computer APIB cable not connected.

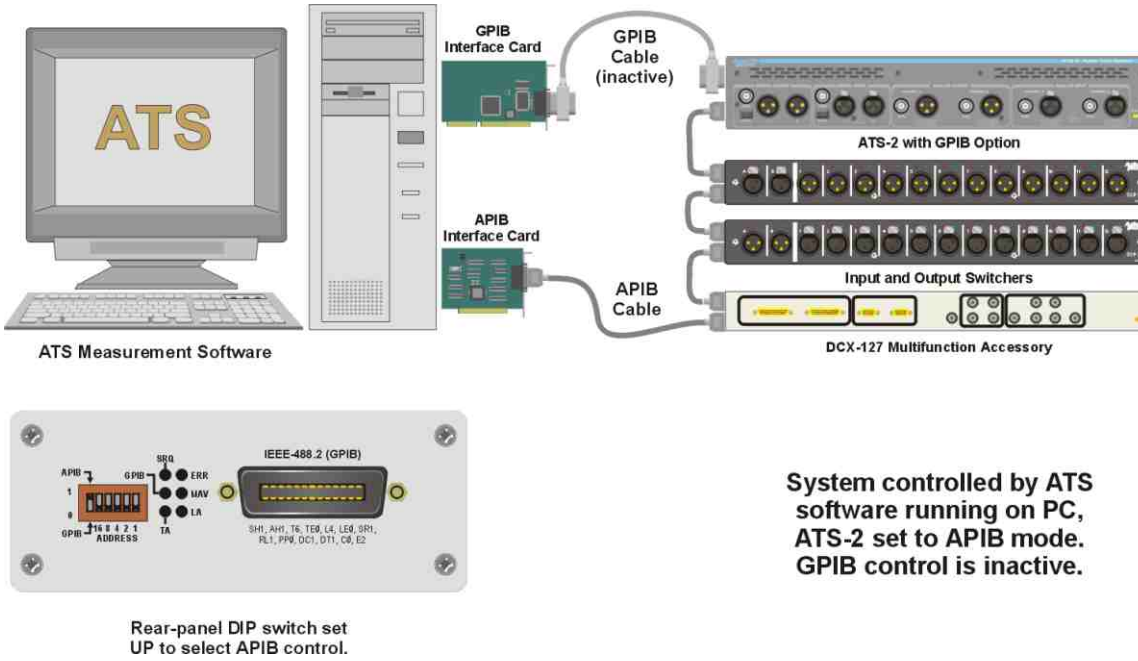


Figure 1-4. APIB controller connections, ATS-2 with GPIB option in APIB control mode. Computer APIB cable connected to DCX-127, then to SWR-2122, then to ATS-2.

Establishing GPIB Communication

GPIB Connection

The ATS-2 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.

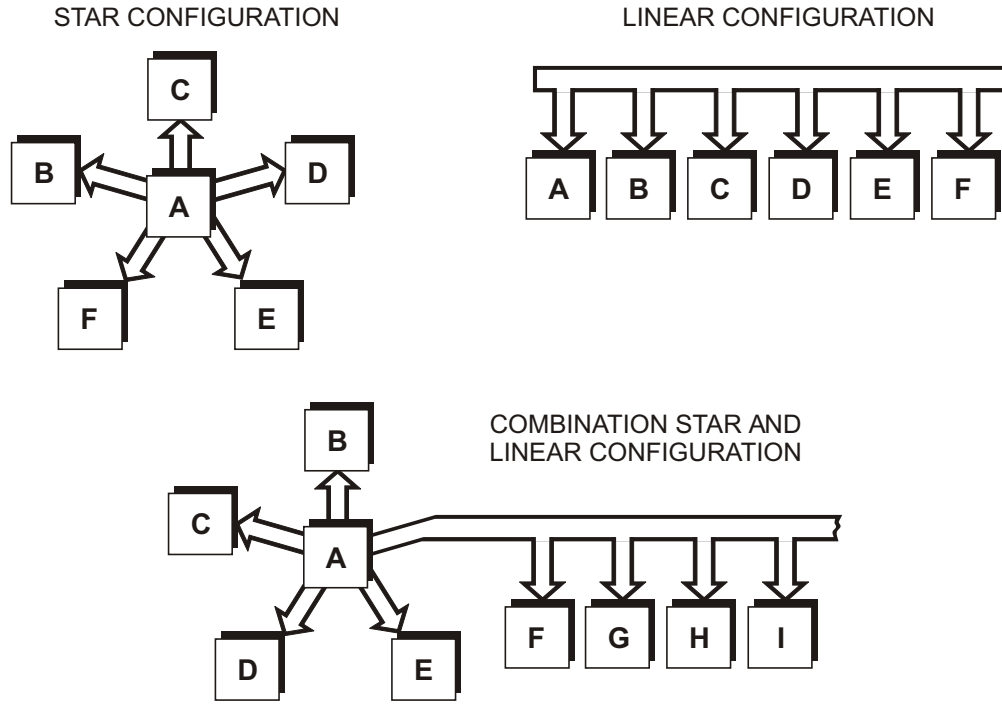


Figure 1-5. GPIB devices may be connected in star, linear, or combination star/linear configurations.

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 ATS-2.

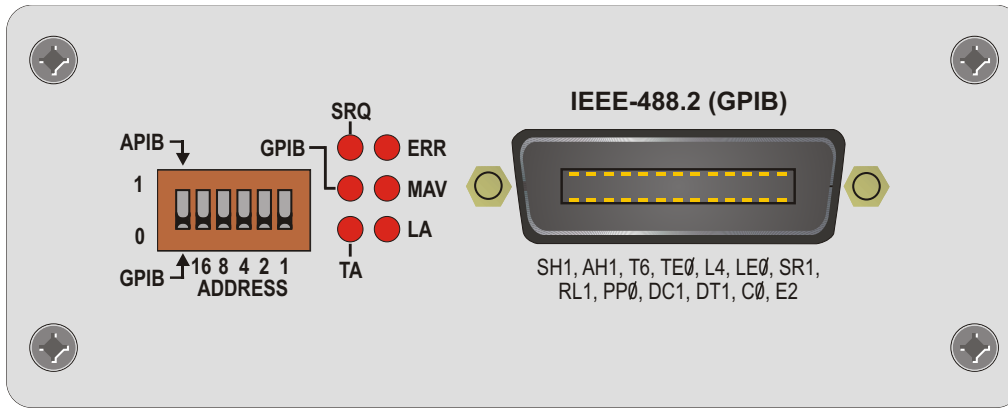


Figure 1-6. ATS-2 Rear Panel Connector for GPIB Option—Address Switch, Status LEDs, and GPIB Port.

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 the 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** below 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. Throughout 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 input coupling command would appear as ":ANLG:COUPling". The short for is ":ANLG:COUP". The long form is ":ANLG:COUPLING".

The instrument does not distinguish between upper and lower case. Therefore the instrument will interpret the command ":ANLG:COUPLING" the same as ":anlg: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.	:ANLG:COUPLING	valid long form
2	:anlg:coup	valid short form
3.	:ANLG: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 that 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 ATS-2 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 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 :ANLG:SOURCE A,GENMON can be concatenated into a single command message: :AGEN:OUTPUT AB;:ANLG:SOURCE 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:DAS:FRQ1 1000HZ into a single command: :AGEN:OUTPUT AB;DAS:FRQ1 1000HZ instead of :AGEN:OUTPUT AB;:AGEN:DAS:FRQ1 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?;DAS:FRQ1? HZ will respond with :AGEN:OUTPUT AB;:AGEN:DAS:FRQ1 1000HZ and the command string :HEADER OFF;:AGEN:OUTPUT?;DAS:FRQ1? HZ will respond with AB;1000HZ.
4. Commands and queries may be concatenated in the same command message. For example: :ANLG:SOURCE A,GENMON;:AGEN:OUTPUT? is a valid message that sets the analog 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 standard IEEE-488.2. Some of these common commands are required by all devices that support the standard. The ATS-2 implements all required common commands and many optional common commands as well.

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 the table below:

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 <PMT> message.

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<PMT>. This type of format does not allow additional messages to be concatenated to the end of the <indefinite length arb block data> argument using the semicolon.

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 E 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, :DSP:DANLR:PHASE? DEG, may respond with :DSP:DANLR:PHASE 9.91E+37DEG,1 if the inputs have no signal.

IEEE-488.1 Interface Functions

The instrument supports the IEEE-488.1 standard interface functions shown in the following table:

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

ATS-2 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).
2. IF an AP defined event occurs THEN an event bit is set in the AESR (7).

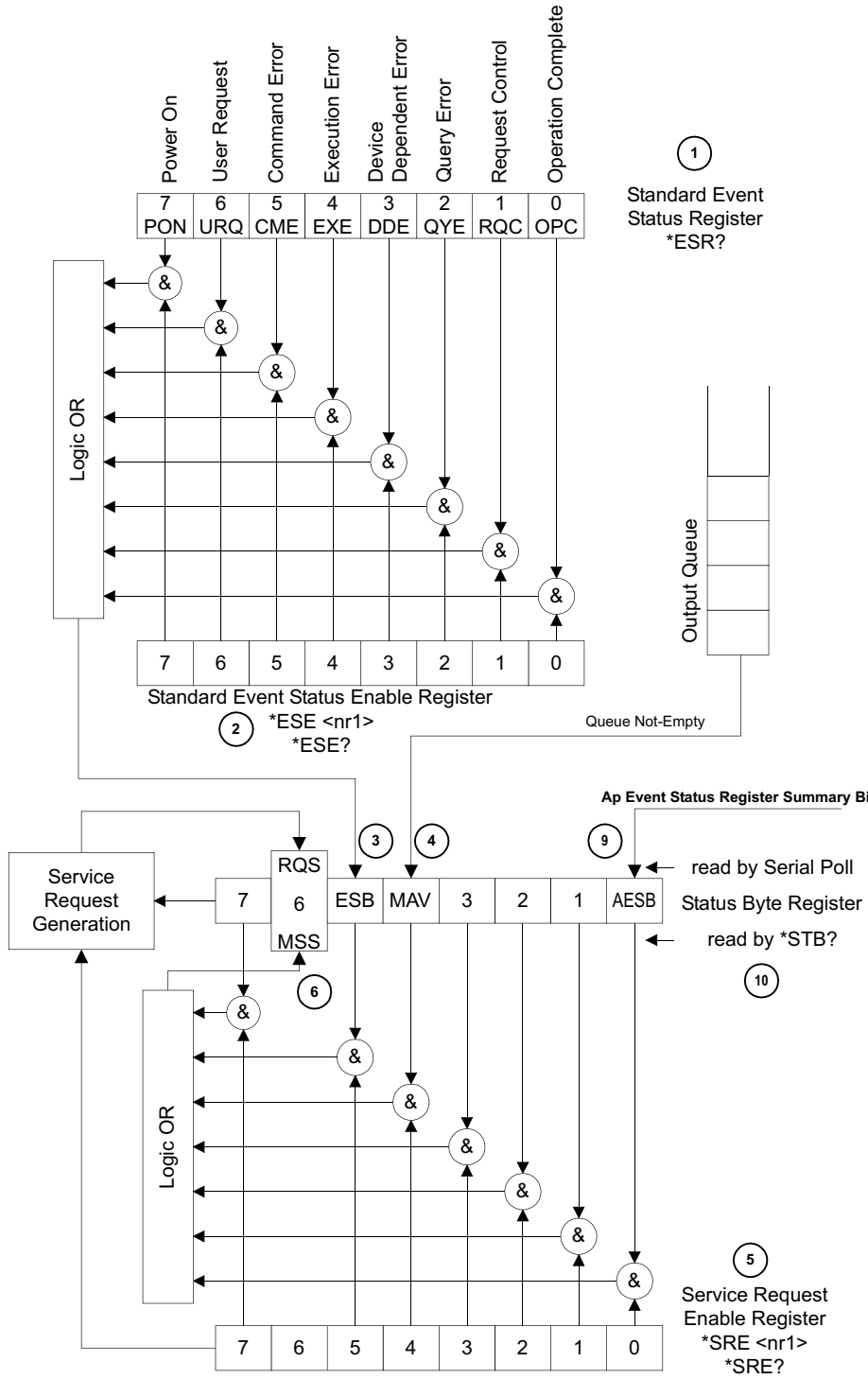


Figure 1-7. IEEE-488 Status Registers and Status Enable Registers.

3. 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.
4. 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.
5. 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.

6. 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.

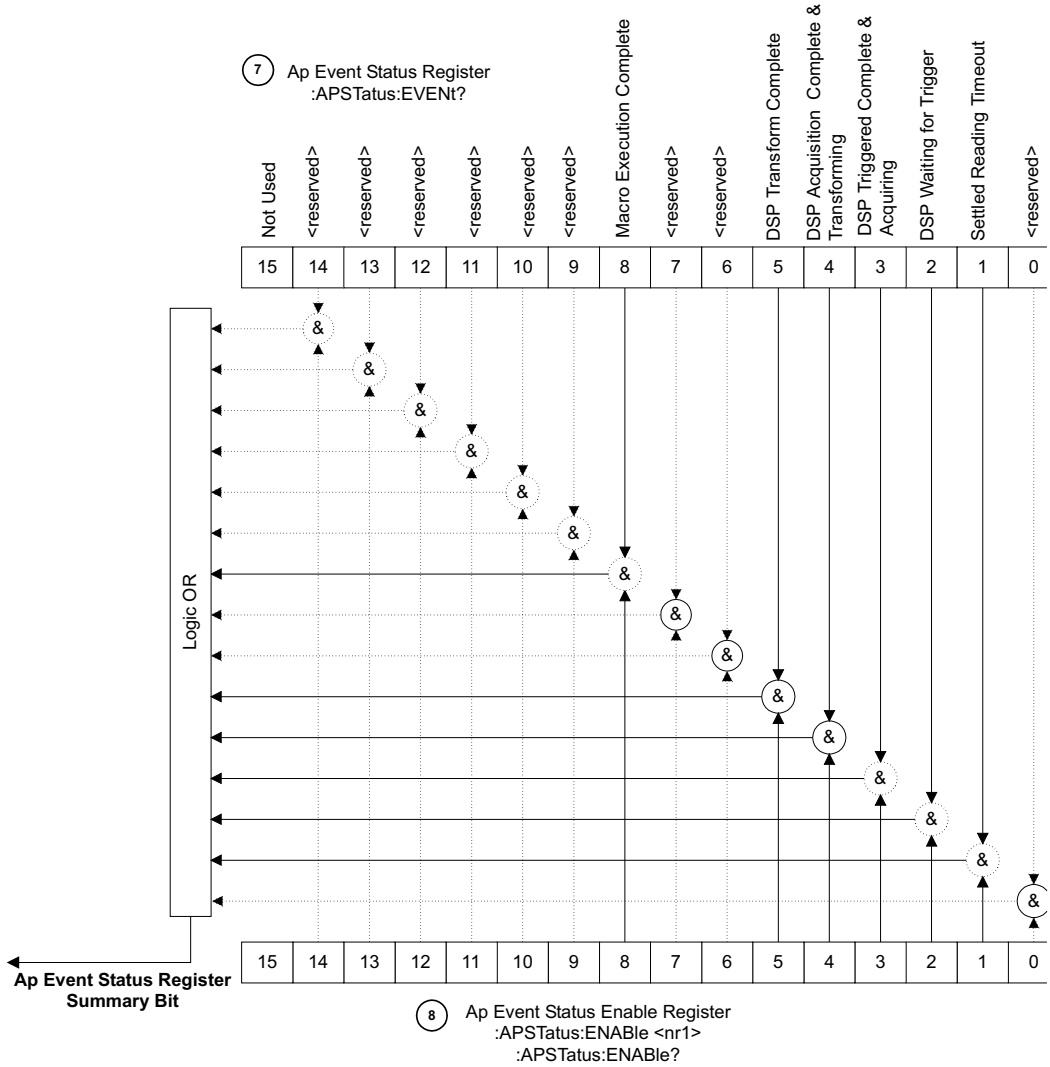


Figure 1-8. AP Event Status Register and Event Status Enable Register.

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

Standard Event Status Register (SESR) Bit Functions	
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.

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						

Status Byte Register (SBR) Bit Functions		
Bit	**Dec Value**	**Function**
7	64	Not used.
6	32	**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	16	**ESB** (Event Status Bit). An enabled event occurred in the SESR.
4		**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.

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.

The 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

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

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.

AP Event Status Enable Register (AESER) 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>

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 ATS-2 may generate in response to error conditions. Errors described in Appendix C are Command Errors or Execution Errors.

Standard Event Status Register Error Codes								
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

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 a 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 SEND, READ, and SERIALPOLL 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
SEND "*ESE 1;*SRE 0"
' Send a level measurement query for the channel A analyzer input level
SEND ":DSP:DANLR: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 ATS-2 GPIB status byte iSPR with a Serial Poll.
    iSPR = SERIALPOLL
    WaitForESB = iSPR And 32 ' True if ESB bit is set
Loop While WaitForESB = False ' Loop While no ESB bit.
' Read the measurement data
LEVEL = READ
' Clear the event status register to clear the OPC bit.
SEND "*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 ATS-2 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 SEND, READ, SERIALPOLL, 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.
SEND "*ESE 1;*SRE 32"
' Send a level measurement query for the channel A analyzer input level
SEND ":DSP:DANLR:LEVEL? A,V;*OPC"
' Wait for a Service Request Interrupt
WaitForSRQ
' Read ATS-2 GPIB status byte iSPR with a Serial Poll.
iSPR = SERIALPOLL
' Read the measurement data if the ESB bit was set in the status byte.
If iSPR And 32 = True Then L
LEVEL = READ
' Clear the event status register to clear the OPC bit.
```

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 SEND, READ, and SERIALPOLL 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
SEND "*ESE 0;*SRE 16"
' Send a level measurement query for the channel A analyzer input level
SEND ":DSP:DANLR: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 ATS-2 GPIB status byte iSPR with a Serial Poll.
    iSPR = SERIALPOLL
    WaitForMAV = iSPR And 16 ' True if MAV bit is set.
Loop While WaitForMAV = False ' Loop While no MAV bit.
' Read the measurement data
LEVEL = READ
```

Chapter 2
Programming Examples

This section provides practical guidelines for writing programs that use unique features of the ATS-2. Each topic is supported with ATS-2 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: "ATS2GSample.bas" and "ATS2G_GPIB_IO.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 "ATS2G Sample" from your Programs menu in your "Start" menu after you have installed the software from the CD ROM. A Visual Basic project file "ATS2GSampleProject.vbp" is also provided with a form file "frmATS2GSample.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\NI-488.2\Languages\Visual Basic).

Error Handling

ATS-2 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:DAS:FRQ1, 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 `:ANLG:RANGE A,-1VP;SOURCE A,XYZ;:ERRS?` is sent, these errors are reported in response to the ERRS? query: **`525,4,":ANLG:RANGE, ANLG, INVALID RANGE VALUE.";502,15,":ANLG:SOURCE, UNKNOWN PARAMETER."`** Note that the

-1VP argument to the **RANGE** command is out of range and the **XYZ** argument to the **SOURCE** command is not a valid setting.

Save, Recall Settings with *SAV and *RCL Commands

ATS-2 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 shown in the Appendix B, **ATS-2 GPIB Default Settings**. 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 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.

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:DAS:FRQ1 $1' is valid, but ':AGEN:DAS:FRQ1 $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 analyzer auto reading rate response frequency and generator frequency and level and send appropriate measurement queries. The variable $1 will specify the analyzer response frequency setting for optimum auto reading rates. The variable $2 will specify the generator channel A output amplitude. The variable $3 will specify the analog generator output frequency. 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 :DSP:DANLR:RESPONSE command, but the second and third macro arguments for the :AGEN:DAS:FRQ1 and :AGEN:AMPL commands require unit suffixes.

Define a macro labeled "GENFRQLVLTHD":

```
*PMC;*EMC 1;*DMC "GENFRQLVLTHD",#0:DSP:DANLR:MODE THDRATIO;RESPONSE
$1;:AGEN:AMPL A,$2;DAS:FRQ1 $3;:DELAY 0.5;:DSP:DANLR:LEV? A,DBV;FREQ?
A,HZ;FUNC? A,PCT
```

Now execute the macro with parameter values as arguments:

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

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 1000,2DBV,1000HZ`** is valid because the macro label **`GENFRQLVLTHD`** is preceded by a colon. However the command message **`:AGEN:OUTPUT AB;GENFRQLVLTHD 1000,2DBV,1000HZ`** 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 ATS-2, 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 #0:AGEN:OUTPUT AB;AMPL A,10DBV;DAS:FRQ1 500HZ;:DSP:DANLR:MODE
THDRATIO;RDGR AUTO,LEVEL,FREQ,FUNCMETER;TUN AGEN;RESP 500;:DELAY
0.75;:DSP:DANLR:LEV? A,DBV;FREQ? A,HZ;FUNC? A,PCT;:AGEN:DAS:FRQ1
100HZ;:DSP:DANLR:RESP 100;:DELAY 0.1;:DSP:DANLR:LEVEL? A,DBV;FREQ?
A,HZ;FUNC? A,PCT
```

Send:

```
*TRG
```

Receive:

```
:DSP:DANLR:LEVEL 9.98352DBV,0;:DSP:DANLR:FREQ 500HZ,0;:DSP:DANLR:FUNCMETER
0.000679858PCT,0;:DSP:DANLR:LEVEL 9.98142DBV,0;:DSP:DANLR:FREQ
100HZ,0;:DSP:DANLR:FUNCMETER 0.000625283PCT,0
```

Save & Recall Waveforms with Batch Mode 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 with the Multitone Analyzer or the FFT Analyzer DSP programs 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. The command **`:HEADER ON;:DSP:DATA? A1`** would result in a response of **`:DSP:DATA A1,#555873`** followed by 55873 bytes of 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 Analyzer 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. This feature is intended for generating synchronous multitone files for use with the Multitone Analyzer. 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. The two generators share the same set of stereo arbitrary waveforms. You can change the arbitrary waveform at any time.

Many multitone arbitrary waveform files are provided with the ATS software provided with ATS-2. These multitone 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 multitone waveform files may be created by a Multitone Creation utility menu in the ATS software. Refer to the ATS-2 User's Manual, "Creating Multitone Waveform Files" for more information.

The process of creating a multitone waveform results in a 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 in the ATS utility. In order to use the multitone arbitrary waveform with the ATS-2 via the GPIB interface, you must

know the frequencies contained in the arbitrary waveform file and you must load the waveform file into the ATS-2 waveform memory.

Managing Generator Waveform Memory

The ATS-2 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, 8192, or 16384. Thus given a sample size of 8192, the ATS-2 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.

Waveform Sample Size	Waveform Byte Count	Maximum Registers
16384	49408	8
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 ATS-2 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 ATS Multitone Creation utility. 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 the Windows Explorer. 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 a 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

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:\ATS\Waveforms\44k\44kSine996-0dB.agm" and "C:\ATS\Waveforms\44k\44kSine996-60dB.agm" into arbitrary waveform registers 1 and 2, and then loads both DSP digital generator buffers with those waveforms. These two waveforms each contain the same frequency sinewave at two different levels. 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:\ATS\Waveforms\44k\44kSine996-0dB.agm" and "C:\ATS\Waveforms\44k\44kSine996-60dB.agm" into registers 1 and 2:

```
:DGEN:ARBSIZE 8192;ARBLOAD
1,#524832<C:\ATS\Waveforms\44k\44kSine996-0dB.agm>
;ARBLOAD 2,#524832<C:\ATS\Waveforms\44k\44kSine996-60dB.agm>
```

The next example sets the output sample rate to 44.1 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 analyzer input to demonstrate that the arbitrary waveform is properly loaded. Both analog and digital generators create the same arbitrary waveforms.

```
:DOUT:RATE 44100HZ;:DGEN:ARBWFM 1,2;OUTPUT AB;WFM ARBITRARY,NONE;AMPL
A,0DBFS;AMPL B,0DBFS;DITHERTYPE TRI;:AGEN:OUTPUT AB;AMPL A,0DBV;AMPL
B,0DBV;WFM DAARBITRARY,NONE;DASR OSR;:MON:SOURCE ABINPUT;VOLUME 50
```

Acquiring Measurements

Measurements may be acquired from measurement meters in the following modules of ATS-2:

- Analog Input
- DCX-127 Multifunction Module
- Audio Analyzer
- Harmonic Distortion Analyzer
- FFT Spectrum Analyzer
- Multitone Audio Analyzer
- Digital Interface Analyzer
- Digital I/O Parameters
- Digital Sync/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 analyzer channel A level meter in DBV units is: **`:DSP:DANLR:LEVEL? A,DBV.`** The response is: **`:DSP:DANLR: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;:DSP:DANLR: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.

Real-time Measurement Meter	Measurement Query	Settling Available
DCX-127 DMM DC Volt Meter	:DCX:DMM? V	Yes
DCX-127 DMM Ohm Meter	:DCX:DMM? O	Yes
DCX-127 Digital Input Data	:DCX:DIN:DATA?	Yes
Analyzer Level A	:DSP:DANLR:LEVEL? A,FFS	Yes
Analyzer Level B	:DSP:DANLR:LEVEL? B,FFS	Yes
Analyzer Frequency A	:DSP:DANLR:FREQ? A,HZ	Yes
Analyzer Frequency B	:DSP:DANLR:FREQ? B,HZ	Yes
Analyzer Function Meter A	:DSP:DANLR:FUNCMETER? A,FFS	Yes

Real-time Measurement Meter	Measurement Query	Settling Available
Analyzer Function Meter B	:DSP:DANLR:FUNCMETER? B,FFS	Yes
Analog Input Audio Peak Level A	:ANLG:PEAK? A,VP	No
Analog Input Audio Peak Level B	:ANLG:PEAK? B,VP	No
Harmonic Distortion Analyzer - Fundamental Ch 1	:DSP:HARMONIC:FAMPLITUDE? 1,FFS	Yes
Harmonic Distortion Analyzer - Fundamental Ch 2	:DSP:HARMONIC:FAMPLITUDE? 2,FFS	Yes
Harmonic Distortion Analyzer - Fundamental Frequency Ch1	:DSP:HARMONIC:FFRQ? 1,HZ	Yes
Harmonic Distortion Analyzer - Fundamental Frequency Ch2	:DSP:HARMONIC:FFRQ? 2,HZ	Yes
Harmonic Distortion Analyzer - Sum1 Ch 1	:DSP:HARMONIC:SUM1? 1,DB	Yes
Harmonic Distortion Analyzer - Sum1 Ch 2	:DSP:HARMONIC:SUM1? 2,DB	Yes
Harmonic Distortion Analyzer - Sum2 Ch 1	:DSP:HARMONIC:SUM2? 1,DB	Yes
Harmonic Distortion Analyzer - Sum2 Ch 2	:DSP:HARMONIC:SUM2? 2,DB	Yes
FFT Spectrum Analyzer Peak Meter 1	:DSP:FFT:PEAK? 1,FFS	No
FFT Spectrum Analyzer Peak Meter 2	:DSP:FFT:PEAK? 2,FFS	No
FASTTEST Multitone Audio Analyzer Peak Meter 1	:DSP:FASTTEST:PEAK? 1,FFS	No
FASTTEST Multitone Audio Analyzer Peak Meter 2	:DSP:FASTTEST:PEAK? 2,FFS	No
Digital Input Peak-to-Peak Interface Pulse 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? UI	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

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.

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

- 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, **`:DSP:DANLR:FREQ? A,DHZ`** requests a frequency measurement (in delta Hz) from the channel A frequency meter (Analyzer). The response could have two forms: **`:DSP:DANLR:FREQ 1000.17HZ,0`** or **`:DSP:DANLR: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 Floor 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 Floor.

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 GPIB interface timeout for the system controller when a read of the ATS-2 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 ATS-2 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 ATS-2. Message units in the ATS-2 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 ATS-2 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 ATS-2. 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 2-1 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 "ATS2G_GPIB_IO.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 0.01 of timeout interval or 0.055 seconds,
    whichever is larger.
  Static iSPR As Integer, fStartTime As Single, fDelay As Single
  fDelay = fTimeout / 100
  If fDelay < 0.055 Then fDelay = 0.055
  fStartTime = Timer
  WaitForESB = False
  Do
    Delay fDelay
    ' Read the ATS-2 status byte iSPR with a Serial Poll.
    ibrsp ATS2G, 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 2-1. Visual Basic WaitForESB Function in file ATS2G_GPIB_IO.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 ATS-2 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/Response Sweeps with Realtime Meters

ATS-2 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 6.0 to illustrate the appropriate ATS-2 GPIB commands and measurement synchronization techniques. This code is provided in the "ATS2Gsample.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 has been executed prior to reading measurement results. This is particularly important when it is desired to read multiple measurement query responses in the ATS-2 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 (for automatic reading rate) and acquires the settled measurements. The

frequency of the distortion notch filter is automatically set to the frequency of the analog generator by the :DSP:DANLR:TUNINGSRC AGEN and :AGEN:DAS:FRQ1 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 generator frequencies from 20 KHz to 20 Hz and measures analyzer channel A frequency in HZ units, Level in dBr units with reference to the level at 1000 Hz, and THD+N Ratio in % units. The test results show the frequency response and distortion of the ATS-2 with XLR cables connected from analog generator outputs to the analog inputs.

```
Sub AGEN_DANLR_FreqRespSwp()
  ' ATS-2
  ' Frequency Response Sweep - Analog Generator Frequency Sweeps with Analog
    Analyzer Measurements
  TEST_TITLE = "AGEN_DANLR_FreqRespSwp"
  ' Send the initial setup commands.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    AINPUT;:AGEN:OUTPUT AB;AMPL A,1V;WFM DASINE,SINE;DAS:FRQ1 1E3HZ"
  GPIBSend ":ANLG:SOURCE AB,XLR;CONVERTER AD1;:DSP:DANLR:INPUT ANLG;MODE
    THDRATIO;DETECTOR RMS;LPF FS_2;TUNINGSRC AGEN;RDGRATE
    AUTO,LEVEL,FREQ,FUNCMETER;RESPONSE 1E3"
  ' SETTLING for FREQ A, LEVEL A, and THD A
  GPIBSend ":SETTLING:DANLR:FUNC A,THDRATIO,NORM,3,1E-5PCT,3,0.0,EXP,0,1"
  GPIBSend ":SETTLING:DANLR:LEVEL CHAA,NORM,3,1E-6V,3,0.0,FLAT,0,1"
  GPIBSend ":SETTLING:DANLR:FREQ A,1,0.01HZ,2,0.01,FLAT,0,1"
  GPIBSend ":SETTLING:TIMEOUT 2"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  Delay 2 ' wait for instrument to settle after initial reset and setup
  ' 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
  GPIBSend "*PMC;*EMC 1;*DMC ""SWPLVLTHD"",#0:DSP:DANLR:RESPONSE
    $1;:AGEN:DAS:FRQ1 $2;:DSP:DANLR:FREQ? A,HZ;LEV? A,DBRA;FUNC? A,DB"
  ' Send the measurement acquisition commands.
  GPIBSend ":AGEN:DAS:FRQ1 1E3HZ;:DELAY 0.2;:DSP:DANLR:LEV?
    A,V;:DSP:REF:SETREFAUTO;:AGEN:DAS:FRQ1 20E3HZ"
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  GPIBSend ":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:DAS:FRQ1 1E3HZ;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(30) = False Then
    UserErrMsg "ATS-2 Measurement Timeout in module " & TEST_TITLE  ' If ESB
    time out then exit.
    GPIBSend "*CLS"   ' Clear status registers.
    Exit Sub ' Quit, do not read response.
  End If
  ' Read the measurement data.
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf & sIO &
    vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ATS2G_Errors
  GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-2. Visual Basic subprogram AGEN_DANLR_FreqRespSwp code.

```
AGEN_DANLR_FreqRespSwp

1.00089V,0

20000HZ,0;-0.00502502DBRA,0;-106.485DB,0;16000HZ,0;-0.00104976DBRA,0;-106.356DB,
0;10000HZ,0;-0.000299919DBRA,0;-105.697DB,0;6299.97HZ,0;-0.00404979DBRA,0;-106.0
21DB,0;4000HZ,0;-0.00329968DBRA,0;-105.967DB,0;2500HZ,0;-0.00262465DBRA,0;-106.1
19DB,0;1600HZ,0;-0.00104976DBRA,0;-105.996DB,0;1000HZ,0;0.0017993DBRA,0;-105.706
DB,0;630HZ,0;-0.000224939DBRA,0;-105.46DB,0;400HZ,0;-0.00164968DBRA,0;-105.704DB
,0;250HZ,0;-0.00269965DBRA,0;-105.854DB,0;160.004HZ,0;-0.00337469DBRA,0;-105.665
DB,0;99.9922HZ,0;-0.00397477DBRA,0;-105.492DB,0;62.9883HZ,0;-0.00247464DBRA,0;-1
05.555DB,0;40.0039HZ,0;-0.00134972DBRA,0;-105.28DB,0;20HZ,0;0.00389801DBRA,0;-10
5.259DB,0;-105.706DB,0;630HZ,0;-0.000224939DBRA,0;-105.46DB,0;400HZ,0;-0.0016496
8DBRA,0;-105.704DB,0;250HZ,0;-0.00269965DBRA,0;-105.854DB,0;160.004HZ,0;-0.00337
469DBRA,0;-105.665DB,0;99.9922HZ,0;-0.00397477DBRA,0;-105.492DB,0;62.9883HZ,0;-0
.00247464DBRA,0;-105.555DB,0;40.0039HZ,0;-0.00134972DBRA,0;-105.28DB,0;20HZ,0;0.
00389801DBRA,0;-105.259DB,0
```

Figure 2-3. Visual Basic subprogram AGEN_DANLR_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 show the output amplitude linearity of the ATS-2 analog generator with XLR cables connected from analog generator outputs to the analog inputs.

```
Sub AGEN_DANLR_GainLinSwp()
  ' Gain Linearity Sweep - Analog Generator Amplitude Sweeps with Analog
    Analyzer Measurements
  TEST_TITLE = "AGEN_DANLR_GainLinSwp"
  ' Send the initial setup commands.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    AFUNC;:AGEN:OUTPUT AB;AMPL A,0DBV;WFM DASINE,SINE;DAS:FRQ1 1E3HZ"
  GPIBSend ":DSP:DANLR:AUTORANGE AB,ON;FAUTORANGE AB,ON;INPUT ANLG;MODE
    BP;DETECTOR RMS;TUNINGSRC FIXED;FILTERFREQ 1E3HZ;RDGRATE R32;RESPONSE
    1E3;:ANLG:SOURCE AB,XLR" ' Add DANLR Settling for THD
  ' SETTLING for Level A meter and Bandpass A function meter
  GPIBSend ":SETTLING:DANLR:FUNC A,BPA,NORM,3,1E-8V,3,0.01,EXP,0,1"
  GPIBSend ":SETTLING:DANLR:LEVEL CHAA,NORM,0.5,1E-6V,3,0.1,FLAT,0,1"
  GPIBSend ":SETTLING:TIMEOUT 2"
  Delay 2 ' wait for instrument to settle after initial reset and setup
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  ' Send a message that purges all macro definitions, enables all macros, and
    defines a sweep macro for generator output amplitude and analyzer
    measurement queries of level and bandpass function meter
  GPIBSend "*PMC;*EMC 1;*DMC ""SWPGAINLIN"",#0:AGEN:AMPL AB,$1;:DSP:DANLR:LEVEL?
    A,DBV;FUNC? A,DBGA"
  ' pre-set the analog generator output level to improve first point measurement
    accuracy and in order to avoid need for a longer settling delay for first
    swept point
  GPIBSend "AGEN:AMPL AB,10DBV"
  Delay 0.2 ' first point pre-sweep delay
  ' Send the measurement acquisition commands.
  GPIBSend "SWPGAINLIN 10DBV;SWPGAINLIN 0DBV;SWPGAINLIN -10DBV;SWPGAINLIN
    -20DBV;SWPGAINLIN -30DBV;SWPGAINLIN -40DBV;SWPGAINLIN -50DBV;SWPGAINLIN
    -60DBV;SWPGAINLIN -70DBV;SWPGAINLIN -80DBV;SWPGAINLIN -90DBV;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(60) = False Then
    UserErrMsg "ATS-2 Measurement Timeout in module " & TEST_TITLE  ' If ESB
    time out then exit.
    GPIBSend "*CLS"   ' Clear status registers.
    Exit Sub ' Quit, do not read response.
  End If
  ' Read the measurement data.
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ATS2G_Errors      ' Check for ATS-2 errors.
  GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit.
End Sub
```

Figure 2-4. Visual Basic subprogram AGEN_DANLR_GainLinSwp code.

```
AGEN_DANLR_GainLinSwp

9.99986DBV,0;-0.0257436DBGA,0;0.00383454DBV,0;0.0212866DBGA,0;-9.98721DBV,0;0.01
7916DBGA,0;-20.0102DBV,0;0.000898811DBGA,0;-30.0112DBV,0;0.00646303DBGA,0;-39.99
43DBV,0;0.0256734DBGA,0;-49.983DBV,0;0.0343302DBGA,0;-59.9944DBV,0;0.0301239DBGA
,0;-69.9959DBV,0;0.0317086DBGA,0;-79.9755DBV,0;0.0337146DBGA,0;-90.0261DBV,0;-0.
0105677DBGA,0
```

Figure 2-5. Visual Basic subprogram AGEN_DANLR_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 **:DSP:DANLR:RDGRATE AUTO** and **:DSP:DANLR:TUNINGSRC FIXED** commands. The **:DSP:DANLR:RDGRATE AUTO** command specifies automatic reading rate for the function meter (the measurement of audio level at the output of the bandpass filter). The **:DSP:DANLR:TUNINGSRC FIXED** command specifies fixed tune mode for the bandpass filter. These two commands in conjunction with the **:DSP:DANLR: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 **:DSP:DANLR:RESPONSE n;FUNC? A,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 show the combined

```
Sub AGEN_DANLR_BPFRNoiseSwp()
  ' Noise Sweep - Analog Analyzer Bandpass Filter Frequency Sweeps
  TEST_TITLE = "AGEN_DANLR_BPFRNoiseSwp"
  ' Send the initial setup commands
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    AFUNC;:AGEN:OUTPUT OFF;AMPL A,0V;WFM DANOISE"
  GPIBSend ":DSP:DANLR:INPUT ANLG;MODE BP;DETECTOR RMS;LPF FS_2;TUNINGSRC
    FIXED;RDGRATE AUTO,FUNCMETER;:ANLG:SOURCE AB,XLR;CONVERTER AD1"
  ' SETTLING for Level A meter and Bandpass A function meter for averaged noise
    measurements
  GPIBSend ":SETTLING:DANL:FUNC A,BPA,NORM,3,1E-8V,3,0.03,AVG,0,1"
  GPIBSend ":SETTLING:DANLR:LEVEL CHAA,NORM,1,1E-6V,3,0.03,AVG,0,1"
  GPIBSend ":SETTLING:TIMEOUT 2"
  Delay 2 ' wait for instrument to settle after initial reset and setup
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  ' Send a message that purges all macro definitions, enables all macros, and
    defines a sweep macro for analyzer measurement query of bandpass function
    meter
  GPIBSend "*PMC;*EMC 1;*DMC ""SWPNOISE"",#0:DSP:DANLR:RESPONSE $1;FILTERFREQ?
    HZ;FUNC? A,DBV"
  ' Pre-tune the bandpass filter to frequency of first sweep point in order to
    avoid need for a longer settling delay for first swept point
  GPIBSend ":DSP:DANLR:RESPONSE 20E3" ' or :DSP:DANLR:FILTERFREQ 20E3HZ
  Delay 0.2 ' first point pre-sweep delay
  ' Send the measurement acquisition commands.
  GPIBSend "SWPNOISE 20E3;SWPNOISE 16E3;SWPNOISE 10E3;SWPNOISE 6.3E3;SWPNOISE
    4E3;SWPNOISE 2.5E3;SWPNOISE 1.6E3;SWPNOISE 1E3;SWPNOISE 630;SWPNOISE
    400;SWPNOISE 250;SWPNOISE 160;SWPNOISE 100;SWPNOISE 63;SWPNOISE 40;SWPNOISE
    20;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(60) = False Then
    UserErrMsg "ATS-2 Measurement Timeout in module " & TEST_TITLE  ' If ESB
    time out then exit.
    GPIBSend "*CLS"   ' Clear status registers.
    Exit Sub ' Quit, do not read response.
  End If
```

Figure 2-6. Visual Basic subprogram AGEN_DANLR_BPFRNoiseSwp code.

```
  ' Read the measurement data.
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ATS2G_Errors       ' Check for ATS-2 errors.
  GPIBSend "*CLS"    ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-6 (cont.). Visual Basic subprogram AGEN_DANLR_BPFRNoiseSwp code.

noise performance of the ATS-2 analog generator and analyzer combined, with XLR cables connected from analog generator outputs to the analog inputs.

```
AGEN_DANLR_BPFRNoiseSwp


20000HZ;-96.8341DBV,0;16000HZ;-131.279DBV,0;10000HZ;-132.836DBV,0;6300HZ;-135.40
6DBV,0;4000HZ;-138.702DBV,0;2500HZ;-140.254DBV,0;1600HZ;-140.831DBV,0;1000HZ;-14
4.606DBV,0;630HZ;-148.17DBV,0;400HZ;-148.763DBV,0;250HZ;-149.756DBV,0;160HZ;-154
.038DBV,0;100HZ;-152.758DBV,0;63HZ;-159.196DBV,0;40HZ;-113.046DBV,0;20HZ;-177.59
1DBV,0
```

Figure 2-7. Visual Basic subprogram AGEN_DANLR_BPFRNoiseSwp measurement results.

Digital Generator Frequency Sweeps with Digital Audio Analyzer Measurements

To sweep the digital generator output frequency and measure digital audio 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:FRQ1`** 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 show the frequency response and distortion of the

```
Sub DGEN_DANLR_FreqRespSwp()
  ' Digital Generator Frequency Sweeps with Digital Analyzer Measurements
  TEST_TITLE = "DGEN_DANLR_FreqRespSwp"
  ' Send the initial setup commands.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    ABINPUTSUM;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;FRQ1 1E3HZ;WFM SINE,SINE"
  GPIBSend ":DIN:FORMAT XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;INVALID
    0;JWFM NONE;PREEMPHASIS OFF;RATE 48000HZ;RESOLUTION 24,BITS"
  GPIBSend ":DSP:REF:DBR1 -10DBFS;DBR2 -10DBFS;:DSP:DANLR:INPUT
    DIGITAL;FAUTORANGE AB,ON;DETECTOR RMS;COUPLING AB,AC;MODE THDRATIO;LPF
    FS_2;TUNINGSRC DGEN;RDGRATE AUTO,LEVEL,FREQ,FUNCMETER;RESPONSE 1E3;:DELAY
    0.2"
  ' SETTLING for FREQ A, LEVEL A, and THD A
  GPIBSend ":SETTLING:DANLR:FUNC A,THDRATIO,NORM,3,1E-5PCT,3,0.03,EXP,0,1"
  GPIBSend ":SETTLING:DANLR:LEVEL CHAD,NORM,1,1E-6V,3,0.03,FLAT,0,1"
  GPIBSend ":SETTLING:DANLR:FREQ A,0.5,0.01HZ,3,0.002,FLAT,0,1"
```

Figure 2-8. Visual Basic subprogram DGEN_DANLR_FreqRespSwp code.

```
  GPIBSend ":SETTLING:TIMEOUT 2"
  Delay 2 ' wait for instrument to settle after initial reset and setup
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  ' Set analyzer DBR1 reference to channel A input level
  GPIBSend ":DSP:DANLR:LEV? A,DBFS;:DSP:REF:SETREFAUTO"
  ' Read the measurement data.
  sIO = GPIBRcv
  ' 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.
  GPIBSend "*PMC;*EMC 1;*DMC ""DIGSWPLVLTHD"",#281:DSP:DANLR:RESPONSE
    $1;:DGEN:FRQ1 $2;:DSP:DANLR:FREQ? A,HZ;LEV? A,DBR1;FUNC? A,DB"
  ' Send the measurement acquisition commands.
  GPIBSend ":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(20) = False Then
    UserErrMsg "ATS-2 Measurement Timeout in module " & TEST_TITLE  ' If ESB
    time out then exit.
    GPIBSend "*CLS"   ' Clear status registers.
    Exit Sub ' Quit, do not read response.
  End If
  ' Read the measurement data.
  sIO = sIO & vbCrLf & GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ATS2G_Errors      ' Check for ATS-2 errors.
  GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-8 (cont.). Visual Basic subprogram DGEN_DANLR_FreqRespSwp code.

ATS-2 with XLR cables connected from the digital XLR output to the digital XLR input connector I.

```
DGEN_DANLR_FreqRespSwp

-9.99966DBFS,0
19999.9HZ,0;0.000370427DBR1,0;-132.701DB,0;16000HZ,0;0.000370427DBR1,0;-132.697D
B,0;10000HZ,0;0.000370427DBR1,0;-132.284DB,0;6300.01HZ,0;0.000370427DBR1,0;-131.
924DB,0;4000HZ,0;0.000296343DBR1,0;-131.812DB,0;2499.99HZ,0;0.000370427DBR1,0;-1
31.426DB,0;1600HZ,0;0.000370427DBR1,0;-131.517DB,0;999.996HZ,0;0.000370427DBR1,0
;-131.582DB,0;629.997HZ,0;0.000370427DBR1,0;-131.201DB,0;400HZ,0;0.000370427DBR1
,0;-131.457DB,0;249.999HZ,0;0.000296343DBR1,0;-131.568DB,0;160HZ,0;0.000148173DB
R1,0;-131.325DB,0;100.001HZ,0;-0.000222264DBR1,0;-131.498DB,0;62.9997HZ,0;0.0014
0754DBR1,0;-129.692DB,0;40HZ,0;0.00348139DBR1,0;-129.841DB,0;20.0014HZ,0;0.01140
19DBR1,0;-129.528DB,0
```

Figure 2-9. Visual Basic subprogram DGEN_DANLR_FreqRespSwp measurement result.

Measurement Sweeps with Batch Mode Programs

FFT, FASTTEST and INTERVU

The batch mode DSP programs FFT and FASTTEST acquire and process stereo waveforms from either the digital audio inputs or from the analog inputs via 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 batch mode 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 before 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 a change to the READING state, and then initiated with one of three commands: ACQX?, XFRM?, and REPROCESS?. The results of acquisition and processing is arrays of measurement data values stored within the ATS-2 DSP engine. The SRCPARAMS command specifies which type of data values are to be provided in response to specific queries, the range of measurement 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 commands (:DSP:BATCh? or :DSP:MEASurement?). If the response is 1, then the process has been aborted due to a timeout and no measurements will be available. The response to these queries must be checked before proceeding to read the data.

Two techniques may be used to read the data available from the DSP engine after a successful acquire, transform, or reprocess: the :DSP:BATCh? query or the :DSP:MEASurement? query. Each DSP program has a specific set of measurements. The :DSP:MEASurement? query provides the first sweep point specified by :DSP:SRCParams. This first point will be the first point in a sweep table if ARB is specified as the last parameter to the :DSP:SRCParams command. The measurements to be read must be specified by the :DSP:OPSTate READ parameter list. The :DSP:MEASurement? query is useful if only one data point is required and a simple response format is desired.

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 fast because the internal microprocessor in the ATS-2 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 response: 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 ATS-2 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 ATS-2 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 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.

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? (Or :DSP:XFRM? or :DSP:REPRocess?).
5. Read the response to :DSP:ACQX? (Or :DSP:XFRM? or :DSP:REPRocess?) and test for 0 it to determine that the acquisition was successful. If successful then proceed, else halt.
6. Use the or the :DSP:MEASurement? or :DSP:BATCh? query to read the measurement(s). These specify the values 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 not only specifies the sweep intervals but also specifies the use of a peak-picking algorithm for frequency domain spectrum sweeps. This parameter is ignored if a time sweep or probability sweep (INTervu pulse and jitter statistics) is specified. The arguments for the last parameter may be ARB, LIN, or LOG. ARB must be used with a sweep table (see :DSP:TABLe and :DSP:TSELect) and disables the peak-pick algorithm. Use ARB if the application requires measurements of specific bins only and if you intend to use a sweep table.

The peak-picking algorithm is enabled with the use LIN or LOG for the last SRCParams argument. 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. 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 next frequency requested. To illustrate this, Figure 2-14 shows a spectrum of a 1010.74 Hz signal driving an analog device at the clipping point. Distortion components appear at the expected 2021.48 Hz second harmonic, intermodulation products are present, and 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:BATCH? ASCII,AMP1,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. 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.

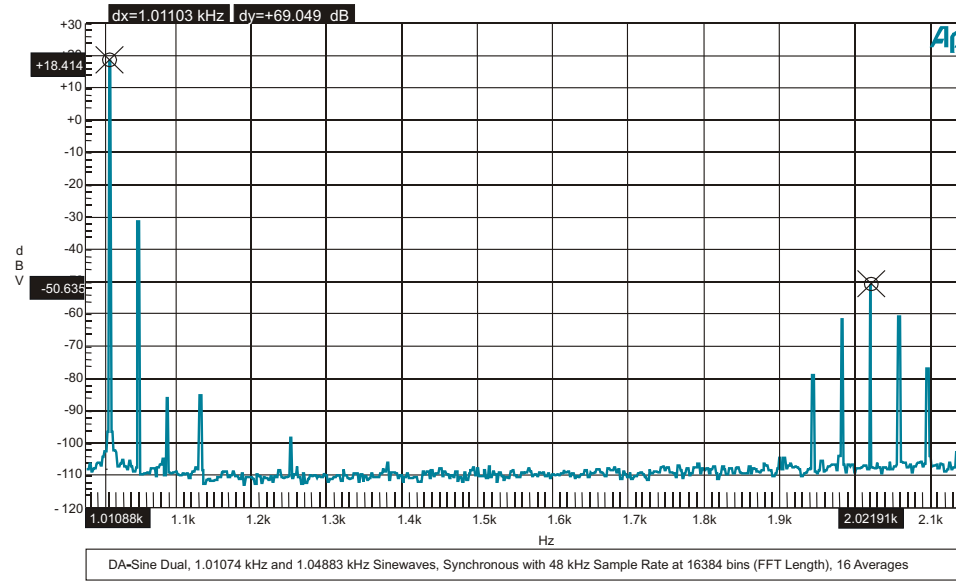


Figure 2-10. Spectrum of signal with distortion components, fundamental at 1010.74 Hz (cursor 1) with second harmonic at 2021.48 Hz (cursor 2).

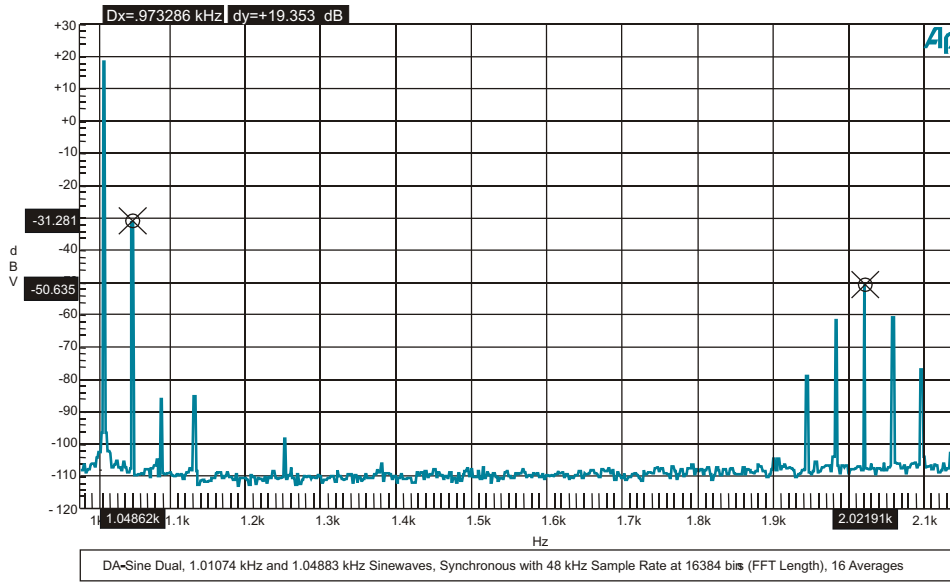


Figure 2-11. Spectrum of signal with distortion components, 1048.83 Hz interference signal at -31.281 dBV (cursor 1) is the highest signal between the fundamental at 1010.74 Hz and its second harmonic at 2021.48 Hz (cursor 2).

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 an array of measurements. The process requires an initial setup, followed by the acquisition of the signal, and then followed by a :DSP:BATCh? query.

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 bin frequencies specified by the :DSP:SRCParams command. The spectrum of interest is 984 Hz to 1017 Hz in order to look at the 1 kHz fundamental signal from the ATS-2 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 waveform buffers. The spectrum data of the new input signal is processed. The data is not displayed, as would be required to prove that the originally acquired signal has been overwritten.

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. This sweep is done with the LOG argument for SRCParams in order to illustrate how the frequency bin spacing is different from LIN. 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.

The :DSP:BATCh? query utilizes 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

```
Sub FFT_SpectrumSwp() ' Acquire new waveform, save waveform to file and recall
    from file into DSP acquisition buffer, then re-transform it
  ' Spectrum Sweeps with the FFT DSP Program
  ' Acquire the signal on digital audio input channel 1.
  TEST_TITLE = "FFT_SpectrumSwp"
  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.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    AINPUT;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;FRQ1 1E3HZ;WFM SINE,SINE;:DIN:FORMAT
    XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;INVALID 0;JWFM
    NONE;PREEMPHASIS OFF;RATE 48000HZ;RESOLUTION 24,BITS"
  GPIBSend ":DSP:PROGRAM FFT;:DSP:FFT:INPUT DIGITAL;ACQLENGTH XLENGTH;AVGS
    1;AVGTYPE 1;COUPLING DC;DELAY 0;MODE INTRPOLATE;START 0;TRIGGER FREE;TSLOPE
    POS;WINDOW EQR;XLENGTH 16384;PKTRIG 1;:DELAY 0.2"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  ' 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.
  GPIBSend ":DSP:OPSTATE SETUP;TIMEOUT 2;SRCP FREQ,HZ,10,10000,9,LIN;OPSTATE
    READ,AMP1;ACQX?;*OPC"
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS" ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If

  ' Save acquired DSP waveforms to a file
  SaveWaveform 1, App.Path & "\ATS2GPIBWav1.aas" ' Save waveform in FFT
    acquisition buffer 1 to a file.
  ATS2G_Errors ' Check for ATS-2 errors.

  ' Acquire a noise signal
  GPIBSend ":DGEN:OUTPUT OFF"
  GPIBSend ":DSP:OPSTATE SETUP;TIMEOUT 2;SRCP FREQ,HZ,10,10000,9,LIN;OPSTATE
    READ,AMP1;ACQX?;*OPC"
```

Figure 2-12. Visual Basic subprogram FFT_SpectrumSwp using the ASCII format for the BATCH? query.

```
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then    ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS" ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If

  ' Retrieve acquired DSP waveforms from the same file
  ' Recall the DSP acquisition waveform from the file it was saved in and
    transform its spectrum.
  RecallWaveform App.Path & "\ATS2GPIBWav1.aas"   ' Recall waveform from a file
    and put into DSP acquisition buffer.
  ' Transform the waveform in the DSP acquisition buffers.
  GPIBSend ":HEADER OFF;:DSP:OPSTATE READ,AMP1;XFRM?;*OPC"
  ' Read the response to the XFRM? query.
  sIO = GPIBRcv
  If sIO <> "0" Then    ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If

  ' Send amplitude measurement queries for channel 1 spectrum bins.
  GPIBSend ":DSP:BATCH? ASCII,AMP1,DBFS"
  ' Read the measurement query responses.
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & "New FFT Acquisition #1 with LIN bin freq
    spacing (with peak-picking)."
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "Format = FREQ, AMP1" & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ' Setup the BATCH sweep parameters to sweep from 10 Hz to 10 KHz with 9 steps
    (10 bins) with LIN spacing.
  GPIBSend ":DSP:OPSTATE SETUP;SRCP FREQ,HZ,10,10000,9,LOG;OPSTATE
    READ,AMP1;REPROCESS?;*OPC"
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then    ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
    ' Send amplitude measurement queries for channel 1 spectrum bins.
  GPIBSend ":DSP:BATCH? ASCII,AMP1,DBFS"
  ' Read the measurement query responses.
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "Reprocess FFT Acquisition #1 with LOG bin freq spacing (with
    peak-picking)."
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "Format = FREQ, AMP1" & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  GPIBSend ":DSP:OPSTATE SETUP"
  ATS2G_Errors       ' Check for ATS-2 errors.
  GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-12 (cont.). Visual Basic subprogram FFT_SpectrumSwp using the ASCII format for the BATCH? query.

returned in the response to the BATCH? query in three possible formats: ASCII, BINARY, and RBINARY.

```
FFT_SpectrumSwp

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

ASCII,10,10,-175.038,1120,-10.2149,2230,-165.983,3340,-166.656,4450,-168.172,556
0,-168.486,6670,-168.231,7780,-166.284,8890,-167.487,10000,-165.739

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

ASCII,10,10,-175.038,21.5443,-171.588,46.4159,-171.015,100,-170.535,215.443,-168
.046,464.159,-168.461,1000,-10.2149,2154.43,-10.8633,4641.59,-166.656,10000,-165
.739
```

Figure 2-13. Visual Basic subprogram FFT_SpectrumSwp measurement results using the ASCII format for the BATCH? query.

The BATCH? Binary Formats

The RBINARY and BINARY binary data formats with the :DSP:BATCh? query provide significantly faster read speeds from the instrument to the GPIB controller. 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 VB code below for the subprogram FFT_StereoSpectrumSwp_RBinary_To_Array shows code that acquires stereo spectrum data. This type of sweep is approximately 2.6 times faster than the ASCII format. The subprogram ReadBatch2 performs the read and parsing of the RBINARY formatted response into a dynamic single float array structure that contains the bin

```
Sub FFT_StereoSpectrumSwp_RBinary_To_Array()
  ' 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 = "FFT_StereoSpectrumSwp_RBinary_To_Array"
  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.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    ABINPUTSUM;:AGEN:AMPL A,1V;AMPL B,1V;WFM DAS,SINE;OUTPUT AB;:DSP:DANLR:INPUT
    ANLG;AUTORANGE A,ON;AUTORANGE B,ON;:ANLG:SOURCE AB,XLR;CONVERTER AD1"
  GPIBSend ":DSP:PROGRAM FFT;:DSP:FFT:XLENGTH 2048;ACQLENGTH XLENGTH;AVGS
    1;AVGTYPE 1;COUPLING SUBH;DELAY 0;MODE INTRPOLATE;START 0;TRIGGER
    FREE;WINDOW EQR;PKTRIG 1;:DELAY 0.2"
  ' Wait for instrument outputs and inputs to settle before acquiring the
    spectrum.
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  Delay 2
```

Figure 2-14. Visual Basic subprogram FFT_StereoSpectrumSwp_RBinary_To_Array code

```
  ' 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.
  GPIBSend ":DSP:OPSTATE SETUP;TIMEOUT 2;SRCP FREQ,HZ,10,22500,999,LOG;OPSTATE
    READ,AMP1,AMP2;ACQX?;*OPC"
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  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.
    frmATS2GSample.txtOutput.Text = frmATS2GSample.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
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "First element array values are: " & vbCrLf & "fSrcCh1Ch2(SRC, 0) = " &
    fSrcCh1Ch2(Src, 0)
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "fSrcCh1Ch2(CH1, 0) = " & fSrcCh1Ch2(Ch1, 0)
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "fSrcCh1Ch2(CH2, 0) = " & fSrcCh1Ch2(Ch2, 0)
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "Highest Signal Level found on Channel 1:" & vbCrLf & "fSrcCh1Ch2(SRC, " &
    lMaxIndex & ") = " & fSrcCh1Ch2(Src, lMaxIndex)
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "fSrcCh1Ch2(CH1, " & lMaxIndex & ") = " & fSrcCh1Ch2(Ch1, lMaxIndex)
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "fSrcCh1Ch2(CH2, " & lMaxIndex & ") = " & fSrcCh1Ch2(Ch2, lMaxIndex)
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "Last element array values are: " & vbCrLf & "fSrcCh1Ch2(SRC, " & lPoints &
    ") = " & fSrcCh1Ch2(0, lPoints)
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "fSrcCh1Ch2(CH1, " & lPoints & ") = " & fSrcCh1Ch2(Ch1, lPoints)
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "fSrcCh1Ch2(CH2, " & lPoints & ") = " & fSrcCh1Ch2(Ch2, lPoints) & vbCrLf
  Else ' Display error message
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & "FFT - Spectrum acquisition failed." & vbCrLf
  End If
  ' Set display to last line
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  GPIBSend ":DSP:OPSTATE SETUP"
  ATS2G_Errors       ' Check for ATS-2 errors.
  GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-14 (cont.). Visual Basic subprogram FFT_StereoSpectrumSwp_RBinary_To_Array code.

```
FFT_StereoSpectrumSwp_RBinary_To_Array

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
fSrcCh1Ch2(CH1, 0) = -106.8371
fSrcCh1Ch2(CH2, 0) = -103.6862
Highest Signal Level found on Channel 1:
fSrcCh1Ch2(SRC, 593) = 976.8639
fSrcCh1Ch2(CH1, 593) = -0.1115728
fSrcCh1Ch2(CH2, 593) = -9.896091E-02
Last element array values are:
fSrcCh1Ch2(SRC, 999) = 22500
fSrcCh1Ch2(CH1, 999) = -135.8004
fSrcCh1Ch2(CH2, 999) = -132.2791
```

Figure 2-15. Visual Basic subprogram FFT_StereoSpectrumSwp_RBinary_To_Array measurement results.

frequency and magnitudes of analog input channels A and B. The output data 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.

The code below for the VB subprogram FFT_THDSpectrumSwp_BinaryTable_To_Array shows use of the BINARY waveform format with BATCH? and uses a sweep table to specify

```
Sub FFT_THDSpectrumSwp_BinaryTable_To_Array()
  ' Harmonic Spectrum Sweeps with the FFT DSP Program. Places harmonic
    amplitudes into a numeric array.
  ' Generates a frequency table for spectrum measurements.
  ' Optional code (commented) reads a frequency table from a *.ATSX file
    generated by ATS software.
  ' Uses the MakeTable function to create a table string for use by ATS-2 with
    the BATCH? query.
  ' Acquire the signal on analog input 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
  ReDim fSwpTable(0 To 1) As Single ' will be redimensioned
  Dim bSuccess As Boolean, lPoints As Long, lIndex As Long, fFundamental As
    Single, iHarmonics As Integer, fHarmonic As Single
  Dim Src As Long, Ch1 As Long, Ch2 As Long, sSwpTable As String, fSumCh1 As
    Single, fTHDCh1 As Single, fSumCh2 As Single, fTHDCh2 As Single
  Src = 0: Ch1 = 1: Ch2 = 2
  fFundamental = 400
  iHarmonics = 20   ' number of frequencies includes fundamental plus harmonics
  TEST_TITLE = "FFT_THDSpectrumSwp_BinaryTable_To_Array"
  ' Send the initial setup commands. Use Flat Top window for maximum accuracy
    for harmonics down to -80 dB (+-0.2dB error), +2 dB error at -90 dB.
  ' Note that a small delay is necessary to permit the DSP program to load
    before the start of an acquisition.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:AGEN:DASINE:FRQ1 " &
    fFundamental & "HZ;:AGEN:AMPL A,1V;AMPL B,1V;OUTPUT AB;WFM
    DASINE,SINE;:DSP:DANLR:INPUT ANLG;AUTORANGE A,ON;AUTORANGE B,ON;:DELAY
    2;:ANLG:SOURCE AB,XLR;CONVERTER AD1"
  GPIBSend ":MON:SOURCE AINPUT;:DSP:PROGRAM FFT;:DSP:FFT:XLENGTH 4096;ACQLENGTH
    XLENGTH;AVGS 4;AVGTYPE 1;COUPLING SUBH;DELAY 0;MODE INTRPOLATE;START
    0;TRIGGER FREE;WINDOW FLAT;PKTRIG 1;:DELAY 0.2"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  ' Compute first iHarmonic frequencies of the fundamental.
  ReDim fSwpTable(1 To iHarmonics)
  For lIndex = 1 To iHarmonics
```

Figure 2-16. Visual Basic subprogram FFT_THDSpectrumSwp_BinaryTable_To_Array code.

```
    fHarmonic = fFundamental * lIndex
    If fHarmonic > 22500 Then
      ReDim Preserve fSwpTable(1 To lIndex - 1) ' Resize array in case harmonic
    freq exceeds bandwidth of FFT
      Exit For  ' Test that harmonic does not exceed bandwidth of FFT
    End If
    fSwpTable(lIndex) = fFundamental * lIndex
  Next lIndex
  iHarmonics = lIndex - 1
  ' Alternative to computing a sweep table: read it from a file.
    ' Read a sweep table array from the data in an ATS "*.atsx" file.
    ' If GetArbWfmSweepArray(App.Path & "\400Hz20Harmonics.atsx", fSwpTable(),
    "Hz") = False Then
    '   UserErrMsg "Sweep Table File Read Error in module " & TEST_TITLE  ' If
    sweep table file read error then exit.
    '   Exit Sub ' Quit, do not read response.
    ' End If
    ' iHarmonics = UBound(fswptable)

  ' Make a BINARY formatted sweep table string for table # 1 based on the array
    of sweep table elements.
  ' Note: change format to ASCII for purpose of debug.
  If MakeTable("BINARY", 1, fSwpTable(), sSwpTable) = False Then
    UserErrMsg "Sweep Table Build Error in module " & TEST_TITLE  ' If sweep
    table string build error then exit.
    Exit Sub ' Quit, do not read response.
  End If

  ' Send the sweep table to the ATS-2
  GPIBSend sSwpTable
  ' Setup the DSP source parameters for a frequency sweep of the FFT bins using
    table 1 and then acquire the signal on channel 1 (A) and process for a
    frequency domain spectrum.
  ' Setup the BATCH sweep parameters to sweep from 400 Hz to 22.4 KHz with 56
    steps (57 bins) using a sweep table.
  ' Peak picking is disabled with arbitrary sweeps using a table.
  sIO = ":DSP:OPSTATE SETUP;TSELECT 1;TIMEOUT 5;SRCP FREQ,HZ," & fSwpTable(1) &
    "," & fSwpTable(iHarmonics) & "," & iHarmonics - 1 & ",ARBITRARY;OPSTATE
    READ,AMP1,AMP2;ACQX?;*OPC"
  GPIBSend sIO
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Send batch mode amplitude measurement queries for channel 1 and channel 2
    spectrum bins and read the measurement query responses.
  bSuccess = ReadBatch2("BINARY", "AMP1", "V", "AMP2", "V", fSrcCh1Ch2())
  ' Set breakpoint on next line to enter VB debug mode in order to "watch"
    values in array fSrcCh1Ch2()
  If bSuccess Then
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & "FFT - THD spectrum acquisition complete:" &
    vbCrLf & "Acquired " & fFundamental & " Hz Fundamental & " & iHarmonics - 1
    & " Harmonics." & vbCrLf
    lPoints = UBound(fSrcCh1Ch2, 2)   ' get last index
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "Fundamental: Freq " & fSrcCh1Ch2(Src, 0) & " Hz, CH1 = " & fSrcCh1Ch2(Ch1,
    0) & " V, CH2 = " & fSrcCh1Ch2(Ch2, 0) & " V"
    For lIndex = 1 To lPoints Step 1
      frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "Harmonic " & lIndex + 1 & ": Freq " & fSrcCh1Ch2(Src, lIndex) & " Hz, CH1 =
    " & fSrcCh1Ch2(Ch1, lIndex) & " V, CH2 = " & fSrcCh1Ch2(Ch2, lIndex) & " V"
      fSumCh1 = fSumCh1 + (fSrcCh1Ch2(Ch1, lIndex) * fSrcCh1Ch2(Ch1, lIndex))
      fSumCh2 = fSumCh2 + (fSrcCh1Ch2(Ch2, lIndex) * fSrcCh1Ch2(Ch2, lIndex))
```

Figure 2-16 (cont.). Visual Basic subprogram FFT_THDSpectrumSwp_BinaryTable_To_Array code.

```
    Next lIndex
    fTHDCh1 = 20 * Log(Sqr(fSumCh1) / fSrcCh1Ch2(Ch1, 0)) / Log(10#)  ' Compute
    Ch1 THD as 20 * Log10((RSS Harmonics)/(Amplitude of Fundamental))
    fTHDCh2 = 20 * Log(Sqr(fSumCh2) / fSrcCh1Ch2(Ch2, 0)) / Log(10#)  ' Compute
    Ch2 THD as 20 * Log10((RSS Harmonics)/(Amplitude of Fundamental))
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    "Ch1 THD = " & fTHDCh1 & " dB" & vbCrLf & "Ch2 THD = " & fTHDCh2 & " dB" &
    vbCrLf
  Else
    ' Display error message if unsuccessful ReadBatch2 function
    frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & "FFT - Spectrum acquisition failed." & vbCrLf
  End If
  ' Set display to last line
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  GPIBSend ":DSP:OPSTATE SETUP"
  ATS2G_Errors       ' Check for ATS-2 errors.
  GPIBSend "*CLS"    ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-16 (cont.). Visual Basic subprogram FFT_THDSpectrumSwp_BinaryTable_To_Array code.

```
FFT_THDSpectrumSwp_BinaryTable_To_Array

FFT - THD spectrum acquisition complete:
Acquired 400 Hz Fundamental & 19 Harmonics.

Fundamental: Freq 400 Hz, CH1 = 1.000882 V, CH2 = 1.002386 V
Harmonic 2: Freq 800 Hz, CH1 = 1.211618E-06 V, CH2 = 5.097748E-07 V
Harmonic 3: Freq 1200 Hz, CH1 = 3.17628E-07 V, CH2 = 7.470109E-07 V
Harmonic 4: Freq 1600 Hz, CH1 = 4.253734E-07 V, CH2 = 4.007148E-07 V
Harmonic 5: Freq 2000 Hz, CH1 = 1.491681E-06 V, CH2 = 1.677082E-06 V
Harmonic 6: Freq 2400 Hz, CH1 = 2.920504E-07 V, CH2 = 3.20735E-07 V
Harmonic 7: Freq 2800 Hz, CH1 = 1.305549E-07 V, CH2 = 2.46924E-07 V
Harmonic 8: Freq 3200 Hz, CH1 = 2.137581E-07 V, CH2 = 2.534534E-07 V
Harmonic 9: Freq 3600 Hz, CH1 = 2.582644E-07 V, CH2 = 3.287788E-07 V
Harmonic 10: Freq 4000 Hz, CH1 = 2.640211E-07 V, CH2 = 1.438762E-07 V
Harmonic 11: Freq 4400 Hz, CH1 = 2.027145E-07 V, CH2 = 1.778478E-07 V
Harmonic 12: Freq 4800 Hz, CH1 = 2.084465E-07 V, CH2 = 2.089636E-07 V
Harmonic 13: Freq 5200 Hz, CH1 = 2.608234E-07 V, CH2 = 2.050015E-07 V
Harmonic 14: Freq 5600 Hz, CH1 = 2.250655E-07 V, CH2 = 1.530717E-07 V
Harmonic 15: Freq 6000 Hz, CH1 = 1.725753E-07 V, CH2 = 3.254945E-07 V
Harmonic 16: Freq 6400 Hz, CH1 = 2.648823E-07 V, CH2 = 1.813071E-07 V
Harmonic 17: Freq 6800 Hz, CH1 = 1.819108E-07 V, CH2 = 1.752229E-07 V
Harmonic 18: Freq 7200 Hz, CH1 = 1.466083E-07 V, CH2 = 2.4467E-07 V
Harmonic 19: Freq 7600 Hz, CH1 = 2.847361E-07 V, CH2 = 1.612494E-07 V
Harmonic 20: Freq 8000 Hz, CH1 = 1.348878E-07 V, CH2 = 2.213115E-07 V
Ch1 THD = -113.2715 dB
Ch2 THD = -113.3997 dB
```

Figure 2-17. Visual Basic subprogram FFT_THDSpectrumSwp_BinaryTable_To_Array measurement data.

the exact frequency bins of interest. This example measures harmonics of a 400 Hz sinewave fundamental frequency and sums them to produce THD data without including other spectral data such as noise (not the same as THD+N).

Waveform Acquisition Sweeps with the FFT DSP Program

Time waveforms may be acquired with the FFT DSP program. Each sample to be measured in the waveform record is specified by the SRCParams command. The process requires an initial setup, followed by the acquisition of the signal, and then followed by a :DSP:BATCh? query measurement response data.

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

```
Sub FFT_WaveformSwp()
  ' Waveform Acquisition Sweeps with the FFT DSP Program
  TEST_TITLE = "FFT_WaveformSwp"
  ' 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.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
AINPUT;:DGEN:OUTPUT AB;AMPL AB,0DBFS;WFM SINE,SINE"
  GPIBSend ":DIN:FORMAT XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;INVALID
0;JWFM NONE;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24,BITS"
  GPIBSend ":DSP:PROGRAM FFT;:DSP:FFT:INPUT DIGITAL;ACQLENGTH L24K;AVGS
1;AVGTYPE 1;COUPLING DC;DELAY 0;MODE RAW;START 0.0;TRIGGER DGEN;TSLOPE
POS;WINDOW EQR;XLENGTH 1024;PKTRIG 1;:DELAY 0.2"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  ' 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 0.25 mS, sweep stop time of 1.25 mS, 0.1 mS intervals
(10 steps, 11 points), with linear spacing.
  GPIBSend ":DSP:OPST SET;TIMEOUT 2;SRCP TIME,SEC,0.00025,.00125,10,LIN;OPST
READ,AMP1;ACQX?;*OPC"
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then    ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"    ' Return to DSP OPSTATE SETUP mode and
clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Send BATCH? query for channel 1 waveform samples in FFS units.
  GPIBSend ":DSP:BATCH? ASCII,AMP1,FFS"
  ' Read the measurement query responses.
  sIO = GPIBRcv
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  GPIBSend ":DSP:OPSTATE SETUP"
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ATS2G_Errors        ' Check for ATS-2 errors.
  GPIBSend "*CLS"    ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-18. Visual Basic code for subprogram FFT_WaveformSwp.

the samples are read with the :DSP:BATCh? query. The time period of interest begins at 0.25 ms after the trigger point and ends 1.25 ms later in order to analyze a cycle of the 1 kHz fundamental signal from the ATS-2 digital generator. For this example, a coarse measurement time interval was selected for simplicity. The subprogram shows the :DSP:BATCh? query with ASCII formatted response data.

```
FFT_WaveformSwp

ASCII,11,0.00025,1.41231,0.00035,1.15101,0.00045,0.336954,0.00055,-0.455078,0.00
065,-1.21809,0.00075,-1.41334,0.00085,-1.16445,0.00095,-0.359606,0.00105,0.43288
9,0.00115,1.20605,0.00125,1.41395
```

Figure 2-19. Visual Basic subprogram 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 pulse impairments. 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, INTervu processes the acquisition buffer and constructs *eye* pattern data for one "cell" of the AES/EBU input pulse train. The :DSP:SRCParams command specifies the time period for the Eye Pattern width. The :DSP:OPSTate READ command specifies both the upper *eye* waveform and 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 example below sets the instrument for a digital input and output, and loads 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

```
Sub INTERVU_EyePatternSwp()
  ' Digital Interface Eye Pattern Sweeps with the Intervu DSP Program
  TEST_TITLE = "INTERVU_EyePatternSwp"
  If Not (ATS2G_IntervuOpt) Then
    MsgBox "ATS-2 High Performance Intervu 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.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    ABINPUTSUM;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;WFM SINE,SINE;:DIN:FORMAT
    XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;INVALID 0;JWFM
    NONE;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24,BITS"
  GPIBSend ":DSP:PROGRAM INTERVU"
  GPIBSend ":DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE EYE;TRIGGER ARCV;WINDOW BH"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  Delay 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.
  ' Sweep start time of 0.0 Sec, sweep stop time of 175 nS, intervals of 5 nS
    (35 steps, 36 points), with linear spacing.
  GPIBSend ":DSP:OPST SET;TIMEOUT 10;SRCP TIME,SEC,0,175E-9,35,LIN;OPST
    READ,LOWER,UPPER;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(10) = False Then
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Send BATCH? query for upper and lower eye pattern waveform samples in Volt
    units.
  GPIBSend ":DSP:BATCH? ASCII,LOWER,V,UPPER,V"
  ' Read the measurement query responses.
```

Figure 2-20. Visual Basic subprogram INTERVU_EyePatternSwp code.

```
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  GPIBSend ":DSP:OPSTATE SETUP"
  ATS2G_Errors       ' Check for ATS-2 errors.
  GPIBSend "*CLS"    ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-20 (cont.). Visual Basic subprogram INTERVU_EyePatternSwp code.

amplitude of the eye pattern samples are read with a :DSP:BATCh? query for the upper and lower eye patterns. The time period of interest begins at 0.0 ns and ends 175 ns later in order to analyze one pattern. For this example a linear time interval of 5 ns per point was selected. This code is provided in the "ATS2Gsample.bas" file on the CD ROM included with this manual. The subprogram "INTERVU_EyePatternSwp" reads the upper and lower eye data using a :DSP:BATCh? query.

```
INTERVU_EyePatternSwp


ASCII,36,0,-0.0434689,0.0241845,5E-09,-0.653635,0.661828,1E-08,-1.06136,1.04676,
1.5E-08,-1.48294,1.47419,2E-08,-1.76761,1.78102,2.5E-08,-1.92711,1.93072,3E-08,-
2.08362,2.08381,3.5E-08,-2.17802,2.18451,4E-08,-2.20731,2.22402,4.5E-08,-2.27775
,2.27432,5E-08,-2.29158,2.3214,5.5E-08,-2.30522,2.32694,6E-08,-2.31779,2.34718,6
.5E-08,-2.32345,2.35039,7E-08,-2.33004,2.35699,7.5E-08,-2.32163,2.37393,8E-08,-2
.35264,2.34985,8.5E-08,-2.34828,2.37318,9E-08,-2.34593,2.37967,9.5E-08,-2.36694,
2.37318,1E-07,-2.36872,2.3883,1.05E-07,-2.36808,2.39554,1.1E-07,-2.3684,2.38298,
1.15E-07,-2.3745,2.38213,1.2E-07,-2.37161,2.35963,1.25E-07,-2.35595,2.35667,1.3E
-07,-2.34711,2.34664,1.35E-07,-2.34561,2.34343,1.4E-07,-2.31385,2.31096,1.45E-07
,-2.2332,2.23393,1.5E-07,-2.06424,2.04302,1.55E-07,-1.86233,1.82779,1.6E-07,-1.4
4375,1.4042,1.65E-07,-0.87737,0.832507,1.7E-07,-0.38344,0.357335,1.75E-07,0,0
```

Figure 2-21. Visual Basic measurement results for subprogram 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 with the :DSP:BATCh? query. In order to measure jitter probability, a series of digital audio frames are 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 sweep of jitter magnitude vs jitter probability.

The example below sets up the instrument for AES/EBU digital input and output, generates a 0.1 UI jitter signal on the AES/EBU digital output, and loads 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, in intervals of 0.5 nS. The VB sample code subprogram "INTERVU_JitterProbabilitySwp" reads the probability data using a :DSP:BATCh? query.

```
Sub INTERVU_JitterProbabilitySwp()
  'Digital Interface Jitter Probability Sweeps with the Intervu DSP Program
  TEST_TITLE = "INTERVU_JitterProbabilitySwp"
  If Not (ATS2G_IntervuOpt) Then
    MsgBox "ATS-2 High Performance Intervu 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.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
ABINPUTSUM;:DGEN:OUTPUT AB;AMPL AB,-10DBFS;WFM SINE,SINE"
  GPIBSend ":DIN:FORMAT XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;INVALID
0;JWFM NONE;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24,BITS"
  GPIBSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE
INTRPOLATE;TRIGGER ARCV;WINDOW BH"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  Delay 1
  ' 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.
  GPIBSend ":DSP:OPSTATE SET;TIMEOUT 4;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
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Send BATCH? query for jitter probability in percent units.
  GPIBSend ":DSP:BATCH? ASCII,PROBABILITY,PCT"
  ' Read the measurement query responses.
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  GPIBSend ":DSP:OPSTATE SETUP"
  ATS2G_Errors       ' Check for ATS-2 errors.
  GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure2-22. Visual Basic subprogram INTERVU_JitterProbabilitySwp code.

```
INTERVU_JitterProbabilitySwp


ASCII,21,-5E-09,0,-4.5E-09,0,-4E-09,0,-3.5E-09,0,-3E-09,0,-2.5E-09,0,-2E-09,0,-1
.5E-09,0,-1E-09,0.00360012,-5.00001E-10,0.34256,0,3.27597,5E-10,3.24354,9.99999E
-10,0.353372,1.5E-09,0.00180006,2E-09,0,2.5E-09,0,3E-09,0,3.5E-09,0,4E-09,0,4.5E
-09,0,5E-09,0
```

Figure 2-23. Visual Basic measurement results for subprogram 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 1572864 samples of the digital interface signal at a sample rate of 80 MHz. The total time period of the acquisition is 0.0196608 seconds. The INTERVU DSP program processes this large acquisition of the interface waveform, reconstructs the clock rate and reference point from zero crossings of the signal, and demodulates the jitter waveform. The jitter waveform sample points are specified in units of time relative to the beginning of the jitter sample interval. The acquisition of jitter magnitude versus time may be triggered in many ways. The example subprogram below triggers on the source of jitter, the jitter generator of ATS-2. This code is provided in the "ATS2Gsample.bas" file on the CD ROM included with this manual. The subprogram "INTERVU_JitterWaveformSwp" reads the jitter waveform data using a :DSP:BATCh? query.

Connect the ATS-2 AES/EBU digital output to the digital input in order to acquire results similar to those shown below. The ATS-2 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.

```
Sub INTERVU_JitterWaveformSwp()
  ' Digital Interface Jitter Waveform Sweeps with the Intervu DSP Program
  TEST_TITLE = "INTERVU_JitterWaveformSwp"
  If Not (ATS2G_IntervuOpt) Then
    MsgBox "ATS-2 High Performance Intervu 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.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    ABINPUTSUM;:DGEN:OUTPUT AB;AMPL AB,0DBFS;WFM SPCIAL,JTEST;:DIN:FORMAT
    XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL 5;INVALID 0;JWFM SINE;JAMPL
    0.3UI;JFREQ 1000;PREEMPHASIS OFF;RATE 44100HZ;RESOLUTION 24,BITS"
  GPIBSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE
    RAW;TRIGGER JITTER;WINDOW BH"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  Delay 1
  ' 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.
  GPIBSend ":DSP:OPST SET;TIMEOUT 4;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
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
```

Figure 2-24. Visual Basic subprogram INTERVU_JitterWaveformSwp code.

```
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Send BATCH? query for jitter waveform in UI units.
  GPIBSend ":DSP:BATCH? ASCII,JITTER,UI"
  ' Read the measurement query responses.
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  GPIBSend ":DSP:OPSTATE SETUP"
  ATS2G_Errors       ' Check for ATS-2 errors.
  GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-24 (cont.). Visual Basic subprogram INTERVU_JitterWaveformSwp code.

```
INTERVU_JitterWaveformSwp


ASCII,27,0.001,0.0163566,0.00105,0.107724,0.0011,0.189587,0.00115,0.254884,0.001
2,0.290039,0.00125,0.298907,0.0013,0.273207,0.00135,0.223704,0.0014,0.152558,0.0
0145,0.0636026,0.0015,-0.032373,0.00155,-0.11918,0.0016,-0.201198,0.00165,-0.259
01,0.0017,-0.293502,0.00175,-0.298799,0.0018,-0.272892,0.00185,-0.22356,0.0019,-
0.149781,0.00195,-0.0631375,0.002,0.0308248,0.00205,0.123738,0.0021,0.205401,0.0
0215,0.270007,0.0022,0.305255,0.00225,0.30864,0.0023,0.287128
```

Figure 2-25. Visual Basic subprogram INTERVU_JitterWaveformSwp measurement results.

Digital Interface Waveform Sweeps with the Intervu DSP Program

The Intervu DSP program may be used to acquire the digital interface squarewave signal in the form of a waveform. INTERVU operates by first acquiring 1572864 samples of the digital interface signal at a sample rate of 80 MHz (for ATS-2). The total time period of the acquisition is 0.0196608 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 "ATS2Gsample.bas" file on the CD ROM included with this manual. The subprogram "INTERVU_WaveformSwp" reads the digital interface waveform data using a :DSP:BATCh? query.

Connect the ATS-2 AES/EBU digital output to the digital input in order to acquire results similar to those shown below. The ATS-2 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. These measurements are acquired beginning at 0.0 ns and ending at 400 ns in increments of 6.25 ns. Note that the waveform points requested between the 12.5 nS sample points are actually interpolated points because the INTERVU MODE is set to INTERPOLATE. Use MODE RAW if you do want non-interpolated sample points.

```
Sub INTERVU_WaveformSwp()
  ' Digital Interface Waveform Sweeps with the Intervu DSP Program
  TEST_TITLE = "INTERVU_WaveformSwp"
  If Not (ATS2G_IntervuOpt) Then
    MsgBox "ATS-2 High Performance Intervu 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.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:MON:SOURCE
    ABINPUTSUM;:DGEN:OUTPUT AB;AMPL AB,0DBFS;FRQ1 997 HZ;WFM
    SINE,SINE;:DIN:FORMAT XLR;SCALEFREQBY MEASURED;:DOUT:FORMAT XLR;AMPL
    5;INVALID 0;JWFM NONE;JAMPL 0.0UI;JFREQ 1000;PREEMPHASIS OFF;RATE
    44100HZ;RESOLUTION 24,BITS"
  GPIBSend ":DSP:PROGRAM INTERVU;:DSP:INTERVU:AVGS 1;JDETECTION ALL;MODE
    INTRPOLATE;TRIGGER XMTBLOCK;TMODE POSTTRIG;WINDOW BH"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  Delay 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 0.0 S, sweep stop time of 400 nS, intervals of 6.25 nS
    (64 steps, 65 points), with linear spacing.
  GPIBSend ":DSP:OPST SET;TIMEOUT 4;SRCP TIME,SEC,0,4E-7,64,LIN;OPST
    READ,AMPL;ACQX?;*OPC"
  ' wait for ESB bit in Status Byte
  If WaitForESB(5) = False Then
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Read the response to the ACQX? query.
  sIO = GPIBRcv
  If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Send BATCH? query for jitter waveform in volts units.
  GPIBSend ":DSP:BATCH? ASCII,AMPL,V"
  ' Read the measurement query responses.
  sIO = GPIBRcv
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  ' Set the DSP back to OPSTATE SETUP after measurements have been read.
  GPIBSend ":DSP:OPSTATE SETUP"
  ATS2G_Errors       ' Check for ATS-2 errors.
  GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit
End Sub
```

Figure 2-26. Visual Basic subprogram INTERVU_WaveformSwp code.

```
INTERVU_WaveformSwp


ASCII,65,0,-2.48198,6.25E-09,-2.49275,1.25E-08,-2.50926,1.875E-08,-2.49914,2.5E-
08,-2.48198,3.125E-08,-2.48919,3.75E-08,-2.50926,4.375E-08,-2.51279,5E-08,-2.509
26,5.625E-08,-2.50388,6.25E-08,-2.48198,6.875E-08,-2.43716,7.5E-08,-2.40017,8.12
5E-08,-2.34315,8.75E-08,-2.15469,9.375E-08,-1.72313,1E-07,-1.06371,1.0625E-07,-0
.283118,1.125E-07,0.490935,1.1875E-07,1.16407,1.25E-07,1.66373,1.3125E-07,1.9791
8,1.375E-07,2.15467,1.4375E-07,2.26473,1.5E-07,2.34557,1.5625E-07,2.40306,1.625E
-07,2.42742,1.6875E-07,2.43708,1.75E-07,2.45467,1.8125E-07,2.48808,1.875E-07,2.5
0923,1.9375E-07,2.49931,2E-07,2.48195,2.0625E-07,2.48384,2.125E-07,2.50923,2.187
5E-07,2.53016,2.25E-07,2.53651,2.3125E-07,2.52595,2.375E-07,2.50923,2.4375E-07,2
.49232,2.5E-07,2.48195,2.5625E-07,2.4404,2.625E-07,2.29105,2.6875E-07,1.94166,2.
75E-07,1.3637,2.8125E-07,0.610154,2.875E-07,-0.190922,2.9375E-07,-0.913443,3E-07
,-1.47283,3.0625E-07,-1.85939,3.125E-07,-2.10013,3.1875E-07,-2.24196,3.25E-07,-2
.31833,3.3125E-07,-2.36465,3.375E-07,-2.40017,3.4375E-07,-2.43462,3.5E-07,-2.454
7,3.5625E-07,-2.45841,3.625E-07,-2.4547,3.6875E-07,-2.46062,3.75E-07,-2.48198,3.
8125E-07,-2.5017,3.875E-07,-2.50926,3.9375E-07,-2.49807,4E-07,-2.48198
```

Figure 2-27. Visual Basic subprogram 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. The digital generator waveform buffers and the FASTTEST DSP program must both be properly configured. Please see the ATS-2 User's Manual 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 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 ATS 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.

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

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

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 output 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 D/A Arbitrary waveform, D/A sample rate set to Output Sample Rate, and turn the output on. Note that the :AGEN:DASR OSR command is required to set the output sample rate to the setting of the digital output sample rate setting because the default DASR setting is 65536, not 48000.
6. Set up the analog input source for XLR inputs with auto-ranging enabled.
7. Set up the headphone monitor to hear the multitone signal on the analog inputs, in stereo mode.
8. 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.
9. 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 with :DSP:BATCh?. Read 31 phase measurements for channel 2.

10. 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 with :DSP:BATCh?.
11. 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 with :DSP:BATCh?.

This code is provided in the "ATS2Gsample.bas" file as subprogram "FASTTEST_MultitoneSwp", on the CD ROM included with this manual. 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 FASTTEST_MultitoneSwp_RQS sets up the ATS-2 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 frmATS2Gsample) to invoke a callback function (GpibNotify1_Notify in form frmATS2Gsample) that serial polls the ATS-2. This happens when the ATS-2 asserts the SRQ line on the GPIB interface when the DSP acquisition has been triggered by the multitone signal on the 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 shows 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.

```
Sub FASTTEST_MultitoneSwp()
  ' Multitone Signal Testing with FASTTEST DSP
  Dim lBufLengthTemp As Long  ' Temporary storage of current value of BUFLENGTH,
    GPIB I/O buffer
  TEST_TITLE = "FASTTEST_MultitoneSwp"
  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.
  GPIBSend "*RCL 0;*ESE 1;*SRE 0;:APST:ENAB 0;:HEADER OFF;:DGEN:ARBSIZE 8192"
  ' 1. FASTTEST DSP setup for signal acquisition and level sweep.
  GPIBSend ":DSP:PROGRAM FASTTEST;FASTTEST:FREQRES 0;INPUT ANLG;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 ISO31.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.
  GPIBSend ":DGEN:ARBWFM 1,1;:DOUT:RATE 48000HZ;:AGEN:AMPL AB,1V;WFM
    DAARBITRARY,NONE;DASR OSR;OUTPUT AB" ' original
  ' Use AD2 converter to sample at the rate set for the digital output, the same
    rate used with the Analog Generator DASR setting of OSR.
```

Figure 2-28. Visual Basic subprogram FASTTEST_MultitoneSwp and FASTTEST_MultitoneSwp_RQS code.

```
GPIBSend ":ANLG:CONV AD2;SOURCE AB,XLR;AUTORANGE AB,ON;:DELAY
  2;:ANLG:AUTORANGE AB,OFF;:MON:SOURCE ABINPUTSUM"
'8. Define a sweep table to sweep the ISO31 frequency bins for FASTTEST
  channels 1 and 2.
GPIBSend ":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,
  2501.95,3152.34,4001.95,4998.04,6351.56,7998.04,10001.95,12498.04,16001.95,1
  9998.04"
GPIBSend ":DSP:TSELECT 2"
' 9. Level and Phase Sweep.
GPIBSend ":DSP:OPSTATE SETUP;TIMEOUT 20;SRCP
  FREQ,HZ,17.578125,19998.046875,60,LOG"
GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
Delay 2 ' wait for ATS2G input queue to be processed.
GPIBSend ":DSP:OPSTATE SETUP;FASTTEST:MODE RESPONSE;:DELAY 0.1;:DSP:OPSTATE
  READ,AMP1,AMP2,PHA2;:DSP:ACQX?"
sIO = GPIBRcv
If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
  TEST_TITLE  ' If DSP Acquisition time out then exit.
  GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
  clear status registers after DSP timeout.
  Exit Sub ' Quit, do not read response.
End If
GPIBSend "*CLS"   ' Clear Event Status Register to clear OPC bit
' Send FASTTEST channel 1 measurement query.
GPIBSend ":DSP:BATCH? ASCII,AMP1,DBV"
' Read Multitone Level measurements.
sLevel1 = GPIBRcv
' Send FASTTEST channel 2 measurement query.
GPIBSend ":DSP:BATCH? ASCII,AMP2,DBV"
' Read Multitone Level measurements.
sLevel2 = GPIBRcv
' Send FASTTEST channel 2 measurement query.
GPIBSend ":DSP:BATCH? ASCII,PHA2,DEG"
' Read Multitone channel 2 phase measurements.
sPhase = GPIBRcv
' 10. Distortion Sweep, setup for distortion processing.
GPIBSend ":DSP:OPSTATE SETUP;FASTTEST:MODE DISTORTION;:DELAY 1.1;:DSP:OPSTATE
  READING,AMP1,AMP2;XFRM?"  ' original line
' Read REPROCESS? query response.
sIO = GPIBRcv
If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
  TEST_TITLE  ' If DSP Acquisition time out then exit.
  GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
  clear status registers after DSP timeout.
  Exit Sub ' Quit, do not read response.
End If
' Send FASTTEST channel 1 measurement query.
GPIBSend ":DSP:BATCH? ASCII,AMP1,DBV"
' Read Multitone channel 1 distortion measurements.
sDistortion1 = GPIBRcv
' Send FASTTEST channel 2 measurement query.
GPIBSend ":DSP:BATCH? ASCII,AMP2,DBV"
' Read Multitone channel 2 distortion measurements.
sDistortion2 = GPIBRcv
' 11. Noise Sweep, setup for noise processing.
GPIBSend ":DSP:OPSTATE SETUP;FASTTEST:MODE NOISE;:DELAY 0.1;:DSP:OPSTATE
  READING,AMP1,AMP2;XFRM?" ' original line
' Read REPROCESS? query response.
sIO = GPIBRcv
If sIO <> "0" Then   ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
  TEST_TITLE  ' If DSP Acquisition time out then exit.
```

Figure 2-28 (cont.). Visual Basic subprogram FASTTEST_MultitoneSwp and FASTTEST_MultitoneSwp_RQS code.

```
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"   ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Send FASTTEST channel 1 measurement query.
  GPIBSend ":DSP:BATCH? ASCII,AMP1,DBV"
  ' Read Multitone channel 1 noise measurements.
  sNoise1 = GPIBRcv
  ' Send FASTTEST channel 2 measurement query.
  GPIBSend ":DSP:BATCH? ASCII,AMP2,DBV"
  ' Read Multitone channel 2 noise measurements.
  sNoise2 = GPIBRcv
  ' 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
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf & sIO & vbCrLf
  sIO = "Interchannel Phase (Phase 2)" & vbCrLf & "Format = FREQ, PHA2, [...]" &
    vbCrLf & vbCrLf & sPhase
  frmATS2GSample.txtOutput.Text = frmATS2GSample.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
  frmATS2GSample.txtOutput.Text = frmATS2GSample.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
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf & sIO &
    vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  GPIBSend ":DSP:OPSTATE SETUP"
  ATS2G_Errors
End Sub

Sub 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
  '     (e.g. tranmitted multitone in a broadcast link).
  ' 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 = "FASTTEST_MultitoneSwp_RQS"
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf &
    TEST_TITLE & vbCrLf & vbCrLf
  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.
  GPIBSend "*RCL 0;:HEADER OFF;:DGEN:ARBSIZE 8192"
  ' Use AD2 converter to sample at the rate set for the digital output, the same
    rate used with the Analog Generator DASR setting of OSR.
  GPIBSend ":ANLG:CONV AD2;SOURCE AB,XLR;AUTORANGE AB,ON;:DELAY
    2;:ANLG:AUTORANGE AB,OFF;:MON:SOURCE ABINPUTSUM"
  ' 1. FASTTEST DSP setup for signal acquisition and level sweep. Use AD2
    converter to sample at the rate set for the digital output, the same rate
    used with the Analog Generator DASR setting of OSR.
  GPIBSend ":DSP:PROGRAM FASTTEST;FASTTEST:FREQRES 0;INPUT ANLG;LENGTH AUTO;MODE
    RESPONSE;PMODE ICHANNEL;PROCESS SYNC;TRGDELAY 0;TRIGGER NORMAL;:DELAY 0.5"
  GPIBSend ":MON:VOLUME " & frmATS2GSample.sliderVOL.value
  ' 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).
```

Figure 2-28 (cont.). Visual Basic subprogram FASTTEST_MultitoneSwp and FASTTEST_MultitoneSwp_RQS code.

```
' 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.
GPIBSend ":DGEN:ARBWFM 1,1;:DOUT:RATE 48000HZ;:AGEN:AMPL A,1V;AMPL B,1V;WFM
  DAARBITRARY,NONE;DASR OSR;OUTPUT AB;:ANLG:SOURCE AB,XLR;AUTORANGE
  AB,ON;:DELAY 2;:ANLG:AUTORANGE AB,OFF;:MON:SOURCE ABINPUTSUM"
' 8. Define a sweep table to sweep the ISO31 frequency bins for FASTTEST
  channels 1 and 2.
GPIBSend ":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,
  2501.95,3152.34,4001.95,4998.04,6351.56,7998.04,10001.95,12498.04,16001.95,1
  9998.04"
GPIBSend ":DSP:TSELECT 2"
' 9. Level and Phase Sweep. Use SRQ interrupt to detect that DSP Transform is
  complete (bit 5 in AESER).
GPIBSend "*SRE 1;:APST:ENAB 32;:DSP:OPSTATE SETUP;TIMEOUT 10;SRCP
  FREQ,HZ,17.578125,19998.046875,31,ARB;OPSTATE READ,AMP1,AMP2,PHA2;ACQX?"
' Wait for SRQ interrupt with timeout. Set timeout to desired value (10
  seconds in this example).
If WaitForSRQ(10) = False Then
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & "FASTTEST
  Acquire Complete SRQ Interrupt Not Detected" & vbCrLf
  ' If DSP Acquisition time out (no SRQ interrupt detected within timeout
  period) then exit option.
  If MsgBox("ATS-2 SRQ Timeout or SRQ Interrupt Handler Not Active",
  vbOKCancel, MSGBOX_TITLE & TEST_TITLE) = vbCancel Then End
Else
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & "FASTTEST
  Acquire Complete SRQ Interrupt Detected" & vbCrLf
End If
' Read ACQX? query response.
sIO = GPIBRcv
If sIO <> "0" Then    ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
  TEST_TITLE  ' If DSP Acquisition time out then exit.
  GPIBSend ":DSP:OPSTATE SETUP;*CLS"    ' Return to DSP OPSTATE SETUP mode and
  clear status registers after DSP timeout.
  Exit Sub ' Quit, do not read response.
End If
' Clear Event Status Register and AP Event Status Register
GPIBSend "*CLS"
' Send FASTTEST measurement query for channels 1 & 2 and interchannel phase
  (channel 2).
GPIBSend ":DSP:BATCH? ASCII,AMP1,DBV,AMP2,DBV,PHA2,DEG"
' Read Multitone Level measurements for channels 1 & 2 and interchannel phase
  (channel 2).
s2ChAmp_Phase2 = GPIBRcv
' 10. Distortion Sweep, setup for distortion processing.
GPIBSend ":DSP:OPSTATE SETUP;FASTTEST:MODE DISTORTION;:DELAY 1.1;:DSP:OPSTATE
  READING,AMP1,AMP2;XFRM?"   ' original line
' Read REPROCESS? query response.
sIO = GPIBRcv
If sIO <> "0" Then    ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
  UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
  TEST_TITLE  ' If DSP Acquisition time out then exit.
  GPIBSend ":DSP:OPSTATE SETUP;*CLS"    ' Return to DSP OPSTATE SETUP mode and
  clear status registers after DSP timeout.
  Exit Sub ' Quit, do not read response.
End If
' Send FASTTEST channel 1 measurement query.
GPIBSend ":DSP:BATCH? ASCII,AMP1,DBV,AMP2,DBV"
```

Figure 2-28 (cont.). Visual Basic subprogram FASTTEST_MultitoneSwp and FASTTEST_MultitoneSwp_RQS code.

```
  ' Read Multitone channel 1 distortion measurements.
  sDistortion = GPIBRcv
  ' 11. Noise Sweep, setup for noise processing.
  GPIBSend ":DSP:OPSTATE SETUP;FASTTEST:MODE NOISE;:DELAY 0.1;:DSP:OPSTATE
    READING,AMP1,AMP2;XFRM?" ' original line
  ' Read REPROCESS? query response.
  sIO = GPIBRcv
  If sIO <> "0" Then    ' ATS-2 DSP ACQX or XFRM or REPROCESS Timeout
    UserErrMsg "ATS-2 DSP ACQX or XFRM or REPROCESS Timeout in module " &
    TEST_TITLE  ' If DSP Acquisition time out then exit.
    GPIBSend ":DSP:OPSTATE SETUP;*CLS"    ' Return to DSP OPSTATE SETUP mode and
    clear status registers after DSP timeout.
    Exit Sub ' Quit, do not read response.
  End If
  ' Send FASTTEST channel 1 measurement query.
  GPIBSend ":DSP:BATCH? ASCII,AMP1,DBV,AMP2,DBV"
  ' Read Multitone channel 1 noise measurements.
  sNoise = GPIBRcv
  ' 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
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf & sIO &
    vbCrLf
  sIO = "Distortion Channels 1 & 2" & vbCrLf & "Format = FREQ, AMP1, AMP2,
    [...]" & vbCrLf & vbCrLf & sDistortion
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf & sIO &
    vbCrLf
  sIO = "Noise Channels 1 & 2" & vbCrLf & "Format = FREQ, AMP1, AMP2, [...]" &
    vbCrLf & vbCrLf & sNoise
  frmATS2GSample.txtOutput.Text = frmATS2GSample.txtOutput.Text & vbCrLf & sIO &
    vbCrLf
  frmATS2GSample.txtOutput.SelStart = Len(frmATS2GSample.txtOutput.Text)
  GPIBSend ":DSP:OPSTATE SETUP;*SRE 0;:APST:ENAB 0"    ' disable SRQ interrupts
    from ATS2G
  ATS2G_Errors
End Sub
```

Figure 2-28 (cont.). Visual Basic subprogram FASTTEST_MultitoneSwp and FASTTEST_MultitoneSwp_RQS code.

```
FASTTEST_MultitoneSwp

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

ASCII,61,17.5781,-25.1502,19.7654,-25.1502,22.2249,-25.152,24.9904,-25.1529,28.1
,-25.1538,31.5966,-25.1546,35.5283,-25.1551,39.9492,-25.1557,44.9202,-25.1561,50
.5097,-25.1565,56.7948,-25.1567,63.8619,-25.157,71.8084,-25.1572,80.7438,-25.157
2,90.791,-25.1573,102.088,-25.1574,114.792,-25.1574,129.075,-25.1574,145.137,-25
.1573,163.196,-25.1572,183.503,-25.1572,206.337,-25.157,232.013,-25.1567,260.883
,-25.1565,293.345,-25.1561,329.847,-25.1557,370.891,-25.1553,417.042,-25.1547,46
8.935,-25.1541,527.286,-25.1536,592.898,-25.153,666.674,-25.1525,749.63,-25.1522
,842.909,-25.1523,947.795,-25.1529,1065.73,-25.1539,1198.34,-25.1553,1347.46,-25
.1565,1515.13,-25.1574,1703.66,-25.1569,1915.65,-25.1552,2154.02,-25.1539,2422.0
5,-25.1528,2723.43,-25.1544,3062.32,-25.1571,3443.37,-25.1556,3871.84,-25.1527,4
353.62,-25.1534,4895.36,-25.1554,5504.5,-25.1563,6189.45,-25.1571,6959.62,-25.15
71,7825.62,-25.1568,8799.39,-25.1551,9894.32,-25.1531,11125.5,-25.1561,12509.9,-
25.1597,14066.5,-25.16,15816.9,-25.1602,17785,-25.1598,19998,-25.1592

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

ASCII,61,17.5781,-25.141,19.7654,-25.141,22.2249,-25.1421,24.9904,-25.1426,28.1,
-25.1432,31.5966,-25.1439,35.5283,-25.1443,39.9492,-25.1447,44.9202,-25.145,50.5
097,-25.1453,56.7948,-25.1455,63.8619,-25.1457,71.8084,-25.1458,80.7438,-25.146,
90.791,-25.146,102.088,-25.146,114.792,-25.146,129.075,-25.146,145.137,-25.146,1
63.196,-25.1458,183.503,-25.1457,206.337,-25.1455,232.013,-25.1453,260.883,-25.1
45,293.345,-25.1447,329.847,-25.1443,370.891,-25.1438,417.042,-25.1432,468.935,-
```

Figure 2-29. Visual Basic subprogram FASTTEST_MultitoneSwp and FASTTEST_MultitoneSwp_RQS measurement results.

```
25.1427,527.286,-25.1421,592.898,-25.1415,666.674,-25.141,749.63,-25.1408,842.90
9,-25.1409,947.795,-25.1414,1065.73,-25.1424,1198.34,-25.1438,1347.46,-25.145,15
15.13,-25.146,1703.66,-25.1454,1915.65,-25.1437,2154.02,-25.1424,2422.05,-25.141
3,2723.43,-25.1429,3062.32,-25.1456,3443.37,-25.1441,3871.84,-25.1413,4353.62,-2
5.1419,4895.36,-25.1438,5504.5,-25.1448,6189.45,-25.1454,6959.62,-25.1454,7825.6
2,-25.1452,8799.39,-25.1435,9894.32,-25.1414,11125.5,-25.1443,12509.9,-25.1477,1
4066.5,-25.1478,15816.9,-25.1478,17785,-25.147,19998,-25.146

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

ASCII,61,17.5781,0.175781,19.7654,0.175781,22.2249,0.131836,24.9904,0.131836,28.
1,0.0878906,31.5966,0.0878906,35.5283,0.0878906,39.9492,0.0878906,44.9202,0.0878
906,50.5097,0.0439453,56.7948,0.0439453,63.8619,0.0439453,71.8084,0.0439453,80.7
438,0,90.791,0.0439453,102.088,0.0439453,114.792,0.0439453,129.075,0.0439453,145
.137,0.0439453,163.196,0.0439453,183.503,0.0439453,206.337,0.0439453,232.013,0.0
439453,260.883,0.0439453,293.345,0,329.847,0,370.891,0.0439453,417.042,0.0439453
,468.935,0,527.286,0,592.898,0,666.674,0,749.63,0,842.909,0,947.795,0,1065.73,0,
1198.34,0,1347.46,0,1515.13,-0.0439453,1703.66,-0.0439453,1915.65,0,2154.02,0,24
22.05,-0.0439453,2723.43,-0.0439453,3062.32,0,3443.37,0,3871.84,-0.0439453,4353.
62,-0.0439453,4895.36,-0.0878906,5504.5,-0.0878906,6189.45,-0.0439453,6959.62,-0
.0439453,7825.62,-0.0878906,8799.39,-0.0878906,9894.32,-0.0878906,11125.5,-0.087
8906,12509.9,-0.131836,14066.5,-0.131836,15816.9,-0.175781,17785,-0.175781,19998
,-0.219727

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

ASCII,61,17.5781,-122.598,19.7654,-134.942,22.2249,-134.942,24.9904,-137.636,28.
1,-137.636,31.5966,-127.337,35.5283,-130.505,39.9492,-125.626,44.9202,-130.164,5
0.5097,-127.668,56.7948,-126.95,63.8619,-126.944,71.8084,-125.423,80.7438,-123.9
75,90.791,-133.707,102.088,-126.674,114.792,-133.961,129.075,-122.559,145.137,-1
31.06,163.196,-123.817,183.503,-133.073,206.337,-126.6,232.013,-130.158,260.883,
-125.018,293.345,-134.608,329.847,-123.46,370.891,-132.454,417.042,-124.51,468.9
35,-131.115,527.286,-123.589,592.898,-129.449,666.674,-123.533,749.63,-132.609,8
42.909,-124.04,947.795,-130.17,1065.73,-123.112,1198.34,-127.287,1347.46,-122.45
8,1515.13,-127.432,1703.66,-123.984,1915.65,-126.796,2154.02,-122.157,2422.05,-1
26.363,2723.43,-122.429,3062.32,-124.182,3443.37,-121.678,3871.84,-123.341,4353.
62,-121.484,4895.36,-122.914,5504.5,-120.015,6189.45,-122.061,6959.62,-119.713,7
825.62,-120.657,8799.39,-118.304,9894.32,-119.926,11125.5,-118.21,12509.9,-117.3
69,14066.5,-117.984,15816.9,-117.132,17785,-115.93,19998,-115.357

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

ASCII,61,17.5781,-115.288,19.7654,-152.443,22.2249,-152.443,24.9904,-129.836,28.
1,-129.836,31.5966,-125.208,35.5283,-127.582,39.9492,-126.826,44.9202,-129.52,50
.5097,-130.829,56.7948,-126.368,63.8619,-128.912,71.8084,-122.268,80.7438,-121.5
08,90.791,-131.642,102.088,-123.736,114.792,-136.134,129.075,-123.077,145.137,-1
26.835,163.196,-122.381,183.503,-131.099,206.337,-122.825,232.013,-127.109,260.8
83,-123.472,293.345,-133.06,329.847,-122.153,370.891,-131.465,417.042,-122.324,4
68.935,-132.925,527.286,-122.877,592.898,-129.504,666.674,-121.897,749.63,-130.1
38,842.909,-121.833,947.795,-129.288,1065.73,-121.973,1198.34,-128.52,1347.46,-1
22.237,1515.13,-126.549,1703.66,-121.914,1915.65,-126.651,2154.02,-121.501,2422.
05,-126.518,2723.43,-121.106,3062.32,-125.049,3443.37,-120.052,3871.84,-123.608,
4353.62,-119.75,4895.36,-123.065,5504.5,-119.373,6189.45,-121.788,6959.62,-119.3
32,7825.62,-120.272,8799.39,-118.285,9894.32,-120.053,11125.5,-117.497,12509.9,-
116.904,14066.5,-118.133,15816.9,-117.529,17785,-115.532,19998,-115.288

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

ASCII,61,17.5781,-121.765,19.7654,-134.942,22.2249,-134.942,24.9904,-137.636,28.
1,-137.636,31.5966,-127.337,35.5283,-128.636,39.9492,-125.626,44.9202,-130.164,5
0.5097,-127.849,56.7948,-126.95,63.8619,-125.734,71.8084,-126.758,80.7438,-122.8
11,90.791,-134.198,102.088,-125.808,114.792,-132.781,129.075,-121.594,145.137,-1
33.278,163.196,-123.189,183.503,-134.166,206.337,-126.6,232.013,-130.762,260.883
,-124.156,293.345,-134.502,329.847,-122.619,370.891,-135.676,417.042,-123.978,46
8.935,-132.643,527.286,-123.19,592.898,-131.295,666.674,-123.162,749.63,-135.346
,842.909,-123.993,947.795,-130,1065.73,-122.951,1198.34,-129.633,1347.46,-122.10
9,1515.13,-126.91,1703.66,-123.993,1915.65,-127.235,2154.02,-122.195,2422.05,-12
6.129,2723.43,-121.7,3062.32,-125.111,3443.37,-121.585,3871.84,-123.549,4353.62,
-121.513,4895.36,-123.323,5504.5,-119.645,6189.45,-122.402,6959.62,-119.828,7825
```

Figure 2-29 (cont.). Visual Basic subprogram FASTTEST_MultitoneSwp and FASTTEST_MultitoneSwp_RQS measurement results.

```
.62,-120.982,8799.39,-118.609,9894.32,-120.004,11125.5,-118.746,12509.9,-117.265
,14066.5,-118.156,15816.9,-117.274,17785,-115.816,19998,-115.37

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

ASCII,61,17.5781,-115.12,19.7654,-152.443,22.2249,-152.443,24.9904,-129.836,28.1
,-129.836,31.5966,-125.208,35.5283,-129.836,39.9492,-126.826,44.9202,-129.52,50.
5097,-130.348,56.7948,-126.368,63.8619,-129.083,71.8084,-123.018,80.7438,-120.43
7,90.791,-130.966,102.088,-123.261,114.792,-137.324,129.075,-122.481,145.137,-12
8.272,163.196,-121.633,183.503,-129.205,206.337,-121.757,232.013,-127.225,260.88
3,-123.141,293.345,-134.59,329.847,-120.989,370.891,-131.761,417.042,-121.734,46
8.935,-133.751,527.286,-122.624,592.898,-130.275,666.674,-121.49,749.63,-129.939
,842.909,-120.907,947.795,-130.268,1065.73,-121.705,1198.34,-129.559,1347.46,-12
2.061,1515.13,-127.163,1703.66,-121.313,1915.65,-127.697,2154.02,-120.961,2422.0
5,-127.071,2723.43,-120.856,3062.32,-125.21,3443.37,-119.854,3871.84,-123.759,43
53.62,-119.495,4895.36,-123.522,5504.5,-119.18,6189.45,-122.58,6959.62,-119.004,
7825.62,-120.346,8799.39,-118.355,9894.32,-120.296,11125.5,-117.051,12509.9,-116
.538,14066.5,-118.243,15816.9,-117.699,17785,-115.496,19998,-115.12

FASTTEST_MultitoneSwp_RQS

FASTTEST Acquire Complete SRQ Interrupt Detected

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

ASCII,31,17.57,-25.1493,-25.1403,0.175781,23.43,-25.1516,-25.1416,0.0878906,29.2
9,-25.1533,-25.1428,0.0878906,41.01,-25.155,-25.1442,0.0878906,52.73,-25.1558,-2
5.1448,0.0439453,64.45,-25.1561,-25.145,0.0439453,82.03,-25.1564,-25.1452,0.0439
453,99.6,-25.1565,-25.1453,0,123.04,-25.1565,-25.1453,0.0439453,158.2,-25.1564,-
25.1451,0,199.21,-25.1561,-25.1449,0,251.95,-25.1556,-25.1444,0,316.4,-25.155,-2
5.1437,0.0439453,398.43,-25.154,-25.1427,0,498.04,-25.1529,-25.1416,0,632.81,-25
.1517,-25.1404,-0.0439453,802.73,-25.1512,-25.1399,0,1001.95,-25.1522,-25.141,0,
1248.04,-25.1549,-25.1436,0,1599.6,-25.157,-25.1457,-0.0439453,1998.04,-25.1536,
-25.1423,0,2501.95,-25.1516,-25.1403,0,3152.34,-25.1568,-25.1455,-0.0439453,4001
.95,-25.1511,-25.1397,-0.0439453,4998.04,-25.1548,-25.1434,-0.0439453,6351.56,-2
5.1563,-25.1449,-0.0878906,7998.04,-25.1559,-25.1444,-0.0878906,10002,-25.152,-2
5.1404,-0.0878906,12498,-25.1588,-25.147,-0.131836,16002,-25.1594,-25.1471,-0.17
5781,19998,-25.1583,-25.1453,-0.219727

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

ASCII,31,17.57,-125.692,-121.153,23.43,-126.51,-122.988,29.29,-120.036,-116.369,
41.01,-125.19,-120.782,52.73,-119.896,-117.714,64.45,-121.639,-119.206,82.03,-12
9.842,-124.354,99.6,-122.329,-120.155,123.04,-122.712,-119.069,158.2,-122.231,-1
19.851,199.21,-123.519,-120.592,251.95,-120.325,-117.913,316.4,-122.177,-120.044
,398.43,-121.786,-119.832,498.04,-121.43,-119.948,632.81,-120.397,-119.398,802.7
3,-121.809,-119.419,1001.95,-120.876,-119.06,1248.04,-121.44,-118.962,1599.6,-12
0.343,-119.209,1998.04,-119.69,-118.611,2501.95,-119.665,-117.973,3152.34,-119.2
21,-118.072,4001.95,-117.926,-117.212,4998.04,-118.039,-117.63,6351.56,-117.045,
-116.283,7998.04,-116.383,-116.092,10002,-115.169,-115.194,12498,-114.903,-114.6
65,16002,-113.747,-113.074,19998,-112.661,-112.378

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

ASCII,31,17.57,-125.628,-119.672,23.43,-126.51,-122.988,29.29,-120.036,-116.369,
41.01,-124.463,-119.873,52.73,-119.33,-116.772,64.45,-120.236,-118.08,82.03,-129
.576,-123.413,99.6,-122.059,-119.307,123.04,-122.323,-117.967,158.2,-122.098,-11
8.812,199.21,-122.625,-119.845,251.95,-119.913,-116.992,316.4,-122.261,-119.23,3
98.43,-120.877,-118.965,498.04,-121.163,-119.593,632.81,-120.238,-118.84,802.73,
-121.406,-118.857,1001.95,-120.34,-118.43,1248.04,-121.126,-118.317,1599.6,-119.
974,-118.678,1998.04,-119.409,-118.101,2501.95,-119.396,-117.566,3152.34,-119.37
8,-117.53,4001.95,-118.194,-116.839,4998.04,-118.001,-117.451,6351.56,-116.803,-
116.193,7998.04,-116.134,-115.941,10002,-115.606,-115.018,12498,-114.701,-114.4,
16002,-113.831,-113.15,19998,-112.932,-112.541
```

Figure 2-29 (cont.). Visual Basic subprogram FASTTEST_MultitoneSwp and FASTTEST_MultitoneSwp_RQS measurement results.

Loading New Firmware into the Instrument

The ATS-2 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 Microsoft Windows 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 ATS-2. A detailed description of the "FTU.EXE" program is provided in Section 3, "Utility Programs," in this manual.

Firmware Download Process

ATS-2 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 0.01,1.001`**). 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 ATS-2 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.

```
' Copyright (c) 2002, Audio Precision Inc.
' All Rights Reserved.
' miniFTU sample program, version 1.0.1, August 30, 2002
' Load a firmware HEX file into the ATS-2 GPIB instrument.

Option Explicit
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 ATS2G idle
Const ERASE_SPR = &H2          ' serial poll response when ATS2G erasing
Const XFER_SPR = &H86          ' serial poll response when ATS2G 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 ATS2G
Const BLOCK_SIZE = 32767       ' size of block of data read from FW file and
    transferred
Const FW_FILE = "ATS2G.HEX"    ' name of firmware file

Sub Main()
    Dim iSPR As Integer, iBoard As Integer, iATS2G 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, iFileHandle As
    Long

    On Error GoTo miniFTU_Error
    iFileHandle = FreeFile
    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
```

Figure 2-30. Visual Basic miniFTU.bas source code.

```
    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
' Check for presence of the firmware file.
' Attempt to open the file. Close it if the file is found, otherwise invoke
the error handler and exit the program.
Open sFwFile For Input As #iFileHandle
lFileSize = FileLen(sFwFile)                    ' get size of file
If lFileSize <= 0 Then                          ' check that file is not
empty
    Close
    MsgBox "Firmware file " & sFwFile & " is empty", vbOKOnly, "miniFTU
Error"
    Exit Sub                                    ' abort program
End If

' put up a message of the required HW & SW configuration
iStatus = MsgBox("WARNING! This program will erase the firmware in your
instrument." & vbCrLf & vbCrLf & _
    "To use this program, verify that..." & vbCrLf & vbCrLf & _
    "1) The ATS-2 is in GPIB mode." & vbCrLf & _
    "2) The ATS-2 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, "ATS2G miniFTU, Ver. " & App.Major & "." &
App.Minor & "." & App.Revision)
If iStatus = vbCancel Then Exit Sub             ' if user clicked CANCEL
then abort

' setup GPIB I/O
iBoard = ilfind("GPIB0")      ' ilfind automatically places board on-line
iStatus = ilsic(iBoard)       ' interface clear
iATS2G = ildev(0, GPIB_ADDR, NO_SAD, T3s, 1, 0)  ' get handle to GPIB_ADDR
iStatus = illn(iATS2G, GPIB_ADDR, NO_SAD, iListener)  ' is there a device
there?
If iListener = 0 Then                           ' no, flag error and
abort
    ' close GPIB I/O
    ibonl 0, 0            ' Put GPIB interface board off-line
    ibonl iATS2G, 0       ' Put GPIB I/O device GPIB_Dev off-line
    iStatus = MsgBox("GPIB Communication Error" & vbCrLf & "No Listener
detected @ address " & GPIB_ADDR, _
                     vbOKOnly, "miniFTU Error")
    Exit Sub
End If

' first do a serial poll to see if bit 1 is set (indicates in update in
process)
iStatus = ilrsp(iATS2G, iSPR)
If (((iSPR And ERASE_SPR) <> ERASE_SPR) And ((iSPR And XFER_SPR) <>
XFER_SPR)) Then
    'ATS2G is idle
    sResponse = Space(100)
    sMsg = "*IDN?"
    iStatus = ilwrt(iATS2G, sMsg, Len(sMsg))          ' what instrument is
this?
    iStatus = ilrd(iATS2G, sResponse, Len(sResponse)) ' get response
    If InStr(1, sResponse, "ATS-2") > 0 Then    ' is "ATS-2" 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)
```

Figure 2-30 (cont.). Visual Basic miniFTU.bas source code.

```
        Next iNdx
        sFwVer = Trim(Right(sResponse, Len(sResponse) - l3rdComma)) ' grab
the rest of the line after the 3rd comma
    Else
        ' close GPIB I/O
        ibonl 0, 0            ' Put GPIB interface board off-line
        ibonl iATS2G, 0       ' Put GPIB I/O device GPIB_Dev off-line
        iStatus = MsgBox("Could not find ATS-2 at address " & vbOKOnly,
"miniFTU Error")
        Exit Sub
    End If

    sMsg = "HEAD ON;FWUP?"                          ' make sure headers are
on and get boot version
    iStatus = ilwrt(iATS2G, sMsg, Len(sMsg))          ' write query for
current boot version
    sBootVer = Space(100)
    iStatus = ilrd(iATS2G, 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(iATS2G, sMsg, Len(sMsg))          ' send update command

    If (iStatus And &H8000) = &H8000 Then           ' if a command error,
exit
        ' close GPIB I/O
        ibonl 0, 0            ' Put GPIB interface board off-line
        ibonl iATS2G, 0       ' Put GPIB I/O device GPIB_Dev off-line
        iStatus = MsgBox("Error in writing FWUPdate command.", vbOKOnly,
"miniFTU Error")
        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(iATS2G, iSPR)                   ' perform serial poll
    DelayStartTime = Timer()                        ' these 4 statements are
just
    Do While Timer() - DelayStartTime < 1           ' a manual delay
        DoEvents                                    '
    Loop                                            '
    iStatus = ilclr(iATS2G)                         ' 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
    ' close GPIB I/O
    ibonl 0, 0            ' Put GPIB interface board off-line
    ibonl iATS2G, 0       ' Put GPIB I/O device GPIB_Dev off-line
    sResponse = MsgBox("Serial Poll Response: " & "0x" & Hex(iSPR),
vbOKOnly, "miniFTU Erase Cycle Timeout")
    Exit Sub              ' abort program
```

Figure 2-30 (cont.). Visual Basic miniFTU.bas source code.

```
    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(iATS2G, EOI_OFF)                ' turn off EOI
    iStatus = iltmo(iATS2G, T1s)                    ' reset timeout to 1 sec

    For iNdx = 1 To lFileSize \ BLOCK_SIZE          ' transferring any whole
    blocks
        sBlock = Input(BLOCK_SIZE, iFileHandle)       ' read up to blocksize
    bytes from #1
        iStatus = ilwrt(iATS2G, sBlock, Len(sBlock))  ' write block to
    instrument

        If (iStatus And &H8000) = &H8000 Then
            Close
            ' close GPIB I/O
            ibonl 0, 0            ' Put GPIB interface board off-line
            ibonl iATS2G, 0       ' Put GPIB I/O device GPIB_Dev off-line
            MsgBox "Aborting transfer, status = " & "0x" & Hex(iStatus),
    vbOKOnly, "miniFTU GPIB Error"
            Exit Sub                                ' abort program
        End If

        iStatus = ilrsp(iATS2G, 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, iFileHandle) ' read last
    partial block from #1
        iStatus = ilwrt(iATS2G, sBlock, Len(sBlock))  ' write block to
    instrument
        If (iStatus And &H8000) = &H8000 Then
            Close                                   ' close s-record file
            ' close GPIB I/O
            ibonl 0, 0            ' Put GPIB interface board off-line
            ibonl iATS2G, 0       ' Put GPIB I/O device GPIB_Dev off-line
            MsgBox "Aborting transfer, status = " & "0x" & Hex(iStatus),
    vbOKOnly, "miniFTU GPIB Error"
            Exit Sub                                ' abort program
        End If
        iStatus = ilrsp(iATS2G, iSPR)               ' serial poll instrument
    End If
    Close                                           ' close s-record file
    iStatus = ileot(iATS2G, EOI_ON)                 ' turn EOI back ON
    If ((iSPR And DONE_SPR) = DONE_SPR) Then        ' if serial poll = DONE
        iStatus = ilwrt(iATS2G, "GO", 2)            ' tell instrument to reboot
    Else                                            ' an error must have
    occurred
        sResponse = MsgBox("Serial Poll Response:" & Str$(iSPR), vbOKOnly, "Xfer
    Cycle Error")
    End If

    ' close GPIB I/O
    ibonl 0, 0            ' Put GPIB interface board off-line
    ibonl iATS2G, 0       ' Put GPIB I/O device GPIB_Dev off-line
    Exit Sub

miniFTU_Error:
    ibonl 0, 0            ' Put GPIB interface board off-line
    ibonl iATS2G, 0       ' Put GPIB I/O device GPIB_Dev off-line
    MsgBox Err.Description, vbCritical + vbOKOnly, "miniFTU Error"
End Sub
```

Figure 2-30 (cont.). 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 ATS-2 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 ATS-2 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.
    3a. Receive the response: AUDIO PRECISION,ATS-2,10047 01-23-2002,1.001 in the Receive String text box.
    3b. 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 0.01,1.001
    5b. Press the Enter key.
6. The ATS-2 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 ATS-2 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\ATS2G1001.hex>.
    7d. Press the Enter key.
    7e. The ATS-2 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 ATS-2 rear panel GPIB LED should be ON, all other LEDs should be OFF.
    8b. The ATS-2 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.

Chapter 3
Utility Software

Audio Precision GPIB Talk & Listen Utility

This utility allows you to interact with ATS-2/GPIB via a convenient windows interface menu. The source code for this utility provides an example of how to implement programmable control of ATS-2 GPIB. This utility is optimized for control of ATS-2, 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 6.0, and uses the following components

- Microsoft Common Dialog Control 6.0 (SP3)
- Microsoft Rich Textbox Control 6.0 (SP3)
- Microsoft Tabbed Dialog Control 6.0 (SP3)
- 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, System Two Cascade Plus, the Portable One family, ATS-1, and ATS-2). 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.

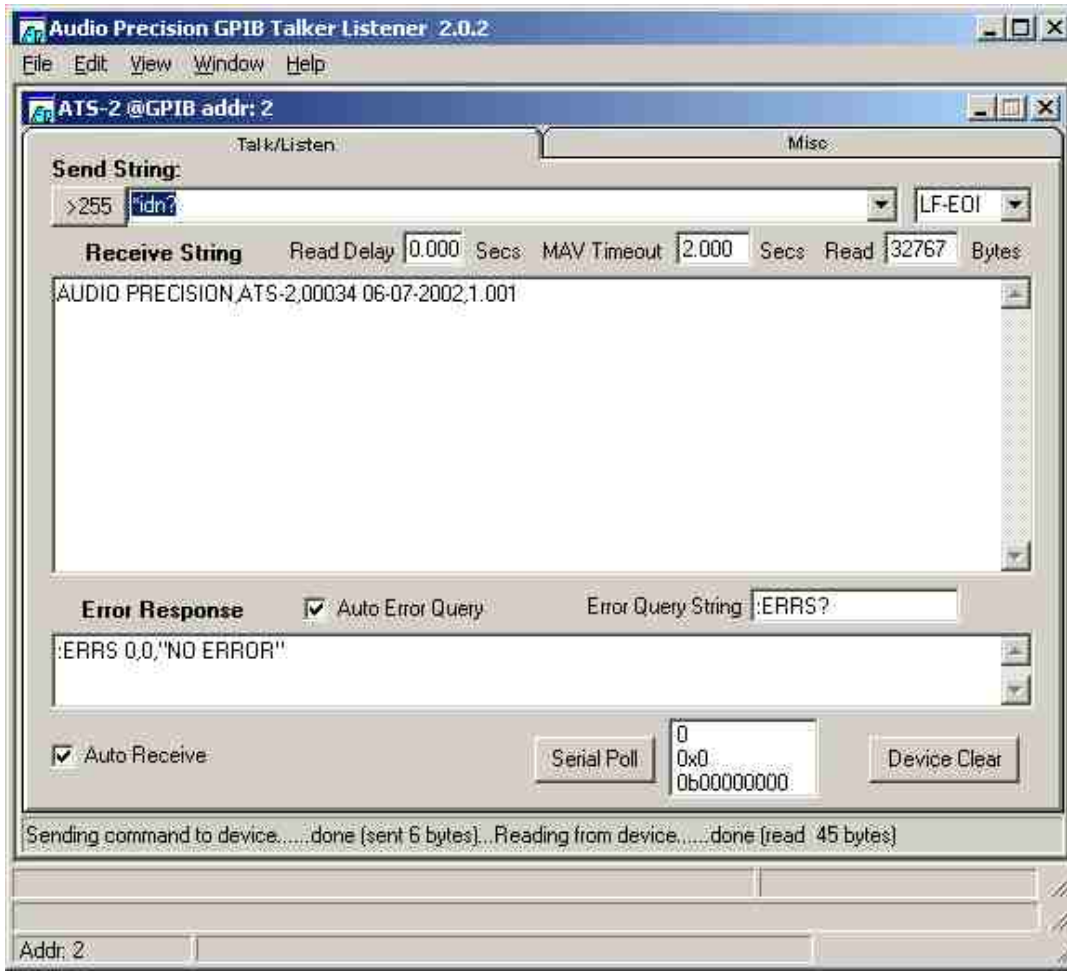


Figure 3-1. Talker/Listener Main Panel.

Menu Bar

File Menu

- **Start Logging to File**—This menu item displays a file browser and allows selection of a log file to be used for logging all operations. Once logging has been started this menu item changes to "Stop Logging to File". Logging will continue until this menu item is selected again. The log filename may not be the file already selected to run with the "Select Log File to Run" menu item below.
- **Select Log File to Run**—This menu item allows selection of a previously saved log file to be executed as a script. Once a log file has been selected to run, this menu item changes to "Unselect Log File to Run". The log file contents will be parsed and command strings sent with the Send String button will be sent to the instrument automatically until the end of the log file is reached. The log filename to run may not be the same file already selected for logging with the "Start Logging to File" menu above.
- **Comment**—This menu item is enabled 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, copyright notice, and a scrollable list box showing control keys and their definitions.

Status Bar

- **Run Log Filename**—The name of the log file previously selected to be run as a macro. Blank if a log file is unselected.
- **Log Filename**—The name of the log file previously selected for logging the current session. Blank if logging is disabled.
- **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

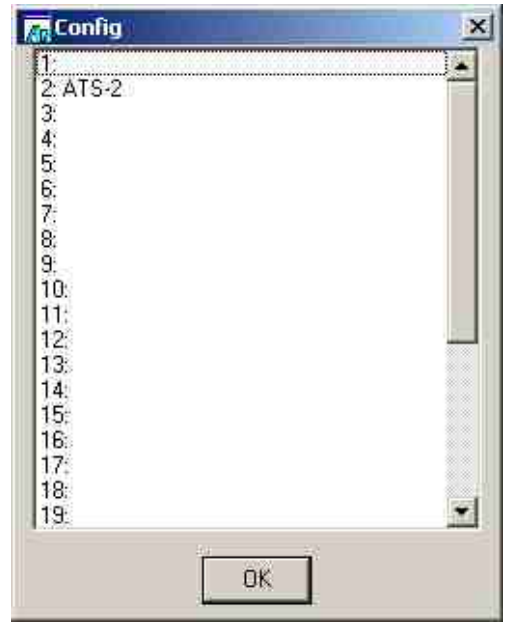


Figure 3-3. Config panel (typical).

- **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

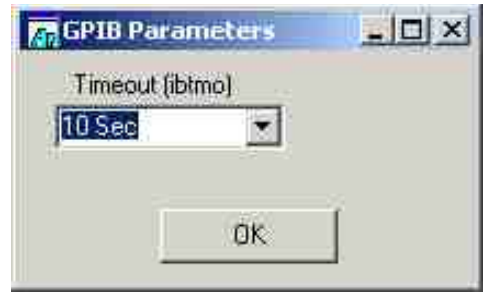


Figure 3-4. GPIB Parameters (Timeout) Panel.

- **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 active.

Instrument Panel(s)

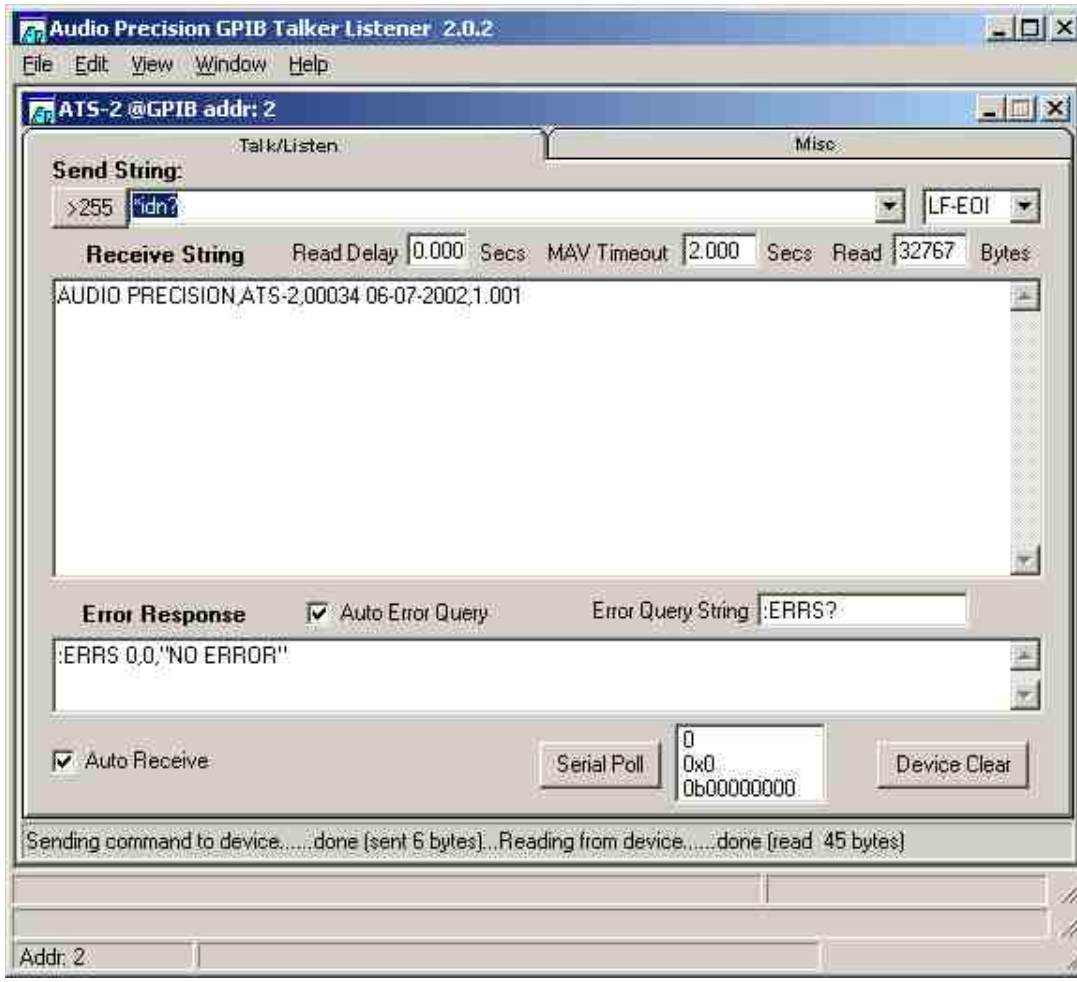


Figure 3-5. Talker/Listener Main Panel with ATS-2 instrument panel.

- **>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.
- **^r (Run Log File in Automatic Mode)**—Sets the Run Log File execution mode to Automatic. This will automatically execute all Send String entries in the selected log file at the next press of the <Enter> key if the cursor is in the Send String edit box. The log file is executed until the end of file is encountered or until File Unselect Log File to Run is selected.
- **^s (Run Log File in Single Step Mode)**—Sets the Run Log File execution mode to Single Step Mode. This is the default. Each Send String entry in the Run Log File will be executed only on the next press of the <Enter> key if the cursor is in the Send String edit box.
- **<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

Read Delay is 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

MAV Timeout is 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)

Read (Bytes) is 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

Error Response—Displays 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 Query

Auto Error Query—Specifies 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

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

Auto Receive—Specifies 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 last condition is necessary for 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—Polls the instrument serially and displays 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

Device Clear—Sends a selective device clear (SDI) bus message to the instrument.

Status Bar

Status Bar—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 bytes sent or read will be one more than the number of bytes in either the Send String control or the Receive String control. This extra byte represents the LF (linefeed) that is automatically appended (Send) or removed (Receive).

Send String >255 Characters Panel

Send String

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—Sends the contents of the Send String to the instrument,

Cancel

Cancel—Closes this panel without sending the contents of the Send String control to the instrument.

ATS-2 GPIB Firmware Transfer Utility

This utility transfers new firmware from a file supplied by Audio Precision into the ATS-2 or 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 ATS-2 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 FLU 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

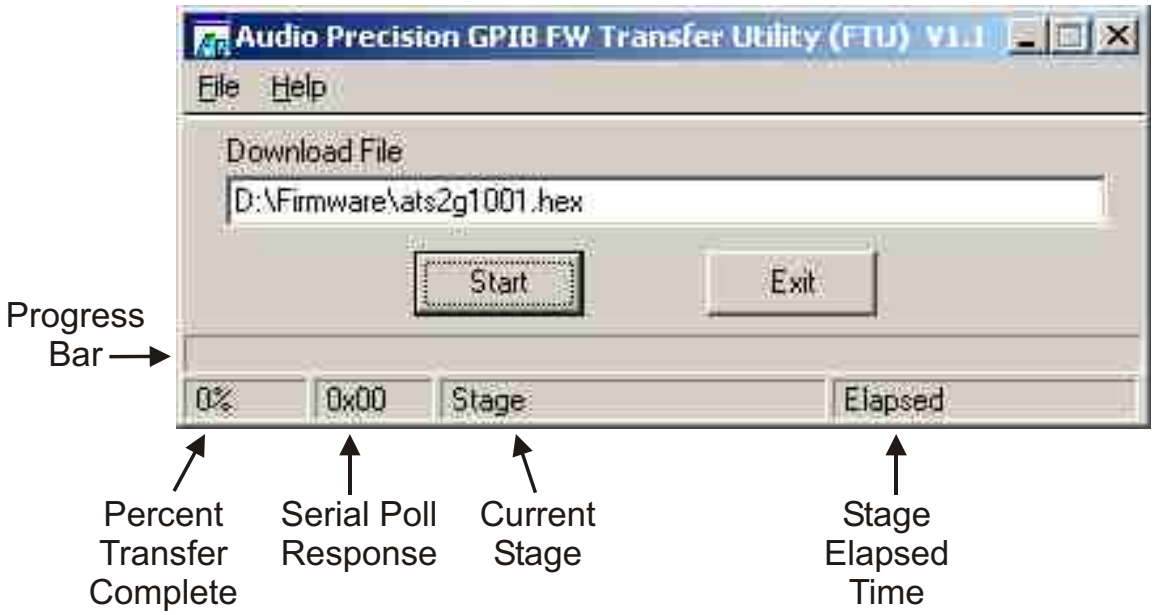


Figure 3-6. GPIB firmware transfer utility main panel.

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

Download File—This field shows the path and filename of the file selected for download to the ATS-2. This indicator is located just above the Start and Exit buttons.

Progress Bar

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 download is complete and 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 depends on the particular instrument model and 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 ATS2G 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

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

Exit

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.

Chapter 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:

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

The command argument is:

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

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

Command Syntax: *DDT **data**

Example 1: `*DDT #231:AGEN:AMPL A,2V;:ANLG:AUT A,OFF`

Example 2: `*DDT #0:AGEN:AMPL A,2V;:ANLG:AUT A,OFF<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: `#2:AGEN:AMPL A,2V;:ANLG:AUT A,OFF`

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” through “9”, “A” through “Z”, “_”, and “a” through “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;AMPL A,1V;AMPL B,2V;DAS:FRQ
3000HZ
```

A macro that implements this series of commands would be:

```
*DMC "SETAGEN",#247:AGEN:OUTPUT AB;AMPL
A,$1;AMPL B,$2;DAS:FRQ1 $3
```

The macro invocation would be:

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

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

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

Macros may be deleted (purged) 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",#247:AGEN:OUTPUT AB;AMPL
A,$1;AMPL B,$2;DAS:FRQ1 $3
```

Example 2:
```
*DMC "SETAGEN",#0:AGEN:OUTPUT AB;AMPL
A,$1;AMPL B,$2;DAS:FRQ1 $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 (F0h).

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:
```
#247:AGEN:OUTPUT AB;AMPL A,$1;AMPL
B,$2;DAS:FRQ1 $3
```

*IDN?

Returns the manufacturer, product model, serial number/adjustment date, and firmware version.

The serial number/adjustment date field consist of an <nr1> encoded numeric serial number delimited with a space character from the MM-DD-YYYY encoded adjustment date (e.g. “10047 1-23-2002").

Response argument(s):

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

Related Commands: <None>

Command Syntax: *IDN?

Response Syntax: **idn**

Example:
```
*IDN?
```

Response:
```
AUDIO PRECISION,ATS-2,10047 1-23-2002,1.001
```

*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?`

Response 1: `"MACRO1","MACRO2", ...`

Response 2: `""  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 if sent back to the instrument 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.

The HEADER command is ignored. The response will always contain command headers.

The sequence of command message groups in the response will be: DIN, DOUT, DIOS, SYNC, ANLG, AGEN, DSP, DGEN, MON, AUX, TRIG, SWR, DCX, SETTLING.

This is equivalent to sending the following queries:
:DIN:SET?;:DOUT:SET?;:DIOS:SET?;:SYNC:SET?;:ANLG:SET?;:AGEN:SET?;:DSP:SET?;:DGEN:SET?;:MON:SET?;:AUX:SET?;:TRIG:SET?;:SWR:SET?;:DCX:SET?;:SETTLING:SET?

Note that the SWR and DCX responses will not be provided if these options are not connected on the APIB bus or are turned off when the instrument is powered on.

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,BITS;DETECTOR AVG;MODE ACTIVE; . . .`

*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 for each field will be 0 if an option is not installed, except that field 2 will always have code 1, 2, or 3.

Field No.	Response Code	Option Installed
1	1	600 ohm Analog Analyzer Input Termination
2	1	150 ohm Analog Generator Output Impedance
	2	200 ohm Analog Generator Output Impedance
	3	600 ohm Analog Generator Output Impedance
3	1	High Bandwidth A/D Converter Hardware (part of High Performance Option)
4	1	Intervu Digital Interface Hardware (part of High Performance Option)
5	1	Always 1 (reserved for future use)
6	1	SWR-2122 (one or more on APIB port)
7	1	DCX-127 (on APIB port)

Response argument(s):

- **opt1** - <nr1>
- **opt2** - <nr1>
- **opt3** - <nr1>
- **opt4** - <nr1>
- **opt5** - <nr1>
- **opt6** - <nr1>
- **opt7** - <nr1>

Related Commands: <None>

Command Syntax: *OPT?

Response Syntax: **opt1, opt2, opt3, opt4, opt5, opt6, opt7**

Example: `*OPT?`

Response: `1,3,1,1,1,0,0<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.
6. Will not affect any digital user filters loaded for the DSP Audio Analyzer program.

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:

- NONE

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 ATS-2 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 Commands

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 505,13,":AGEN:DAS:FRQ1, AGEN, BELOW MINIMUM FREQUENCY.";502,15,":DGEN:WFM, UNKNOWN PARAMETER."`

: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: > 0, < 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`

Chapter 5
Analog Generator Commands

The header path for the analog generator is :AGEN:. These commands set parameters for the analog generator.

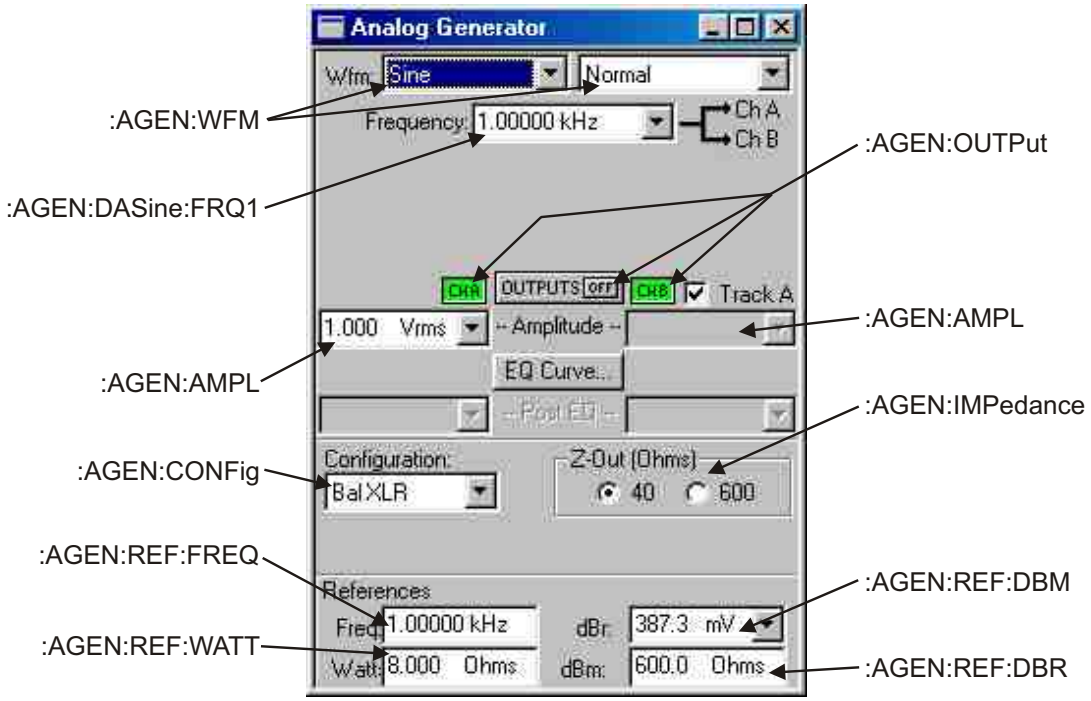


Figure 5-1. ATS software Analog Generator control panel AGEN GPIB commands.

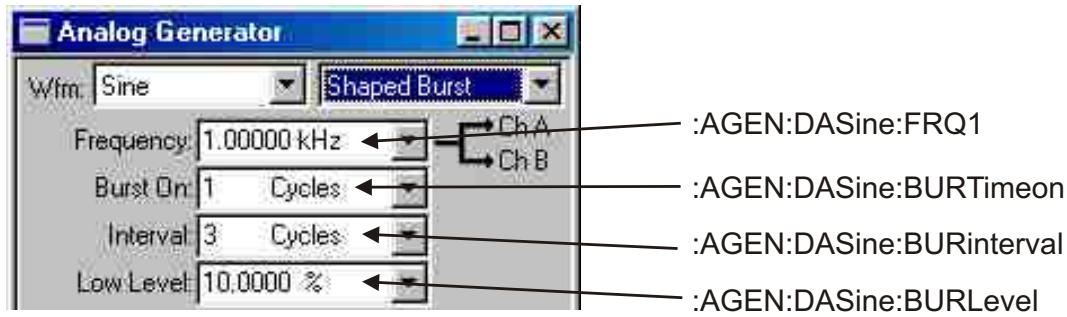


Figure 5-2. AGEN shaped burst commands.

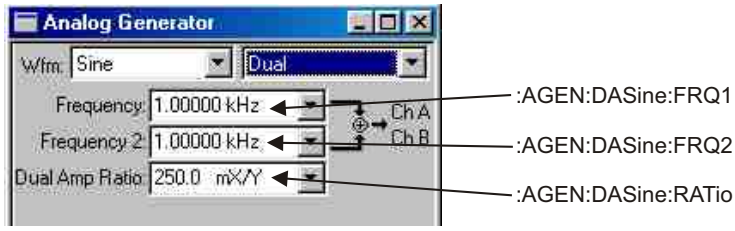


Figure 5-3. AGEN Sine Dual commands.

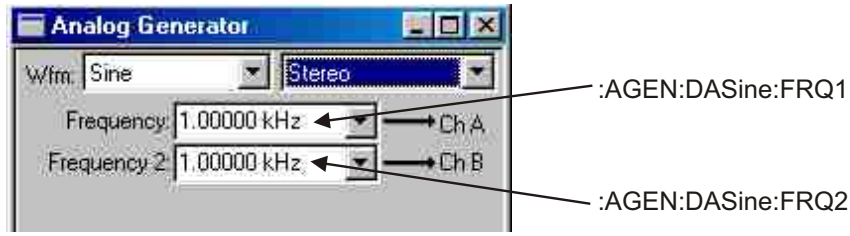


Figure 5-4. AGEN Sine Stereo commands.

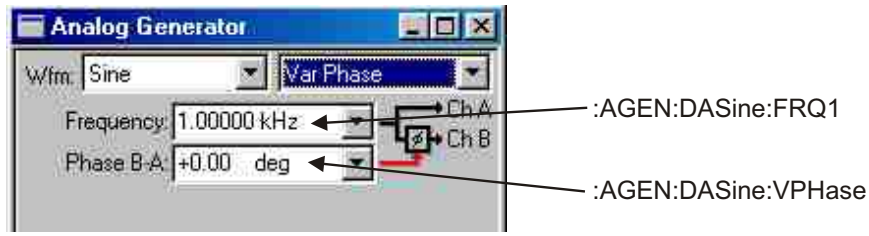


Figure 5-5. AGEN Sine Variable phase commands

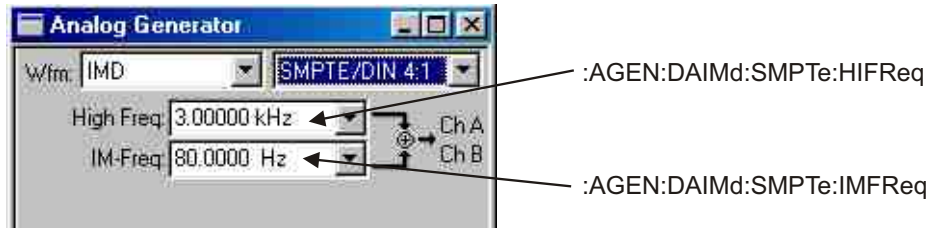


Figure 5-6. AGEN IMD SMPTE/DIN commands

:AGEN:AMPL

This command 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 output configuration.

The maximum amplitude for the balanced output configuration is 16v RMS.

The maximum amplitude for the unbalanced or common mode test output configuration is 8v RMS.

The command arguments are:

- **channel** - { AB | A | B }
- **ampl** - <nrf> (range: see text above) { V | DBM | DBR | DBU | DBV | 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: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 | DBM | DBR | DBU | DBV | IDBR | VP | VPP | W }

Response argument(s):

- **response_channel** - { A | B }
- **ampl** - <nrf> { V | DBM | DBR | DBU | DBV | IDBR | VP | VPP | W }

Related Commands: :AGEN:AMPL

Command Syntax: :AGEN:AMPL? **channel, units**

Response Syntax: :AGEN:AMPL **response_channel, ampl**

Example: `:AGEN:AMPL? A,DBU`

Response: `:AGEN:AMPL A,8.239DBU`

:AGEN:CONFig

This command sets the output configuration. The possible settings are balanced (BAL), unbalanced (UNBal), and common mode test (CMTSt).

The command argument is:

- **configuration** - { BAL | CMTSt | UNBal }

Default: BAL

Related Commands: :AGEN:CONFig?

Command Syntax: :AGEN:CONFig **configuration**

Example: `:AGEN:CONFIG CMTST`

:AGEN:CONFig?

This query returns the current output configuration setting.

Response argument(s):

- **configuration** - { BAL | CMTSt | UNBal }

Related Commands: :AGEN:CONFig

Command Syntax: :AGEN:CONFig?

Response Syntax: :AGEN:CONFig **configuration**

Example: `:AGEN:CONFIG?`

Response: `:AGEN:CONFIG CMTST`

:AGEN:DAIMd:SMPTe Compound Command Header

These commands set the IMD (D/A) parameters for the digitally generated SMPTE waveform provided through ATS-2 D/A converters. These parameters are used whenever the IMD (D/A) waveform is selected with the :AGEN:WFM DAIMd,SMP1 or SMP2 command.

: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, ≤ 61665 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 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:DASine Compound Command Header

These commands control the parameters for the digitally generated D/A Sinewave functions provided by ATS-2 D/A converters via the analog generator output.

:AGEN:DASine:BURinterval

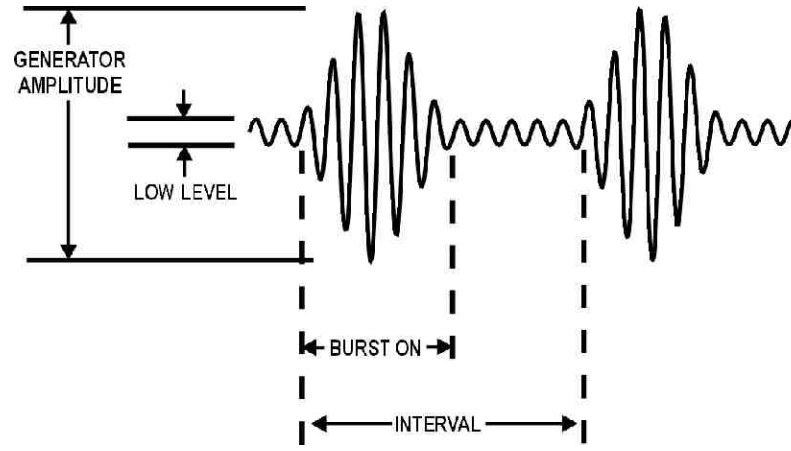
This command sets the length of the shaped burst interval in cycles or seconds.

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 range in seconds is dependent on the output frequency set by the :AGEN:DASine:FRQ1 command.

Seconds are rounded up to the nearest whole cycle.

Range @ Max Freq (61665 Hz)	CYCLes	SEC
Minimum	2	32.43331 µs
Maximum	65535	1.062775

Range @ Min Freq (2 Hz)	CYCLes	SEC
Minimum	2	1.0
Maximum	65535	32768



The command argument is:

- **interval** - <nrf> (range: see table above) { CYCLes | SEC }

Default: 3CYCLES

Related Commands: :AGEN:DASine:BURinterval?,:AGEN:DASine:BURTimeon, :AGEN:WFM, :AGEN:DASINE:FRQ1

Command Syntax: :AGEN:DASine:BURinterval **interval**

Example: `:AGEN:DASINE:BURINTERVAL 1000CYCLES`

:AGEN:DASine:BURinterval?

Returns the length (in cycles) of the burst interval (see diagram under :AGEN:DASINE:BURINTERVAL).

The command arguments are:

- **units** - { CYCLes | SEC }

Response argument(s):

- **interval** - <nrf> { CYCLes | SEC }

Related Commands: :AGEN:DASine:BURinterval

Command Syntax: :AGEN:DASine:BURinterval? **units**

Response Syntax: :AGEN:DASine:BURinterval **interval**

Example: `:AGEN:DASINE:BURINTERVAL? CYCLES`

Response: `:AGEN:DASINE:BURINTERVAL 1000CYCLES`

:AGEN:DASine:BURLevel

This command sets the amplitude of the burst signal during the "burst off time". This amplitude is expressed as a ratio of the "burst on time" amplitude.

The command argument is:

- **level** - <nrf> (range: ≥ 0.00001192, ≤ 100) { PCT | DB | PPM | X_Y }

Default: 10.0 PCT

Related Commands: :AGEN:AMPL, :AGEN:DASine:BURinterval, :AGEN:DASine:BURTimeon, :AGEN:DASine:BURLevel?, :AGEN:WFM

Command Syntax: :AGEN:DASine:BURLevel **level**

Example: `:AGEN:DASine:BURLEVEL 12.5PCT`

:AGEN:DASine:BURLevel?

This query returns the burst off time amplitude (as a percentage of 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:DASine:BURLevel

Command Syntax: :AGEN:DASine:BURLevel? **units**

Response Syntax: :AGEN:DASine:BURLevel **level**

Example: `:AGEN:DASine:BURLEVEL? PCT`

Response: `:AGEN:DASine:BURLEVEL 12.5PCT`

:AGEN:DASine:BURTimeon

Sets the length of the burst on time in cycles or seconds. 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 range in seconds is dependent on the output frequency set by the :AGEN:DASine:FRQ1 command.

Seconds are rounded up to the nearest whole cycle.

Range @ Max Freq (61665 Hz)	CYCLes	SEC
Minimum	1	1.54643E-5
Maximum	65535	1.0627585

Range @ Min Freq (2 Hz)	CYCLes	SEC
Minimum	1	0.5
Maximum	65535	32767.5

The command argument is:

- **timeon** - <nrf> (range: see table above) { CYCLes | SEC }

Default: 1CYCLES

Related Commands: :AGEN:DASine:BURinterval, :AGEN:WFM, :AGEN:DASine:BURTimeon?, :AGEN:DASINE:FRQ1

Command Syntax: :AGEN:DASine:BURTimeon **timeon**

Example: `:AGEN:DASINE:BURTIMEON 50CYCLES`

:AGEN:DASine:BURTimeon?

Returns the length of the burst on time in cycles or seconds.

Response argument(s):

- **timeon** - <nrf> { CYCLes | SEC }

Related Commands: :AGEN:DASine:BURTimeon

Command Syntax: :AGEN:DASine:BURTimeon?

Response Syntax: :AGEN:DASine:BURTimeon **timeon**

Example: `:AGEN:DASINE:BURTIMEON?`

Response: `:AGEN:DASINE:BURTIMEON 50CYCLES`

:AGEN:DASine:FRQ1

This command sets a frequency for the digitally generated sinewaves available with the :AGEN:WFM DASine functions (SINE, VPHase, STEReo, DUAL, SHAPed) and DASPecial function POLarity, and the DASQuare function. The frequency of this sinewave is defined for each function in the table below.

DASine,SINE	Single sinewave frequency for both channels A & B
DASine,VPHase	Single sinewave frequency for both channels A & B with variable phase for channel B.
DASine,STEReo	Channel A sinewave frequency
DASine,DUAL	Two tone sinewave signal, controls the frequency of the higher amplitude sinewave.
DASine,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.
DASPecial,POLarity	Frequency of the polarity waveform, max frequency range is 30832.3 Hz.
DASQuare	Fundamental frequency of the square waveform, maximum frequency range is 20019 Hz.

The command argument is:

- **frequency** - <nrf> (range: ≥ 2, ≤ 61665) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz}

Default: 1000 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, special polarity waveform, and square waveform available with the :AGEN: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: :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: ≥ 2, ≤ 61603.8) { HZ | CENT | DECS | DHZ | DPCT | DPPM | F_R | OCTS | PCTHz }

Default: 1000 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: -138.474 dB to 0 dB, 0.00001192% to 100%, 0.1192 to 1E+6 PPM, 119.2E-9 to 1.0 X/Y.

The command argument is:

- **ratio** - <nrf> (range: ≥ 1.192E-7, ≤ 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:DASR

This command specifies the sample rate of the D/A converter when the DAARbitrary waveform is selected.

An error will be generated if the currently selected waveform is not the DAARbitrary waveform when this command is received.

The command argument is:

- **rate** - { F65K | F131k | ISR | OSR }

Default: F65K

Related Commands: :AGEN:WFM, :AGEN:DASR?

Command Syntax: :AGEN:DASR **rate**

Example: `:AGEN:DASR F131K`

:AGEN:DASR?

This query returns the sample rate of the D/A converter. The response is valid only if the current AGEN waveform type

DAARBITRARY,NONE.

The response is invalid if the current waveform type is any other waveform.

Response argument(s):

- **rate** - { F65K | F131k | ISR | OSR }

Related Commands: :AGEN:WFM, :AGEN:DASR, :AGEN:SET?, *LRN?

Command Syntax: :AGEN:DASR?

Response Syntax: :AGEN:DASR **rate**

Example: `:AGEN:DASR?`

Response: `:AGEN:DASR F131K`

:AGEN:IMPedance

This command sets the analog generator output impedance. Sending an invalid impedance for the current output configuration will generate an execution error. The valid choices are detailed in the following table. Note that the 150, 200, and 600 ohm settings are mutually exclusive and will generate an execution error "Conflict with installed options" if the required hardware option is not installed (Standard, EuroZ, or 600).

	Impedance(s)		
Output Configuration	**Standard**	**Euro Z**	**600 Ω**
Balanced, CMTST	40, 150	40, 200	40, 600
Unbalanced	20, 50	20, 50	20, 50

The command argument is:

- **impedance** - { Z20 | Z40 | Z50 | Z150 }
  for Euro Z option { Z20 | Z40 | Z50 | Z200 }
  for 600 Ω option { Z20 | Z40 | Z50 | Z600 }

Default: Z40

Related Commands: :AGEN:CONFig, :AGEN:IMPedance?, *OPT?

Command Syntax: :AGEN:IMPedance **impedance**

Example: `:AGEN:IMPEDANCE Z20`

:AGEN:IMPedance?

This query returns the analog generator output impedance setting.

Response argument(s):

- **impedance** - { Z20 | Z40 | Z50 | Z150 }
  for Euro Z option { Z20 | Z40 | Z50 | Z200 }
  for 600 Ω option { Z20 | Z40 | Z50 | Z600 }

Related Commands: :AGEN:IMPedance

Command Syntax: :AGEN:IMPedance?

Response Syntax: :AGEN:IMPedance **impedance**

Example: `:AGEN:IMPEDANCE?`

Response: `:AGEN:IMPEDANCE Z20`

: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 Compound Command Header

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?, :AGEN:AMPL

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?, :AGEN:AMPL

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?, :AGEN:DASine:FRQ1, :AGEN:DASine:FRQ2

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?, :AGEN:AMPL

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.

The response will contain the :AGEN:DASR message unit only if :AGEN:WFM is set to DAARbitrary (see the :AGEN:DASR command).

Response argument(s):

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

Related Commands: (see all other generator commands)

Command Syntax: :AGEN:SET?

Response Syntax: :AGEN:**settings**

Example:
```
:AGEN:SET?
```

Response:
```
:AGEN:REF:DBM 600;
DBR 0.3873V;
WATT 8;
:AGEN:OUTPUT OFF;
CONFIG UNBAL;
IMPEDANCE Z40;
AMPL A,1.2V;
AMPL B,7.5V;
```

```
WFM DAARBITRARY,NONE;
DAIMD:SMPTE:HIFREQ 3000HZ;
IMFREQ 80;
:AGEN:DASINE:FRQ1 1000HZ;
FRQ2 1000HZ;
BURINTERVAL 3CYCLES;
BURLEVEL 12.5PCT;
BURTIMEON 1CYCLES;
RATIO 0.25X_Y;
VPHASE 0;
:AGEN:DASR F65K
```

:AGEN:WFM

This command specifies a generator waveform type.

Some waveform types may be specified with only one argument, while others require both arguments. The first argument specifies the type of waveform. The second argument selects a specific waveform sub-type.

In the case of D/A Arbitrary, D/A Noise, and D/A Square 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.

If arbitrary waveform is specified, a generator arbitrary waveform must be selected with the :DGEN:ARBWfm command in order to provide an output signal. The output will be muted (no waveform loaded into DSP memory) unless a generator arbitrary waveform is selected. See the :DGEN:ARBWfm command.

This command interacts with the :DGEN:WFM ARBitrary command when waveform type DAARbitrary is selected. Both commands may change the generator waveforms currently loaded into the DSP generator memory, thus affecting the signal generated by the other. This command will function normally, but will automatically inherit the waveforms which are in use by the digital generator.

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)	SMPTE/DIN 1:1 (SMP1), SMPTE/DIN 4:1 (SMP4)
D/A Sine (DASine)	Dual (DUAL), Shaped Burst (SHAPed), Normal (SINE), Stereo (STEReo), Variable Phase (VPHase)
D/A Noise (DANoise)	no argument required (but will accept NONE)
Special D/A (DASPecial)	Polarity (POLarity) , Pass Thru (PASSthru)
D/A Square (DASQuare)	no argument required (but will accept NONE)

The command arguments are:

- **type** - { DAARbitrary | DAIMd | DANoise | DASine | DASPecial | DASQuare }
- **subtype** - { SINE | DUAL | NONE | PASSthru | POLarity | SHAPed | SMP1 | SMP4 | STEReo | VPHase }

Default: SINE, SINE

Related Commands: :DGEN:ARBWfm, :DGEN:WFM, :AGEN:WFM?

Command Syntax: :AGEN:WFM **type[, subtype]**

Example 1:
```
:AGEN:WFM DASQUARE
```

Example 2:
```
:AGEN:WFM DAIMD,SMP1
```

:AGEN:WFM?

Returns the current type of waveform (and optional specific waveform) being generated. There will be only one argument for DAARbitrary, DANoise, and DASQuare waveform types. There will be two arguments for all other types of waveforms.

Response argument(s):

- **type** - { DAARbitrary | DAIMd | DANoise | DASine | DASPecial | DASQuare }
- **subtype** - { SINE | DUAL | NONE | PASSthru | POLarity | SHAPed | SMP1 | SMP4 | STEReo | VPHase }

Related Commands: :AGEN:WFM

Command Syntax: :AGEN:WFM?

Response Syntax: :AGEN:WFM **type[, subtype]**

Example 1:
```
:AGEN:WFM?
```

Response 1:
```
:AGEN:WFM DASQUARE
```

Example 2:
```
:AGEN:WFM?
```

Response 2:
```
:AGEN:WFM DAIMD,SMP1
```

Chapter 6
Analog Input Commands

The compound header for the Analog Input is :ANLG:.

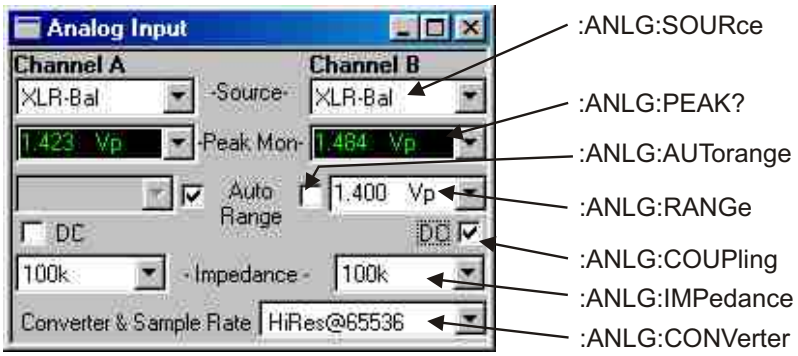


Figure 6-1. ATS software Analog Input control panel ANLG GPIB commands.

:ANLG:AUTorange

This command enables/disables auto-ranging 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 auto-ranging are OFF and ON.

If the range command (:ANLG:RANGe) is received for a channel that is currently in autorange, auto-ranging will be turned OFF for that channel.

The command argument is:

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

Default: AB, ON

Related Commands: :ANLG:AUTorange?, :ANLG:RANGe

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

Example: `:ANLG:AUTORANGE A,OFF`

:ANLG:AUTorange?

This query returns the auto-ranging setting for the indicated analog input channel. The valid channel settings are A and B. The valid autorange responses are OFF and ON.

Command argument:

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

Response argument(s):

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

Related Commands: :ANLG:AUTorange

Command Syntax: :ANLG:AUTorange? **channel**

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

Example: `:ANLG:AUTORANGE? A`

Response: `:ANLG:AUTORANGE A,OFF`

:ANLG:CONVerter

This command selects the analog input converter. The converters are: AD1 (HiRes A/D @65536), AD2 (HiRes A/D @OSR), AD3 (HiBW A/D @131072), AD4 (HiBW A/D @262144), and AD5 (HiBW A/D @2xOSR).

An error will be generated if AD3, AD4, or AD5 is selected and the High Performance option is not installed.

The command argument is:

- **converter** - { AD1 | AD2 | AD3 | AD4 | AD5 }

Default: AD1

Related Commands: :ANLG:CONVerter?, :DOUT:RATE

Command Syntax: :ANLG:CONVerter **converter**

Example: `:ANLG:CONVERTER AD2`

:ANLG:CONVerter?

This query returns the analog input converter setting. The possible responses are: AD1 (HiRes A/D @65536), AD2 (HiRes A/D @OSR), AD3 (HiBW A/D @131072), AD4 (HiBW A/D @262144), and AD5 (HiBW A/D @2xOSR).

Response argument(s):

- **converter** - { AD1 | AD2 | AD3 | AD4 | AD5 }

Related Commands: :ANLG:CONVerter, :DOUT:RATE

Command Syntax: :ANLG:CONVerter?

Response Syntax: :ANLG:CONVERTER **converter**

Example: `:ANLG:CONVERTER?`

Response: `:ANLG:CONVERTER AD2`

:ANLG: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** - { A | B | AB }
- **coupling** - { AC | DC }

Default: AB, AC

Related Commands: :ANLG:COUPling?

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

Example: `:ANLG:COUPLING A, AC`

:ANLG: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: :ANLG:COUPling

Command Syntax: :ANLG:COUPling? **channel**

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

Example: `:ANLG:COUPLING? A`

Response: `:ANLG:COUPLING A,AC`

:ANLG:IMPedance

This command specifies the input impedance of the indicated channel(s). The possible argument values are: 600 ohms (Z600) and high impedance 100k ohms (ZHI). The 600 ohm impedance setting will generate an execution error if the required Analog Analyzer 600 Input Termination hardware option is not installed.

The command arguments are:

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

Default: ZHI

Related Commands: :ANLG:IMPedance?

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

Example: `:ANLG:IMPEDANCE A, ZHI`

:ANLG:IMPedance?

This query returns the input impedance of the indicated channel.

The command argument is:

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

Response argument(s):

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

Related Commands: :ANLG:IMPedance

Command Syntax: :ANLG:IMPedance? **channel**

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

Example: `:ANLG:IMPEDANCE? A`

Response: `:ANLG:IMPEDANCE A,ZHI`

:ANLG:PEAK?

This query returns an unsettled reading from the analog input peak monitor for the specified channel. This query automatically triggers a new reading for the specified channel and returns the reading value when the reading is ready.

The command arguments are:

- **channel** - { A | B }
- **units** - { DBU | DBV | VP }

Response argument(s):

- **peak** - <nrf> { DBU | DBV | VP }

Related Commands:

Command Syntax: :ANLG:PEAK? **channel, units**

Response Syntax: :ANLG:PEAK? **peak**

Example: `:ANLG:PEAK? A,VP`

Response: `:ANLG:PEAK 9.929VP`

:ANLG:RANGe

This command sets the attenuators and selectable gain amplifiers for the indicated channel(s). The inputs to this command are the channel and the expected input signal level in units of volts peak (VP), dBu (DBU), and dBv (DBV). The

ATS-2 will determine the appropriate range setting from this value.

The ranges (VP) are: 200, 90, 45, 22.4, 11.2, 5.6, 2.8, 1.4, 0.71, and 0.355.

If a setting value does not match a range exactly then the next higher range will be used. Any value larger than 200 (Volts peak) will generate an Execution error and the range will remain unchanged.

If the :ANLG:AUTorange is ON when this command is received, it will be reset to OFF for the indicated channel(s).

The command arguments are:

- **channel** - { A | B | AB }
- **setting** - <nrf> (table range: > 0VP, 200VP see text) { DBU | DBV | VP }

Related Commands: :ANLG:RANGe?, :ANLG:AUTorange

Command Syntax: :ANLG:RANGe **channel, setting**

Example: `:ANLG:RANGE A,5VP`

:ANLG:RANGe?

This query returns the channel range setting in the specified units. This is also true if auto-ranging is turned on.

The command arguments are:

- **channel** - { A | B }
- **units** - { DBU | DBV | VP }

Response argument(s):

- **response_channel** - { A | B }
- **setting** - <nrf> { DBU | DBV | VP }

Related Commands: :ANLG:RANGe

Command Syntax: :ANLG:RANGe? **channel, units**

Response Syntax: :ANLG:RANGe **response_channel, setting**

Example: `:ANLG:RANGE? A,VP`

Response: `:ANLG:RANGE A,5.6VP`

:ANLG:SET?

This query returns all analog input settings.

Response 1 illustrates the response message units provided when the AUTorange command is set to ON. The RANGe setting for a channel is not included in the response. This response may be sent back to the instrument and would preserve the autorange condition for each channel.

Response 2 illustrates the response message units provided when the AUTorange commands are set to OFF. The RANGe setting for a channels is included in the response. This response may be sent back to the instrument and would set the range to fixed settings with auto-ranging turned off for each channel.

Response 3 illustrates the response message units provided when the AUTorange command for channel A is set ON and the AUTorange command for channel B is set OFF. The RANGe setting for channels A is not included in the response but the range setting for channel B is included in the response. This response may be sent back to the instrument and would set the range to a fixed setting for channel B only, with auto-ranging turned on for channel A.

Response argument(s):

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

Related Commands: (see all other Analog Input setting commands)

Command Syntax: :ANLG:SET?

Example:
```
:ANLG:SET?
```

Response 1:
```
:ANLG:AUTORANGE A,ON;
AUTORANGE B,ON;
CONVERTER AD1;
COUPLING A,AC;
COUPLING B,AC;
IMPEDANCE A,ZHI;
IMPEDANCE B,ZHI;
SOURCE A,XLR;
SOURCE B,XLR
```

Response 2:
```
:ANLG:AUTORANGE A,OFF;
AUTORANGE B,OFF;
CONVERTER AD1;
COUPLING A,AC;
COUPLING B,AC;
IMPEDANCE A,ZHI;
IMPEDANCE B,ZHI;
RANGE A,1.4VP;
RANGE B,1.4VP;
SOURCE A,XLR;
SOURCE B,XLR
```

Response 3:
```
:ANLG:AUTORANGE A,ON;
AUTORANGE B,OFF;
CONVERTER AD1;
COUPLING A,AC;
COUPLING B,AC;
IMPEDANCE A,ZHI;
IMPEDANCE B,ZHI;
RANGE B,1.4VP;
SOURCE A,XLR;
```

```
SOURCE B,XLR
```

:ANLG:SOURce

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

The command arguments are:

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

Default: AB, XLR

Related Commands: :ANLG:SOURce?

Command Syntax: :ANLG:SOURce **channel, source**

Example: `:ANLG:SOURCE A,XLR`

:ANLG:SOURce?

This query 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: :ANLG:SOURce

Command Syntax: :ANLG:SOURce? **channel**

Response Syntax: :ANLG:SOURce **response_channel, source**

Example: `:ANLG:SOURCE? A`

Response: `:ANLG:SOURCE A,XLR`

Chapter 7
Digital Generator Commands

These commands set parameters for the digital generator. The header-path for the digital generator is :DGEN:.

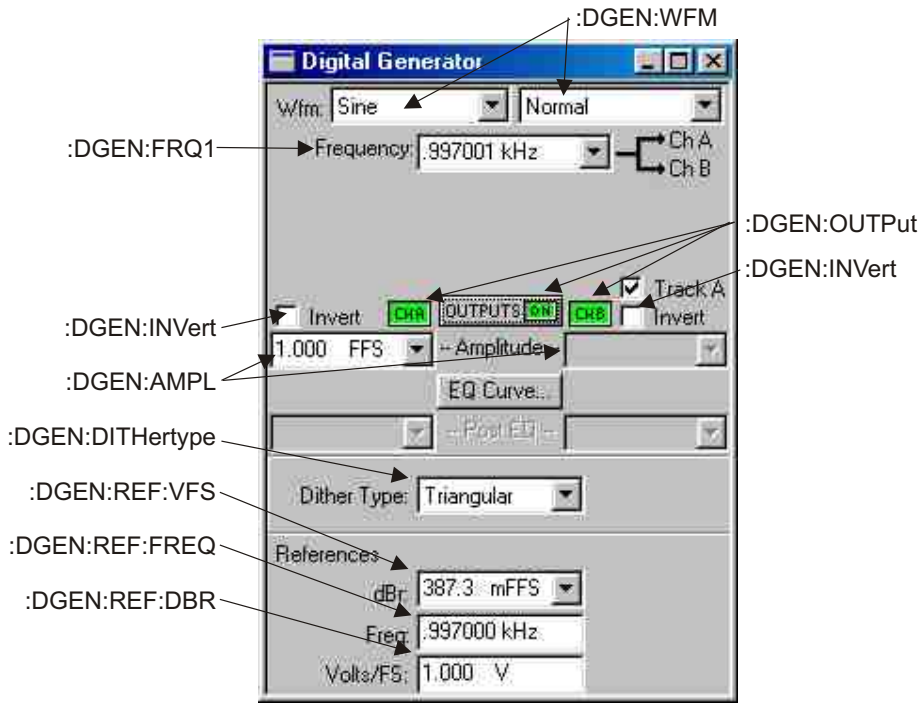


Figure 7-1. ATS software Digital Generator control panel DGEN GPIB commands.

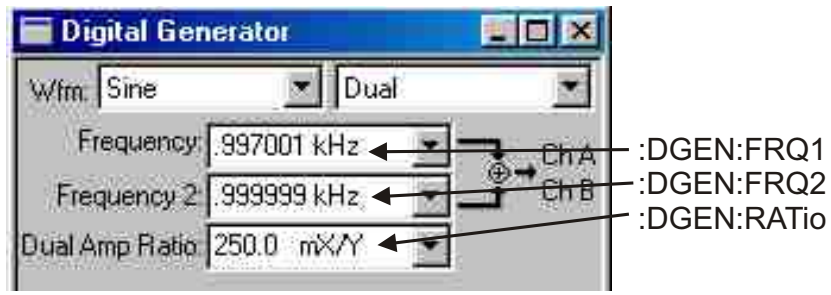


Figure 7-2. DGEN Sine Dual commands.

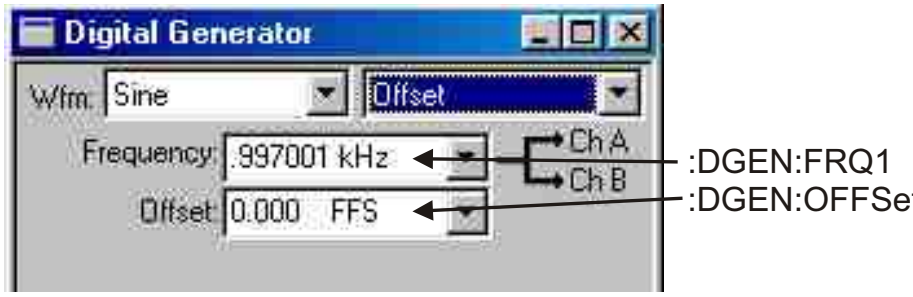


Figure 7-3. DGEN Sine Offset commands.

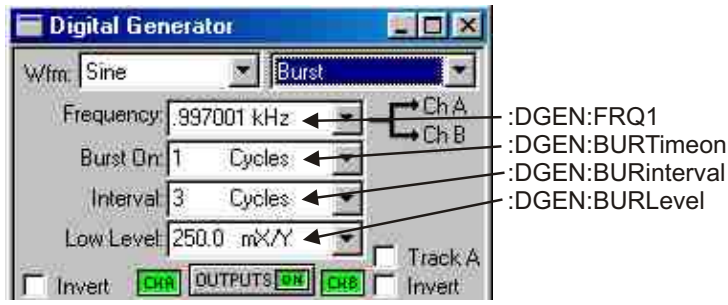


Figure 7-4. DGEN Sine Burst commands.

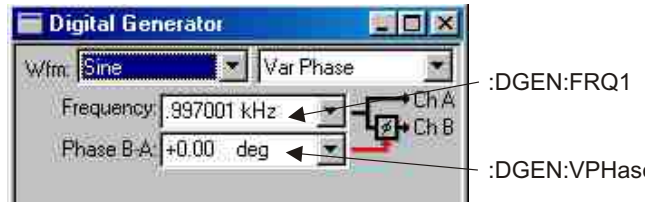


Figure 7-5. DGEN Sine Variable Phase commands.

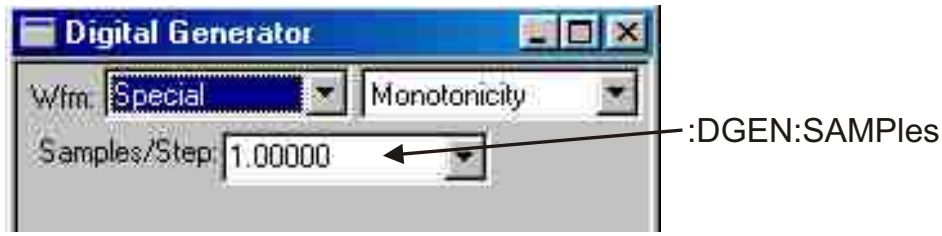


Figure 7-6. DGEN Special Monotonicity commands.

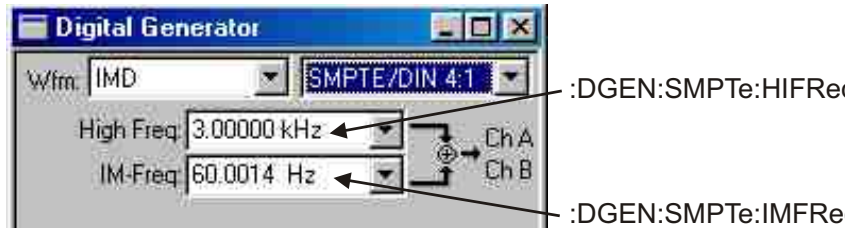


Figure 7-7. DGEN IMD SMPTE/DIN commands.

: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)

{ BITS | FFS | DBFS | DBR | DBU | DBV | DEC | IDBR | PCTFs | V | VP | VPP }

Default: 0.999756 FFS

Related Commands: :DGEN:AMPL?

Command Syntax: :DGEN:AMPL **channel, amplitude**

Example: `:DGEN:AMPL A,1FFS`

:DGEN:AMPL?

Returns the output amplitude of the specified digital generator channel in the specified units.

The command arguments are:

- **channel** - { A | B }
- **units** - { BITS | FFS | DBFS | DBR | DBU | DBV | DEC | IDBR | PCTFs | V | VP | VPP }

Response argument(s):

- **response_channel** - { A | B }
- **amplitude** - <nrf> (range: ≥ 0, ≤ 1.0 FFS)

{ BITS | FFS | DBFS | DBR | DBU | DBV | DEC | 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 firmware memory for the specified waveform 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 8192, the response will be based on 16384 points.

The query argument is:

- **points** - <nr1>

(range: 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384)

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 mono waveform file (*.agm) that was generated by the ATS Multitone Creation utility software (Utilties menu).

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 waveform size (number of points) 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:RATe, :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. An error will be generated if the register is not valid (see :DGEN:ARBSize).

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 (wfm file size is the size of a *.agm file created by the ATS Multitone Creation utility):

Size	Number of registers	Wfm File Size
16384	8	49408
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 will not affect a waveform already loaded into the digital generator waveform memory. If the requested size is the same as the current size, the waveform registers will not be cleared.

The command argument is:

- **points** - <nr1> (range: 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384)

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 | 16384)

Related Commands: :DGEN:ARBSize

Command Syntax: :DGEN:ARBSize?

Response Syntax: :DGEN:ARBSize **points**

Example: `:DGEN:ARBSIZE?`

Response: `:DGEN:ARBSIZE 1024`

:DGEN:ARBWfm

Specifies arbitrary waveform registers to load into the digital generator waveform buffers. The contents of a waveform register (see :DGEN:ARBLoad) can be changed without affecting the output signal.

The digital generator has two waveform buffers (channel 1 and channel 2). This command requires specification of both channels. The first parameter specifies the waveform register to load into channel 1 and the second parameter specifies the waveform register to load into channel 2.

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.

ATS-2 has one stereo pair of arbitrary waveform buffers to for both digital and analog (D/A) outputs. An arbitrary waveform may be generated 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 generated in the digital domain while any other analog waveform is generated by the analog outputs. However if an arbitrary waveform is desired at both analog and digital outputs simultaneously, they will be the same arbitrary waveform at the same sample rate.

Up to 388 single-channel generator arbitrary waveforms may be stored in waveform registers using the :DGEN:ARBLoad command (depending on the size of each waveform register, see :DGEN:ARBSize).

An execution error will be generated if the specified register is empty (has not been loaded using the ARBLoad command) or if an invalid register is specified.

The normal sequence of commands is: 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 digital generator memory (:DGEN:ARBWfm), and output the select the arbitrary waveform 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 digital generator arbitrary waveform memory for each output channel. If the digital generator arbitrary waveform memory for a channel has not been loaded the response will be 0 (zero). If the waveform registers are resized after a waveform has been transferred, all registers will be cleared and the response to the ARBWfm? will be 0. The contents of the digital generator arbitrary waveform memory will not be changed.

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**

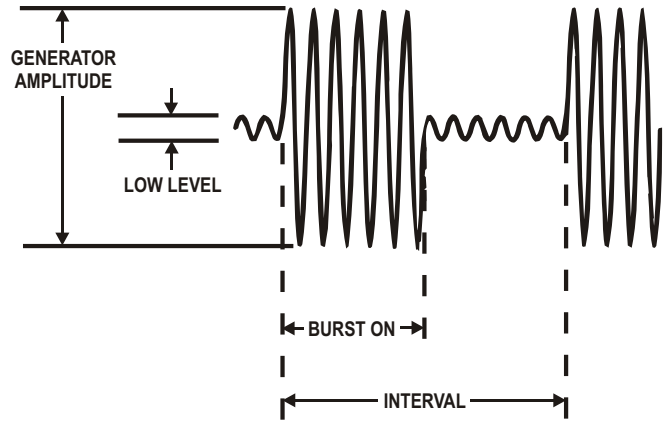
Example: `:DGEN:ARBWFM?`

Response: `:DGEN:ARBWFM 3,2`

:DGEN:BURinterval

This command sets the length of the burst interval for the Sine Burst and Sine Shaped digital generator waveforms in units of cycles or seconds. Seconds are rounded up to the reciprocal of

the nearest whole cycle. 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.

Range @ Max Freq (54000 Hz) & Sample Rate (108000 samples/s)	CYCLes	SEC
Minimum	2	37.037037 µs
Maximum	65536	1.2136296

Range @ Min Freq (2 Hz)	CYCLes	SEC
Minimum	2	1.0
Maximum	65536	32768

The command argument is:

- **interval** - <nrf> (range: see table above) { CYCLes | SEC }

Default: 3CYCLES

Related Commands: :DGEN:BURinterval?, :DGEN:BURTimeon, :DGEN:BURLevel, :DGEN:WFM, :DGEN:FRQ1

Command Syntax: :DGEN:BURinterval **interval**

Example: `:DGEN:BURINTERVAL 1000CYCLES`

:DGEN:BURinterval?

Returns the length of the burst interval in units of cycles or seconds (see diagram under :DGEN:BURinterval).

The command arguments are:

- **units** - { CYCLes | SEC }

Response argument(s):

- **interval** - <nrf> (range: see table in :DGEN:BURinterval)

Related Commands: :DGEN:BURinterval

Command Syntax: :DGEN:BURinterval? **units**

Response Syntax: :DGEN:BURinterval **interval**

Example: `:DGEN:BURINTERVAL? CYCLES`

Response: `:DGEN:BURINTERVAL 1000CYCLES`

: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. This setting does not affect the 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 in units of cycles or seconds.

Seconds are rounded up to the reciprocal of the nearest whole cycle.

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. See the diagram for :DGEN:BURinterval.

Range @ Max Freq (54000 Hz) & Sample Rate (108000 samples/s)	CYCLes	SEC
Minimum	1	18.518519 µs
Maximum	65535	1.2136296

Range @ Min Freq (2 Hz)	CYCLes	SEC
Minimum	1	0.5
Maximum	65535	32767.5

The command argument is:

- **timeon** - <nrf> (range: see table above) { CYCLes | SEC }

Default: 1CYCLES

Related Commands: :DGEN:BURinterval, :DGEN:BURLevel, :DGEN:WFM, :DGEN:BURTimeon?, :DGEN:FRQ1

Command Syntax: :DGEN:BURTimeon **timeon**

Example: `:DGEN:BURTIMEON 50CYCLES`

:DGEN:BURTimeon?

Returns the length of the burst on time in cycles or seconds.

The command arguments are:

- **units** - { CYCLes | SEC }

Response argument(s):

- **timeon** - <nrf> (range: see table in :DGEN:BURTimeon) { CYCLes | SEC }

Related Commands: :DGEN:BURTimeon

Command Syntax: :DGEN:BURTimeon? **units**

Response Syntax: :DGEN:BURTimeon **timeon**

Example: `:DGEN:BURTIMEON? CYCLES`

Response: `:DGEN:BURTIMEON 50CYCLES`

:DGEN:DITHertype

Disables digital generator dither (NONE) or selects from one of three choices of dither 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 ½ 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 SINE, SHAPed SPCial 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: ≥ 2.002, 49.9999% of :DOUT:RATE in Hz

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

e.g., 53.9999 Hz at 108000 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: ≥ 8.99935, 49.9999% of :DOUT:RATE in Hz

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

e.g., 53999.9 Hz at 108000 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 rms.

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)

{ BITS | FFS | DBFS | DBR | DBU | DBV | DEC | IDBR | PCTFs | V | VP | VPP }

Default: 0.0 FFS

Related Commands: :DGEN:OFFSet?, :DGEN:AMPL, :DGEN:WFM

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** - { BITS | FFS | DBFS | DBR | DBU | DBV | DEC | IDBR | PCTFs | V | VP | VPP }

Response argument(s):

- **offset** - <nrf> { BITS | FFS | DBFS | DBR | DBU | DBV | DEC | 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:DITHer

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 less than or equal to the FRQ1 sinewave amplitude by this ratio.

The command argument is:

- **ratio** - <nrf> (range: ≥ 1.192E-7, ≤ 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 Compound Command Header

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 ATS-2 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)

{ BITS | DEC | 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 0.8FFS`

: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:

$$REF_{DBR}\ (V) = REF_{VFS}\ (V) * REF_{DBR}\ (FFS)$$

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** - { BITS | DEC | FFS | DBFS | DBU | DBV | PCTFs | V | VP | VPP }

Response argument(s):

- **setting** - <nrf> { BITS | DEC | 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 Volts per Full Scale (V/FS) digital 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?, :DGEN:AMPL

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, ≤ 65536)

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:**settings**

Example:
```
:DGEN:SET?
```

Response:
```
:DGEN:REF:FREQ 997;DBR 0.3873FFS;VFS
1;:DGEN:OUTPUT OFF;INVERT A,OFF;INVERT
B,OFF;AMPL A,0.999756FFS;AMPL
B,0.999756FFS;BURTIMEON 1CYCLES;BURLEVEL
0.25X_Y;BURINTERVAL 3CYCLES;RATIO
0.25X_Y;DITHERTYPE TRI;ARBSIZE 8192;FRQ1
997.001HZ;FRQ2 999.999HZ;VPHASE 0;OFFSET
0FFS;WFM SINE,SINE;SAMPLES 1;SMPTE:HIFREQ
3000HZ;IMFREQ 60
```

:DGEN:SMPTe Compound Command Header

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 SMP1 or SMP4 command.

:DGEN:SMPTe:HIFReq

Sets the high frequency (upper frequency) of the two tone digitally generated SMP1 and SMP4 intermodulation test waveforms.

The command argument is:

- **frequency** - <nrf>

(range: ≥ 2000, ≤ 53999.9 49.999% 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

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 SMP1 or SMP4 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 SMP1 and SMP4 intermodulation test waveforms.

The command argument is:

- **frequency** - <nrf> (range: ≥ 40, ≤ 500)

Default: 60

Related Commands: :DGEN:WFM, DGEN:SMPTe:HIFReq

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 SMP1 or SMP4 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: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 SINE,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 SINE,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:SMPTe:HIFReq command can set the SMPTE high frequency, but this setting will not be used until the waveform type is set to IMD,SMP1 or IMD,SMP4.

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 ARBitrary, NOISe, and SQUare waveform types, a second argument is not required. However, a second parameter of NONE is optional. This mechanism is 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)	SMPTE/DIN 1:1 (SMP1), SMPTE/DIN 4:1 (SMP4)
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)	No argument required (but will accept NONE)
Special (SPCial)	Constant Value (CONStant), Jitter Test Signal (JTESt), Monotonicity (MONOtonic), PassThru (PASSthru), Polarity (POLarity), Psuedo-Random Bit Test (RANDom), Walking Zeros (WLK0), Walking Ones (WLK1)
Square (SQUare)	no argument required (but will accept NONE)

The command argument is:

- **type** - { SINE | ARBitrary | IMD | NOISe | SPCial | SQUare }
- **subtype** - { SINE | BURSt | CONStant | DCSine | DUAL | JTESt | MONotonic | NONE | PASSthru | POLarity | RANDom | SHAPed | SMP1 | SMP4 | STEReo | VPHase | 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 | NOISe | SPCial | SQUare }
- **subtype** - { SINE | BURSt | CONStant | DCSine | DUAL | JTESt | MONotonic | NONE | PASSthru | POLarity | RANDom | SHAPed | SMP1 | SMP4 | STEReo | VPHase | WLK0 | WLK1 }

Related Commands: :DGEN:WFM

Command Syntax: :DGEN:WFM?

Response Syntax: :DGEN:WFM **type[, subtype]**

Example:
```
:DGEN:WFM?
```

Response 1:
```
:DGEN:WFM SINE,DUAL
```

Response 2:
```
:DGEN:WFM SQUARE
```

Chapter 8
Realtime Analyzer Commands

These commands set parameters for the realtime DSP analyzer programs. The header-path for the DSP programs is :DSP:.

The realtime DSP programs in ATS-2 are the Audio Analyzer and the Harmonic Distortion Analyzer.

- Audio Analyzer (DANLr)—Realtime audio signal analyzer.
- HARMONIC Distortion Analyzer—Realtime THD distortion measurements with user selected harmonics of the fundamental signal frequencies.

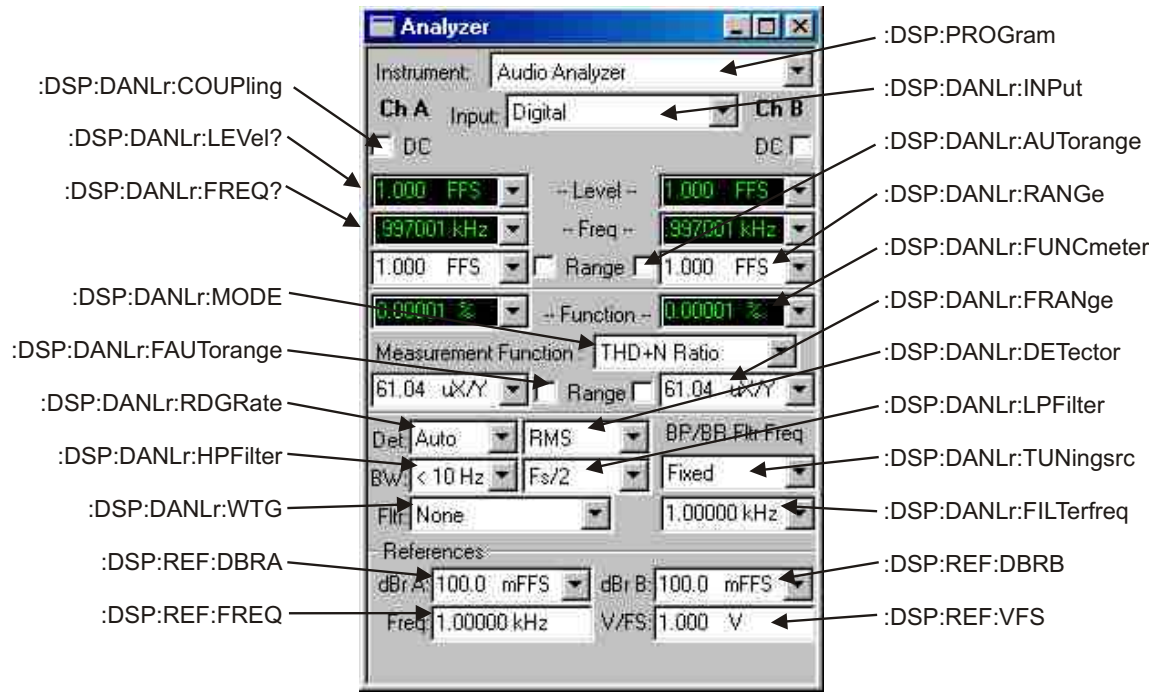


Figure 8-1. ATS software Audio Analyzer control panel DANLR GPIB commands.

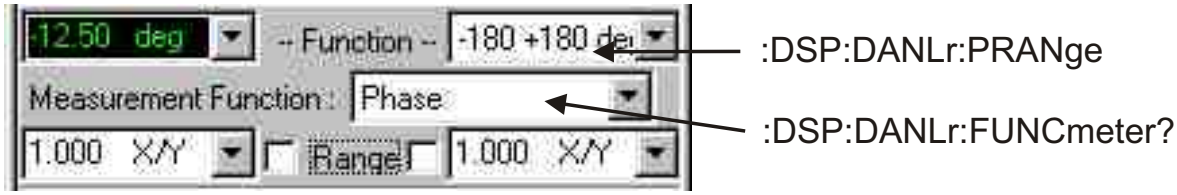


Figure 8-2. DANLR Phase commands

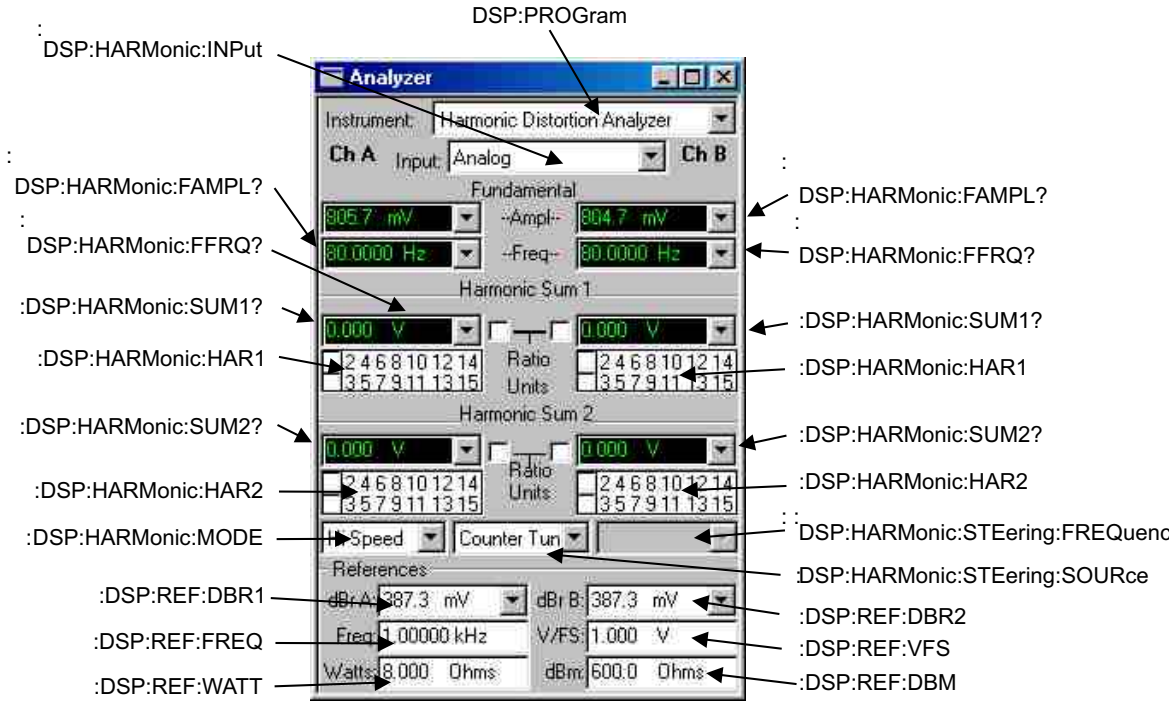


Figure 8-3. ATS software HARMONIC Analyzer control panel GPIB commands.

:DSP:DANLr Compound Command Header

Compound header for Audio Analyzer program commands.

:DSP:DANLr:AUTorange

This command controls autoranging of the indicated channel(s) inputs. 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 input channel. The valid channel settings are A and B. The valid autorange responses are OFF and ON.

Command argument:

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

Response argument(s):

- **channel** - { 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:COUPling

This command sets the coupling for the indicated channel(s) when the Input is set to Digital (:DSP:DANL:INPut DIGital). The valid settings for coupling are AC and DC.

An error will be generated if the input is set to analog (:DSP:DANLr:INPut ANLG) when this command is received.

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 autoranging for a function meter. 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:

- **channel** - { A | AB | B }
- **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 autoranging for a function meter. The possible states are ON and OFF.

The command argument is:

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

Response argument(s):

- **channel** - { A | B }
- **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 measurement 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 10 Hz to 47% of the sample rate, for example, 10 Hz to 30801.92 kHz at a 65536 Hz sample rate.

The command argument is:

- **frequency** - <nrf> (range: ≥ 10 Hz, ≤ 47% 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.

ATS-2 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. Range values will be rounded up to the next higher setting (with a maximum of 1 X/Y or 0.0 dB).

The ranges are:

X/Y	dB	PCT	PPM
1.0	0.000	100	1000000
0.5	-6.021	50	500000
0.25	-12.041	25	250000
0.125	-18.062	12.5	125000
0.0625	-24.082	6.25	62500
0.03125	-30.103	3.125	31250
0.01563	-36.124	1.563	15830
0.007813	-42.144	0.7813	7813
0.003906	-48.165	0.3906	3906
0.001953	-54.185	0.1953	1953
0.0009766	-60.206	0.09766	976.6
0.0004883	-66.227	0.04883	488.3
0.0002441	-72.247	0.02441	244.1
0.0001221	-78.268	0.01221	122.1
0.00006104	-84.288	0.006104	61.04
0.000030517	-90.309	0.003517	30.517

The command argument is:

- **channel** - { A | B }
- **range** - <nrf> (range: *see* table above) {DB | PCT | PPM | X_Y }

Default: 1 FFS

Related Commands: :DSP:DANLr:FRANge?, :DSP:DANLr:FAUTorange

Command Syntax: :DSP:DANLr:FRANge **channel, range**

Example: `:DSP:DANLR:FRANGE A,-20DB`

:DSP:DANLr:FRANge?

This query returns the current function meter gain range setting.

The command argument is:

- **channel** - { A | B }
- **units** - { DB | PCT | PPM | X_Y }

Response argument(s):

- **channel** - { A | B }
- **range** - <nrf> { DB | PCT | PPM | X_Y }

Related Commands: :DSP:DANLr:FRANge

Command Syntax: :DSP:DANLr:FRANge? **channel, units**

Response Syntax: :DSP:DANLr:FRANge **channel, range**

Example: `:DSP:DANLR:FRANGE? A,DB`

Response: `:DSP:DANLR:FRANGE A,-18.0618DB`

:DSP:DANLr:FREQ?

This query specifies the 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 Digital Analyzer Frequency 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 response parameter will be a 0. If settling is enabled then the last response parameter 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: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 one of the two function meters in the specified units. If the specified units are not appropriate for the function meter mode and type of input, then an error will be generated. The appropriate units for each function meter mode and type of input are listed in the table below.

Mode	INPUT	Units
Amplitude, Bandpass, THD+N Amplitude	ANLG	V, DBV, DBU, DBRA, DBRB, DBM, DBGA, DBGB, W, FFS, PCTFs (% FS), DBFS
	DIGital	BITS, FFS, PCTFs (%FS), DBFS, V, DBU, DBV, DBR1, DBR2
2-Channel Ratio, Crosstalk, THD+N Ratio, SMPTE/DIN	ANLG and DIGital	DB, PCT (%), PPM, X_Y
Phase	ANLG and DIGital	DEG (Degrees)

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

The 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 response 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:

- **channel** - { A | B }
- **units** - { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | DEG | FFS | PCT | PCTFs | PPM | V | W | X_Y }

Response argument(s):

- **rdg** - <nrf> { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | DEG | FFS | PCT | PCTFs | PPM | V | W | X_Y }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DSP:DANLr:MODE, :DSP:DANLr:INPut, :SETTling:DANLr, :DSP:REF:DBM, :DSP:REF:DBR1, :DSP:REF:DBR2, :DSP:REF:DBRA, :DSP:REF:DBRB, :DSP:REF:VFS, :DSP:REF:WATT

Command Syntax: :DSP:DANLr:FUNCmeter? **channel, units**

Response Syntax: :DSP:DANLr:FUNCmeter **rdg, settle_timeout**

Example:
```
:DSP:DANLR:FUNCMETER? A,FFS
```

Response:
```
:DSP:DANLR:FUNCMETER 0.168785FFS,0
```

:DSP:DANLr:HPFilter

This command specifies the frequency of the high pass filter in the function meters. This filter does not affect channel A and B level meters or the frequency meters. There are five filter settings: 10 Hz (F10), 22 Hz (F22), 100 Hz (F100), 400 Hz (F400), and User Downloadable (USER).

The command argument is:

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

Default: F10

Related Commands: :DSP:DANLr:HPFilter?, :DSP:DANLr:USRLoad

Command Syntax: :DSP:DANLr:HPFilter **frequency**

Example: `:DSP:DANLR:HPFILTER F10`

:DSP:DANLr:HPFilter?

This query returns the setting of the high pass filter.

Response argument(s):

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

Related Commands: :DSP:DANLr:HPFilter

Command Syntax: :DSP:DANLr:HPFilter?, :DSP:DANLr:USRLoad

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 both channels. The input sources are Digital (DIG) or Analog (ANLG).

The input source determines which set of units are valid for the function meters and the level meters.

The command arguments are:

- **source** - { ANLG | DIGital }

Default: DIGital

Related Commands: :DSP:DANLr:INPut?

Command Syntax: :DSP:DANLr:INPut **source**

Example: `:DSP:DANLR:INPUT ANLG`

:DSP:DANLr:INPut?

This query returns the input source setting for both channels. The possible responses are Digital (DIG) or Analog (ANLG).

Response argument(s):

- **source** - { ANLG | DIGital }

Related Commands: :DSP:DANLr:INPut

Command Syntax: :DSP:DANLr:INPut?

Response Syntax: :DSP:DANLr:INPut **source**

Example: `:DSP:DANLR:INPUT?`

Response: `:DSP:DANLR:INPUT ANLG`

: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 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 ANLG or :DSP:DANLR:INPUT DIGital). This dependency is shown in the following table.

INPUT	Valid Units
ANLG	DBM, DBGA, DBGB, DBRA, DBRB, DBU, DBV, V, W, FFS, PCTFs, DBFS
DIGital	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V

The command argument is:

- **channel** - { A | B }
- **units** - { BITS | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCTFs | V | W }

Response argument(s):

- **level** - <nrf> { BITS | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCTFs | V | W }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :SETTling:DANLr, :DSP:REF:DBM, :DSP:REF:DBR1, :DSP:REF:DBR2, :DSP:REF:DBRA, :DSP:REF:DBRB, :DSP:REF:VFS, :DSP:REF:WATT, :DSP:DANLr:SETRefauto

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 low pass filter for both function meters. This filter does not affect channel A and B level meters or the frequency meters.

There are four filter settings: full bandwidth (FS_2), 15 kHz (F15K), 20 kHz (F20K), and User Downloadable (USER).

An error will be generated if USER is selected but no lowpass filter has been downloaded with the :DSP:DANLr:USRLoad command.

The command argument is:

- **frequency** - { FS_2 | F15K | F20K | USER }

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 low pass filter setting for the function meters.

Response argument(s):

- **frequency** - { FS_2 | F15K | F20K | USER }

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 both of the function meters. The valid modes are Amplitude (AMPLitude), Bandpass (BP), Phase (PHASe), 2-Channel Ratio (RATio), THD+N Amplitude (THDampl), THD+N Ratio (THDRatio), SMPTE/DIN (SMPTe), and Crosstalk (XTALk).

The command argument is:

- **mode** - { AMPLitude | BP | PHASe | RATio | THDampl | THDRatio | SMPTe | XTALk }

Default: AMPLitude

Related Commands: :DSP:DANLr:MODE?, :DSP:DANLr:FUNC?

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 meters.

Response argument(s):

- **mode** - { AMPLitude | BP | PHASe | RATio | THDampl | THDRatio | SMPTe | 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:PRANge

This command sets the range of the phase meter. The ranges are: Auto (AUTO), -180/+180 (R180), -90/+270 (R270), and 0/+360 (R360).

This command will be accepted when the Function Meter Mode is not PHASE. The range setting will be used when Function Meter Mode is changed to PHASE.

The command argument is:

- **range** - { AUTO | R180 | R270 | R360 }

Default: AUTO

Related Commands: :DSP:DANLr:PRANge?, :DSP:DANLr:MODE

Command Syntax: :DSP:DANLr:PRANge **range**

Example: `:DSP:DANLR:PRANGE R180`

:DSP:DANLr:PRANge?

This query returns the phase meter range setting (regardless of the current Function Meter Mode).

Response argument(s):

- **range** - { AUTO | R180 | R270 | R360 }

Related Commands: :DSP:DANLr:PRANge

Command Syntax: :DSP:DANLr:PRANge?

Response Syntax: :DSP:DANLr:PRANge range

Example: `:DSP:DANLR:PRANGE?`

Response: `:DSP:DANLR:PRANGE R180`

: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 one of four possible units. When this command is received autoranging will be turned OFF (if it was ON) and the specified range will be set.

ATS-2 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	-84.228
30.516 µ	0.00305	-90.309

The command arguments are:

- **channel** - { A | B | AB }
- **setting** - <nrf> (range: ≥ 1.0E-6 FFS, ≤ 1.0 FFS) { BITS | DBFS | FFS | PCTFs }

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.

The command arguments are:

- **channel** - { A | B }
- **units** - { BITS | DBFS | FFS | PCTFs }

Response argument(s):

- **response_channel** - { A | B }
- **setting** - <nrf> { BITS | DBFS | FFS | PCTFS }

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, 128/sec, and 256/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 | R256 | 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.

Response argument(s):

- **rate** - { R8 | R4 | R16 | R32 | R64 | R128 | R256 | 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 Audio Analyzer filter.frequency is set to the specified frequency if the function meter is in a tuning mode (BP (bandpass), THDampl, THDRatio or XTALk).

If the function meter is in a tuning mode (i.e., BP (bandpass), THDampl, THDRatio or XTALk), the 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 DANLR Audio Analyzer settings. The response consists of a sequence of the 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.

If :DSP:DANLr:MODE is set to Amplitude (AMPLitude), Phase (PHASe), SMPTE IMD (SMPTe), or Ratio (RATio) then the response will NOT include the message units for TUNINGSRC and FILTERFREQ, in order that the response is legal to send back to the instrument without causing a settings conflict error. This is illustrated in Response 1 below.

If :DSP:DANLr:MODE is set to crosstalk (XTALk), THD+N ratio (THDRatio), THD+N amplitude (THDampl), or bandpass (BP) then the response will include the message units for TUNINGSRC and FILTERFREQ. This is illustrated in Response 2 below.

If :DSP:DANLr:AUTORANGE is set to ON for either channel A or channel B, then the response will include the AUTORANGE message unit for the corresponding channel and will not include the RANGE message unit for that channel. This is illustrated in Response 1 below.

If :DSP:DANLr:AUTORANGE is set to OFF for either channel A or channel B, then the response will include the AUTORANGE message unit for the corresponding channel and will also include the RANGE message unit for that channel. This is illustrated in Response 2 below.

If :DSP:DANLr:FAUTORANGE is set to ON for either channel A or channel B, then the response will include the FAUTORANGE message unit for the corresponding channel but will not include the FRANGE message unit for that channel. This is illustrated in Response 1.

If :DSP:DANLr:FAUTORANGE is set to OFF for either channel A or channel B, then the response will include the FAUTORANGE message unit for the corresponding channel but will also include the FRANGE message unit for that channel. This is illustrated in Response 2 below.

Response argument(s):

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

Related Commands:

Command Syntax: :DSP:DANLr:SET?

Response Syntax: :DSP:DANLr:**settings**

Example:
```
:DSP:DANLR:SET?
```

Response 1:
```
:DSP:DANLR:AUTORANGE A,ON;AUTORANGE
B,ON;COUPLING A,AC;COUPLING B,AC;DETECTOR
FRMS;HPFILTER F10;INPUT DIGITAL;LPFILTER
FS_2;MODE AMPLITUDE;RESPONSE 1000;WTG
UNWT;RDGRATE R8;FAUTORANGE A,ON;FAUTORANGE
B,ON;PRANGE AUTO
```

Response 2:
```
:DSP:DANLR:AUTORANGE A,OFF;AUTORANGE
B,OFF;RANGE A,0.00305FFS;RANGE
B,0.00305FFS;COUPLING A,AC;COUPLING
B,AC;DETECTOR FRMS;HPFILTER F10;INPUT
DIGITAL;LPFILTER FS_2;MODE THDRATIO;RESPONSE
1000;WTG UNWT;RDGRATE R8;TUNINGSRC
FIXED;FILTERFREQ 1000HZ;FAUTORANGE
A,OFF;FAUTORANGE B,OFF;FRANGE
A,0.0004883X_Y;FRANGE B,0.0004883X_Y;PRANGE
AUTO
```

: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:USRLoad

This command loads user defined filter data into the DSP Analyzer program for highpass, lowpass, and weighting filters. One filter may be selected to receive the filter definition data: the highpass filter (HP), the lowpass filter (LP), or the Weighting filter (WTG).

Each filter (HP, LP, or WTG) may be loaded separately with this command. A filter may be changed by loading new data. The previous filter data will be erased when new data is loaded.

The data must be obtained from a filter definition file created with filter development software provided by Audio Precision, part of the optional ATS software.

The data must be formatted as definite length arbitrary block data. An error will be generated and the filter will not be loaded if the data is greater than 8192 bytes.

The command argument is:

- **filter** - { HP | LP | WTG }
- **data** - <definite length arb block data>

Default: <none>

Related Commands: :DSP:DANLR:HPFilter, :DSP:DANLR:LPFilter, :DSP:DANLR:WTG

Command Syntax: :DSP:DANLr:USRLoad **filter, data**

Example: `:DSP:DANLR:USRLOAD WTG, #43328bb...`

:DSP:DANLr:WTG

This command specifies weighting filters to be used with the function meters. These filters are typically used in noise and THD+N measurements.

The available weighting filters, for AMPLitude and 2-Ch Ratio modes are UNWT (unweighted), AWTG (A-weighting), CCIR (CCIR Weighting), FWTG (F weighting), CCITt (CCITT Weighting), CMSG (C-message Weighting), and User Downloadable (USER).

The available weighting filters for THD+N AMPL and THD+N RATIO modes are UNWT (unweighted), AWTG (A-weighting), CCIR (CCIR Weighting), FWTG (F weighting), CCITt (CCITT Weighting), CMSG (C-message Weighting), HI2 (HI-2 Harmonic Weighting), and User Downloadable (USER).

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).

An error will be generated if a weighting filter is specified when the Function Meter mode is SMPTE/DIN or Phase.

An error will be generated if the USER filter is selected but no user filter has been downloaded. See the following table.

Ampl, 2-Ch Ratio	THD+N Ampl, THD+N Ratio	Crosstalk	Bandpass
Unweighted (UNWTD)	Unweighted (UNWTD)	Narrow, Freq x1 (FX1)	Narrow, Freq x1 (FX1)
A-weighting (AWTG)	A-weighting (AWTG)		Narrow, Freq x2 (FX2)
CCIR (CCIR)	CCIR (CCIR)		Narrow, Freq x3 (FX3)
F-weighting (FWTG)	F-weighting (FWTG)		Narrow, Freq x4 (FX4)
CCITT weighting (CCITt)	CCITT weighting (CCITt)		Narrow, Freq x5 (FX5)
C-message weighting (CMSG)	C-message weighting (CMSG)		
User	User		
	HI-2		

The command argument is:

- **weighting** - { UNWT | AWTG | CCIR | CCITt | CMSG | FWTG | FX1 | FX2 | FX3 | FX4 | FX5 | HI2 | USER }

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 | HI2 | USER }

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:HARMonic Compound Command Header

Compound command header for Harmonic Distortion Analyzer commands.

:DSP:HARMonic:FAMPlitude?

This query returns a fundamental amplitude measurement for the selected channel. The desired channel 1 (A) or channel 2 (B) and the units must be specified for the measurement.

The valid units depend on the type of input set by the :DSP:HARMonic:INP command. Digital input units may be Fraction Full Scale (FFS), Percent Full Scale (PCTFs), dB Full Scale (DBFS), Bits (BITS), Volts (V), dBu (DBU), dBV (DBV), dB Ref 1 (DBR1), and dB Ref 2 (DBR2).

The valid units for Analog input are: Fraction Full Scale (FFS), Percent Full Scale (PCTFs), dB Full Scale (DBFS), Volts (V), dBu (DBU), dBV (DBV), dB Ref A (DBRA), and dB Ref B (DBRB), dB Generator A (DBGA), dB Generator B (DBGB), dBm (DBM), and watts (W).

Cross-Domain Units

Cross-domain units are a means of expressing digital domain measurements in terms of the analog domain signal that would cause that digital domain level when converted by an A/D converter with an arbitrary full-scale sensitivity value. The cross-domain units available when Harmonic Analyzer is set for Digital input are Volts, dBu, and dBV. These cross-domain units are derived by scaling the actual digital domain amplitude measurement into the analog domain by use of the V/FS Reference value specified by :DSP:REF:VFS. For example, V/FS should be set to 10.00 Volts when measurements are being made of the digital output of an A/D converter whose full-scale signal value is 10.0 Volts. A digital domain value of 100 mFFS (-20 dBFS) would then be displayed as 1.00 Volt rms, +2.22 dBu, or 0.0 dBV.

Cross-domain units are also a means of expressing analog domain measurements in terms of the digital domain signal that would cause that analog domain level when converted by a D/A converter with an arbitrary full-scale sensitivity value.

The cross-domain units available when Harmonic Analyzer is set for an analog input with :DSP:HARMonic:INPut are FFS, %FS, and dBFS. These cross-domain units are derived by scaling the actual analog domain amplitude measurement into the digital domain by use of the V/FS Reference value specified by :DSP:REF:VFS.

For example, V/FS should be set to 10.00 Volts when measurements are being made of the analog output of a D/A converter whose full-scale signal value is 10.0 Volts. An analog

domain value of 1 V would then be displayed as 0.1 FFS, 10 %FS, or -20 dBFS.

Note: Cross-Domain UNITS Conversions use the V/FS reference :DSP:REF:VFS for BOTH Digital and Analog inputs specified with :DSP:HARMonic:INP. Thus, if input is ANLG then only FFS, PCTFS, and DBFS are converted from volts using the V/FS reference. If input is DIG then only V, dBV, and dBU are converted from FFS using the V/FS reference.

Settling

The distortion fundamental amplitude meter settling is enabled/disabled by the algorithm parameter of the :SETTLING:HARMonic:FAMPlitude command. The power-on and *RST default is settling enabled.

The last parameter in the response indicates the settling status. If settling is enabled, a zero response (0) indicates that the measurement settled and did not time out. The first parameter in the response will contain the last (newest) reading in the settling buffer that met the settling criteria.

A one (1) in the last parameter indicates a settling timeout. 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 a non-settled measurement was requested, then a zero response (0) will always be provided for the last parameter because the settling function was disabled. The first parameter in the response will contain the next reading (after receipt of the query).

The command argument is:

- **channel** - <nr1> (range: 1 or 2)
- **unit** - { BITS | DBFS | DBRA | DBRB | DBGA | DBGB | DBM | DBR1 | DBR2 | DBU | DBV | FFS | PCTFs | V | W }

Response Argument(s):

- **amplitude** - <nrf> { BITS | DBFS | DBRA | DBRB | DBGA | DBGB | DBM | DBR1 | DBR2 | DBU | DBV | FFS | PCTFs | V | W }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DSP:REF:DBM, :DSP:REF:DBR1, :DSP:REF:DBR2, :DSP:REF:DBRA, :DSP:REF:DBRB, :DSP:REF:VFS, :DSP:REF:WATT, :SETTling:HARMonic

Command Syntax: :DSP:HARMonic:FAMPlitude? **channel, unit**

Response Syntax: :DSP:HARMonic:FAMPlitude **amplitude, settle_timeout**

Example: `:DSP:HARMONIC:FAMPLITUDE? 1,DBV`

Response: `:DSP:HARMONIC:FAMPLITUDE -15.341DBV,0`

:DSP:HARMonic:FFRQ?

This query returns the fundamental frequency measurement for the selected channel. The desired channel 1 (A) or channel 2 (B) and the units must be specified for the measurement.

Units may be Cents, Decades, Delta Hz, Delta Hertz Percent, Parts Per Million, Frequency Ratio, Hertz, Octaves, and Percent Hz. All units except Hz are referenced to the DSP Frequency Reference value (:DSP:REF:FREQ).

The distortion fundamental frequency meter settling is enabled/disabled by the algorithm parameter of the :SETTLING:HARMonic:FFReq command. The power-on and *RST default is settling enabled.

The last parameter in the response indicates the settling status. If settling is enabled, a zero response (0) indicates that the measurement settled and did not time out. The first parameter in the response will contain the last (newest) reading in the settling buffer that met the settling criteria.

A one (1) in the last parameter indicates a settling timeout. 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 a non-settled measurement was requested, then a zero response (0) will always be provided for the last parameter because the settling function was disabled. The first parameter in the response will contain the next reading (after receipt of the query).

The command argument is:

- **channel** - <nr1> (range: 1 or 2)
- **units** - { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz }

Response Argument(s):

- **freq** <nrf> - { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DSP:REF:FREQ, :SETTling:HARMonic

Command Syntax: :DSP:HARMonic:FFRQ? **channel, units**

Response Syntax: :DSP:HARMonic:FFRQ **freq, settle_timeout**

Example: `:DSP:HARMONIC:FFRQ? 1,HZ`

Response: `:DSP:HARMONIC:FFRQ 1250HZ,0`

:DSP:HARMonic:HAR1

This command uses two arguments to specify the Sum 1 harmonics for each channel. The first argument specifies the channel, and the second argument specifies the harmonics to be enabled. The channel argument selects either channel 1 or channel 2. The harmonics argument is a decimal formatted positive integer. The value is a bit-map that specifies which of the 14 harmonics (from harmonic 2 to harmonic 15) will be enabled for Sum 1 measurements on the specified channel. The argument range is 0 to 16383.

All harmonics can be enabled with the value 16383 (all bits set). All harmonics can be disabled with the value 0. All Even harmonics can be enabled with the decimal value 5461 (1+4+16+64+256+1024+4096), equivalent to the binary value 1010101010101. All Odd harmonics can be enabled with the decimal value 10922 (2+8+32+128+512+2048+8192) equivalent to the binary value 10101010101010.

The bits in the harmonics argument are interpreted as shown in the table below:

Harmonic	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Bit Number	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Value	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

The command arguments are:

- **channel** - <nr1> (range: 1 or 2)
- **harmonics** - <nr1> (range: ≥ 0, ≤ 16383)

Default: 1,0

Related Commands: :DSP:HARMonic:HAR1?, :DSP:HARMonic:SUM1?

Command Syntax: :DSP:HARMonic:HAR1 **channel, harmonics**

Example: `:DSP:HARMonic:HAR1 1,10922`

:DSP:HARMonic:HAR1?

This query returns the setting of the Sum 1 harmonics for each channel.

The command argument is:

- channel - <nr1> (range: 1 or 2)

The query response arguments are:

- channel - <nr1> (range: 1 or 2)
- harmonics - <nr1> (range: ≥ 0, ≤ 16383)

Related Commands: :DSP:HARMonic:HAR1

Command Syntax: :DSP:HARMonic:HAR1? channel

Response Syntax: :DSP:HARMonic:HAR1 channel, harmonics

Example: `:DSP:HARMonic:HAR1? 1`

Response: `:DSP:HARMonic:HAR1 1,10922`

:DSP:HARMonic:HAR2

This command uses two arguments to specify the Sum 2 harmonics for each channel. The first argument specifies the channel, and the second argument specifies the harmonics to be enabled. The channel argument selects either channel 1 or channel 2. The harmonics argument is a decimal formatted positive integer. The value is a bit-map that specifies which of the 14 harmonics (from harmonic 2 to harmonic 15) will be enabled for Sum 2 measurements on the specified channel. The argument range is 0 to 16383.

All harmonics can be enabled with the value 16383 (all bits set). All harmonics can be disabled with the value 0. All Even harmonics can be enabled with the decimal value 5461 (1+4+16+64+256+1024+4096), equivalent to the binary value 1010101010101. All Odd harmonics can be enabled with the decimal value 10922 (2+8+32+128+512+2048+8192) equivalent to the binary value 10101010101010.

The bits in the harmonics argument are interpreted as shown in the table below:

Harmonic	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Bit Number	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Value	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

The command arguments are:

- **channel** - <nr1> (range: 1 or 2)
- **harmonics** - <nr1> (range: ≥ 0, ≤ 16383)

Default: 1,0

Related Commands: :DSP:HARMonic:HAR2?,

Command Syntax: :DSP:HARMonic:HAR2 **channel, harmonics**

Example: `:DSP:HARMonic:HAR2 1,10922`

:DSP:HARMonic:HAR2?

This query returns the setting of the Sum 2 harmonics for each channel.

The command argument is:

- **channel** - <nr1> (range: 1 or 2)

The query response arguments are:

- **channel** - <nr1> (range: 1 or 2)
- **harmonics** - <nr1> (range: ≥ 0, ≤ 16383)

Related Commands: :DSP:HARMonic:HAR2; :DSP:HARMonic:

Command Syntax: :DSP:HARMonic:HAR2? **channel**

Response Syntax: :DSP:HARMonic:HAR2 **channel, harmonics**

Example: `:DSP:HARMonic:HAR2? 1`

Response: `:DSP:HARMonic:HAR2 1,10922`

:DSP:HARMonic:INPut

This command sets the input source. The input sources are DIGital (Digital at input sample rate) and ANLG (Analog Input).

The command argument is:

- **source** - { ANLG | DIGital }

Default: ANLG

Related Commands: :DSP:HARMonic:INPut?

Command Syntax: :DSP:HARMonic:INPut **source**

Example: `:DSP:HARMonic:INPUT ANLG`

:DSP:HARMonic:INPut?

This command returns the input source setting, either DIGital (Digital at input sample rate) or ANLG (Analog Input).

The command argument is:

- **source** - { ANLG | DIGital }

Related Commands: :DSP:HARMonic:INPut

Command Syntax: :DSP:HARMonic:INPut?

Response Syntax: :DSP:HARMonic:INPut **source**

Example: `:DSP:HARMonic:INPUT?`

Response: `:DSP:HARMonic:INPUT ANLG`

:DSP:HARMonic:MODE

This command sets the measurement mode between high measurement speed and high measurement accuracy. The High Speed selection (HISPeed) is recommended when measuring distortion products no more than 90 to 100 dB down (higher than 0.003% to 0.001%). To obtain the most accurate measurements of distortion products lower than this (typically occurring only in the digital domain), the High Accuracy selection (HIACcuracy) is recommended but measurement speed will be lower.

The command argument is:

- **mode** - { HIACcuracy | HISPeed }

Default: HISPeed

Related Commands: :DSP:HARMonic:MODE?

Command Syntax: :DSP:HARMonic:MODE **mode**

Example: `:DSP:HARMONIC:MODE HIACCURACY`

:DSP:HARMonic:MODE?

This query returns the measurement mode setting, either High Speed (HISPeed) or High Accuracy (HIACcuracy).

Response Argument(s):

- **mode** - { HIACcuracy | HISPeed }

Related Commands: :DSP:HARMonic:MODE

Command Syntax: :DSP:HARMonic:MODE?

Response Syntax: :DSP:HARMonic:MODE **mode**

Example: `:DSP:HARMonic:MODE?`

Response: `:DSP:HARMONIC:MODE HIACCURACY`

:DSP:HARMonic:SET?

This query returns all DSP Harmonic Distortion settings. It is equivalent to sending the commands: :DSP:HARMONIC:HAR1? 1;HAR1? 2;HAR2? 1;HAR2? 2;INPUT?;MODE?;:DSP:HARMONIC:STEERING:FREQUENCY? HZ;SOURCE?

Related Commands: :DSP:HARMONIC:HAR1?, :DSP:HARMONIC:HAR2?, :DSP:HARMONIC:INPUT?, :DSP:HARMONIC:MODE?, :DSP:HARMONIC:STEERING:FREQ?, :DSP:HARMONIC:STEERING:SOURCE?

Command Syntax: :DSP:HARMonic:SET?

Response Syntax: <see individual query response syntax>

Example: `:DSP:HARMONIC:SET?`

Response: `:DSP:HARMONIC:HAR1 1,10922;HAR1 2,10922;HAR2 1,10922;HAR2 2,10922;INPUT ANLG;MODE HIACCURACY;:DSP:HARMONIC:STEERING:FREQUENCY 1000HZ;SOURCE CNTR`

:DSP:HARMonic:STEering:FREQuency

This command sets the fixed steering frequency (fundamental frequency) for the distortion meter. This value is used only when the steering source is set to FIXed.

The frequency value may be specified in frequency units of Cents, Decades, Delta Hz, Delta Hertz Percent, Parts Per Million, Frequency Ratio, Hertz, Octaves, and Percent Hz.

All units except Hz are referenced to the DSP Frequency Reference value (:DSP:REF:FREQ).

The minimum command range is 0.0 Hz. The maximum command range is limited to 0.45883 * sample rate (e.g. exactly 30070 Hz when the sample rate is set to 65536 Hz with input AD1). Note that the sample rate will be dependent on the selection of the input and A/D converter. If input is DIG then the rate is dependent on the digital input sample rate and freq reference settings.

The command argument is:

- **freq** - <nrf> (range: 0 to 0.45883 SR) { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz }

Default: 1000HZ

Related Commands: :DSP:HARMonic:STEering:FREQuency?, :DSP:REF:FREQ, :DSP:HARMonic:INPUT, :DSP:HARMonic:STEering:SOURCE, :ANLG:CONVerter

Command Syntax: :DSP:HARMonic:STEering:FREQuency **freq**

Example:
```
:DSP:HARMONIC:STEERING:FREQUENCY 5000HZ
```

:DSP:HARMonic:STEering:FREQuency?

This query returns the Steering Frequency setting used when the steering source is set to FIXed.

The command argument is:

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

Response Argument(s):

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

Related Commands: :DSP:HARMonic:STEering:FREQuency

Command Syntax: :DSP:HARMonic:STEering:FREQuency? unit

Response Syntax: :DSP:HARMonic:STEering:FREQuency freq

Example:
```
:DSP:HARMONIC:STEERING:FREQUENCY? HZ
```

Response:
```
:DSP:HARMONIC:STEERING:FREQUENCY 5000HZ
```

:DSP:HARMonic:STEering:SOURce

This command selects the steering source to determine how the fundamental frequency will be determined for distortion measurements. The possible choices are AGEN, CNTR, DGEN, and FIXed.

AGEN selects the frequency that has been set for the analog generator. This frequency will be the setting of :AGEN:DASine:FRQ1 or :AGEN:DAIMd:SMPTe:HIFReq,

depending on which type of waveform is currently selected for the analog generator.

DGEN selects the frequency that has been set for the digital generator. This frequency will be the setting of :DGEN:FRQ1, :DGEN:SMPTe:HIFReq, or :DGEN:CCIF:CFReq, depending on which type of waveform is currently selected for the digital generator. This mode would normally be used when sweeping digital input devices with stimulus coming from ATS-2 digital generator.

Counter (CNTR) selects the current frequency reading of the Harmonic Distortion frequency meter. This function would be selected when making harmonic distortion measurements from an external signal such as playback of a Compact Disc or audio tape or reception of a signal from a distant source.

FIXed selects the frequency set by the Steering Frequency command.

The command argument is:

- **source** - { AGEN | CNTR | DGEN | FIXed }

Default: CNTR

Related Commands: :DSP:HARMonic:STEering:SOURce?, :DSP:HARMonic:STEering:FREQuency, :AGEN:DASine:FRQ1, :AGEN:DAIMd:SMPTe:HIFReq, :DGEN:FRQ1, :DGEN:SMPTe:HIFReq, :DGEN:CCIF:CFReq

Command Syntax: :DSP:HARMonic:STEering:SOURce **source**

Example:
```
:DSP:HARMONIC:STEERING:SOURCE AGEN
```

:DSP:HARMonic:STEering:SOURce?

This query returns the source setting for the distortion steering frequency.

Response Argument(s):

- **source** - { AGEN | CNTR | DGEN | FIXed }

Related Commands: :DSP:HARMonic:STEering:SOURce

Command Syntax: :DSP:HARMonic:STEering:SOURce?

Response Syntax: :DSP:HARMonic:STEering:SOURce **source**

Example:
```
:DSP:HARMONIC:STEERING:SOURCE?
```

Response:
```
:DSP:HARMONIC:STEERING:SOURCE AGEN
```

:DSP:HARMonic:SUM1?

This query returns the Sum 1 distortion measurement for the selected channel. The desired channel 1 or channel 2 and the units must be specified for the measurement.

The valid units depend on the type of input set by the :DSP:HARMonic:INP command. Digital input units may be Fraction Full Scale (FFS), Percent Full Scale (PCTFs), dB Full Scale (DBFS), Bits (BITS), Volts (V), dBu (DBU), dBV (DBV), dB Ref 1 (DBR1), dB Ref 2 (DBR2), Percent (PCT), X/Y Ratio (X_Y), dB (DB), and Parts-per-Million (PPM).

The valid units for Analog input are: Fraction Full Scale (FFS), Percent Full Scale (PCTFs), dB Full Scale (DBFS), Volts (V), dBu (DBU), dBV (DBV), dB Ref A (DBRA), and dB Ref B (DBRB), dB Generator A (DBGA), dB Generator B (DBGB), dBm (DBM), watts (W), Percent (PCT), X/Y Ratio (X_Y), dB (DB), and Parts-per-Million (PPM).

Note that PCT, X_Y, DB, and PPM are Ratio Units normally selected with a checkbox in ATS software.

Cross-Domain Units

Cross-domain units are a means of expressing digital domain measurements in terms of the analog domain signal that would cause that digital domain level when converted by an A/D converter with an arbitrary full-scale sensitivity value. The cross-domain units available when DSP Harmonic Analyzer is set for Digital input are Volts, dBu, and dBV. These cross-domain units are derived by scaling the actual digital domain amplitude measurement into the analog domain by use of the V/FS Reference value specified by :DSP:REF:VFS. For example, V/FS should be set to 10.00 Volts when measurements are being made of the digital output of an A/D converter whose full-scale signal value is 10.0 Volts. A digital domain value of 100 mFFS (-20 dBFS) would then be displayed as 1.00 Volt rms, +2.22 dBu, or 0.0 dBV.

Cross-domain units are also a means of expressing analog domain measurements in terms of the digital domain signal that would cause that analog domain level when converted by a D/A converter with an arbitrary full-scale sensitivity value. The cross-domain units available when DSP Harmonic Analyzer is set for an analog input with :DSP:HARMonic:INPut are FFS, %FS, and dBFS. These cross-domain units are derived by scaling the actual analog domain amplitude measurement into the digital domain by use of the V/FS Reference value specified by :DSP:REF:VFS.

For example, V/FS should be set to 10.00 Volts when measurements are being made of the analog output of an D/A converter whose full-scale signal value is 10.0 Volts. An analog domain value of 1 V would then be displayed as 0.1 FFS, 10 %FS, or -20 dBFS.

[NOTE: Cross-Domain UNITS Conversions use the V/FS reference :DSP:REF:VFS for BOTH Digital and Analog inputs specified with :DSP:HARMonic:INP. Thus, if input is ANLG then only FFS, PCTFS, and DBFS are converted from volts using the

V/FS reference. If input is DIG then only V, dBV, and dBU are converted from FFS using the V/FS reference.]

Settling

The Sum 1 distortion measurement meter settling is enabled/disabled by the algorithm parameter of the :SETTLING:HARMonic command. The power-on and *RST default is settling enabled.

The last parameter in the response indicates the settling status. If settling is enabled, a zero response (0) indicates that the measurement settled and did not time out. The first parameter in the response will contain the last (newest) reading in the settling buffer that met the settling criteria.

A one (1) in the last parameter indicates a settling timeout. 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 a non-settled measurement was requested, then a zero response (0) will always be provided for the last parameter because the settling function was disabled. The first parameter in the response will contain the next reading (after receipt of the query).

The command argument is:

- **channel** - <nr1> (range: 1 or 2)
- **unit** - { BITS | DB | DBFS | DBRA | DBRB | DBGA | DBGB | DBM | DBR1 | DBR2 | DBU | DBV | FFS | PCT | PCTFs | PPM | V | W | X_Y }

Response Argument(s):

- **amplitude** - <nrf> { BITS | DB | DBFS | DBRA | DBRB | DBGA | DBGB | DBM | DBR1 | DBR2 | DBU | DBV | FFS | PCT | PCTFs | PPM | V | W | X_Y }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DSP:REF:DBM, :DSP:REF:DBR1, :DSP:REF:DBR2, :DSP:REF:DBRA, :DSP:REF:DBRB, :DSP:REF:VFS, :DSP:REF:WATT, :SETTling:HARMonic

Command Syntax: :DSP:HARMonic:SUM1? **channel, unit**

Response Syntax: :DSP:HARMonic:SUM1 **amplitude, settle_timeout**

Example: `:DSP:HARMONIC:SUM1? 1,DB`

Response: `:DSP:HARMONIC:SUM1 -90.005DB,1`

:DSP:HARMonic:SUM2?

This query returns the Sum 2 distortion measurement for the selected channel. The desired channel 1 or channel 2 and the units must be specified for the measurement.

The valid units depend on the type of input set by the :DSP:HARMonic:INP command. Digital input units may be Fraction Full Scale (FFS), Percent Full Scale (PCTFs), dB Full Scale (DBFS), Bits (BITS), Volts (V), dBu (DBU), dBV (DBV), dB Ref 1 (DBR1), dB Ref 2 (DBR2), Percent (PCT), X/Y Ratio (X_Y), dB (DB), and Parts-per-Million (PPM).

The valid units for Analog input are: Fraction Full Scale (FFS), Percent Full Scale (PCTFs), dB Full Scale (DBFS), Volts (V), dBu (DBU), dBV (DBV), dB Ref A (DBRA), and dB Ref B (DBRB), dB Generator A (DBGA), dB Generator B (DBGB), dBm (DBM), watts (W), Percent (PCT), X/Y Ratio (X_Y), dB (DB), and Parts-per-Million (PPM).

Note that PCT, X_Y, DB, and PPM are Ratio Units normally selected with a checkbox in ATS software.

Cross-Domain Units

Cross-domain units are a means of expressing digital domain measurements in terms of the analog domain signal that would cause that digital domain level when converted by an A/D converter with an arbitrary full-scale sensitivity value. The cross-domain units available when DSP Harmonic Analyzer is set for Digital input are Volts, dBu, and dBV. These cross-domain units are derived by scaling the actual digital domain amplitude measurement into the analog domain by use of the V/FS Reference value specified by :DSP:REF:VFS. For example, V/FS should be set to 10.00 Volts when measurements are being made of the digital output of an A/D converter whose full-scale signal value is 10.0 Volts. A digital domain value of 100 mFFS (-20 dBFS) would then be displayed as 1.00 Volt rms, +2.22 dBu, or 0.0 dBV.

Cross-domain units are also a means of expressing analog domain measurements in terms of the digital domain signal that would cause that analog domain level when converted by a D/A converter with an arbitrary full-scale sensitivity value. The cross-domain units available when DSP Harmonic Analyzer is set for an analog input with :DSP:HARMonic:INPut are FFS, %FS, and dBFS. These cross-domain units are derived by scaling the actual analog domain amplitude measurement into the digital domain by use of the V/FS Reference value specified by :DSP:REF:VFS.

For example, V/FS should be set to 10.00 Volts when measurements are being made of the analog output of an D/A converter whose full-scale signal value is 10.0 Volts. An analog domain value of 1 V would then be displayed as 0.1 FFS, 10 %FS, or -20 dBFS.

[NOTE: Cross-Domain UNITS Conversions use the V/FS reference :DSP:REF:VFS for BOTH Digital and Analog inputs specified with :DSP:HARMonic:INP. Thus, if input is ANLG then only FFS, PCTFS, and DBFS are converted from volts using the

V/FS reference. If input is DIG then only V, dBV, and dBU are converted from FFS using the V/FS reference.]

Settling

The Sum 2 distortion measurement meter settling is enabled/disabled by the algorithm parameter of the :SETTLING:HARMonic command. The power-on and *RST default is settling enabled.

The last parameter in the response indicates the settling status. If settling is enabled, a zero response (0) indicates that the measurement settled and did not time out. The first parameter in the response will contain the last (newest) reading in the settling buffer that met the settling criteria.

A one (1) in the last parameter indicates a settling timeout. 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 a non-settled measurement was requested, then a zero response (0) will always be provided for the last parameter because the settling function was disabled. The first parameter in the response will contain the next reading (after receipt of the query).

The command argument is:

- **channel** - <nr1> (range: 1 or 2)
- unit - { BITS | DB | DBFS | DBRA | DBRB | DBGA | DBGB | DBM | DBR1 | DBR2 | DBU | DBV | FFS | PCT | PCTFs | PPM | V | W | X_Y }

Response Argument(s):

- **amplitude** - <nrf> { BITS | DB | DBFS | DBRA | DBRB | DBGA | DBGB | DBM | DBR1 | DBR2 | DBU | DBV | FFS | PCT | PCTFs | PPM | V | W | X_Y }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DSP:REF:DBM, :DSP:REF:DBR1, :DSP:REF:DBR2, :DSP:REF:DBRA, :DSP:REF:DBRB, :DSP:REF:VFS, :DSP:REF:WATT, :SETTling:HARMonic

Command Syntax: :DSP:HARMonic:SUM2? **channel, unit**

Response Syntax: :DSP:HARMonic:SUM2 a**mplitude, settle_timeout**

Example: `:DSP:HARMONIC:SUM2? 1,DBV`

Response: `:DSP:HARMONIC:SUM2 -79.819DB,1`

:DSP:PROGram

This command specifies the DSP program to be loaded into the DSP processing system. DANLr is loaded at power-on.

There are 5 DSP programs available for ATS-2: Audio Analyzer (DANLr), Multitone Audio Analyzer (FASTtest), FFT Spectrum Analyzer (FFT), Harmonic Distortion Analyzer (HARMonic), and Digital Interface Analyzer (INTervu).

The command argument is:

- **program** - { DANLr | FASTtest | FFT | HARMonic | INTervu }

Default: DANLr

Related Commands: :DSP:PROGram

Command Syntax: :DSP:PROGram program

Example: `:DSP:PROGRAM FFT`

:DSP:PROGram?

This query returns the name of the currently loaded DSP program.

Response argument(s):

- **program** - { DANLr | FASTtest | FFT | HARMonic | INTervu }

Related Commands: DSP:PROGram

Command Syntax: :DSP:PROGram?

Response Syntax: :DSP:PROGram **program**

Example: `:DSP:PROGRAM?`

Response: `:DSP:PROGRAM FFT`

:DSP:REF Compound Command Header

These commands set the parameters for the DSP analyzer units (both real time and batch mode) which require reference values.

:DSP:REF:DBM

This command sets the DBM unit impedance reference value.The reference value is in units of ohms.

The command argument is:

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

Default: 600

Related Commands: :DSP:REF:DBM?

Command Syntax: :DSP:REF:DBM **impedance**

Example: `:DSP:REF:DBM 300`

:DSP:REF:DBM?

This query returns the current DBM unit impedance reference setting for the analyzer. The response value is ohms.

Response argument(s):

- **impedance** - <nrf>

Related Commands: :DSP:REF:DBM

Command Syntax: :DSP:REF:DBM?

Response Syntax: :DSP:REF:DBM **impedance**

Example: `:DSP:REF:DBM?`

Response: `:DSP:REF:DBM 300`

:DSP:REF:DBR1

This command sets the DBR1 unit reference value. This reference value is only used for DANLR, FASTTEST, FFT, and HARMonic when the INPut is DIGital.

The minimum range is 0 FFS or PCTFS, or -1E+34 for DBFS or BITS units.

The command argument is:

- **setting** - <nrf> (range: ≥ -1E34, ≤ 1E34) { BITS | DBFS | FFS | PCTFs }

Default: 0.100 FFS

Related Commands: :DSP:REF:DBR1?, :DSP:REF:VFS

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.

The command arguments are:

- **units** - { BITS | DBFS | FFS | PCTFs }

Response argument(s):

- **setting** - <nrf> { BITS | DBFS | FFS | PCTFs }

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. This reference value is only used for DANLR, FASTTEST, FFT, and HARMonic when the INPut is DIGital.

The minimum range is 0 FFS or PCTFS, or -1E+34 for DBFS or BITS units.

The command argument is:

- **setting** - <nrf> (range: ≥ -1E34, ≤ 1E34) { BITS | DBFS | FFS | PCTFs }

Default: 0.100 FFS

Related Commands: :DSP:REF:DBR2?

Command Syntax: :DSP:REF:DBR2 **setting**

Example: `:DSP:REF:DBR2 0.80FFS`

:DSP:REF:DBR2?

This query returns the current DBR2 unit reference setting for the DSP analyzer in the specified units.

The command arguments are:

- **units** - { BITS | DBFS | FFS | PCTFs }

Response argument(s):

- **setting** - <nrf> { BITS | DBFS | FFS | PCTFs }

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:DBRA

This command sets the analyzer DBRA unit reference value. This reference value can also be set using the :DSP:REF:SETRefauto command.

This reference value is only used for DANLR, FFT, FASTTEST, and HARMonic when the INPut is ANAlog.

The minimum range is 0 V or -1E+34 for DBU or DBV units.

The command argument is:

- **setting** - <nrf> (range: > 0, 1E34) V, { V | DBU | DBV }

Default: 0.3873V

Related Commands: :DSP:REF:DBRA?, :DSP:REF:SETRefauto

Command Syntax: :DSP:REF:DBRA **setting**

Example: `:DSP:REF:DBRA 1 DBV`

:DSP:REF:DBRA?

This query returns the current DBRA unit reference setting for the analyzer in the specified units.

The command arguments are:

- **units** - { V | DBU | DBV }

Response argument(s):

- **setting** - <nrf> { V | DBU | DBV }

Related Commands: :DSP:REF:DBRA

Command Syntax: :DSP:REF:DBRA? **units**

Response Syntax: :DSP:REF:DBRA **setting**

Example: `:DSP:REF:DBRA? V`

Response: `:DSP:REF:DBRA 1.0V`

:DSP:REF:DBRB

This command sets the analyzer DBRB unit reference value. This reference value can also be set using the :DSP:REF:SETRefauto command.

This reference value is only used for DANLR, FFT, FASTTEST, and HARMonic when the INPut is ANAlog.

The minimum range is 0 V or -1E+34 for DBU or DBV units.

The command argument is:

- **setting** - <nrf> (table range: > 0, = 1E34) V { V | DBU | DBV }

Default: 0.3873V

Related Commands: :DSP:REF:DBRB?, :DSP:REF:SETRefauto

Command Syntax: :DSP:REF:DBRB **setting**

Example: `:DSP:REF:DBRB 1 DBV`

:DSP:REF:DBRB?

This query returns the current DBRB unit reference setting for the analyzer in the specified units.

The command arguments are:

- **units** - { V | DBU | DBV }

Response argument(s):

- **setting** - <nrf> { V | DBU | DBV }

Related Commands: :DSP:REF:DBRB

Command Syntax: :DSP:REF:DBRB? **units**

Response Syntax: :DSP:REF:DBRB **setting**

Example: `:DSP:REF:DBRB? V`

Response: `:DSP:REF:DBRB 1.0V`

:DSP:REF:FREQ

This command sets the FREQ unit reference value for the DSP analyzer. The unit is hertz.

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:SETRefauto

This command sets the reference values for dBrA, dBrB, dBr1, and dBr2 if the DANLR or HARMONIC DSP program is loaded with the :DSP:PROG command. Nothing happens if any other DSP programs are loaded (FFT, FASTTEST, or INTERVU).

If the DSP program's INPut is set to ANLG then the DBRA reference value is set with the reading of the the DANLR channel A Level meter (or HARMonic channel A FAMPlitude meter), and the DBRB reference value is set with the DANLR channel B Level meter (or HARMonic channel B FAMPlitude meter).

If the DSP program's INPut is set to DIGital then the DBR1 reference value is set with the reading of the DANLR channel A Level meter (or HARMonic channel A FAMPlitude meter), and the DBR2 reference value is set with the reading of the DANLR channel B Level meter (or HARMonic channel B FAMPlitude meter).

Related Commands: :DSP:REF:DBRA, :DSP:REF:DBRB, :DSP:REF:DBR1 :DSP:REF:DBR2

Command Syntax: :DSP:REF:SETRefauto

Example: `:DSP:REF:SETREFAUTO`

:DSP:REF:VFS

This command sets the V/FS unit reference value for the DSP analyzer. The implicit 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 implicit 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:REF:WATT

This command 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: :DSP:REF:WATT?

Command Syntax: :DSP:REF:WATT **impedance**

Example: `:DSP:REF:WATT 8`

:DSP:REF:WATT?

This query returns the WATT unit impedance reference value for the analog analyzer. The response unit is ohms.

Response argument(s):

- **impedance** - <nrf>

Related Commands: :DSP:REF:WATT

Command Syntax: :DSP:REF:WATT?

Response Syntax: :DSP:REF:WATT **impedance**

Example: `:DSP:REF:WATT?`

Response: `:DSP:REF:WATT 8.0`

:DSP:SET?

This query returns all instrument DSP related settings. The SRCParams and TIMeout message units are not included in the response unless a batch-mode DSP program is loaded (FASTtest, FFt, or INTervu).

Response 1 illustrates the case when a batch-mode DSP program is not loaded (DANLR).

Response 2 illustrates the case when a batch-mode DSP program is loaded (FASTTEST).

Response argument(s):

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

Related Commands:

Command Syntax: :DSP:SET?

Response Syntax: :DSP:**settings**

Example:
```
:DSP:SET?
```

Response 1:
```
:DSP:REF:DBM 50;DBR1 0.1FFS;DBR2 0.1FFS;DBRA
0.1V;DBRB 0.1V;FREQ 1000;VFS 1;WATT
4;:DSP:PROGRAM DANLR;:DSP:DANLR:AUTORANGE
A,ON;AUTORANGE B,ON;COUPLING A,AC;COUPLING
B,AC;DETECTOR FRMS;HPFILTER F10;INPUT
DIGITAL;LPFILTER FS_2;MODE
AMPLITUDE;RESPONSE 1000;WTG UNWT;RDGRATE
R8;FAUTORANGE A,ON;FAUTORANGE B,ON;PRANGE
AUTO
```

Response 2:
```
:DSP:REF:DBM 600;DBR1 0.1FFS;DBR2
0.1FFS;DBRA 0.3873V;DBRB 0.3873V;FREQ
1000;VFS 1;WATT 8;:DSP:PROGRAM
FASTTEST;TIMEOUT 10.000000;SRCPARAMS
FREQ,HZ,20,20000,30,LOG;:DSP:FASTTEST:FREQRE
S 0;INPUT ANLG;LENGTH AUTO;MODE
SPECTRUM;PMODE INDEPENDENT;PROCESS
SYNC;TRGDELAY 0;TRIGGER DGEN;PKTRIG 1
```

Chapter 9
Batch Mode Analyzer Commands

These commands set parameters for the batch mode digital analyzer programs.

The batch mode DSP programs in ATS-2 are the Multitone Audio Analyzer (FASTTEST), the Spectrum Analyzer (FFT) and the Digital Interface Analyzer (INTERVU). The INTERVU commands are invalid and will cause execution errors if used with an ATS-2 without the Perfomance Upgrade Option, which provides the Digital Interface Analyzer hardware.

- Multitone Audio Analyzer (FASTTEST)—Provides frequency domain analysis of a multitone signal and can provide a time domain display of any waveform. This program requires a multitone waveform to be loaded into the digital generator and selected as an arbitrary waveform. Prerecorded multitone signals may be measured.
- Spectrum Analyzer (FFT)—Provides general-purpose time domain analysis of waveforms or frequency domain analysis of signals, including the received jitter signal on the digital inputs.
- Digital Interface Analyzer (INTERVU)—analyzes the AES3 / IEC60958 digital interface input signal via an 80 MHz sample rate A/D converter. This capability is not included in the base ATS-2 hardware and must be added as part of the Performance Upgrade Option.

Commands with the compound command header :DSP:REF set references for both real time and batch mode analysis. These commands are listed in Chapter 8.

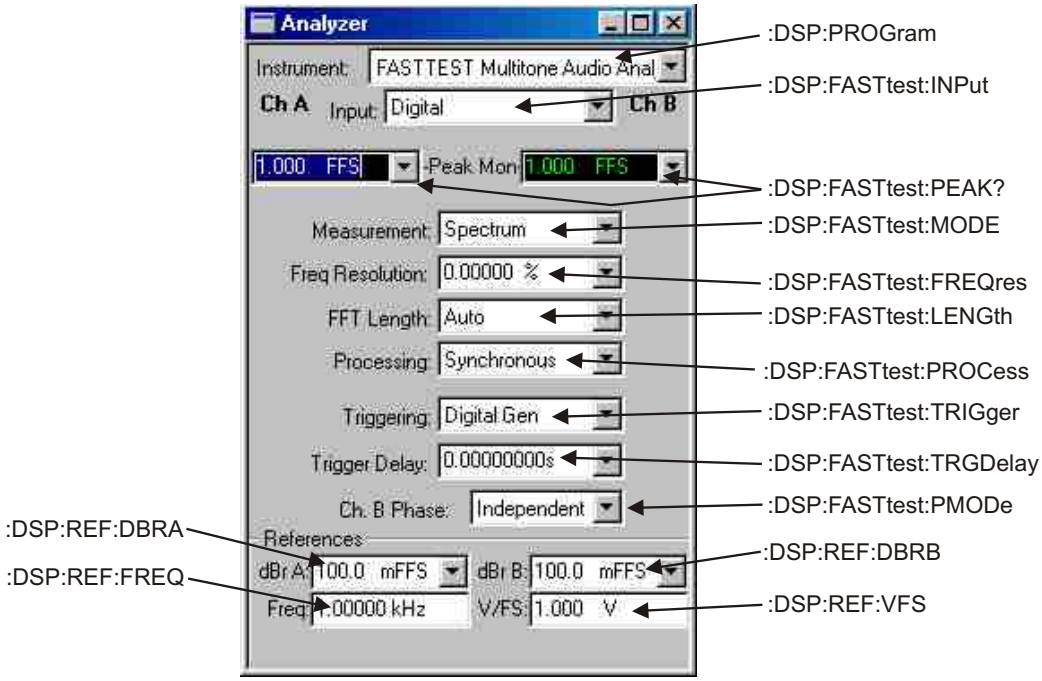


Figure 9-1. ATS software Multitone Analyzer control panel FASTTEST GPIB commands.

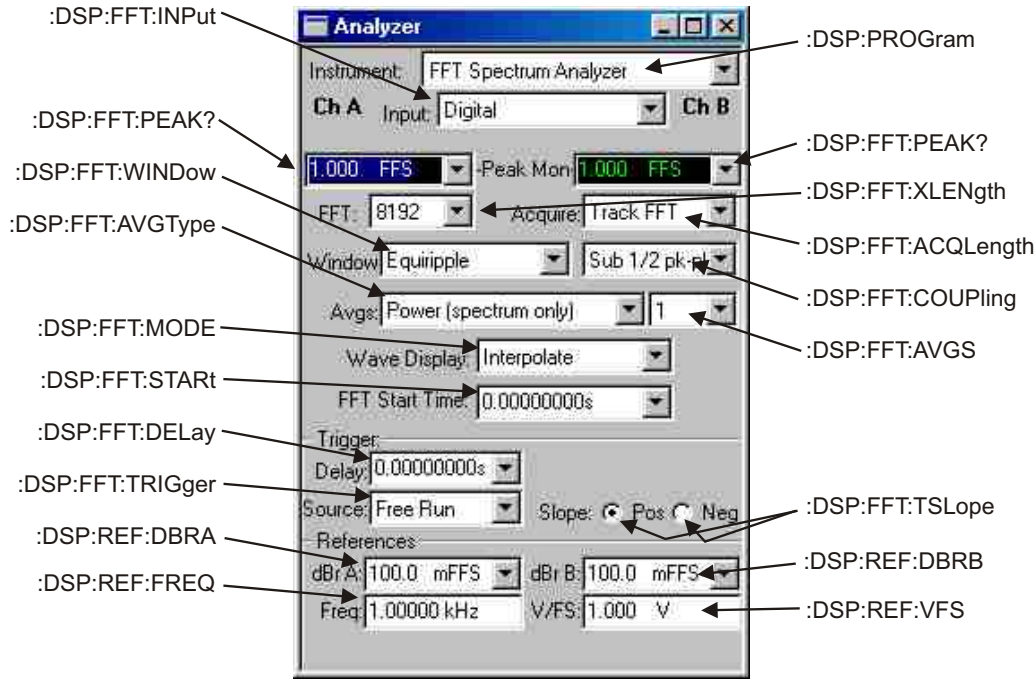


Figure 9-2. ATS software FFT Analyzer control panel FFT GPIB commands.

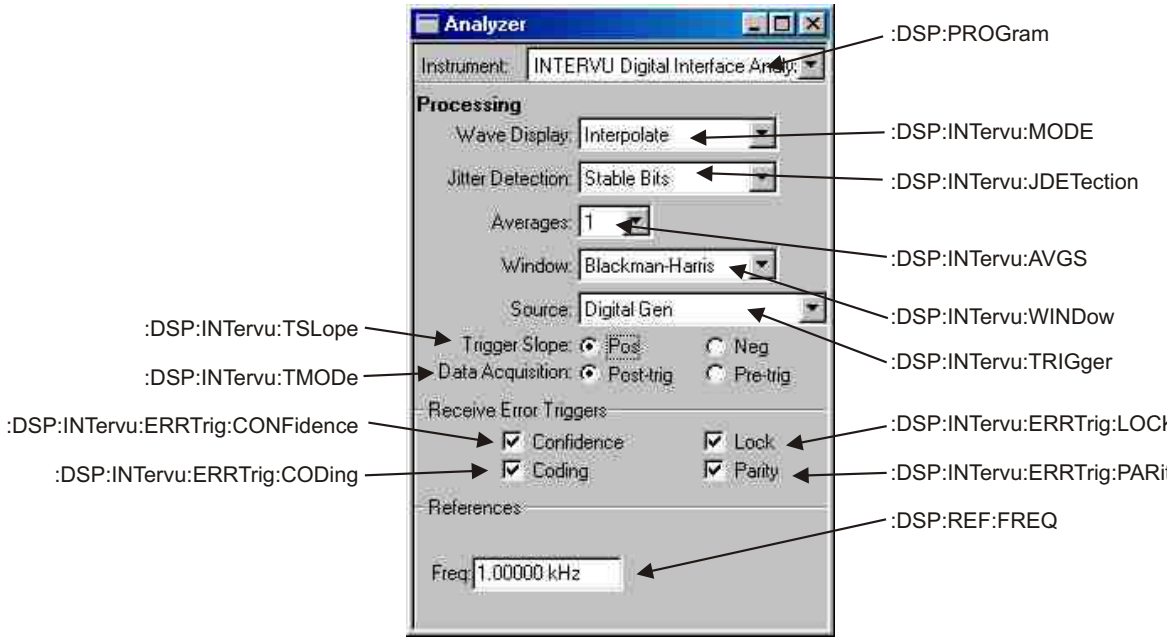


Figure 9-3. ATS software Digital Interface Analyzer control panel INTERVU GPIB commands.

Operating States

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 or Intervu batch mode DSP programs are loaded and is invalid if received when the ANALYZER or Harmonic real-time DSP programs are 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.

The DSP program must be in the SETUP state before most DSP 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:MEAS? or :DSP:BATCh?) 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. 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 DSP commands listed below. Note that these commands are invalid in the SETUP operating state.

- :DSP:ACQX?
- :DSP:BATCH?

- :DSP:MEAS?
- :DSP:REPRocess?
- :DSP:XFRM?

The application programmer must set the appropriate DSP operating state when sending commands to ATS-2. 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? (or XFRM? or REPRocess?) 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 ANLG
: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
REM Load a multitone waveform into the digital generator and select the
    arbitrary waveform, see :DGEN:ARBLoad and :AGEN:WFM
:DSP:PROGram FASTtest
:DSP:FASTtest:INPut ANLG
:DSP:SRCParams FREQ,HZ,17.57,19998.04,30,ARBitrary
:DSP:TABLe
    1,ASCII,31,#017.57,23.43,29.29,41.01,52.73,64.45,82.03,99.6,123.04,158.2,
    199.21,251.95,316.4,398.43,498.04,632.81,802.73,1001.91,1248.04,1599.6,19
    98.04,2501.95,3152.34,4001.95,4998.04,6351.56,7998.04,10001.95,12498.04,1
    6001.95,19998.04
:DSP:TSELect 1
REM Setup Fasttest response and interchannel phase measurements for ch1 &
    ch2.
:DSP:FASTtest:MODE RESPonse;PMODe ICHannel
:DSP:TIMeout 10;OPSTate READing,AMP1,AMP2,PHA2
REM The following acquire will time out unless an arb wfm is loaded
:DSP:ACQX?
REM Read the response
:DSP:BATCh? ASCii,AMP1,V,AMP2,V,PHA2,DEG
REM Read the measurement 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
:DSP:OPSTate SETup
```

Command Interaction

The DSP goes through three distinct phases while acquiring FFT spectral data. 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 9-5).

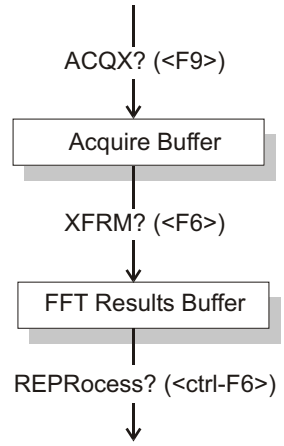


Figure 9-4. DSP flow diagram.

Each of the DSP programs processes the data differently depending on its setup parameters. In most cases, the data can be acquired one time and processed multiple times thereafter without another acquisition. The ACQX? command accomplishes the initial acquisition of data. The XFRM? command can be used to retransform the acquired data, (e.g. if it is desired to change the FFT window function or process a different portion of the acquisition buffer for purposes of time windowing the data). The REPRocess? command is typically used to get measurement data for a different range of frequencies after a new range has been specified with the :DSP:SRCParams command.

Note that ACQX? command always performs the XFRM and REPROCESS steps of the sequence. The XFRM? command always performs the REPROCESS step. The REPRocess? command does not cause a new transform step.

: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 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 (FFT, FASTTEST 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), then a time-out did not occur. If the response is 1 (one) then a time-out occurred and data is not available to be read with the :DSP:BATCh? or :DSP:MEASurement queries.

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.

Response argument(s):

- **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`

:DSP:BATCh?

This query returns an entire batch data array.

If the :DSP:SRCParams “mode” parameter specifying the batch is LINear or LOG, the source data points are calculated when this command is received. If the parameter is ARBitrary, the source data points are provided by the user via the :DSP:TABLe command.

Note that the combination of channel and measurement must have been enabled with the :DSP:OPSTate READing command prior to issuing this command.

The source domain and unit, start, stop, number of points, and increment algorithm are specified with the :DSP:SRCParams command.

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. The difference between the BINary and RBINary formats is in the byte sequence: BINary is big-endian, per IEEE488.2, RBINary is little-endian, standard Intel x86 floating point representation.

The 'meas' parameter must be appropriate for the DSP program currently loaded:

	FASTTEST	FFT	INTERVU
AMP1	x	x	
AMP2	x	x	
PHA1	x	x	
PHA2	x	x	
AMPLitude			x
JITTer			x
LOWer			x
UPPer			x
PROBability			x

:DSP:BATCH? measurements and units valid for each DSP batch mode program are shown in the table below.

DSP Program	Input Type	Measurement Parameter	Valid Units
FASTTEST	ANLG	AMP1 & AMP2	DBFS, DBGA, DBGB, DBRA, DBRB, DBM, DBU, DBV, FFS, PCTFs, V, W
	DIGital	AMP1 & AMP2	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
	ANLG & DIGital	PHA1 & PHA2	DEG
FFT	ANLG	AMP1 & AMP2	DBFS, DBGA, DBGB, DBRA, DBRB, DBM, DBU, DBV, FFS, PCTFs, V, W
	JITTer	AMP1	DBFS, DBGA, DBGB, DBRA, DBRB, DBM, DBU, DBV, FFS, PCTFs, V, W
		AMP2	DBUI, UI
	JITSec	AMP1	DBFS, DBGA, DBGB, DBRA, DBRB, DBM, DBU, DBV, FFS, PCTFs, V, W
		AMP2	SEC
	DIGital	AMP1 & AMP2	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
	ANLG & DIGital	PHA1 & PHA2	DEG
INTERVU	–	AMPLitude	DBV, V
		LOWer or UPPer	DBV, V
		JITTer	DBUI, SEC, UI
		PROBability	DB, PCT, PPM, X_Y

The query arguments are:

- **format** - { ASCii | BINary | RBINary }

- **meas1** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **unit1** - { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBUI | DBV | DEG | FFS | PCT | PCTFs | PPM | SEC | UI | V | W | X_Y }
- **meas2** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **unit2** - { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBUI | DBV | DEG | FFS | PCT | PCTFs | PPM | SEC | UI | V | W | X_Y }
- **meas3** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **unit3** - { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBUI | DBV | DEG | FFS | PCT | PCTFs | PPM | SEC | UI | V | W | X_Y }
- **meas4** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **unit4** - { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBUI | DBV | DEG | FFS | PCT | PCTFs | PPM | SEC | UI | V | W | X_Y }

The response arguments are:

- **format** - { ASCII | BINARY | RBINARY }
- **pointcount** - <nr1>
- **data** - <definite length arb block data>

Related Commands: :DSP:PROGram, :DSP:OPSTate, :DSP:SRCParams, :DSP:TABLe

Command Syntax: :DSP:BATCH? **format, meas1, unit1 [,meas2, unit2, meas3, unit3, meas4, unit4]**

Response Syntax: :DSP:BATCH **format, pointcount, data**

Example 1:
```
:DSP:BATCH? ASCII,AMP1,FFS,PHA1,DEG
```

Response 1:
```
:DSP:BATCH ASCII,512,0.125,0.125,...
```

Example 2:
```
:DSP:BATCH? BINARY,JITTER,SEC
```

Response 2:
```
:DSP:BATCH BINARY,512,#44096<binary data>
```

:DSP:DATA

This command downloads data into either of the DSP acquisition buffers when the loaded DSP program is

FASTTEST or FFT. Use the :DSP:INTervu:DATA command to perform this function with the INTERVU DSP program.

This data must have been originally uploaded from an ATS-2 acquisition buffer or transfer buffer using the :DSP:DATA? query.

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 acquisition or transform buffer contents for the FFT or FASTTEST DSP programs. The FASTTEST DSP program does not provide transform buffer data.

DSP Program	Acquisition Buffers	Transform Buffers
Intervu (INTervu	No	No
FASTTEST (FASTtest)	Yes	No
FFT (FFT)	Yes	Yes

:DSP:Data? will cause a command error if invoked while DANLr, HARMonic, or INTervu DSP programs are 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 relevant 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).

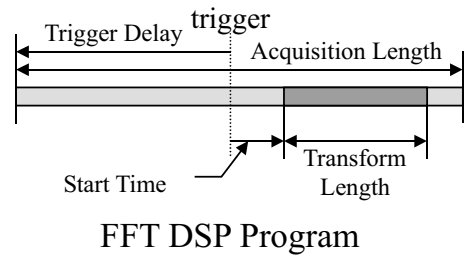


Figure 9-5. FFT Trigger time relationships

In FASTTEST the acquisition length is equal to the transform length.

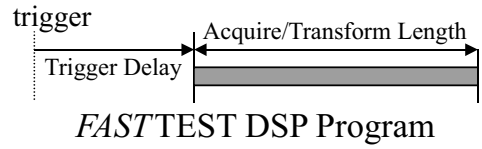


Figure 9-6. FASTtest Trigger time relationships

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

Compound command header for Multitone Audio Analyzer commands.

:DSP:FASTtest:FREQres

This command specifies the percent of frequency error correction when :DSP:FASTtest:PROCess is set to FREQ. It also specifies the Frequency Resolution used in Response and Distortion measurement modes.

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 buffers. The range of error correction is set by the FREQres argument value. Error correction occurs during acquisition of the signal.

In the Response measurement mode, the amplitudes of all FFT bins within plus and minus the Frequency Resolution value of each sweep source value (specified by :DSP:SRCParams) are combined in RSS (root-sum-square) fashion and provided 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 or flutter or other low frequency modulation. Wow and flutter in tape playback mechanisms spreads the energy from a single tone across a narrow spectral band.

In Distortion mode, 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. The Distortion mode defines all signals other than the fundamental tones as distortion and noise. Entering a non-zero value of Frequency Resolution causes flutter sidebands to be excluded from 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 :DSP:ACQX? and then use :DSP:XFRM? to re-transform 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 sets the input source to Digital (DIGital) or Analog (ANLG). When the input is ANLG the analog input signal is routed through the A/D converters selected with the :ANLG:CONVerter command.

The command argument is:

- **input** - { DIGital | ANLG }

Default: ANLG

Related Commands: :DSP:FASTtest:SRC1, :DSP:FASTtest:SRC2, :DSP:FASTtest:INPut?

Command Syntax: :DSP:FASTtest:INPut **input**

Example:
```
:DSP:FASTTEST:INPUT DIGITAL
```

:DSP:FASTtest:INPut?

This query returns the setting of the INPut command.

Response argument(s):

- **input** - { DIGital | ANLG }

Related Commands: :DSP:FASTtest:INPut

Command Syntax: :DSP:FASTtest:INPut?

Response Syntax: :DSP:FASTtest:INPut **input**

Example:
```
:DSP:FASTTEST:INPUT?
```

Response:
```
:DSP:FASTTEST:INPUT DIGITAL
```

: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 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 mode of FASTTEST.

The valid lengths are Auto (AUTO), 32768 (L32K), 16384 (L16K), 8192 (L8K), 4096 (L4K), 2048 (L2K), 1024 (L1K), and 512 (L512) samples.

The command argument is:

- **length** - { AUTO | L512 | L1K | L2K | L4K | L8K | L16K | L32K }

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 | L512 | L1K | L2K | L4K | L8K | L16K | L32K }

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 ATS-2 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.

Note that channel 1 is the A input, channel 2 is the B input.

The command arguments are:

- **channel** - <nr1> (range: 1 or 2)
- **units** - { BITS | DBFS | FFS | PCTFs }

Response argument(s):

- **peak** - <nrf>

Related Commands: :DSP:FASTtest:PKTRig

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: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 DSP is in :DSP: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 with FASTTEST.

Synchronous mode (SYNC) should be used when utilizing synchronous multitone waveforms that have no frequency shift. Synchronous multitone waveforms may be created with the ATS software Multitone Creation utility. 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 windowed 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:**settings**

Example:
```
:DSP:FASTTEST:SET?
```

Response:
```
:DSP:FASTTEST:FREQRES 0;INPUT DIGITAL;LENGTH
AUTO;MODE SPECTRUM;PMODE INDEPENDENT;PROCESS
SYNC;TRGDELAY 0;TRIGGER DGEN;PKTRIG 1
```

: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)

Values up to 291.271 seconds (e.g. OSR of 28.8 Ks/S) may be entered for Trigger Delay.

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, analog 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 ATS-2. 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 (:DSP:TIMeout). 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. It is the signal connected to the rear panel TRIG IN BNC connector. The trigger circuitry is edge-sensitive, and an acquisition will trigger on a positive-going edge on the signal at this connector.

Digital generator mode is used when the signal source driving the device under test is an ATS-2 arbitrary waveform generated by the Digital Generator arbitrary waveform.

Analog generator mode (AGEN) is used when the signal source derived from an arbitrary waveform generated by the analog generator. FASTTEST will trigger at the first sample point in the analog generator arbitrary waveform buffer.

Off mode (free run) provides an immediate acquisition when ACQX? is received.

The command argument is:

- **source** - { AGEN | DGEN | EXTernal | LOOSe | NORMal | OFF | TIGHt }

Default: OFF

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** - { AGEN | 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

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).

See the :DSP:DATA? query for a more detailed explanation of the relationship between start time, acquisition length, trigger delay, and transform length.

The argument to this command specifies that the acquisition buffer length will either be the same size as the FFT transform length (XLENgth) (see :DSP:FFT:XLENgth command) or is a

fixed length of 800, 1.5k, 2.5k, 5k, 10k, 19k, 24k, 36k, 72k, 144k, or 256k.

The exact acquisition lengths in points are:

ACQLength setting	Acquisition Points
L800	811
L1500	1361
L2500	2467
L5000	4813
L10000	9491
L19000	18539
L24K	24571
L36K	36857
L72K	71999
L144K	143999
L256K	262139

The command argument is:

- **length** - { XLENgth | L800 | L1500 | L2500 | L5000 | L10000 | L19000 | L24K | L36K | L72K | L144K | L256K }

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 | L800 | L1500 | L2500 | L5000 | L10000 | L19000 | L24K | L36K | L72K | L144K | L256K }

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: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. The table describes the setting number and the windows that are valid with that setting.

Setting number and valid windows		
Type	**Meaning**	**Valid Windows**
1	Power (spectrum only)	<all>
2	Sync, re-align	<all>
3	Sync	<all>
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: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.

The range in seconds is +- 8388607 divided by the input sample rate (e.g. measured sample rate at digital input if INPut is DIGital or sample rate of A/D converter if INPut is ANLG). The number 8388607 is 2 ^ 23 - 1.

DELay range example:

Sample Rate = 48000

DELay = +-8388607/48000

= -174.76265 sec to 174.76265 sec

The minimum range may be set to the negative value described above, however the DSP processor will compute the actual minimum range for negative values when an acquisition is attempted based on other parameters that may have been set after this DELay command was received (e.g. sample rate).

The DSP computes the minimum negative value according to the formula: acquisition length (samples)- FFT Size (samples) divided by the input sample rate.

Minimum range example:

Sample Rate = 48000

ACQLength = 18539 (L19000)

XLENgth FFT Size = 8192

Minimum DELay = -(18539 - 8192) / 48000

= -0.2155625 sec

The command argument (in seconds) is:

- **time** - <nrf> (range: -1E+34, 1E+34, see text above)

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 -0.05`

:DSP:FFT:INPut

This command sets the input source to Digital (DIGital), Analog (ANLG), Ch A Analog/Ch B Jitter Seconds (JITSec), and Ch A Analog/Ch B Jitter UI (JITTer).

The command argument is:

- **input** - { ANLG | DIGital | JITSec | JITTer }

Default: ANLG

Related Commands: :DSP:FFT:INPut?

Command Syntax: :DSP:FFT:INPut **input**

Example: `:DSP:FFT:INPUT ANLG`

:DSP:FFT:INPut?

This query returns the setting of the :DSP:FFT:INPUT command.

Response argument(s):

- **input** - { ANLG | DIGital | JITSec | JITTer }

Related Commands: :DSP:FFT:INPut

Command Syntax: :DSP:FFT:INPut?

Response Syntax: :DSP:FFT:INPut **input**

Example: `:DSP:FFT:INPUT?`

Response: `:DSP:FFT:INPUT ANLG`

: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** - { BITS | DBFS | FFS | PCTFs }

Response argument(s):

- **peak** - <nrf> { BITS | DBFS | FFS | PCTFs }

Related Commands: :DSP:FFT:PKTRig

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`

:DSP: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?`

Response: `:DSP:FFT:PKTRIG 1`

:DSP:FFT:SENS

This command specifies the trigger sensitivity when :DSP:FFT:TRIGger is set to C1F or C2F. An error will be generated if :DSP:FFT:TRIGGER is set to any other value when this command is received.

The command argument is:

- **sens** - <nrf> (range: ≥ 0, ≤ 1 FFS) { 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 SENSitivity and TLEVel settings are not included in the response unless the TRIGger setting is appropriate (see Response 1).

The SENS response message will be included if the response if TRIGger is set to C1Fixed or C2Fixed (see Response 2).

The TLEVel response message will be included in the response if TRIGger is set to C1Level or C2Level (see Response 3).

Response argument(s):

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

Related Commands:

Command Syntax: :DSP:FFT:SET?

Response Syntax: :DSP:FFT:**settings**

Example: `:DSP:FFT:SET?`

Response1:
```
:DSP:FFT:ACQLENGTH XLENGTH;AVGS 1;AVGTYPE
1;COUPLING DC;DELAY 0;MODE INTRPOLATE;INPUT
ANLG;START 0;TRIGGER FREE;TSLOPE POS;WINDOW
EQR;XLENGTH 8192;PKTRIG 1
```

Response2:
```
:DSP:FFT:ACQLENGTH XLENGTH;AVGS 1;AVGTYPE
1;COUPLING DC;DELAY 0;MODE INTRPOLATE;INPUT
ANLG;START 0;TRIGGER C1FIXED;TSLOPE
POS;WINDOW EQR;XLENGTH 8192;SENS
0.0001FFS;PKTRIG 1
```

Response3:
```
:DSP:FFT:ACQLENGTH XLENGTH;AVGS 1;AVGTYPE
1;COUPLING DC;DELAY 0;MODE INTRPOLATE;INPUT
ANLG;START 0;TRIGGER C1LEVEL;TSLOPE
POS;WINDOW EQR;XLENGTH 8192;TLEV
0.001FFS;PKTRIG 1
```

:DSP:FFT:STARt

This command specifies the start point (in 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) and less than the value computed according to the formula below:

Delay (sec) - ((Acquisition length - FFT Size) / input sample rate)

where Delay is the :DSP:FFT:DELay setting in seconds,

Acquisition length is :DSP:FFT:ACQLength in samples, FFT Size is :DSP:FFT:XLENgth in samples.

Maximum range example:

Sample Rate = 48000

ACQLength = 18539 (L19000)

XLENength FFT Size = 8192

Delay = 10 seconds

STARt = 10 + ((18539 - 8192) / 48000)

= 10.215563 seconds

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:TLEVel

This command specifies the triggering level when the FFT Trigger Source :DSP:FFT:TRIGger command is set to C1Level or C2Level. An error will be generated if :DSP:FFT:TRIGger is set to any other value.

The command argument is:

- **level** - <nrf> (range: ≥ -1, ≤ 1 FFS) { FFS | DBFS | PCTFs }

Default: 0.001FFS

Related Commands: :DSP:FFT:TRIGger

Command Syntax: :DSP:FFT:TLEVel **level**

Example: `:DSP:FFT:TLEVel 0.1FFS`

:DSP:FFT:TLEVel?

This query returns the triggering level that will be used when the FFT Trigger Source :DSP:FFT:TRIGger command is set to C1Level or C2Level.

The command argument is:

- **unit** - { FFS | DBFS | PCTFs }

Response argument(s):

- **level** - <nrf> (table range: ≥-1, ≤ 1) { FFS | DBFS | PCTFS }

Default: 1E-3FFS

Related Commands: :DSP:FFT:TRIGger

Command Syntax: :DSP:FFT:TLEVel? **unit**

Response Syntax: :DSP:FFT:TLEVEL **level**

Example: `:DSP:FFT:TLEVel? FFS`

Response: `:DSP:FFT:TLEVEL 0.1FFS`

: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? query is received or may wait for a trigger event, depending upon the setting of the Trigger.

The selections are FREE (Free Running), C1Auto (Channel #1 Auto), C2Auto (Channel #2 Auto), C1Fixed (Channel #1 Fixed), C2Fixed (Channel #2 Fixed), C1Level (Channel #1 Level), C2Level (Channel #2 Level), DGEN (Digital Generator), ACMains (AC Mains), AGEN (Analog Generator), EXTernal (External), and JGEN (Jitter Generator).

FREE (Free Running) causes acquisition immediately after the :DSP:ACQX? query is received, regardless of signal amplitude. This is the typical operating mode with steady-state test signals.

C1Fixed (Channel #1 Fixed) and C2Fixed (Channel #2 Fixed) use a fixed threshold set by the :DSP:FFT:SENS command, referred to as the triggering threshold. An FFT acquisition will trigger on the first zero crossing of the selected slope (positive or negative) that occurs after the signal amplitude is sufficient to swing both through zero and the Sens value. If the signal contains a sufficient DC offset such that it does not swing through zero, no triggering will take place. In this case, use one of the quasi-AC coupling modes (Subtract Avg or Sub ½ pk-pk) that will cause the processed signal to pass through zero and permit triggering to function.

C1Level (Channel #1 Level) and C2Level (Channel #2 Level) use a variable trigger level set by the :DSP:FFT:TLEVel command, and will trigger on the first signal excursion of the selected slope (Positive or Negative) through that level. This operates identically to conventional oscilloscope triggering. An acquisition will be triggered the first time the signal with the specified slope (positive or negative) passes through this level.

C1Auto (Channel #1 Auto) and C2Auto (Channel #2 Auto) will cause triggering at one-half the peak-to-peak value if the selected channel has a signal amplitude greater than digital infinity zero.

DGEN (Digital Generator) triggers on the digital generator waveform. If the Digital Generator is outputting any waveform except an Arbitrary Waveform, a Digital Generator trigger occurs at each cycle of the waveform. If the Digital Generator is outputting an Arbitrary Waveform (that is, a signal from a waveform file), a Digital Generator trigger occurs at the first sample of the waveform.

ACMains (AC Mains) provides a trigger at each cycle of the AC waveform of the mains line powering ATS-2.

AGEN (Analog Generator) triggers on the analog generator waveform. If the Analog Generator is outputting any waveform

except an Arbitrary Waveform, an Analog Generator trigger occurs at each cycle of the waveform. If the Analog Generator is outputting an Arbitrary Waveform (that is, a signal from a waveform file), an Analog Generator trigger occurs at the first sample of the waveform.

EXTernal (External) refers to the TRIG IN BNC connector on the rear of the ATS-2 chassis. The trigger circuitry is edge-sensitive, and an FFT acquisition will trigger on a positive-going or negative-going edge on the signal at this connector, depending upon the setting of the trigger Slope.

JGEN (Jitter Generator) provides a trigger at each cycle of the waveform selected by the digital output jitter generator (:DOUT:JWFM). This provides stable acquisition of the jitter waveform generated by ATS-2.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 | C1Level | C2Auto | C2Fixed | C2Level | 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 | C1Level | C2Auto | C2Fixed | C2Level | 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:TSLope

This command specifies the acquisition triggering Slope (positive or negative). Selecting Positive causes triggering to

occur on a positive transition of the trigger signal. Negative causes triggering on a negative transition of the trigger 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: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 for 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 (BH), Gaussian (GAUSsian), Hamming (HAMMing), Hann (HANN), Flat-top (FLAT), Equiripple (EQR), None (NONE), None Move to BIN Center (NMTBc), Rife-Vincent 4 (RV4), and Rife-Vincent 5 (RV5).

The command argument is:

- **window** - { BH | EQR | FLAT | GAUSsian | HANN | HAMMing | NMTBc | NONE | RV4 | RV5 }

Default: EQR

Related Commands: :DSP:FFT:WINDow?

Command Syntax: :DSP:FFT:WINDow **window**

Example: `:DSP:FFT:WINDOW GAUSSIAN`

:DSP:FFT:WINDow?

This query returns the setting of the :DSP:FFT:WINDOW command.

Response argument(s):

- **window** - { BH | EQR | FLAT | GAUSsian | HANN | HAMMing | NMTBc | NONE | RV4 | RV5 }

Related Commands: :DSP:FFT:WINDow

Command Syntax: :DSP:FFT:WINDow?

Response Syntax: :DSP:FFT:WINDow **window**

Example: `:DSP:FFT:WINDOW?`

Response: `:DSP:FFT:WIND GAUSSIAN`

: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 where the transform region starts in the DSP acquisition buffer (relative to the trigger point).

The argument to this command will be rounded up to the nearest legitimate setting: 256, 512, 1024, 2048, 4096, 8192, 16384, 32768.

The command argument is:

- **length** - <nr1> (range: ≥ 0, ≤ 32768)

Default: 8192

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, ≤ 32768)

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

Compound command header for Digital Interface Analyzer commands (INTervu).

: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:DATA

This command loads previously acquired waveform data into the Intervu waveform acqisition buffer for processing by the Intervu DSP program. The data must be previously acquired by the Intervu DSP program and read from the instrument with :DSP:INTERVU:DATA? query or with the ATS software (the <definite length arb block data> header must be pre-pended to the file data if saved by ATS software).

After loading the data you may reprocess the waveform with the :DSP:REPRocess? or :DSP:XFRM? queries.

The data is erased from the DSP memory if a DSP program is loaded with the :DSP:PROGram command (including Intervu).

The command argument is:

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

Default: <none>

Related Commands: :DSP:PROGram, :DSP:INTervu:DATA?

Command Syntax: :DSP:INTervu:DATA **data**

Example: `:DSP:INTERVU:DATA #71573138bb...`

:DSP:INTervu:DATA?

This query returns data from the Intervu waveform acquisition buffer.

The response is <definite length arbitrary block data>. The waveform data imbedded in the block is formatted in a proprietary binary format. You may save the entire data block (unaltered) to a computer file and use the contents of the file as an argument to the :DSP:INTervu:DATA command in order to download the data into Intervu for reprocessing with the :DSP:REPRocess? or :DSP:XFRM? commands.

A command error will be generated if Intervu has not acquired a digital interface waveform with the :DSP:ACQX? query or if the data has not been loaded into the acquisition buffer with the :DSP:INTervu:DATA command.

Response Argument(s):

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

Related Commands: :DSP:INTervu:DATA, :DSP:ACQX?

Command Syntax: :DSP:INTervu:DATA?

Response Syntax: :DSP:INTervu:DATA **data**

Example: `:DSP:INTERVU:DATA?`

Response: `:DSP:INTERVU:DATA #71573138bb...`

:DSP:INTervu:ERRTrig Compound Command Header

These commands enable Intervu Error Triggering for data coding errors, data confidence errors, data parity errors, or interface lock errors when the Intervu trigger (:DSP:INTervu:TRIGger) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

:DSP:INTervu:ERRTrig:CODing

This command enables Intervu Error Triggering when a data coding error occurs if the Intervu trigger (:DSP:INTervu:TRIGger) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

The command argument is:

- **error** - { OFF | ON }

Default: ON

Related Commands: :DSP:INTervu:ERRTrig:CODing?, :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:ERRTrig:CODing **error**

Example: `:DSP:INTERVU:ERRTRIG:CODING OFF`

:DSP:INTervu:ERRTrig:CODing?

This query returns the Intervu Data Coding Error Triggering setting.

Response Argument(s):

- **error** - { OFF | ON }

Related Commands: :DSP:INTervu:ERRTrig:CODing, :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:ERRTrig:CODing?

Response Syntax: :DSP:INTervu:ERRTrig:CODing **error**

Example: `:DSP:INTERVU:ERRTRIG:CODING?`

Response: `:DSP:INTERVU:ERRTRIG:CODING OFF`

:DSP:INTervu:ERRTrig:CONFidence

This command enables Intervu Error Triggering when a data confidence error occurs if the Intervu trigger :DSP:INTervu:TRIGger) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

The command argument is:

- **error** - { OFF | ON }

Default: ON

Related Commands: :DSP:INTervu:ERRTrig:CONFidence?, :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:ERRTrig:CONFidence **error**

Example: `:DSP:INTERVU:ERRTRIG:CONFIDENCE OFF`

:DSP:INTervu:ERRTrig:CONFidence?

This query returns the Intervu Data Confidence Error Triggering setting.

Response Argument(s):

- **error** - { OFF | ON }

Related Commands: :DSP:INTervu:ERRTrig:CONFidence

Command Syntax: :DSP:INTervu:ERRTrig:CONFidence?

Response Syntax: :DSP:INTervu:ERRTrig:CONFidence **error**

Example: `:DSP:INTERVU:ERRTRIG:CONFIDENCE?`

Response: `:DSP:INTERVU:ERRTRIG:CONFIDENCE OFF`

:DSP:INTervu:ERRTrig:LOCK

This command enables Intervu Error Triggering when an interface lock error occurs if the Intervu trigger (:DSP:INTervu:TRIGger) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

The command argument is:

- **error** - { OFF | ON }

Default: ON

Related Commands: :DSP:INTervu:ERRTrig:LOCK?, :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:ERRTrig:LOCK **error**

Example:
```
:DSP:INTERVU:ERRTRIG:LOCK OFF
```

:DSP:INTervu:ERRTrig:LOCK?

This query returns the Intervu Interface Lock Error Triggering setting.

Response Argument(s):

- **error** - { OFF | ON }

Related Commands: :DSP:INTervu:ERRTrig:LOCK, :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:ERRTrig:LOCK?

Response Syntax: :DSP:INTervu:ERRTrig:LOCK **error**

Example:
```
:DSP:INTERVU:ERRTRIG:LOCK?
```

Response:
```
:DSP:INTERVU:ERRTRIG:LOCK OFF
```

:DSP:INTervu:ERRTrig:PARity

This command enables Intervu Error Triggering when a data parity error occurs if the Intervu trigger :DSP:INTervu:TRIGger) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

The command argument is:

- **error** - { OFF | ON }

Default: ON

Related Commands: :DSP:INTervu:ERRTrig:PARity?, :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:ERRTrig:PARity **error**

Example:
```
:DSP:INTERVU:ERRTRIG:PARITY OFF
```

:DSP:INTervu:ERRTrig:PARity?

This query returns the Intervu Parity Error Triggering setting.

Response Argument(s):

- **error** - { OFF | ON }

Related Commands: :DSP:INTervu:ERRTrig:PARity

Command Syntax: :DSP:INTervu:ERRTrig:PARity?

Response Syntax: :DSP:INTervu:ERRTrig:PARity **error**

Example: `:DSP:INTERVU:ERRTRIG:PARITY?`

Response: `:DSP:INTERVU:ERRTRIG:PARITY OFF`

: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 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 | FALL | RISE }

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: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 ATS-2), 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:BATCh? and :DSP:MEASurement? 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.

	Probability	Jitter	Amplitude	Upper Eye	Lower Eye
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**
Valid combinations of :DSP:INTERVU:MODE and INTERVU measurement queries					

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:SET?

This query returns the state of the Intervu DSP program settings. Note that the TSLope setting will be included in the response only if the TRIGger setting is set appropriately (see :DSP:INTERVU:TSLope).

Response1 illustrates the response string when TRIGGER is set to ARCV. TSLOPE is not provided.

Response2 illustrates the response string when TRIGGER is set to JITTER. TSLOPE is provided.

Response argument(s):

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

Related Commands: :DSP:INTervu:SET

Command Syntax: :DSP:INTervu:SET?

Response Syntax: :DSP:INTervu:**settings**

Example:
```
:DSP:INTERVU:SET?
```

Response1:
```
:DSP:INTERVU:AVGS 1;JDETECTION STABLE;MODE
INTRPOLATE;TMODE POSTTRIG;TRIGGER
ARCV;WINDOW BH;ERRTRIG:CODING ON;CONFIDENCE
ON;LOCK ON;PARITY ON
```

Response2:
```
:DSP:INTERVU:AVGS 1;JDETECTION STABLE;MODE
INTRPOLATE;TMODE POSTTRIG;TSLOPE POS;TRIGGER
JITTER;WINDOW BH;ERRTRIG:CODING
ON;CONFIDENCE ON;LOCK ON;PARITY ON
```

:DSP:INTervu:TMODe

This command sets the Intervu data acquisition mode that acquires the digital interface waveform data prior to the trigger event (pre-trigger) (PRETrig) or after the trigger event (post-trigger) (POSTtrig).

The command argument is:

- **mode** - { POSTtrig | PRETrig }

Default: POSTtrig

Related Commands: :DSP:INTervu:TMODe?, :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:TMODe **mode**

Example:
```
:DSP:INTERVU:TMODE PRETRIG
```

:DSP:INTervu:TMODe?

This query returns the Intervu data acquisition mode setting, either pre-trigger (PRETrig) or post-trigger (POSTtrig).

Response Argument(s):

- **mode** - { POSTtrig | PRETrig }

Related Commands: :DSP:INTervu:TMODe

Command Syntax: :DSP:INTervu:TMODe?

Response Syntax: :DSP:INTervu:TMODe **mode**

Example:
```
:DSP:INTERVU:TMODE?
```

Response:
```
:DSP:INTERVU:TMODE PRETRIG
```

:DSP:INTervu:TRIGger

This command controls when acquisition of digital interface signal begins into the ATS-2 INTERVU acquisition buffer, along with the Trigger Slope command (:DSP:INTervu:TSLope), the Trigger Mode command (:DSP:INTervu:TMODe), and the four :DSP:INTervu:ERRTrig Receive Error Trigger commands (Confidence, Coding, Lock, and Parity) in the case of the Receive Error or Sync Error Trigger.

The Digital Interface Analyzer has an extensive trigger source list with the additional capability of triggering on up to four error conditions from either the digital input or the SYNC/REF input.

When :DSP:ACQX? is sent, the timing of the beginning of the Digital Interface Analyzer acquisition is determined by the :DSP:INTervu:TRIGger command and the :DSP:INTervu:ERRTrig commands, the :DSP:INTervu:TSLope command and the :DSP:INTervu:TMODe command. The various trigger source choices include several points in the digital interface from the received digital input, the transmitted digital output or the Sync/Ref received input; from an external clock reference, the AC mains frequency and internal generators. The trigger from the digital interface signals can be extracted from the X- and Z-preambles (Channel A subframe), the Y-preambles (Channel B subframe) or the “block” Z-preambles (Block 192 frames).

Most of these alternative triggering sources will not cause any difference of either spectrum analysis or waveform of the jitter signal, or of spectrum analysis of the interface signal waveform. The differences will be seen only when displaying the interface signal waveform (time domain) with a narrow span (a few microseconds) between the :DSP:SRCParams Start and Stop times so that the 3-UI, 2-UI, and 1-UI pulse widths that make up the preamble can be distinguished.

On the various preamble trigger sources, the trigger operation is such that the trailing edge of the first 3-UI pulse of the preamble occurs nominally at time zero. The first information displayed after time zero in these cases will be the remaining 5 UIs of the selected preamble, followed by the first bit in the data area. Depending on the nature of the interface signal, that could be the LSB of the audio signal if full 24-bit resolution audio is transmitted, or the beginning of the 4-bit Auxiliary data area if audio is restricted to 20 bits or fewer.

The transmit preamble selections have the same triggering characteristics. This triggering selection permits measurement of time delay through a digital device or system under test.

The Transmit and Receive Block selections cause signal to be acquired at the first Channel Status Block Preamble transmitted or received. The Receive Block is delayed by two full frames by the AES receiver.

The Jitter Generator signal triggers at every cycle of the sinewave signal generated by the DIO jitter generator. This selection provides a stable acquisition of the received jitter waveform when measuring through a digital device.

Triggering for squarewave acquisitions

None of the serial digital interface trigger sources (receive and transmit block, error and sub-frame sources) are useful when measuring squarewave jitter. For stable triggering on a squarewave signal, split the connection with a BNC "T" adapter and connect the two resultant lines to the ATS-2 DIGITAL INPUT and the TRIG IN rear panel connection. Select Trig In (EXTernal) as the Trigger Source.

The Jitter Generator trigger selection works only when Sinewave Jitter is turned on with the :DOUT:JWFM command. This trigger mode can be useful when looking at jitter on a squarewave clock that is derived from an AES3 signal fed from ATS-2's digital generator output.

The Trigger choices are:

- Analog Generator Audio Signal (AGEN),
- Digital Generator Audio Signal (DGEN),
- Jitter Generator (JITTer),
- External Trigger Input (EXTernal),
- AC Line Mains (ACMains),
- Ch. A Receive Sub-Frame (Preamble) (ARCV),
- Ch. B Receive Sub-Frame (Preamble) (BRCV),
- Ch. A Receive Sub-Frame DeJitt (Preamble) (DJARcv),
- Ch. B Receive Sub-Frame (Preamble) (DJBRcv),
- Receive Block (192 frames) (RCVBlock),
- Receive Error (RCVerr),
- Ch. A Transmit Sub-Frame (Preamble) (AXMT),
- Ch. B Transmit Sub-Frame (Preamble) (BXMT),
- Ch. A Transmit Sub-Frame DeJitt (Preamble) (DJAXmt),
- Ch. B Transmit Sub-Frame DeJitt (Preamble) (DJBXmt),
- Transmit Block (XMTBlock),
- Ch A Sync/Ref Receive Sub-Frame (Preamble) (ASYNc),
- Ch B Sync/Ref Receive Sub-Frame (Preamble) (BSYNc),
- Sync/Ref Rcv Block (192 frames) (BLKSync),
- Sync/Ref Rcv Error (ESYNc)

Note that the Receive choices pertain to the signal at the I or II front-panel input connector selected by the :DIN:CONNector command if the input is XLR (see :DIN:FORMat command).

Analog Gen (AGEN)

A trigger transition is generated for each cycle of the current Analog Generator waveform.

Digital Gen (DGEN)

A trigger transition is generated for each cycle of the current Digital Generator waveform.

Jitter Gen (JITTer)

A trigger transition is generated for each cycle of the current Jitter Generator waveform.

Trig In (EXTernal)

A trigger transition is generated for each external trigger transition received at the TRIG IN BNC connector.

Line (ACMains)

A trigger transition is generated for each cycle of the AC mains line voltage.

ChA Rcv Sub-Frame (ARCV)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal received at the digital input. If there is jitter in the received waverform, that jitter will affect the trigger.

ChB Rcv Sub-Frame (BRCV)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal received at the digital input. If there is jitter in the received waverform, that jitter will affect the trigger.

ChA Rcv Sub-Frame DeJitt (DJARcv)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal received at the digital input, after the signal has be re-clocked to remove jitter.

ChB Rcv Sub-Frame DeJitt (DJBRcv)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal received at the digital input, after the signal has be re-clocked to remove jitter.

Rcv Block (192 frames) (RCVBlock)

A trigger transition is generated at the beginning of each Status Block frame (each Z-preamble) in the interface signal received at the digital input. This occurs once every 192 frames.

Rcv Error (RCVerr)

Four “receive error” trigger conditions in the interface waveform are defined. See :DSP:INTervu:ERRTrig. When any of these selected conditions occur in the interface signal received at the the main digital input, a trigger transition will be generated.

ChA Xmit Subframe (AXMT)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal transmitted at the digital output. If jitter has been added to the digital output, that jitter will affect the trigger.

ChB Xmit Subframe (BXMT)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal transmitted at the digital output. If jitter has been added to the digital output, that jitter will affect the trigger.

ChA Xmit Subframe DeJitt (DJAXmt)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal transmitted at the digital output, before the addition of any jitter.

ChB Xmit Subframe DeJitt (DJBXmt)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal transmitted at the digital output, before the addition of any jitter.

Xmit Block (192 frames) (XMTBlock)

A trigger transition is generated at the beginning of each Status Block frame (each Z-preamble) in the interface signal transmitted at the digital output. This occurs once every 192 frames.

ChA Sync/Ref Rcv Sub-Frame (ASYNc)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal received at the SYNC/REF IN input.

ChB Sync/Ref Rcv Sub-Frame (BSYNc)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal received at the SYNC/REF IN input.

Sync/Ref Rcv Block (192 frames) (BLKSync)

A trigger transition is generated at the beginning of each Status Block frame (each Z-preamble) in the interface signal received at the SYNC/REF IN input. This occurs once every 192 frames.

Sync/Ref Rcv Error (ESYNc)

Four “receive error” conditions in the interface waveform are defined and flagged. See :DSP:INTervu:ERRTrig. When any of these trigger conditions occur in the interface signal received at the SYNC/REF input, a trigger transition will be generated.

The trigger slope :DSP:INTERVU:TSLOPE is set to POS if :DSP:INTervu:TRIGger is set to trigger on an error, sub-frame, or block.

This command will change the setting of the :TRIGger:SOURce comman (both commands change the same hardware trigger circuitry). Changing the :TRIGger:SOURce command will make an identical change to this command.

The command argument is:

- **type** - { ACMains | AGEN | ARCV | ASYNc | AXMT | BLKSync | BRCV | BSYNc | BXMT | DGEN | DJARcv | DJAXmt | DJBRcv | DJBXmt | ESYNc | EXTernal | JITTer | RCVBlock | RCVerr | XMTBlock }

Default: ARCV

Related Commands: :DSP:INTervu:TRIGger?, :DSP:INTervu:TSLope, :DSP:INTervu:ERRTrig, :TRIGger:SOURce

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** - { ACMains | AGEN | ARCV | ASYNc | AXMT | BLKSync | BRCV | BSYNc | BXMT | DGEN | DJARcv | DJAXmt | DJBRcv | DJBXmt | ESYNc | EXTernal | JITTer | RCVBlock | RCVerr | XMTBlock }

Related Commands: :DSP:INTervu:TRIGger, :TRIGger:SOURce

Command Syntax: :DSP:INTervu:TRIGger?

Response Syntax: :DSP:INTervu:TRIGger **type**

Example: `:DSP:INTERVU:TRIGGER?`

Response: `:DSP:INTERVU:TRIGGER ARCV`

:DSP:INTervu:TSLope

This command specifies the Intervu acquisition trigger slope when the the :DSP:INTervu:TRIGger command is set to AGEN, DGEN, JITTer, ACMains, or EXTernal.

An error is generated (513,13,":DSP:INTERVU:TSLOPE, INCOMPATIBLE TRIGGER, SLOPE SET TO POS") if this command is received when :DSP:INTervu:TRIGger is set to an error, sub-frame, or block, e.g. { ARCV | ASYNc | AXMT | BLKSync | BRCV | BSYNc | BXMT | DJARcv | DJAXmt |

DJBRcv | DJBXmt | ESYNc | RCVBlock | RCVerr | XMTBlock }.

The trigger slope is automatically set to POS if :DSP:INTervu:TRIGger is set to trigger on an error, sub-frame, or block, e.g. { ARCV | ASYNc | AXMT | BLKSync | BRCV | BSYNc | BXMT | DJARcv | DJAXmt | DJBRcv | DJBXmt | ESYNc | RCVBlock | RCVerr | XMTBlock }.

The command argument is:

- **slope** - { NEG | POS }

Default: POS

Related Commands: :DSP:INTervu:TRIGger, :DSP:INTervu:TSLope?

Command Syntax: :DSP:INTervu:TSLope **slope**

Example: `:DSP:INTERVU:TSLOPE NEG`

:DSP:INTervu:TSLope?

This command returns the Intervu acquisition trigger slope setting.

Response Argument(s):

- **slope** - { NEG | POS }

Related Commands: :DSP:INTervu:TSLope, :DSP:INTervu:TRIGger

Command Syntax: :DSP:INTervu:TSLope?

Response Syntax: :DSP:INTervu:TSLope **slope**

Example: `:DSP:INTERVU:TSLOPE?`

Response: `:DSP:INTERVU:TSLOPE NEG`

: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 for effective close-in selectivity (signal spillover into nearby FFT bins) and for 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:MEASurement?

This query returns a single measurement at the first sweep point specified in the :DSP:SRCParams command. The response is a single ASCII formatted numeric measurement value with a unit suffix.

If the :DSP:SRCParams “mode” parameter is LINear or LOG, the source data point is the sweep start point.

If :DSP:SRCParams “mode” parameter is ARBitrary, the source data point is the first point found in the table selected by :DSP:TSEL, specified by the :DSP:TABLe command.

Note that the “meas” parameter (combination of channel and measurement) must have been previously enabled with the :DSP:OPSTate READing command prior to receipt of this command. For example, if the currently loaded DSP program is FFT and :DSP:OPSTATE is set to READING,PHA1,PHA2, then only PHA1 or PHA2 would be valid settings for the :DSP:MEASurement “meas” parameter. The valid setting of the “unit” parameter would then be limited to DEG only. See the tables below for legal units for a specific “meas” parameter.

The 'meas' parameter must be appropriate for the DSP program currently loaded according to this table:

	FASTTEST	FFT	INTERVU
AMP1	x	x	
AMP2	x	x	
PHA1	x	x	
PHA2	x	x	
AMPLitude			x

JITTer			x
LOWer			x
UPPer			x
PROBability			x

Units valid for each DSP program used with :DSP:MEASurement? query are shown in the table below for each measurement parameter. These are the same measurements and units available with the :DSP:BATCh? query.

DSP Program	Input Type	Measurement Parameter	Valid Units
FASTTEST	ANLG	AMP1 & AMP2	DBFS, DBGA, DBGB, DBRA, DBRB, DBM, DBU, DBV, FFS, PCTFs, V, W
	DIGital	AMP1 & AMP2	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
	ANLG & DIGital	PHA1 & PHA2	DEG
FFT	ANLG	AMP1 & AMP2	DBFS, DBGA, DBGB, DBRA, DBRB, DBM, DBU, DBV, FFS, PCTFs, V, W
	JITTer	AMP1	DBFS, DBGA, DBGB, DBRA, DBRB, DBM, DBU, DBV, FFS, PCTFs, V, W
		AMP2	DBUI, UI
	JITSec	AMP1	DBFS, DBGA, DBGB, DBRA, DBRB, DBM, DBU, DBV, FFS, PCTFs, V, W
		AMP2	SEC
	DIGital	AMP1 & AMP2	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
	ANLG & DIGital	PHA1 & PHA2	DEG
INTERVU	–	AMPLitude	DBV, V
		LOWer or UPPer	DBV, V
		JITTer	DBUI, SEC, UI
		PROBability	DB, PCT, PPM, X_Y

Errors

This query is legal only if :DSP:OPSTATE is REAding. An error will be generated if :DSP:OPSTATE is SETup.

An error will be generated if the setting of the “meas” parameter is inconsistent with the setting(s) of the :DSP:OPSTate READing command arguments (that are dependent on the DSP program currently loaded).

An error will be generated if this command is recieved when the DSP Program currently loaded is DANLr or HARMonic.

The query arguments are:

- **meas** - { AMP1 | AMP2 | AMPLitude | JITTer | LOWer | PHA1 | PHA2 | PROBability | UPPer }
- **unit** - { BITs | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBUI | DBV | DEG | FFS | PCT | PCTFs | PPM | SEC | UI | V | W | X_Y }

The response arguments are:

- **data** - <nrf> { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBUI |

DBV | DEG | FFS | PCT | PCTFs | PPM | SEC | UI | V | W | X_Y }

Related Commands: :DSP:BATCH?, :DSP:PROGram, :DSP:OPSTate, :DSP:SRCParams, :DSP:TABLe

Command Syntax: :DSP:MEASurement? **meas**, **unit**

Response Syntax: :DSP:MEASurement **data**

Example: `:DSP:MEASUREMENT? AMP1,FFS`

Response: `:DSP:MEASUREMENT 0.125FFS`

: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.

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 with :DSP:BATCH? or :DSP:MEASurement? (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.

The readings must be appropriate for the DSP program being used.

See the table below for valid measurements for each batch mode DSP program. These may be used as optional parameters following the READing parameter. For example, AMPLitude, JITTer, UPPer, LOWer and PROBability are only valid when :DSP:PROGram is set to INTervu, but AMP1, AMP2, PHA1 and PHA2 are NOT valid for INTervu.

	FASTTEST	FFT	INTERVU
AMP1	x	x	
AMP2	x	x	
PHA1	x	x	
PHA2	x	x	
AMPLitude			x
JITTer			x
LOWer			x
UPPer			x
PROBability			x

This command will also cause an execution error if the current :DSP:PROG is either DANLr or HARMonic. For additional discussion on the use 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:BATCh?, :DSP:PROGram, :DSP:MEAsurement?, :DSP:OPSTate?

Command Syntax: :DSP:OPSTate **mode[, enable1[, enable2[, enable3[, enable4]]]]**

Example: `:DSP:OPSTATE READING,AMP1,PHA1`

: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,AMP1,PHA1`

Response: `:DSP:OPSTATE SETUP`

:DSP:PROGram

This command specifies the DSP program to be loaded into the DSP processing system. By default, the Audio Analyzer DANLR program is loaded at power-on.

There are 5 DSP programs available for ATS-2: the Audio Analyzer (DANLr), the Spectrum Analyzer (FFT), the Harmonic Distortion Analyzer (HARMonic), the Multitone Audio Analyzer (FASTtest) and the optional Digital Interface Analyzer (INTervu).

The command argument is:

- **program** - { DANLr | FASTtest | FFT | HARMonic | INTervu }

Default: DANLR

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** - { DANLr | FASTtest | FFT | HARMonic | INTervu }

Related Commands: DSP:PROGram

Command Syntax: :DSP:PROGram?

Response Syntax: :DSP:PROGram **program**

Example: `:DSP:PROGRAM?`

Response: `:DSP:PROGRAM FFT`

:DSP:REF Compound Command Header

These commands set the parameters for the DSP analyzer units (both real time and batch mode) which require reference values. These commands are duplicated here for convenience.

:DSP:REF:DBM

This command sets the DBM unit impedance reference value.The reference value is in units of ohms.

The command argument is:

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

Default: 600

Related Commands: :DSP:REF:DBM?

Command Syntax: :DSP:REF:DBM **impedance**

Example: `:DSP:REF:DBM 300`

:DSP:REF:DBM?

This query returns the current DBM unit impedance reference setting for the analyzer. The response value is ohms.

Response argument(s):

- **impedance** - <nrf>

Related Commands: :DSP:REF:DBM

Command Syntax: :DSP:REF:DBM?

Response Syntax: :DSP:REF:DBM **impedance**

Example: `:DSP:REF:DBM?`

Response: `:DSP:REF:DBM 300`

:DSP:REF:DBR1

This command sets the DBR1 unit reference value. This reference value is only used for DANLR, FASTTEST, FFT, and HARMonic when the INPut is DIGital.

The minimum range is 0 FFS or PCTFS, or -1E+34 for DBFS or BITS units.

The command argument is:

- **setting** - <nrf> (range: ≥ -1E34, ≤ 1E34) { BITS | DBFS | FFS | PCTFs }

Default: 0.100 FFS

Related Commands: :DSP:REF:DBR1?, :DSP:REF:VFS

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.

The command arguments are:

- **units** - { BITS | DBFS | FFS | PCTFs }

Response argument(s):

- **setting** - <nrf> { BITS | DBFS | FFS | PCTFs }

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. This reference value is only used for DANLR, FASTTEST, FFT, and HARMonic when the INPut is DIGital.

The minimum range is 0 FFS or PCTFS, or -1E+34 for DBFS or BITS units.

The command argument is:

- **setting** - <nrf> (range: ≥ -1E34, ≤ 1E34) { BITS | DBFS | FFS | PCTFs }

Default: 0.100 FFS

Related Commands: :DSP:REF:DBR2?

Command Syntax: :DSP:REF:DBR2 **setting**

Example:
```
:DSP:REF:DBR2 0.80FFS
```

:DSP:REF:DBR2?

This query returns the current DBR2 unit reference setting for the DSP analyzer in the specified units.

The command arguments are:

- **units** - { BITS | DBFS | FFS | PCTFs }

Response argument(s):

- **setting** - <nrf> { BITS | DBFS | FFS | PCTFs }

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:DBRA

This command sets the analyzer DBRA unit reference value. This reference value can also be set using the :DSP:REF:SETRefauto command.

This reference value is only used for DANLR, FFT, FASTTEST, and HARMonic when the INPut is ANAlog.

The minimum range is 0 V or -1E+34 for DBU or DBV units.

The command argument is:

- **setting** - <nrf> (range: > 0, 1E34) V, { V | DBU | DBV }

Default: 0.3873V

Related Commands: :DSP:REF:DBRA?, :DSP:REF:SETRefauto

Command Syntax: :DSP:REF:DBRA **setting**

Example: `:DSP:REF:DBRA 1 DBV`

:DSP:REF:DBRA?

This query returns the current DBRA unit reference setting for the analyzer in the specified units.

The command arguments are:

- **units** - { V | DBU | DBV }

Response argument(s):

- **setting** - <nrf> { V | DBU | DBV }

Related Commands: :DSP:REF:DBRA

Command Syntax: :DSP:REF:DBRA? **units**

Response Syntax: :DSP:REF:DBRA **setting**

Example: `:DSP:REF:DBRA? V`

Response: `:DSP:REF:DBRA 1.0V`

:DSP:REF:DBRB

This command sets the analyzer DBRB unit reference value. This reference value can also be set using the :DSP:REF:SETRefauto command.

This reference value is only used for DANLR, FFT, FASTTEST, and HARMonic when the INPut is ANAlog.

The minimum range is 0 V or -1E+34 for DBU or DBV units.

The command argument is:

- **setting** - <nrf> (table range: > 0, = 1E34) V { V | DBU | DBV }

Default: 0.3873V

Related Commands: :DSP:REF:DBRB?, :DSP:REF:SETRefauto

Command Syntax: :DSP:REF:DBRB **setting**

Example: `:DSP:REF:DBRB 1 DBV`

:DSP:REF:DBRB?

This query returns the current DBRB unit reference setting for the analyzer in the specified units.

The command arguments are:

- **units** - { V | DBU | DBV }

Response argument(s):

- **setting** - <nrf> { V | DBU | DBV }

Related Commands: :DSP:REF:DBRB

Command Syntax: :DSP:REF:DBRB? **units**

Response Syntax: :DSP:REF:DBRB **setting**

Example: `:DSP:REF:DBRB? V`

Response: `:DSP:REF:DBRB 1.0V`

:DSP:REF:FREQ

This command sets the FREQ unit reference value for the DSP analyzer. The unit is hertz.

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 implicit 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 implicit 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:REF:WATT

This command sets the WATT unit impedance reference value for the analog analyzer. The unit is ohms.

The command argument is:

- **impedance** - <nrf> (range: > 0, 1E34)

Default: 8.0

Related Commands: :DSP:REF:WATT?

Command Syntax: :DSP:REF:WATT **impedance**

Example: `:DSP:REF:WATT 8`

:DSP:REF:WATT?

This query returns the WATT unit impedance reference value for the analog analyzer. The response unit is ohms.

Response argument(s):

- **impedance** - <nrf>

Related Commands: :DSP:REF:WATT

Command Syntax: :DSP:REF:WATT?

Response Syntax: :DSP:REF:WATT **impedance**

Example: `:DSP:REF:WATT?`

Response: `:DSP:REF:WATT 8.0`

:DSP:REPRocess?

This query causes the DSP to do post-processing of measurement results using the current DSP settings and to return a timeout status. This query performs the last setps of the three processing steps performed when a DSP:ACQX? query is executed. 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 (see :DSP:OPSTate).

A query response will be generated when either the reprocess completes or times out (see :DSP:TIMeout).

The response argument indicates time-out status (whether the query completed in the specified time). If the response is 0 (zero), then a time-out did not occur. If the response is a 1 (one) then a time-out occurred and data is not available for the :DSP:BATCh? or :DSP:MEASurement? queries.

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:ACQX?, :DSP:OPSTate, :DSP:PROGram, :DSP:XFRM?

Command Syntax: :DSP:REPRocess?

Response Syntax: :DSP:REPRocess **status**

Example: `:DSP:REPROCESS?`

Response: `:DSP:REPROCESS 0`

:DSP:SET?

This query returns instrument DSP settings for the currently loaded DSP program and related acquisition parameters if appropriate.

Response 1 illustrates the response elements when a real-time DSP program is loaded (DANLR). DSP Timeout and SRCParams settings are not provided.

Response 2 illustrates the response elements when a batch-mode DSP program is loaded (FASTTEST). DSP Timeout and SRCParams settings are provided.

Response argument(s):

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

Related Commands:

Command Syntax: :DSP:SET?

Response Syntax: :DSP:**settings**

Example:

```
:DSP:SET?
```

Response 1:

```
:DSP:REF:DBM 50;DBR1 0.1FFS;DBR2 0.1FFS;DBRA
0.1V;DBRB 0.1V;FREQ 1000;VFS 1;WATT
4;:DSP:PROGRAM DANLR;:DSP:DANLR:AUTORANGE
A,ON;AUTORANGE B,ON;COUPLING A,AC;COUPLING
B,AC;DETECTOR FRMS;HPFILTER F10;INPUT
DIGITAL;LPFILTER FS_2;MODE
AMPLITUDE;RESPONSE 1000;WTG UNWT;RDGRATE
R8;FAUTORANGE A,ON;FAUTORANGE B,ON;PRANGE
AUTO
```

Response 2:

```
:DSP:REF:DBM 600;DBR1 0.1FFS;DBR2
0.1FFS;DBRA 0.3873V;DBRB 0.3873V;FREQ
1000;VFS 1;WATT 8;:DSP:PROGRAM
FASTTEST;TIMEOUT 10.000000;SRCPARAMS
FREQ,HZ,20,20000,30,LOG;:DSP:FASTTEST:FREQRE
S 0;INPUT ANLG;LENGTH AUTO;MODE
SPECTRUM;PMODE INDEPENDENT;PROCESS
SYNC;TRGDELAY 0;TRIGGER AGEN;PKTRIG 1
```

:DSP:SRCParams

This command sets up the currently loaded ATS-2 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.

The table below shows the valid source parameters available for each DSP program and the valid units for each source parameter.

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

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).

This command will cause an execution error 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: ≥ −1E+34, ≤ 1E+34)
- **stop** - <nrf> (range: ≥ −1E+34, ≤ 1E+34)
- **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?, :DSP:MEASurement?

Command Syntax: :DSP:SRCParams **parameter, units, start, stop, steps, mode**

Example: `:DSP:SRCPARAMS AMPLITUDE,V,-3,3,600,LINEAR`

Example: `:DSP:SRCPARAMS TIME,SEC,-0.1,1.5,1599,LINEAR`

Example: `:DSP:SRCPARAMS FREQ,HZ,100,10000,20,LOG`

: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 the previous table }
- **start** - <nrf> (range: ≥ -1E+34, ≤ 1E+34)
- **stop** - <nrf> (range: ≥ -1E+34, ≤ 1E+34)
- **steps** - <nr1> (range: ≥ 1, ≤ 16383)
- **mode** - { LINear | LOG | ARBitrary }

Default: FREQ,HZ, 20, 20000, 30, LOG

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,-3,3,600,LINEAR`

:DSP:TABLe

This command loads a sweep source table into one of two table registers for eventual use by the :DSP:BATCh? or :DSP:MEASurement? queries when the last parameter of the :DSP:SRCParams command is set to ARBitrary. 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:SRCParams command sets the last parameter to ARBITRARY (the table will not be used if the last 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 erased and replaced 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 bock data block (if the specified 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 arbitrary block data> or <indefinite length arbitrary block data>. If it is formatted as <indefinite length arbitrary block data>, then it must be terminated with <PMT>.

Default: none

Related Commands: :DSP:BATCh?, :DSP:MEASurement?, :DSP:SRCParams, :DSP:TSELect

Command Syntax: :DSP:TABLe **table, format, numpoints, arbblockdata**

Example:

```
: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.
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 <PMT>. 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 arbitrary 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,#017.57,23.43,29.29,41
.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.
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 timeout parameter in seconds units for the :DSP:ACQX?, :DSP:XFRM?, or :DSP:REPRocess? queries.

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:XFRM?, :DSP:REPRocess?, :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? Or :DSP:MEASurement queries when the last :DSP:SRCParams parameter is set to ARBITRARY. The sweep table will not be used if LIN or LOG is specified instead of ARBITRARY.

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? or :DSP:MEASurement? queries are received if the selected sweep table is not already defined.

The command argument is:

- **table** - <nr1> (range: 1 or 2)

Default: 1

Related Commands: :DSP:TABLe, :DSP:BATCh?

Command Syntax: :DSP:TSELect **table**

Example: `:DSP:TSELECT 2`

:DSP:TSELect?

This query returns the active sweep table.

The response argument is:

- **table** - <nr1> (range: 1 or 2)

Related Commands: :DSP:TSELect

Command Syntax: :DSP:TSELect? **table**

Response Syntax: :DSP:TSELect **table**

Example: `:DSP:TSELECT?`

Response: `:DSP:TSELECT 2`

:DSP:XFRM?

This query causes the DSP to do a transform of data without a new acquisition. If the FFT DSP program is loaded, then an FFT transform is performed on data in the acquisition buffer, data that have been acquired with an earlier :DSP:ACQX? query. The transform will be performed using the current DSP settings and return a timeout status. This command is valid only in OPSTate READing mode.

An execution error will be generated if the batch mode DSP program is in OPSTate SETup mode.

A query response will be generated when either the transform process completes or a time-out occurs (see :DSP:TIMeout).

The response argument (status) indicates a time-out status (whether the transform completed in the specified time). If the response is 0 (zero), then a time-out did not occur. If the response is 1 (one) then a time-out did occur and data is not available to be read with the :DSP:BATCh? or :DSP:MEASurement queries.

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)

Default: <None>

Related Commands: :DSP:BATCh?, :DSP:MEASurement?, :DSP:TIMeout

Command Syntax: :DSP:XFRM?

Response Syntax: :DSP:XFRM? **status**

Example:
```
:DSP:XFRM?
```

Response:
```
:DSP:XFRM 0
```

Chapter 10
Digital Output Commands

The header-path for Digital Output commands is :DOUT.

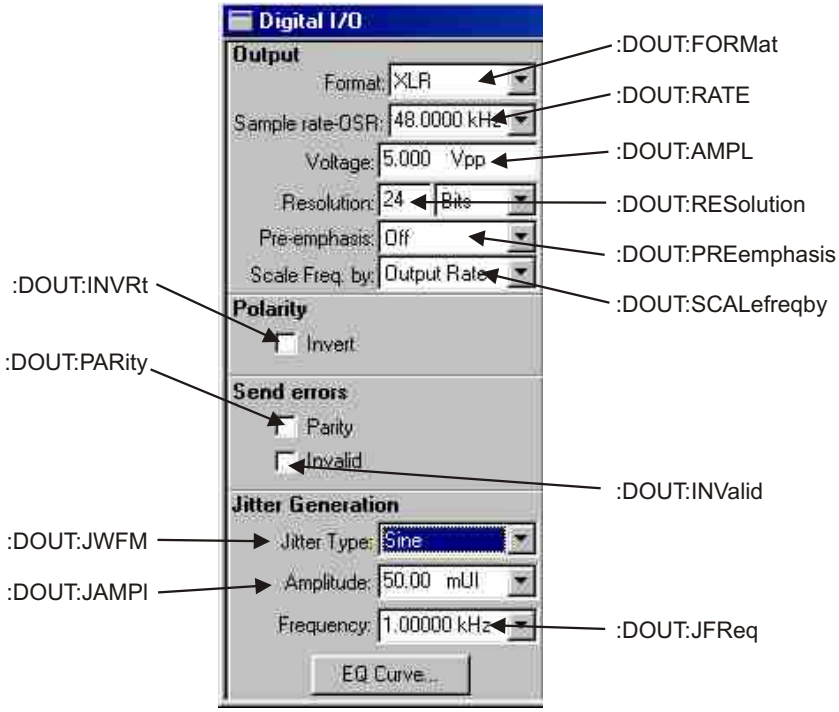


Figure 10-1. ATS software Digital Output control panel DOUT GPIB commands.

: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. 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.

The actual output voltage will agree with the setting determined by :DOUT:AMPL only at the connector selected by :DOUT:FORMat, and only if that connector is properly

terminated (110 Ohms for the XLR, 75 Ohms for the BNC). The voltage at the other connectors will be in error.

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 Optical output intensity is not variable.

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, ≤ 5.1 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 selected XLR or BNC outputs. The response has units of Vpp.

Response argument(s):

- **setting** - <nrf> (range: ≥ 0, ≤ 5.1 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:FORMat

Specifies the output for the digital signal. The valid outputs are:

Output	Parameter	Description
XLR (bal)	XLR	Front panel XLR digital output connector, balanced
BNC (unbal)	BNC	Front panel BNC digital output connector, unbalanced
Optical	OPTical	Front panel Toslink optical output connector
Dual XLR	XLRDual	Front panel dual XLR (balanced) digital output
Dual XLR @ 2x OSR	XLR2xdual	Front panel dual XLR (balanced) digital output connectors, operate at twice output sample rate

The command argument is:

- **source** - { BNC | OPTical | XLR | XLRDual | XLR2xdual }

Default: XLR

Related Commands: :DOUT:FORMat?

Command Syntax: :DOUT:FORMat **source**

Example: `:DOUT:FORMAT BNC`

:DOUT:FORMat?

Returns the currently selected digital output.

Response argument(s):

- **source** - { BNC | OPTical | XLR | XLRDual | XLR2xdual }

Related Commands: :DOUT:FORMat

Command Syntax: :DOUT:FORMat?

Response Syntax: :DOUT:FORMat **source**

Example: `:DOUT:FORMAT?`

Response: `:DOUT:FORMAT BNC`

: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 subframes 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 or 1)

Default: 0

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 or 1)

Related Commands: :DOUT:INValid

Command Syntax: :DOUT:INValid?

Response Syntax: :DOUT:INValid **state**

Example: `:DOUT:INVALID?`

Response: `:DOUT:INVALID 1`

:DOUT:INVRt

This command inverts the signal polarity of the digital output data stream at the AES/EBU output connectors.

The command argument is:

- **invert** - { OFF | ON }

Default: OFF

Related Commands: :DOUT:INVRt?

Command Syntax: :DOUT:INVRt **invert**

Example: `:DOUT:INVRT ON`

:DOUT:INVRt?

This query returns the AES/EBU outputs invert setting.

Response argument(s):

- **invert** - { OFF | ON }

Related Commands: :DOUT:INVRt

Command Syntax: :DOUT:INVRt?

Response Syntax: :DOUT:INVRt **invert**

Example: `:DOUT:INVRT?`

Response: `:DOUT:INVRT ON`

:DOUT:JAMPl

Specifies the injected jitter signal amplitude when the jitter waveform is sine. Units are Seconds (SEC), Unit Intervals (UI), or dB relative to 1 Unit Interval (DBUI).

The command argument is:

- **amplitude** - <nrf> (range: ≥ 0, ≤ 12.75 UI, depends on internal sample rate for seconds units) { DBUI | SEC | UI}

Default: 0.0 UI

Related Commands: :DOUT:JAMPl?, :DOUT:JWFM

Command Syntax: :DOUT:JAMPl **amplitude**

Example: `:DOUT:JAMPL 5.5UI`

:DOUT:JAMPl?

Returns the injected jitter amplitude setting (in the specified units).

The command argument is:

- **units** - { DBUI | SEC | UI }

Response argument(s):

- **amplitude** - <nrf> { DBUI | SEC | UI }

Related Commands: :DOUT:JAMPl

Command Syntax: :DOUT:JAMPl? **units**

Response Syntax: :DOUT:JAMPl **amplitude**

Example:
```
:DOUT:JAMPL? UI
```

Response:
```
:DOUT:JAMPL 5.5UI
```

:DOUT:JFReq

Specifies the frequency of the injected sinewave jitter signal.

The command argument (in hertz) is:

- **frequency** - <nrf> (range: ≥ 2.0, ≤ 200001)

Default: 998.644

Related Commands: :DOUT:JFReq?

Command Syntax: :DOUT:JFReq **frequency**

Example:
```
:DOUT:JFREQ 1500
```

:DOUT:JFReq?

Returns current jitter sinewave frequency. The response is in hertz.

Response argument(s):

- **frequency** - <nrf>

Related Commands: :DOUT:JFReq

Command Syntax: :DOUT:JFReq?

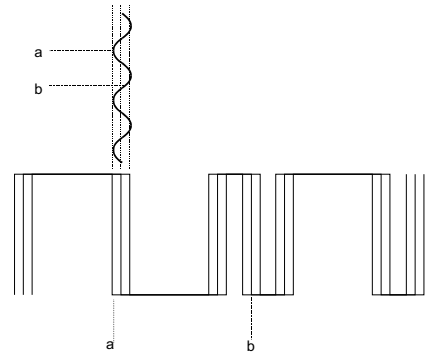
Response Syntax: :DOUT:JFReq **frequency**

Example:
```
:DOUT:JFREQ?
```

Response:
```
:DOUT:JFREQ 1500
```

:DOUT:JWFM

Disables the interface jitter signal or specifies the jitter sinewave waveform.



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 | SINE }

Default: NONE

Related Commands: :DOUT:JWFM?

Command Syntax: :DOUT:JWFM **type**

Example: `:DOUT:JWFM SINE`

:DOUT:JWFM?

Returns the current injected jitter waveform setting.

Response argument(s):

- **type** - { NONE | SINE }

Related Commands: :DOUT:JWFM

Command Syntax: :DOUT:JWFM?

Response Syntax: :DOUT:JTYPe **type**

Example: `:DOUT:JWFM?`

Response: `:DOUT:JWFM SINE`

:DOUT:PARity

This command sets the state of the digital output parity error flag. The parity flag indicates a parity error in a subframe.

A setting of 1 indicates a parity error, a setting of 0 indicates no parity error.

The command argument is:

- **parity** - <nr1> (range: 0 or 1)

Default: 0

Related Commands: :DOUT:PARity?

Command Syntax: DOUT:PARity **parity**

Example: `DOUT:PARITY 1`

:DOUT:PARity?

This query returns the setting of the digital output parity error flag. 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> (range: 0 or 1)

Related Commands: :DOUT:PARity

Command Syntax: :DOUT:PARity?

Response Syntax: :DOUT:PARity **parity**

Example: `:DOUT:PARITY?`

Response: `:DOUT:PARITY 1`

: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 (J17).

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 of CD+10db (CDGain) and J17+20db (J17Gain) 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: ≥ 28800, ≤ 108000) { 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 in bits if the second parameter 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.

If the second argument is ALAW, then the output word will be set to a fixed resolution of 13 bits and A-Law companded to 8 bits at the digital output. The first parameter will be ignored.

If the second argument is ULAW, then the output word will be set to a fixed resolution of 14 bits and u-Law companded to 8 bits at the digital output. The first parameter will be ignored.

The command argument is:

- **bits** - <nr1> (range: ≥ 8, ≤ 24)
- **mode** - { ALAW | BITS | ULAW }

Default: 24

Related Commands: :DOUT:RESolution?

Command Syntax: :DOUT:RESolution **bits, mode**

Example: `:DOUT:RESOLUTION 20,BITS`

:DOUT:RESolution?

Returns the resolution of the imbedded digital audio signal. If the second parameter is BITS then the first parameter is the bits of resolution of the output digital signal, otherwise the first parameter is irrelevent.

Response argument(s):

- **bits** - <nr1> (range: ≥ 8, ≤ 24)
- **mode** - { ALAW | BITS | ULAW }

Related Commands: :DOUT:RESolution

Command Syntax: :DOUT:RESolution?

Response Syntax: :DOUT:RESolution **bits, mode**

Example: `:DOUT:RESOLUTION?`

Response: `:DOUT:RESOLUTION 20,BITS`

:DOUT:SCALefreqby

This command sets the digital output mode for scaling output audio frequency.

Measured Input Rate (INMeas) uses the sample rate measured at the digital input as a basis for scaling the output audio frequency.

Measured Output Rate (OUTMeas) uses the sample rate measured at the digital output for scaling the output audio frequency.

Output Rate (OUTRate) uses the programmed output sample rate as the basis for scaling the output audio frequency.

DIO Rate Reference (REF) uses a user-entered reference sample rate as the basis for scaling the digital output audio frequency.

The command argument is:

- **source** - { INMeas | OUTMeas | OUTRate | REF }

Default: OUTRate

Related Commands: :DOUT:SCALefreqby?, :DOUT:RATe?, :DIN:REF

Command Syntax: :DOUT:SCALefreqby **source**

Example: `:DOUT:SCALEFREQBY REF`

:DOUT:SCALefreqby?

This query returns the setting of the digital output mode for scaling output audio frequency.

Response argument(s):

- **source** - { INMeas | OUTMeas | OUTRate | REF }

Related Commands: :DOUT:SCALefreqby

Command Syntax: :DOUT:SCALefreqby?

Response Syntax: :DOUT:SCALefreqby **source**

Example: `:DOUT:SCALEFREQBY?`

Response: `:DOUT:SCALEFREQBY REF`

: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.

Response argument(s):

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

Related Commands: see other :DOUT setting commands

Command Syntax: :DOUT:SET?

Response Syntax: :DOUT:**settings**

Example:
```
:DOUT:SET?
```

Response:
```
:DOUT:AMPL 5;FORMAT XLR;INVALID 0;INVRT
ON;JWFM SINE;JAMPL 0UI;JFREQ 998.644;JWFM
NONE;PARITY 0;PREEMPHASIS OFF;RATE
48000HZ;RESOLUTION 24,BITS;SCALEFREQBY
OUTRATE
```

Chapter 11
Digital Input Commands

The header-path for Digital Input commands is :DIN.

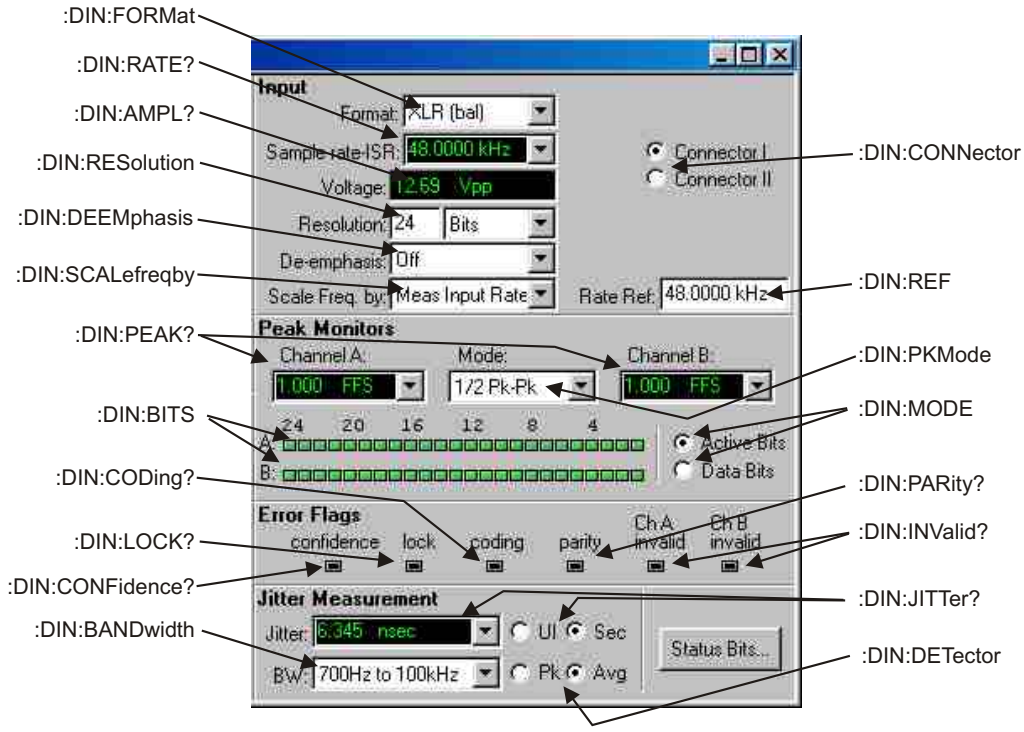


Figure 11-1. ATS software Digital Input control panel DIN GPIB commands.

:DIN:AMPL?

Returns the peak-to-peak digital interface signal amplitude in volts (Vpp) at the front panel XLR or BNC connector specified in the :DIN:FORMat command.

The digital interface signal amplitude is not calibrated for optical inputs. If the digital input format is Optical the :DIN:PEAK? query will not return a reading but will return a NAN. The settling timeout parameter will be set to 0. An error will be generated ":DIN:AMPL?, DIO, INPUT FORMAT IS OPTICAL".

The first parameter in the response is the measurement.

The 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, :DIN:FORMat

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**

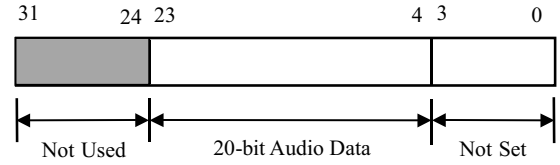
Example: `:DIN:BANDWIDTH?`

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 during the most recent measurement interval (approximately 1/2 second). 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> (range: 0 or 1)

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> (range: 0 or 1)

Related Commands:

Command Syntax: :DIN:CONFidence?

Response Syntax: :DIN:CONFidence **conf**

Example: `:DIN:CONFIDENCE?`

Response: `:DIN:CONFIDENCE 1`

:DIN:CONNector

This command selects which AES/EBU (XLR) input connector (1 or 2) and audio channels (A, B, C, and D) are used for certain digital domain measurements. These two connectors may be used in two ways; for connection to devices that operate at double sample rate (96 kHz, 88.2 kHz, etc.) by carrying a monaural signal at half the frame rate on each of two cables, or as built-in two-cable digital switchers during normal digital operation when a single cable carries a multiplexed stereo signal.

The command argument is:

- **connector** - <nr1> (range: 1 or 2)

Default: 1

Related Commands: :DIN:CONNector?

Command Syntax: :DIN:CONNector **connector**

Example: `:DIN:CONNector 2`

:DIN:CONNector?

This query returns the connector setting for which AES/EBU (XLR) input connector (1 or 2) is used for single-channel digital domain measurements.

Response arguments(s):

- **connector** - <nr1> (range: 1 or 2)

Related Commands: :DIN:CONNector

Command Syntax: :DIN:CONNector?

Response Syntax: :DIN:CONNector **connector**

Example: `:DIN:CONNECTOR?`

Response: `:DIN:CONNECTOR 2`

: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. See :DOUT:PREemphasis for more information.

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 input 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	Setting	Meaning
XLR (bal)	XLR	Front panel XLR digital input connector, balanced
BNC (unbal)	BNC	Front panel BNC digital input connector, unbalanced
Optical	OPTical	Front panel Toslink optical input connector
Gen Mon	GENMon	Digital generator XLR or BNC output connector
Dual XLR	XLRDual	Front panel dual XLR (balanced) digital input

Note that if the format is Optical the :DIN:PEAK? query will return a NAN because the input level is not calibrated for optical inputs.

Dual Connector Mode

Typically the alternate data subframes in the AES3 signal carry the left and right audio channels. In the Dual XLR mode, each AES3 interface signal carries a monaural signal whose sub-frame rate is double the frame rate. Two connections (I and II) are required for 2-channel audio. Normally, Connector I carries the “A” or left audio channel and Connector II carries the “B” or right audio channel.

The command argument is:

- **source** - { BNC | GENMon | OPTical | XLR | XLRDual }

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), and Dual XLR (XLRDual).

Response argument(s):

- **source** - { BNC | GENMon | OPTical | XLR | XLRDual }

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 XLR settings (XLR and XLRDual) share the same input termination setting (either 110 ohm or high impedance), and the BNC has its own input termination setting (either 75 ohm or high impedance).

Connector	Input Termination(s)
XLR	110 ohms or high impedance
BNC	75 ohms or 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> (range: 0 or 1)

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 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.

The query argument is:

- **unit** - { SEC | UI }

Response argument(s):

- **jitter** - <nrf> - { SEC | UI }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DIN:SETTling

Command Syntax: :DIN:JITTer? **unit**

Response Syntax: :DIN:JITTer **jitter, settle_timeout**

Example: `:DIN:JITTER? UI`

Response: `:DIN:JITTER 3.5UI,0`

:DIN:LOCK?

Returns the state of the locked error flag of the digital input. 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> (range: 0 or 1)

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 changed between logical one and zero during the most recent measurement interval (approximately 1/2 second). 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.

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: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> (range: 0 or 1)

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 to detect possible signal clipping.

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. The :DIN:REF setting is used as the reference for all units except HZ.

The first parameter in the response is the measurement.

The Digital Input Sample Rate 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 response 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> - { CENT | DECS | DHZ | DPCT | DPPM | F_R | HZ | OCTS | PCTHz }
- **settle_timeout** - <nr1> (range: 0 or 1)

Related Commands: :DIN:REF, :SETTling:DIN

Command Syntax: :DIN:RATE? **unit**

Response Syntax: :DIN:RATE **rate, settle_timeout**

Example: `:DIN:RATE? DPPM`

Response: `:DIN:RATE 158DPPM,0`

:DIN:REF

Specifies the reference frequency used for the :DIN:RATE? query and for scaling audio frequency measurements when the :DIN:SCALefreqby setting is REF. The value is in units of hertz.

The command argument is:

- **freq** - <nrf> (range: ≥ 28800, ≤ 216000)

Default: 48000

Related Commands: :DIN:RATE, :DIN:SCALefreqby, :DIN:REF?

Command Syntax: :DIN:REF **freq**

Example: `:DIN:REF 44100`

:DIN:REF?

Returns the current digital input frequency reference value. The response units are hertz.

Response argument(s):

- **freq** - <nrf> (range: ≥ 28800, ≤ 216000)

Related Commands: :DIN:REF

Command Syntax: :DIN:REF?

Response Syntax: :DIN:REF **freq**

Example: `:DIN:REF?`

Response: `:DIN:REF 44100`

:DIN:RESolution

Specifies digital input data resolution in bits or a fixed resolution and companding mode. The first parameter of this command specifies digital input data resolution in bits if the second parameter is 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.

If the second argument is ALAW, then the input word will be set to a fixed resolution of 13 bits and A-Law companded from 8 bits at the digital input. The first parameter will be ignored.

If the second argument is ULAW, then the input word will be set to a fixed resolution of 14 bits and (-Law companded from 8 bits at the digital input. The first parameter will be ignored.

The command argument (in bits) is:

- **bits** - <nr1> (range: ≥ 8, ≤ 24)
- **mode** - { ALAW | BITS | ULAW }

Default: 24

Related Commands: :DIN:BITS?, :DIN:RESolution?

Command Syntax: :DIN:RESolution **bits, mode**

Example: `:DIN:RESOLUTION 20,BITS`

:DIN:RESolution?

This query returns the resolution setting of the imbedded digital audio signal and the companding mode. If the second parameter is BITS then the first parameter is the bits of resolution of the input digital signal, otherwise the first parameter is irrelevent.

Response argument(s):

- **bits** - <nr1> (range: ≥ 8, ≤ 24)
- **mode** - { ALAW | BITS | ULAW }

Related Commands: :DIN:RESolution

Command Syntax: :DIN:RESolution?

Response Syntax: :DIN:RESolution **bits, mode**

Example: `:DIN:RESOLUTION?`

Response: `:DIN:RESOLUTION 20,BITS`

: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

follow any changes in sample rate of the input source. The Output Rate selection can be used to measure the frequency-shifting 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?

Returns 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:**settings**

Example: `:DIN:SET?`

Response: `:DIN:REF 48000;BANDWIDTH 700;RESOLUTION 24,BITS;DETECTOR AVG;MODE ACTIVE;PKMODE HALF;SCALEFREQBY MEASURED;DEEMPHASIS OFF;FORMAT XLR;IMPEDANCE BNC,Z75;IMPEDANCE XLR,Z110;CONNECTOR 1`

Chapter 12

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:.

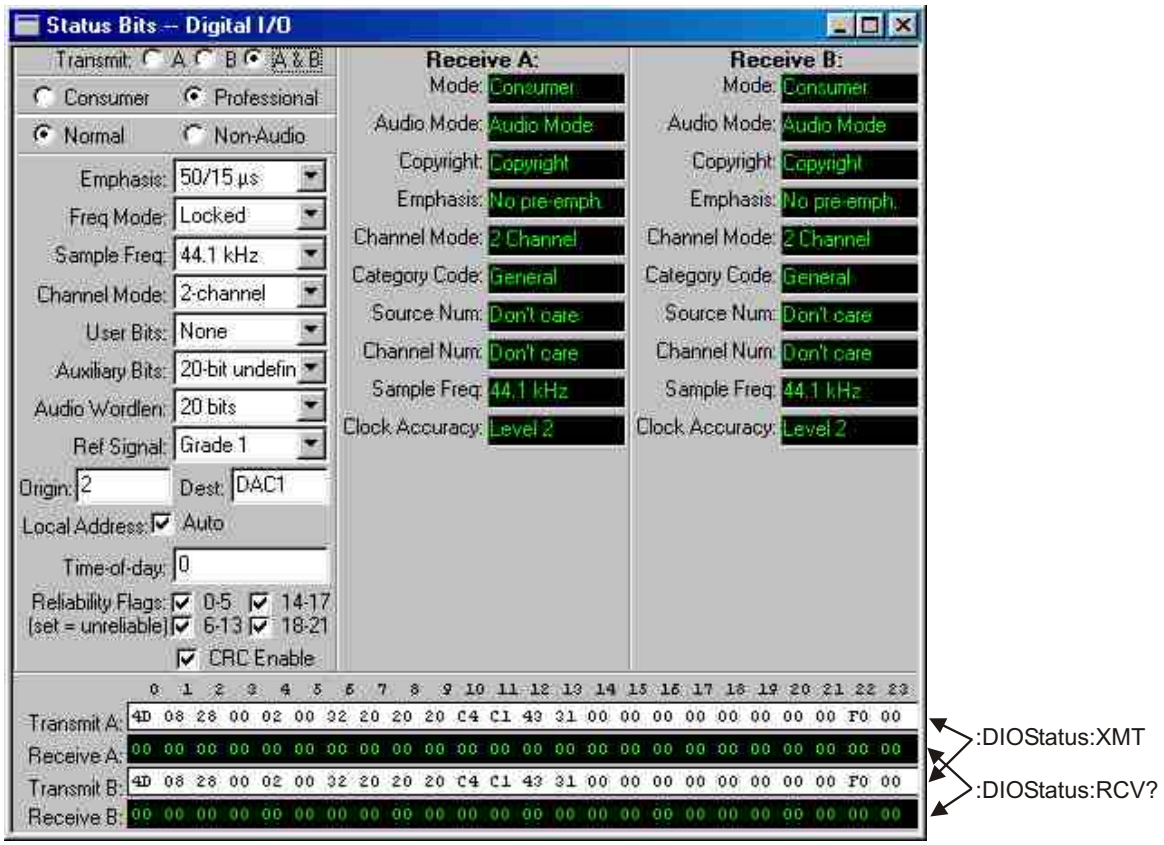


Figure 12-1. ATS software Status Bits - DIO control panel DIOSTATUS GPIB commands.

:DIOStatus:RCV?

This query returns the 192 bits of data for either Channel A or B AES/EBU 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,00,02,00,41,50,53,32,41,50,53,32,8
0,F6,8B,00,D2,04,00,00,00,7E"
```

:DIOStatus:XMT

This command sets the 192 AES/EBU 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 Transmit section of ATS-2 Status Bits panel after setting the bits with the high level controls. This technique simplifies the process of encoding the hex bytes by using the ATS control software 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
,00,00,00,00,00,00,00,00,00,00,00,00";B,"04,00,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
,00,00,00,00,00,00"
```

Related Commands: :DIOStatus:XMT?

Command Syntax: :DIOStatus:XMT **channel, data**

Example:
```
:DIOSTATUS:XMT
A,"8D,82,2C,00,02,00,41,50,53,32,41,50,53,32
,80,F6,8B,00,D2,04,00,00,00,7E"
```

:DIOStatus:XMT?

This query returns the current setting of the AES/EBU 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,
41,50,53,32,80,F6,8B,00,D2,04,00,00,00,7E"
```

Chapter 13
Sync/Ref Commands

The header-path for the Sync/Reference commands is :SYNC:.

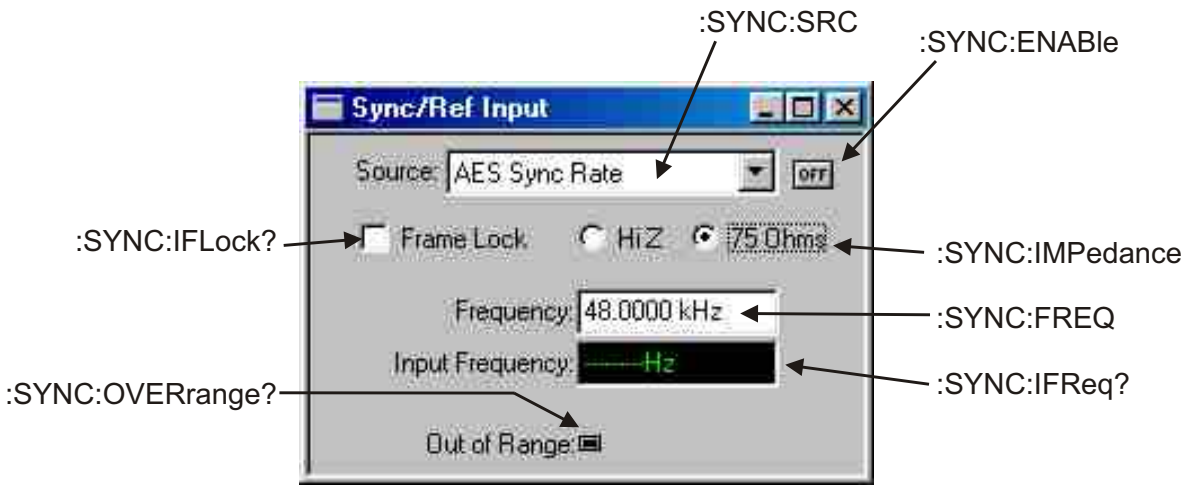


Figure 13-1. ATS software Sync/Ref Input control panel SYNC GPIB commands.

: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 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 ATS-2 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 maximum range is dependent on the Sync Source setting; 108E+3 if the input source is AES-EBU or NTSC or PAL, 25.6E+6 if source is Squarewave input.

Units are hertz.

The command argument is:

- **frequency** - <nrf> (range: ≥ 8000, ≤ 1E+34 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 Reference Frequency setting.

Response argument(s):

- **frequency** - <nrf> (range: ≥ 8000, ≤ 25.6E +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:IFLock

If the :SYNC:SRC command is set to AESebu, then this command causes the ATS-2 digital output frames to be synchronized with both sample rate and frame (Sync Reference Input Frame Lock) of the digital signal connected to the rear panel AES Reference input.

The command argument is:

- **lock** - { OFF | ON }

Default: OFF

Related Commands: :SYNC:IFLock?, :SYNC:SRC

Command Syntax: :SYNC:IFLock **lock**

Example:
```
:SYNC:IFLOCK ON
```

:SYNC:IFLock?

This query returns the setting for the Sync Reference Input Frame Lock for the sync reference input signal connected to the rear panel AES Reference input if the :SYNC:SRC command is set to AESebu.

Response argument(s):

- **lock** - { OFF | ON }

Related Commands: :SYNC:IFLock, :SYNC:SRC

Command Syntax: :SYNC:IFLock?

Response Syntax: :SYNC:IFLock **lock**

Example:
```
:SYNC:IFLOCK?
```

Response:
```
:SYNC:IFLOCK ON
```

:SYNC:IFReq?

This query returns the measured frequency (Hz) of the AES/EBU digital input signal connected to the SYNC/REF IN BNC connector if :SYNC:SRC is set to OFF.

If :SYNC:SRC is not OFF this query will return 9.91E+37, the Not-A-Number number. The response value is in units of hertz.

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 IN BNC connector.

The ZLO argument sets the input impedance to 75 Ohms.

The ZHI argument sets the input impedance to > 5K 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 IN BNC connector.

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:OVERrange?

This query returns a 1 if the actual Sync Input signal frequency (:SYNC:IFReq?) is greater than +/-15 ppm from the Sync Frequency (:SYNC:FREQ), or outside the amplitude range required for reliable operation. A 0 is returned if the sync/ref input circuit is locked to the sync/ref input signal.

This measurement may require several seconds to indicate an out of range condition or a locked condition.

The possible responses are 1 (overange) and 0 (locked).

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:**settings**

Example:
```
:SYNC:SET?
```

Response:
```
:SYNC:ENABLE OFF;SRC AESEBU;IMPEDANCE
ZLO;FREQ 48000;IFLOCK OFF
```

:SYNC:SRC

This command specifies the input for synchronizing the internal sample rate clock to an external reference signal. The input choices are shown in the table below. Refer to the ATS-2 User's Manual for more information.

AES Sync Rate (AESebu)

This setting will synchronize the ATS-2 master clock to the sample rate of a properly formatted AES3 or IEC60958 signal applied to the SYNC/REF IN BNC.This synchronizes the sample rates, but does not ensure that the signal frames are aligned in time. To align the frames, the ATS-2 digital output signal must be set to the same sample rate as the reference signal and :SYNC:IFLock must be set to ON.

Squarewave (SQUare)

This setting will synchronize the ATS-2 master clock to a TTL squarewave signal or other repetitive signal applied to the SYNC/REF IN BNC. This signal may be between 8 kHz to 10 MHz. A bit clock output from a digital device applied to the SYNC/REF IN connection in Squarewave mode will synchronize the ATS-2 sample rate to the device sample rate. See Squarewave frequency ranging, below.

NTSC Video Sync Horiz Rate (NTSC)

This setting will synchronize the ATS-2 master clock to the horizontal sync pulse (nominally 15.7343 kHz) of an NTSC analog video signal applied to the SYNC/REF IN BNC.

PAL / SECAM Video Sync Horiz Rate (PAL)

This setting will synchronize the ATS-2 master clock to the horizontal sync pulse (nominally 15.6250 kHz) of a PAL / SECAM analog video signal applied to the SYNC/REF IN BNC.

Squarewave frequency ranging

To cover a wide frequency range, ATS-2 performs automatic frequency range switching when Squarewave is selected as the Sync/Ref input source.

Due to this ranging, the measured input frequency reading can be in error if the input frequency differs by a large amount from the nominal frequency entered on the Sync/Ref panel. Use :SYNC:FREQ to set a nominal frequency close to the actual input frequency.

If you do not know the squarewave input frequency, set a nominal frequency of 8 kHz. The measured frequency (:SYNC:IFREQ?) will be correct, although with slightly reduced accuracy. Now use :SYNC:FREQ to set the nominal frequency to this measured frequency, and then the new measured frequency (:SYNC:IFREQ?) will be correct, with full accuracy.

The command argument is:

- **signal** - { AESebu | NTSC | PAL | SQUare }

Default: AESebu

Related Commands: :SYNC:SRC?, :SYNC:ENABle, :SYNC:IFREQ?, :SYNC:FREQ

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`

Chapter 14
Trigger Commands

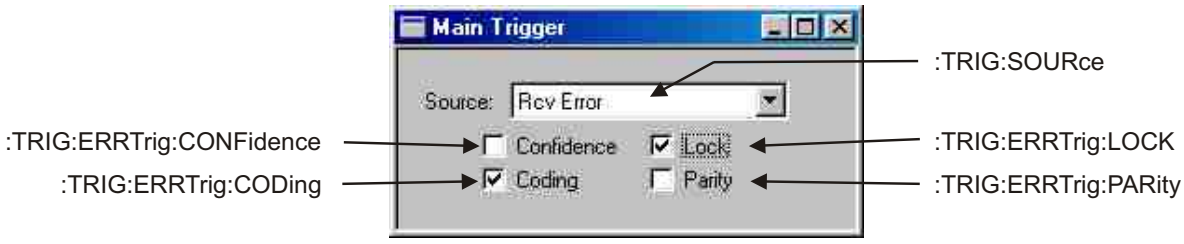


Figure 14-1. ATS software Main Trigger control panel TRIG GPIB commands.

:TRIG: Compound Command Header

The ATS-2 trigger output signal appears at the TRIG OUT BNC connector on the rear panel. This signal is intended to trigger an oscilloscope or other monitoring device at the occurrence of a specific condition in the course of measurement. These selections are identical to the trigger choices to the Digital Interface Analyzer (Intervu) trigger sources. Settings control the same functions interactively.

The Trigger Output is a 5 V TTL/CMOS logic signal with a 50 ohm source impedance. The rising edge of the signal transition is alway the trigger instant.

:TRIG:ERRTrig Compound Command Header

These commands enable trigger generation for data coding errors, data confidence errors, data parity errors, or interface lock errors when the Intervu trigger (:TRIG:SOURce) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

:TRIG:ERRTrig:CODing

This command enables trigger generation when a data coding error occurs if the trigger (:TRIG:SOURce) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

The command argument is:

- **error** - { OFF | ON }

Default: ON

Related Commands: :TRIG:ERRTrig:CODing?, :TRIG:SOURce

Command Syntax: :TRIG:ERRTrig:CODing **error**

Example: `:TRIG:ERRTRIG:CODING OFF`

:TRIG:ERRTrig:CODing?

This query returns the Coding Error Trigger setting.

Response Argument(s):

- **error** - { OFF | ON }

Related Commands: :TRIG:ERRTrig:CODing, :TRIG:SOURce

Command Syntax: :TRIG:ERRTrig:CODing?

Response Syntax: :TRIG:ERRTrig:CODing **error**

Example: `:TRIG:ERRTRIG:CODING?`

Response: `:TRIG:ERRTRIG:CODING OFF`

:TRIG:ERRTrig:CONFidence

This command enables trigger generation when a data confidence error occurs if the trigger (:TRIG:SOURce) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

The command argument is:

- **error** - { OFF | ON }

Default: ON

Related Commands: :TRIG:ERRTrig:CONFidence?, :TRIG:SOURce

Command Syntax: :TRIG:ERRTrig:CONFidence **error**

Example: `:TRIG:ERRTRIG:CONFIDENCE OFF`

:TRIG:ERRTrig:CONFidence?

This query returns the Confidence Error Triggering setting.

Response Argument(s):

- **error** - { OFF | ON }

Related Commands: :TRIG:ERRTrig:CONFidence

Command Syntax: :TRIG:ERRTrig:CONFidence?

Response Syntax: :TRIG:ERRTrig:CONFidence **error**

Example: `:TRIG:ERRTRIG:CONFIDENCE?`

Response: `:TRIG:ERRTRIG:CONFIDENCE OFF`

:TRIG:ERRTrig:LOCK

This command enables trigger generation when an interface lock error occurs if the trigger (:TRIG:SOURce) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

The command argument is:

- **error** - { OFF | ON }

Default: ON

Related Commands: :TRIG:ERRTrig:LOCK?, :TRIG:SOURce

Command Syntax: :TRIG:ERRTrig:LOCK **error**

Example: `:TRIG:ERRTRIG:LOCK OFF`

:TRIG:ERRTrig:LOCK?

This query returns the Lock Error Triggering setting.

Response Argument(s):

- **error** - { OFF | ON }

Related Commands: :TRIG:ERRTrig:LOCK, :TRIG:SOURce

Command Syntax: :TRIG:ERRTrig:LOCK?

Response Syntax: :TRIG:ERRTrig:LOCK **error**

Example: `:TRIG:ERRTRIG:LOCK?`

Response: `:TRIG:ERRTRIG:LOCK OFF`

:TRIG:ERRTrig:PARity

This command enables trigger generation when a data parity error occurs if the trigger (:TRIG:SOURce) is set to Sync Error (ESYNc) or Receive Error (RCVerr).

The command argument is:

- **error** - { OFF | ON }

Default: ON

Related Commands: :TRIG:ERRTrig:PARity?, :TRIG:SOURce

Command Syntax: :TRIG:ERRTrig:PARity **error**

Example: `:TRIG:ERRTRIG:PARITY OFF`

:TRIG:ERRTrig:PARity?

This query returns the Parity Error Triggering setting.

Response Argument(s):

- **error** - { OFF | ON }

Related Commands: :TRIG:ERRTrig:PARity

Command Syntax: :TRIG:ERRTrig:PARity?

Response Syntax: :TRIG:ERRTrig:PARity **error**

Example: `:TRIG:ERRTRIG:PARITY?`

Response: `:TRIG:ERRTRIG:PARITY OFF`

:TRIG:SOURce

This command selects a trigger source.

The ATS-2 has an extensive trigger source list with the additional capability of generating triggers on up to four error conditions from either the digital input or the SYNC/REF IN BNC connector.

The various trigger source choices include several points in the digital interface from the received digital input, the transmitted digital output or the Sync/Ref received input; from an external clock reference, the AC mains frequency and internal generators. The trigger from the digital interface signals can be extracted from the X- and Z-preambles (Channel A subframe), the Y-preambles (Channel B subframe) or the “block” Z-preambles (Block 192 frames).

On the various preamble trigger sources, the trigger operation is such that the trailing edge of the first 3-UI pulse of the preamble occurs nominally at time zero. The first information displayed after time zero in these cases will be the remaining 5 UIs of the selected preamble, followed by the first bit in the data area. Depending on the nature of the interface signal, that could be the LSB of the audio signal if full 24-bit resolution audio is transmitted, or the beginning of the 4-bit Auxiliary data area if audio is restricted to 20 bits or fewer.

The transmit preamble selections have the same triggering characteristics. This triggering selection permits measurement of time delay through a digital device or system under test.

The Transmit and Receive Block selections cause signal to be acquired at the first Channel Status Block Preamble transmitted or received. The Receive Block is delayed by two full frames by the AES receiver.

The Jitter Generator signal triggers at every cycle of the sinewave signal generated by the digital output jitter generator. This selection provides a stable acquisition of the received jitter waveform when measuring through a digital device.

Triggering for squarewave acquisitions

None of the serial digital interface trigger sources (receive and transmit block, error and sub-frame sources) are useful

when measuring squarewave jitter with the Digital Interface Analyzer (INTervu). For stable triggering on a squarewave signal, split the connection with a BNC "T" adapter and connect the two resultant lines to the ATS-2 DIGITAL INPUT and the TRIG IN rear panel connection. Select Trig In (EXTernal) as the Trigger Source. For correct voltage readings, terminate only one of the two inputs.

The Jitter Generator trigger selection works only when Sinewave Jitter is turned on with the :DOUT:JWFM command. This trigger mode can be useful when looking at jitter on a squarewave clock that is derived from an AES3 signal fed from ATS-2's digital generator output.

The Trigger choices are:

- Analog Generator Audio Signal (AGEN),
- Digital Generator Audio Signal (DGEN),
- Jitter Generator (JITTer),
- External Trigger Input (EXTernal),
- AC Line Mains (ACMains),
- Ch. A Receive Sub-Frame (Preamble) (ARCV),
- Ch. B Receive Sub-Frame (Preamble) (BRCV),
- Ch. A Receive Sub-Frame DeJitt (Preamble) (DJARcv),
- Ch. B Receive Sub-Frame (Preamble) (DJBRcv),
- Receive Block (192 frames) (RCVBlock),
- Receive Error (RCVerr),
- Ch. A Transmit Sub-Frame (Preamble) (AXMT),
- Ch. B Transmit Sub-Frame (Preamble) (BXMT),
- Ch. A Transmit Sub-Frame DeJitt (Preamble) (DJAXmt),
- Ch. B Transmit Sub-Frame DeJitt (Preamble) (DJBXmt),
- Transmit Block (XMTBlock),
- Ch A Sync/Ref Receive Sub-Frame (Preamble) (ASYNc),
- Ch B Sync/Ref Receive Sub-Frame (Preamble) (BSYNc),
- Sync/Ref Rcv Block (192 frames) (BLKSync),
- Sync/Ref Rcv Error (ESYNc)

Note that the Receive choices pertain to the signal at the I or II front-panel input connector selected by the :DIN:CONNector command if the input is XLR (see :DIN:FORMat command).

Analog Gen (AGEN)

A trigger transition is generated for each cycle of the current Analog Generator waveform.

Digital Gen (DGEN)

A trigger transition is generated for each cycle of the current Digital Generator waveform.

Jitter Gen (JITTer)

A trigger transition is generated for each cycle of the current Jitter Generator waveform.

Trig In (EXTernal)

A trigger transition is generated for each external trigger transition received at the TRIG IN BNC connector.

Line (ACMains)

A trigger transition is generated for each cycle of the AC mains line voltage.

ChA Rcv Sub-Frame (ARCV)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal received at the digital input. If there is jitter in the received waverform, that jitter will affect the trigger.

ChB Rcv Sub-Frame (BRCV)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal received at the digital input. If there is jitter in the received waverform, that jitter will affect the trigger.

ChA Rcv Sub-Frame DeJitt (DJARcv)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal received at the digital input, after the signal has be re-clocked to remove jitter.

ChB Rcv Sub-Frame DeJitt (DJBRcv)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal received at the digital input, after the signal has be re-clocked to remove jitter.

Rcv Block (192 frames) (RCVBlock)

A trigger transition is generated at the beginning of each Status Block frame (each Z-preamble) in the interface signal received at the digital input. This occurs once every 192 frames.

Rcv Error (RCVerr)

Four “receive error” trigger conditions in the interface waveform are defined. See :TRIG:ERRTrig. When any of these selected conditions occur in the interface signal received at the the main digital input, a trigger transition will be generated.

ChA Xmit Subframe (AXMT)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal transmitted at the digital output. If jitter has been added to the digital output, that jitter will affect the trigger.

ChB Xmit Subframe (BXMT)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal transmitted at the digital output. If jitter has been added to the digital output, that jitter will affect the trigger.

ChA Xmit Subframe DeJitt (DJAXmt)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal transmitted at the digital output, before the addition of any jitter.

ChB Xmit Subframe DeJitt (DJBXmt)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal transmitted at the digital output, before the addition of any jitter.

Xmit Block (192 frames) (XMTBlock)

A trigger transition is generated at the beginning of each Status Block frame (each Z-preamble) in the interface signal transmitted at the digital output. This occurs once every 192 frames.

ChA Sync/Ref Rcv Sub-Frame (ASYNc)

A trigger transition is generated at the beginning of each Channel A subframe (each X-preamble plus each Z-preamble) in the interface signal received at the SYNC/REF IN input.

ChB Sync/Ref Rcv Sub-Frame (BSYNc)

A trigger transition is generated at the beginning of each Channel B subframe (each Y-preamble) in the interface signal received at the SYNC/REF IN input.

Sync/Ref Rcv Block (192 frames) (BLKSync)

A trigger transition is generated at the beginning of each Status Block frame (each Z-preamble) in the interface signal received at the SYNC/REF IN input. This occurs once every 192 frames.

Sync/Ref Rcv Error (ESYNc)

Four “receive error” conditions in the interface waveform are defined and flagged. See :TRIG:ERR commands. When any of these trigger conditions occur in the interface signal received at the SYNC/REF input, a trigger transition will be generated.

This command will change the setting of the :TRIG:INTervu:TRIGger command (both commands change the same hardware trigger circuitry). Changing the :DSP:INTervu:TRIGger command will make an identical change to this command.

The command argument is:

- **type** - { ACMains | AGEN | ARCV | ASYNc | AXMT | BLKSync | BRCV | BSYNc | BXMT | DGEN | DJARcv | DJAXmt | DJBRcv | DJBXmt | ESYNc | EXTernal | JITTer | RCVBlock | RCVerr | XMTBlock }

Default: ARCV

Related Commands: :TRIG:SOURce?

Command Syntax: :TRIG:SOURce **type**

Example: `:TRIG:SOURce ARCV`

:TRIG:SOURce?

This query returns the interface condition which triggers the Intervu acquisition of the digital interface signal. See :TRIG:SOURce for a more detailed explanation.

Response argument(s):

- **type** - { ACMains | AGEN | ARCV | ASYNc | AXMT | BLKSync | BRCV | BSYNc | BXMT | DGEN | DJARcv | DJAXmt | DJBRcv | DJBXmt | ESYNc | EXTernal | JITTer | RCVBlock | RCVerr | XMTBlock }

Related Commands: :TRIG:SOURce

Command Syntax: :TRIG:SOURce?

Response Syntax: :TRIG:SOURce **type**

Example: `:TRIG:SOURCE?`

Response: `:TRIG:SOURCE ARCV`

:TRIG:SET?

This query returns all trigger input settings.

Response argument(s):

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

Related Commands: (see all other Trigger setting commands)

Command Syntax: :TRIG:SET?

Example: :TRIG:SET?

Response: :TRIG:ERRTRIG:CODING ON;CONFIDENCE ON;LOCK ON;PARITY ON;:TRIG:SOURCE ARCV

Chapter 15
Auxiliary Control Commands

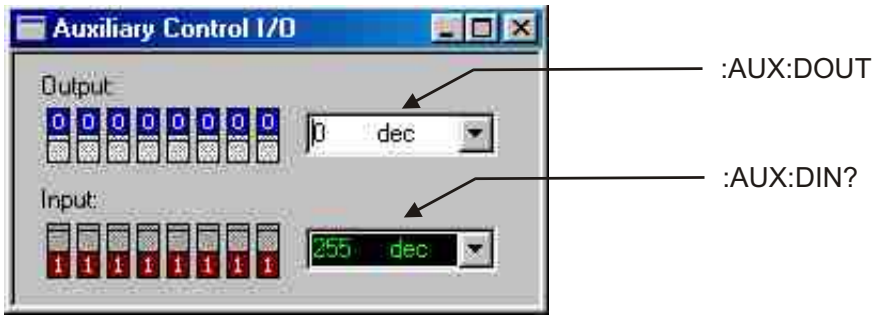


Figure 15-1. ATS software Auxiliary Control I/O control panel AUX GPIB commands.

:AUX:DOUT

This command sets the output on auxiliary 8-bit digital control port. The range of values is 0 through 255 (decimal). The example sets all of the control pins high.

The command arguments are:

- **setting** - <nr1> (range: 0, 255)

Default: 0

Related Commands: :AUX:DOUT?

Command Syntax: :AUX:DOUT **setting**

Example: `:AUX:DOUT 255`

:AUX:DOUT?

This query returns the state of the auxiliary digital control output port (in decimal).

Response argument(s):

- **setting** - <nr1>

Related Commands: :AUX:DOUT

Command Syntax: :AUX:DOUT?

Response Syntax: :AUX:DOUT **setting**

Example: `:AUX:DOUT?`

Response: `:AUX:DOUT 255`

:AUX:DIN?

This query returns the 8 bit digital word at the auxiliary digital input port (decimal).

Response argument(s):

- **setting** - <nr1>

Related Commands: :AUX:DOUT

Command Syntax: :AUX:DIN?

Response Syntax: :AUX:DIN **setting**

Example: `:AUX:DIN?`

Response: `:AUX:DIN 255`

:AUX:SET?

This query returns the current setting of all the AUX parameters.

Response argument(s):

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

Related Commands: :AUX:DOUT?

Command Syntax: :AUX:SET?

Response Syntax: :AUX:**settings**

Example: `:AUX:SET?`

Response: `:AUX:DOUT 255`

Chapter 16

Monitor Headphone/Speaker Commands

Monitor commands apply to the internal speaker and to the monitor jack on the rear panel of ATS-2. The header-path for the speaker/headphone commands is :MON:

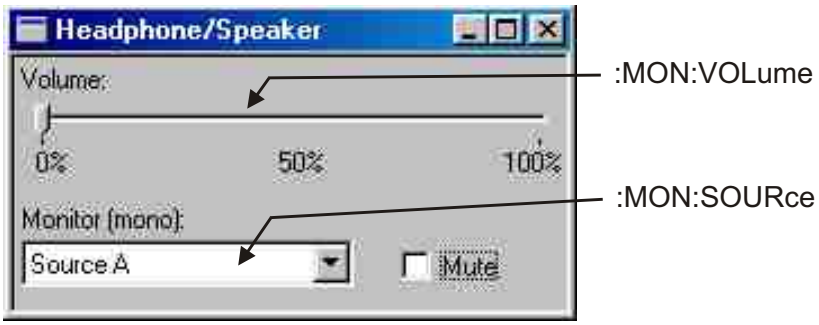


Figure 16-1. ATS software Headphone / Speaker control panel MON GPIB commands.

:MON:SOURce

This command specifies the instrument channel input for the headphone/speaker monitor.

Audio monitor source choices:

Source A (AINPut)
This is the audio signal as it appears at the input to the Analyzer instrument in use, after ranging--the point where the Channel A Level or Peak meter readings are taken. If the INTERVU DSP program is loaded, the monitored signal is the Channel A embedded audio.

Source B (BINPut)
This is the audio signal as it appears at the input to the Analyzer instrument in use, after ranging--the point where the Channel B Level or Peak meter readings are taken. If INTERVU is the instrument, the monitored signal is the Channel B embedded audio.

Function A (AFUNc)
The channel A signal as it appears after a given Audio Analyzer function is applied. Signal is only available at this point when using the Audio Analyzer DSP program.

Function B (BFUNc)
The channel B signal as it appears after a given Audio Analyzer function is applied. Signal is only available at this point when using the Audio Analyzer DSP program.

Source A + Source B (ABINputsum)
The sum of these signals.

Function A + Function B (ABFuncsum)
The sum of these signals.

Source A + Function A (ASUM)
The sum of these signals.

Source B + Function B (BSUM)
The sum of these signals.

The command argument is:

- **source** - { ABINputsum | ABFuncsum | AFUNc | AINPut | ASUM | BFUNc | BINPut | BSUM }

Default: AINPut

Related Commands: :MON:SOURce?, :MON:VOLume

Command Syntax: :MON:SOURce **source**

Example: `:MON:SOURCE AFUNC`

:MON:SOURce?

This command returns the instrument channel source for the headphone/speaker monitor signal.

Response argument(s):

- **source** - { ABINputsum | ABFuncsum | AFUNc | AINPut | ASUM | BFUNc | BINPut | BSUM }

Related Commands: :MON:SOURce, :MON:VOLume, :MON:MUTe

Command Syntax: :MON:SOURce?

Response Syntax: :MON:SOURce **source**

Example: `:MON:SOURce AFUNC`

:MON:SET?

This query returns settings of the headphone/speaker monitor. The settings reported are: mute, source, and volume.

Response argument(s):

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

Related Commands: :MON:SOURCE?, :MON:VOLume?

Command Syntax: :MON:SET?

Response Syntax: :MON:**settings**

Example: `:MON:SET?`

Response: `:MON:SOURCE AFUNC;VOLUME 70`

:MON:VOLume

This command sets the headphone/speaker volume. The <nr1> integer argument represents the volume in percent of full scale range from 0 to 100.

Set VOLume to 0 to mute the output.

The command argument is:

- **volume** - <nr1> (range: 0, 100)

Default: 0

Related Commands: :MON:VOLume?

Command Syntax: :MON:VOLume **volume**

Example: `:MON:VOLUME 70`

:MON:VOLume?

This query returns the headphone/speaker volume setting. The <nr1> integer response argument represents the volume in percent of full scale range from 0 to 100.

Response argument(s):

- **volume** - <nr1> (range: 0, 100)

Related Commands: :MON:VOLume

Command Syntax: :MON:VOLume?

Response Syntax: :MON:VOLume **volume**

Example: `:MON:VOLUME?`

Response: `:MON:VOLUME 70`

Chapter 17
Settling Commands

Most measurements provided by ATS-2 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 ATS-2. 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 the figure below, <meter?> means any meter query, for example :DSP:DANLR:LEVel? A,V.

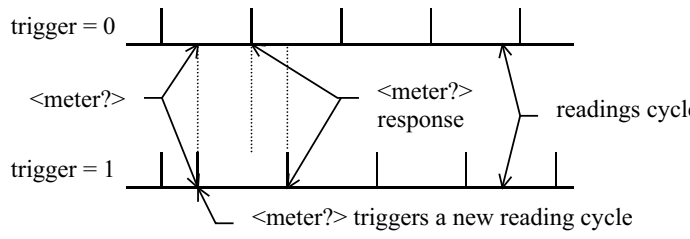


Figure 17-1. Settling illustration.

See Chapter 2 of this manual for a more complete discussion of settling.

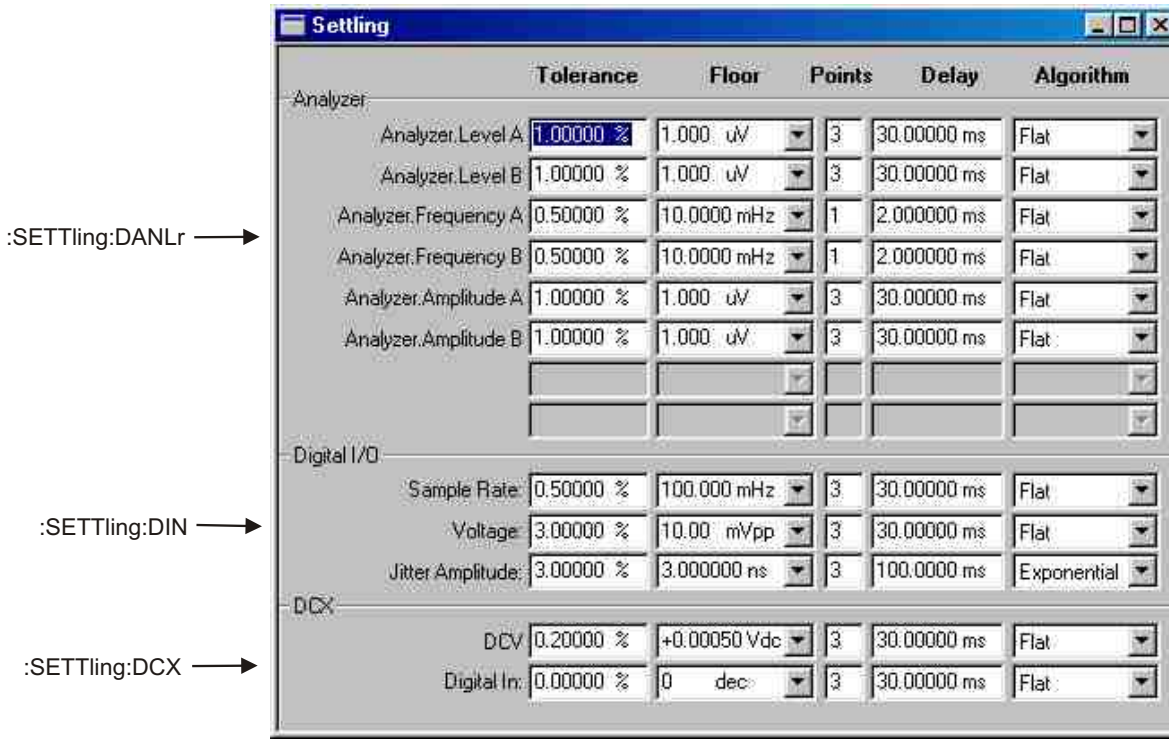


Figure 17-2. ATS software Settling control panel SETTLING GPIB commands.

:SETTling:DANLr Compound Command Header

Compound header for Audio Analyzer settling commands. The Audio Analyzer has separate settling commands (and queries) for the frequency meter, the level meter and the function meter.

: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.

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 first argument specifies the channel, either channel A or channel B.

The choices for the second 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),
- PHASe (phase mode, same settings are used for analog and digital input),
- SMPTe (SMPTE/DIN, same settings are used for analog and 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 third 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 channel a with amplitude mode with analog domain input using the FastRMS detector mode. A different settling parameter setting could be specified (and stored) for channel a with amplitude mode with analog domain input using a normal detector mode. Yet another set of settling parameters could be specified for channel a with amplitude mode with digital input using the FastRMS detector.

Command	Param 1 Channel	Param 2 (Meter)	Meter/ Mode	Input	Param 3 (Detector)
:SETTling:DANLr:FUNC	*A*	AMPA	*Ampl*	*Analog*	NORMal
	B				
	A				FRMS
	B				
	A	AMPD		*Digital*	NORMal
	B				
	A				FRMS
	B				
	A	THDA	*THD+n Ampl*	*Analog*	NORMal
	B				
	A				FRMS
	B				
	A	THDD		*Digital*	NORMal
	B				
	A				FRMS
	B				
	A	BPA	*Bandpass*	*Analog*	NORMal
	B				
	A				FRMS
	B				
	A	BPD		*Digital*	NORMal
	B				
	A				FRMS
	B				
	A	PHASe	*Phase*		NORMal
	B				
	A				FRMS
	B				
	A	SMPTe	*SMPTE/DIN*		NORMal
	B				
	A				FRMS
	B				
	A	THDRatio	*THD+N Ratio*		NORMal
	B				
	A				FRMS
	B				
	A	RATio	*2-Chan Ratio*		NORMal
	B				
	A				FRMS
	B				
	A	XTALk	*Crosstalk*		NORMal
	B				
	A				FRMS
	B				

The following table details the valid units for the floor parameter for each function meter mode.

Meter/Mode	Input	Units
Amplitude (AMPA), Bandpass (BPA), THD+N Amplitude (THDA)	Analog	V (V), dBm (DBM), dBV (DBV), dBu (DBU), dBr A (DBRA), dBr B (DBRB), dBg A ()DBGA), dBg B (DBGB), Watts (W), FFS (FFS), %FS (PCTFs), dBFS (DBFS)
Amplitude (AMPD), Bandpass (BPD), THD+N Amplitude (THDD)	Digital	FFS (FFS), %FS (PCTFs), dBFS (DBFS), Bits (BITS), V (V), dBu (DBU), dBV (DBV), dBr 1 (DBR1), dBr 2 (DBR2)
2-Ch Ratio (RATio), Crosstalk (XTALk), THD+N Ratio (THDRatio), SMPTE/DIN (SMPTe)	Analog or Digital	dB (DB), % (PCT), PPM (PPM), X/Y (X_Y)
Phase (PHASe)	Analog or Digital	Degrees (DEG)

The command arguments are:

- **channel** - { A | B }
- **meter** - { AMPA | AMPD | BPA | BPD | PHASe | RATio | SMPTe | 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) { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | DEG | FFS | PCT | PCTFs | 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 (max. is approximately 24.855 days)
- **trigger** - <nr1> (range: 0 or 1)

Default: A,AMPA,FRMS,1,1E-6 V,1,0.001,FLAT,0,1

B,AMPA,FRMS,1,1E-6 V,1,0.001,FLAT,0,1

A,AMPA,NORM,1,1E-6V,3,0.03,FLAT,0,1

B,AMPA,NORM,1,1E-6V,3,0.03,FLAT,0,1

A,AMPD,FRMS,1,1E-7FFS,1,0.001,FLAT,0,1

B,AMPD,FRMS,1,1E-7FFS,1,0.001,FLAT,0,1

A,AMPD,NORM,1,1E-6FFS,3,0.03,FLAT,0,1

B,AMPD,NORM,1,1E-6FFS,3,0.03,FLAT,0,1

A,BPA,FRMS,3,1E-8V,2,0.02,EXP,0,1

B,BPA,FRMS,3,1E-8V,2,0.02,EXP,0,1

A,BPA,NORM,3,1E-8V,3,0.1,EXP,0,1

B,BPA,NORM,3,1E-8V,3,0.1,EXP,0,1

A,BPD,FRMS,3,1E-8FFS,2,0.02,EXP,0,1
B,BPD,FRMS,3,1E-8FFS,2,0.02,EXP,0,1
A,BPD,NORM,3,1E-8FFS,3,0.1,EXP,0,1
B,BPD,NORM,3,1E-8FFS,3,0.1,EXP,0,1
A,PHASe,NORM,0,0.2DEG,2,0.02,FLAT,0,1
B,PHASe,NORM,0,0.2DEG,2,0.02,FLAT,0,1
A,PHASe,FRMS,0,0.2DEG,2,0.02,FLAT,0,1
B,PHASe,FRMS,0,0.2DEG,2,0.02,FLAT,0,1
A,THDA,FRMS,3,1E-7V,2,0.02,FLAT,0,1
B,THDA,FRMS,3,1E-7V,2,0.02,FLAT,0,1
A,THDA,NORM,3,1E-7V,3,0.1,EXP,0,1
B,THDA,NORM,3,1E-7V,3,0.1,EXP,0,1
A,THDD,FRMS,3,1E-7FFS,2,0.02,FLAT,0,1
B,THDD,FRMS,3,1E-7FFS,2,0.02,FLAT,0,1
A,THDD,NORM,3,1E-7FFS,3,0.1,EXP,0,1
B,THDD,NORM,3,1E-7FFS,3,0.1,EXP,0,1
A,THDRatio,NORM,3,1E-5PCT,3,0.1,EXP,0,1
B,THDRatio,NORM,3,1E-5PCT,3,0.1,EXP,0,1
A,THDRatio,FRMS,3,1E-5PCT,2,0.02,FLAT,0,1
B,THDRatio,FRMS,3,1E-5PCT,2,0.02,FLAT,0,1
A,RATio,NORM,3,1E-4PCT,3,0.03,FLAT,0,1
B,RATio,NORM,3,1E-4PCT,3,0.03,FLAT,0,1
A,RATio,FRMS,1,1E-4PCT,1,0.001,FLAT,0,1
B,RATio,FRMS,1,1E-4PCT,1,0.001,FLAT,0,1
A,SMPTe,NORM,3,1E-5PCT,3,0.1,EXP,0,1
B,SMPTe,NORM,3,1E-5PCT,3,0.1,EXP,0,1
A,SMPTe,FRMS,3,1E-5PCT,2,0.02,FLAT,0,1
B,SMPTe,FRMS,3,1E-5PCT,2,0.02,FLAT,0,1
A,XTALk,NORM,3,1E-5PCT,3,0.1,EXP,0,1
B,XTALk,NORM,3,1E-5PCT,3,0.1,EXP,0,1
A,XTALk,FRMS,3,1E-5PCT,2,0.02,EXP,0,1
B,XTALk,FRMS,3,1E-5PCT,2,0.02,EXP,0,1

Related Commands: :SETTling:DANLr:FUNC?, :DSP:REF:DBM, :DSP:REF:DBR1, :DSP:REF:DBR2, :DSP:REF:DBRA, :DSP:REF:DBRB, :DSP:REF:VFS, :DSP:REF:WATT

Command Syntax: :SETTling:DANLr:FUNC **channel, meter, detector, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example: `:SETTLING:DANLR:FUNC A,BPD,NORMAL,3,1e-8FFS,3,0.1,EXP,0,1`

:SETTling:DANLr:FUNC?

Returns the settling parameter settings for the individual 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 command arguments are:

- **channel** - { A | B }
- **meter** - { AMPA | AMPD | BPA | BPD | PHASe | RATio | SMPTe | THDA | THDD | THDRatio | XTALk }
- **detector** - { FRMS | NORMal }
- **units** - { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | DEG | FFS | PCT | PCTFs | PPM | V | W | X_Y }

Response argument(s):

- **channel** - { A | B }
- **response_meter** - { AMPA | AMPD | BPA | BPD | PHASe | RATio | SMPTe | 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) { BITS | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | DEG | FFS | PCT | PCTFs | 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:DANLr:FUNC

Command Syntax: :SETTling:DANLr:FUNC? **channel, meter, detector, units**

Response Syntax: :SETTling:DANLr:FUNC **channel, response_meter, detector, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example: `:SETTLING:DANLR:FUNC? A,BPD,NORMAL,FFS`

Response: `:SETTLING:DANLR:FUNC A,BPD,NORMAL,5,1e-8FFS,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. The detector parameter (second in list), specifies the type of detector, either NORMal (Q-Peak or RMS) or FRMS (FastRMS).

Command	Channel	Param 1 (Channel and Input Domain)	Input Domain	Param 2 (Detector)
:SETTling:DANLr:LEVel	*A*	CHAA	*Analog*	NORMal
				FRMS
		CHAD	*Digital*	NORMal
				FRMS
	B	CHBA	*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), dBm (DBM), dBV (DBV), dBu (DBU), dBr A (DBRA), dBr B (DBRB), dBg A (DBGA), dBg B (DBGB), Watts (W), FFS (FFS), %FS (PCTFS), dBFS (DBFS)
Digital Input: Ch A Level (CHAD), Ch B Level (CHBD)	FFS (FFS), %FS (PCTFs), dBFS (DBFS), Bits (BITS), V (V), 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) { BITS | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCTFs | V | W }
- **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,FRMS,1,1E-6V,1,0.001,FLAT,0,1

CHAA,NORM,1,1E-6V,3,0.03,FLAT,0,1

CHAD,FRMS,1,1E-7FFS,1,0.001,FLAT,0,1

CHAD,NORM,1,1E-6FFS,3,0.03,FLAT,0,1

CHBA,FRMS,1,1E-6V,1,0.001,FLAT,0,1

CHBA,NORM,1,1E-6V,3,0.03,FLAT,0,1

CHBD,FRMS,1E-7FFS,1,0.001,FLAT,0,1

CHBD,NORM,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,FLAT,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), dBm (DBM), dBV (DBV), dBu (DBU), dBr A (DBRA), dBr B (DBRB), dBg A (DBGA), dBg B (DBGB), Watts (W), FFS (FFS), %FS (PCTFS), dBFS (DBFS)
Digital Input: Ch A Level (CHAD), Ch B Level (CHBD)	FFS (FFS), %FS (PCTFs), dBFS (DBFS), Bits (BITS), V (V), dBu (DBU), dBV (DBV), dBr 1 (DBR1), dBr 2 (DBR2)

The command arguments are:

- **channel** - { CHAA | CHAD | CHBA | CHBD }
- **detector** - { FRMS | NORMal }
- **units** - { BITS | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCTFs | V | W }

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) { BITS | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | FFS | PCTFs | V | W }
- **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 (**V**), 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,0.2,5e-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 (**V**), 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 (:DIN:AMPL?) is a peak-to-peak signal amplitude measurement at the front panel XLR or BNC connectors.

The following units apply to the available meters:

Meter	Units
Amplitude (:DIN:AMPL?)	Vpp
Interface Jitter (:DIN:JITTer?)	Seconds, UI
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
JITTer,3,3e-9SEC,3,0.1,EXP,0,1
RATE,0.5,0.1HZ,3,0.03,FLAT,0,1

Related Commands: :DIN:AMPL?, :DIN:RATe?, :DIN:JITTer?, :SETTling:DIN?

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 | UI | 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 | 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
- **trigger** - <nr1> (range: 0 or 1)

Related Commands: :SETTling:DIN

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:HARMonic

This command specifies the settling parameters for the Harmonic Distortion Analyzer meters for channels 1 and 2. Channel nomenclature is different from the ATS software panels. Channel 1 is channel A. Channel 2 is channel B.

Channel

The channel parameter specifies the channel and the input domain (analog or digital) to provide settling settings unique to the channel and domain. Valid channel specifiers are Channel 1

Analog (CH1A), Channel 1 Digital (CH1D), Channel 2 Analog (CH2A), and Channel 2 Digital (CH2D). For example, the settling settings sent for channel CH1D for meter FAMP will only be used for the fundamental amplitude meter on channel 1 when the input is set to digital, but will not be used for any other meter or channel or input domain.

Meter

The Harmonic Distortion Analyzer meters are the Fundamental Amplitude meters (FAMPlitude), the Fundamental Frequency meters (FFRQ), and the Harmonic Sum meters (SUM1 and SUM2).

Floor

Settling Floor units for the FAMPlitude meters on channel 1 and 2 are Fraction Full Scale (FFS), Percent Full Scale (PCTFs), dB Full Scale (DBFS), Bits (BITS), Volts (V), dBu (DBU), dBV (DBV), dB Ref 1 (DBR1), and dB Ref 2 (DBR2), dBr A (DBRA), dBr B (DBRB), dBg A (DBGA), dBg B (DBGB), dBm (DBM), and Watts (W).

Settling Floor units for the FFRQ meters on channel 1 and 2 are Cents (CENT), Decades (DECS), Delta Hz (DHZ), Delta Herz Percent (DPCT), Delta Parts Per Million (DPPM), Frequency Ratio (F_R), Herz (HZ), Octaves (OCTS), and Percent Hz (PCTHz).

Non-Ratio Settling Floor units for the SUM1 and SUM2 meters on channels 1 and 2 in both analog and digital domains for non-ratio measurements are Fraction Full Scale (FFS), Percent Full Scale (PCTFs), dB Full Scale (DBFS), Bits (BITS), Volts (V), dBu (DBU), dBV (DBV), dB Ref 1 (DBR1), and dB Ref 2 (DBR2), dBr A (DBRA), dBr B (DBRB), dBg A (DBGA), dBg B (DBGB), dBm (DBM), and Watts (W). These settings are stored independently from the Ratio settling settings.

Ratio Settling floor units for the SUM1 and SUM2 meters on channels 1 and 2 in both analog and digital domains are Percent (PCT), X/Y Ratio (X_Y), dB (DB), and Parts-per-Million (PPM). These settings are stored independently from the Non-Ratio settling settings.

Cross-domain floor units are converted to the unit requested for a measurement using the setting of :DSP:REF:VFS. For example, if :DSP:HARMonic:INPut is set to DIGital, and a :DSP:HARMonic:FAMPL? 1,FFS measurement is requested, then a settling floor specified in units of V will be converted internally to FFS for settling. For example, the command :DSP:REF:VFS 0.5;:SETTLING:HARMONIC CH1D,FAMP,1.0,0.1V,3,3E-2,FLAT,0,0;:DSP:HARMONIC:FAMPL? 1,FFS will result in a floor setting of 0.2FFS.

Cross-domain conversion of floor units will depend on the :DSP:HARMonic:INP setting and current value of the :DSP:REF:VFS setting.

For FFRQ meter, CH1A and CH1D are equivalent, and CH2A and CH2D are equivalent because the frequency settling parameter applies to both analog and digital measurements domains.

For SUM1 and SUM2 meters with a Ratio unit for the floor parameter (DB, PCT, PPM, and X_Y), CH1A and CH1D are equivalent, and CH2A and CH2D are equivalent because ratio measurements ignore the input domain.

Channels	Meter	Floor Units
CH1A	FAMP	DBFS, DBM, DBRA, DBRB, DBGA, DBGB, DBU, DBV, FFS, PCTFs, V, W
CH1D	FAMP	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
CH2A	FAMP	DBFS, DBM, DBRA, DBRB, DBGA, DBGB, DBU, DBV, FFS, PCTFs, V, W
CH2D	FAMP	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
CH1A/CH1D	FFRQ	CENT, DECS, DHZ, DPCT, DPPM, F_R, HZ, OCTS, PCTHz
CH2A/CH2D	FFRQ	CENT, DECS, DHZ, DPCT, DPPM, F_R, HZ, OCTS, PCTHz
Ch1A (Non-Ratio)	SUM1	DBFS, DBM, DBRA, DBRB, DBGA, DBGB, DBU, DBV, FFS, PCTFs, V, W
Ch1D (Non-Ratio)	SUM1	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
Ch2A (Non-Ratio)	SUM1	DBFS, DBM, DBRA, DBRB, DBGA, DBGB, DBU, DBV, FFS, PCTFs, V, W
Ch2D (Non-Ratio)	SUM1	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
Ch1A/CH1D (Ratio)	SUM1	DB, PCT, PPM, X_Y
Ch2A/CH2D (Ratio)	SUM1	DB, PCT, PPM, X_Y
Ch1A (Non-Ratio)	SUM2	DBFS, DBM, DBRA, DBRB, DBGA, DBGB, DBU, DBV, FFS, PCTFs, V, W
Ch1D (Non-Ratio)	SUM2	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
Ch2A (Non-Ratio)	SUM2	DBFS, DBM, DBRA, DBRB, DBGA, DBGB, DBU, DBV, FFS, PCTFs, V, W
Ch2D (Non-Ratio)	SUM2	BITS, DBFS, DBR1, DBR2, DBU, DBV, FFS, PCTFs, V
Ch1A/CH1D (Ratio)	SUM2	DB, PCT, PPM, X_Y
Ch2A/CH2D (Ratio)	SUM2	DB, PCT, PPM, X_Y

The command arguments are:

- **channel** - { CH1A | CH1D | CH2A | CH2D }
- **meter** - { FAMPlitude | FFRQ | SUM1 | SUM2 }
- **tolerance** - <nrf> (range: ≥ 0, (1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument meter and unit) { BITS | CENT | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | DECS | DHZ | DPCT | DPPM | FFS | F_R | HZ | OCTS | PCT | PCTFs | PCTHz | PPM | V | X_Y | W }
- **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, (1)

Defaults: CH1A,FAMP,1.0,1E-6V,3,0.03,FLAT,0,0
CH1D,FAMP,1.0,1E-6FFS,3,0.03,FLAT,0,0
CH2A,FAMP,1.0,1E-6V,3,0.03,FLAT,0,0
CH2D,FAMP,1.0,1E-6FFS,3,0.03,FLAT,0,0
CH1A,FFRQ,0.5,0.01HZ,1,0.002,FLAT,0,0
CH2A,FFRQ,0.5,0.01HZ,1,0.002,FLAT,0,0
CH1A,SUM1,0.5,0.01V,1,0.002,FLAT,0,0
CH1D,SUM1,0.5,0.01FFS,1,0.002,FLAT,0,0
CH2A,SUM1,0.5,0.01V,1,0.002,FLAT,0,0
CH2D,SUM1,0.5,0.01FFS,1,0.002,FLAT,0,0
CH1A,SUM1,0.5,0.01X_Y,1,0.002,FLAT,0,0
CH2A,SUM1,0.5,0.01X_Y,1,0.002,FLAT,0,0
CH1A,SUM2,0.5,0.01V,1,0.002,FLAT,0,0
CH1D,SUM2,0.5,0.01FFS,1,0.002,FLAT,0,0
CH2A,SUM2,0.5,0.01V,1,0.002,FLAT,0,0
CH2D,SUM2,0.5,0.01FFS,1,0.002,FLAT,0,0
CH1A,SUM2,0.5,0.01X_Y,1,0.002,FLAT,0,0
CH2A,SUM2,0.5,0.01X_Y,1,0.002,FLAT,0,0

Related Commands: :SETTling:HARMonic?, :DSP:HARMonic:FAMPlitude?, :DSP:HARMonic:FFRQ?, :DSP:HARMonic:SUM1?, :DSP:HARMonic:SUM2?, :DSP:REF:VFS

Command Syntax: :SETTling:HARMonic **channel, meter, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example:
```
:SETTling:HARMonic CH1A,FAMP,1.0,1E-6V,3,
3E-2,FLAT,4.0,0
```

:SETTling:HARMonic?

This query returns the settling parameter settings for the DSP Harmonic Distortion Analyzer meters for channels 1 and 2. See :SETTling:HARMonic.

Channel

Channel nomenclature is different from the ATS software panels. Channel 1 is channel A. Channel 2 is channel B.

The channel parameter specifies the channel and the input domain (analog or digital) to provide settling settings unique to the channel and domain. Valid channel specifiers are Channel 1 Analog (CH1A), Channel 1 Digital (CH1D), Channel 2 Analog (CH2A), and Channel 2 Digital (CH2D). For example, the settling settings sent for channel CH1D for meter FAMP will only be used for the fundamental amplitude meter on channel

1 when the input is set to digital, but will not be used for any other meter or channel or input domain.

Meter

The Harmonic Distortion Analyzer meters are the Fundamental Amplitude meters (FAMPlitude), the Fundamental Frequency meters (FFRQ), and the Harmonic Sum meters (SUM1 and SUM2).

Floor Units

See the discussion on Floor Units above for the :SETTling:HARMonic command.

For FFRQ meter the response will always be for the analog domain but will apply to the digital domain as well.

For SUM1 and SUM2 meters the response will always be for the analog domain but will apply to the digital domain as well.

The command arguments are:

- **channel** - { CH1A | CH1D | CH2A | CH2D }
- **meter** - { FAMPlitude | FFRQ | SUM1 | SUM2 }
- **units** - { BITS | CENT | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | DECS | DHZ | DPCT | DPPM | FFS | F_R | HZ | OCTS | PCT | PCTFs | PCTHz | PPM | V | X_Y | W }

Response Argument(s):

- **channel** - { CH1A | CH1D | CH2A | CH2D }
- **meter** - { FAMPlitude | FFRQ | SUM1 | SUM2 }
- **tolerance** - <nrf> (range: ≥ 0, (1E34) in units of percent
- **floor** - <nrf> (range: depends on instrument meter and unit) { BITS | CENT | DB | DBFS | DBGA | DBGB | DBM | DBR1 | DBR2 | DBRA | DBRB | DBU | DBV | DECS | DHZ | DPCT | DPPM | FFS | F_R | HZ | OCTS | PCT | PCTFs | PCTHz | PPM | V | X_Y | W }
- **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, (1)

Related Commands: :SETTling:HARMonic

Command Syntax: :SETTling:HARMonic? **channel, meter, units**

Response Syntax: :SETTling:HARMonic **channel, meter, tolerance, floor, points, delay, algorithm, timeout, trigger**

Example 1:
```
:SETTling:HARMonic? CH1A,FAMP,V
```

Response 1: `:SETTLING:HARMONIC CH1A,FAMPLITUDE,1,1E-06V,3,0.03,FLAT,0,1`

Example 2: `:SETTLING:HARMonic? CH1D,FFRQ,HZ`

Response 2: `:SETTLING:HARMONIC CH1D,FFRQ,0.5,0.01HZ,1,0.002,FLAT,0,1`

Example 3: `:SETTling:HARMonic? CH1D,SUM1,DB`

Response 3: `:SETTLING:HARMONIC CH1D,SUM1,0.5,-80DB,1,0.002,FLAT,0,1`

Example 4: `:SETTling:HARMonic? CH2D,SUM2,DBFS`

Response 4: `:SETTLING:HARMONIC CH2D,SUM2,0.5,-40DBFS,1,0.002,FLAT,0,1`

: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:**settings**

Example: `:SETTLING:SET?`

Response:

```
:SETTLING: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 JITTER,3,3E-09SEC,3,0.1,EXP,0,1;
DIN RATE,0.5,0.1HZ,3,0.03,FLAT,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 A,AMPA,FRMS,1,1E-06V,1,0.001,FLAT,0,1;
FUNC B,AMPA,FRMS,1,1E-06V,1,0.001,FLAT,0,1;
FUNC A,AMPA,NORMAL,1,1E-06V,3,0.03,FLAT,0,1;
FUNC B,AMPA,NORMAL,1,1E-06V,3,0.03,FLAT,0,1;
FUNC A,AMPD,FRMS,1,1E-07FFS,1,0.001,FLAT,0,1;
FUNC B,AMPD,FRMS,1,1E-07FFS,1,0.001,FLAT,0,1;
FUNC A,AMPD,NORMAL,1,1E-06FFS,3,0.03,FLAT,0,1;
FUNC B,AMPD,NORMAL,1,1E-06FFS,3,0.03,FLAT,0,1;
FUNC A,BPA,FRMS,3,1E-08V,2,0.02,EXP,0,1;
FUNC B,BPA,FRMS,3,1E-08V,2,0.02,EXP,0,1;
FUNC A,BPA,NORMAL,3,1E-08V,3,0.1,EXP,0,1;
FUNC B,BPA,NORMAL,3,1E-08V,3,0.1,EXP,0,1;
FUNC A,BPD,FRMS,3,1E-08FFS,2,0.02,EXP,0,1;
```

```
FUNC B,BPD,FRMS,3,1E-08FFS,2,0.02,EXP,0,1;
FUNC A,BPD,NORMAL,3,1E-08FFS,3,0.1,EXP,0,1;
FUNC B,BPD,NORMAL,3,1E-08FFS,3,0.1,EXP,0,1;
FUNC A,PHASE,NORMAL,0,0.2DEG,2,0.02,FLAT,0,1;
FUNC A,THDA,FRMS,3,1E-07V,2,0.02,FLAT,0,1;
FUNC B,THDA,FRMS,3,1E-07V,2,0.02,FLAT,0,1;
FUNC A,THDA,NORMAL,3,1E-07V,3,0.1,EXP,0,1;
FUNC B,THDA,NORMAL,3,1E-07V,3,0.1,EXP,0,1;
FUNC A,THDD,FRMS,3,1E-07FFS,2,0.02,FLAT,0,1;
FUNC B,THDD,FRMS,3,1E-07FFS,2,0.02,FLAT,0,1;
FUNC A,THDD,NORMAL,3,1E-07FFS,3,0.1,EXP,0,1;
FUNC B,THDD,NORMAL,3,1E-07FFS,3,0.1,EXP,0,1;
FUNC
A,THDRATIO,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
FUNC
B,THDRATIO,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
FUNC
A,THDRATIO,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1;
FUNC
B,THDRATIO,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1;
FUNC
A,RATIO,NORMAL,3,0.0001PCT,3,0.03,FLAT,0,1;
FUNC
B,RATIO,NORMAL,3,0.0001PCT,3,0.03,FLAT,0,1;
FUNC A,RATIO,FRMS,1,0.0001PCT,1,0.001,FLAT,0,1;
FUNC B,RATIO,FRMS,1,0.0001PCT,1,0.001,FLAT,0,1;
FUNC A,SMPTE,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
FUNC B,SMPTE,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
FUNC A,SMPTE,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1;
FUNC B,SMPTE,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1;
FUNC A,XTALK,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
FUNC B,XTALK,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
FUNC A,XTALK,FRMS,3,1E-05PCT,2,0.02,EXP,0,1;
FUNC
B,XTALK,FRMS,3,1E-05PCT,2,0.02,EXP,0,1;LEVEL
CHAD,NORMAL,1,1E-06FFS,3,0.03,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 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,FLAT,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:HARMONIC
CH1A,FAMPLITUDE,1,1E-06V,3,0.03,FLAT,0,1;
HARMONIC
CH1D,FAMPLITUDE,1,1E-06FFS,3,0.03,FLAT,0,1;
HARMONIC
CH2A,FAMPLITUDE,1,1E-06V,3,0.03,FLAT,0,1;
HARMONIC
CH2D,FAMPLITUDE,1,1E-06FFS,3,0.03,FLAT,0,1;
HARMONIC CH1A,FFRQ,0.5,0.01HZ,1,0.002,FLAT,0,1;
HARMONIC CH2A,FFRQ,0.5,0.01HZ,1,0.002,FLAT,0,1;
HARMONIC CH1A,SUM1,0.5,0.01V,1,0.002,FLAT,0,1;
```

```
HARMONIC
CH1D,SUM1,0.5,0.01FFS,1,0.002,FLAT,0,1;
HARMONIC CH2A,SUM1,0.5,0.01V,1,0.002,FLAT,0,1;
HARMONIC
CH2D,SUM1,0.5,0.01FFS,1,0.002,FLAT,0,1;
HARMONIC
CH1A,SUM1,0.5,0.01X_Y,1,0.002,FLAT,0,1;
HARMONIC
CH2A,SUM1,0.5,0.01X_Y,1,0.002,FLAT,0,1;
HARMONIC CH1A,SUM2,0.5,0.01V,1,0.002,FLAT,0,1;
HARMONIC
CH1D,SUM2,0.5,0.01FFS,1,0.002,FLAT,0,1;
HARMONIC CH2A,SUM2,0.5,0.01V,1,0.002,FLAT,0,1;
HARMONIC
CH2D,SUM2,0.5,0.01FFS,1,0.002,FLAT,0,1;
HARMONIC
CH1A,SUM2,0.5,0.01X_Y,1,0.002,FLAT,0,1;
HARMONIC
CH2A,SUM2,0.5,0.01X_Y,1,0.002,FLAT,0,1;
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) 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) seconds

Related Commands: :SETTling:TIMeout

Command Syntax: :SETTling:TIMeout?

Response Syntax: :SETTling:TIMeout **timeout**

Example: `:SETTLING:TIMEOUT?`

Response: `:SETTLING:TIMEOUT 0`

Chapter 18
DCX-127 Commands

The header path for the DCX-127 commands is :DCX:.

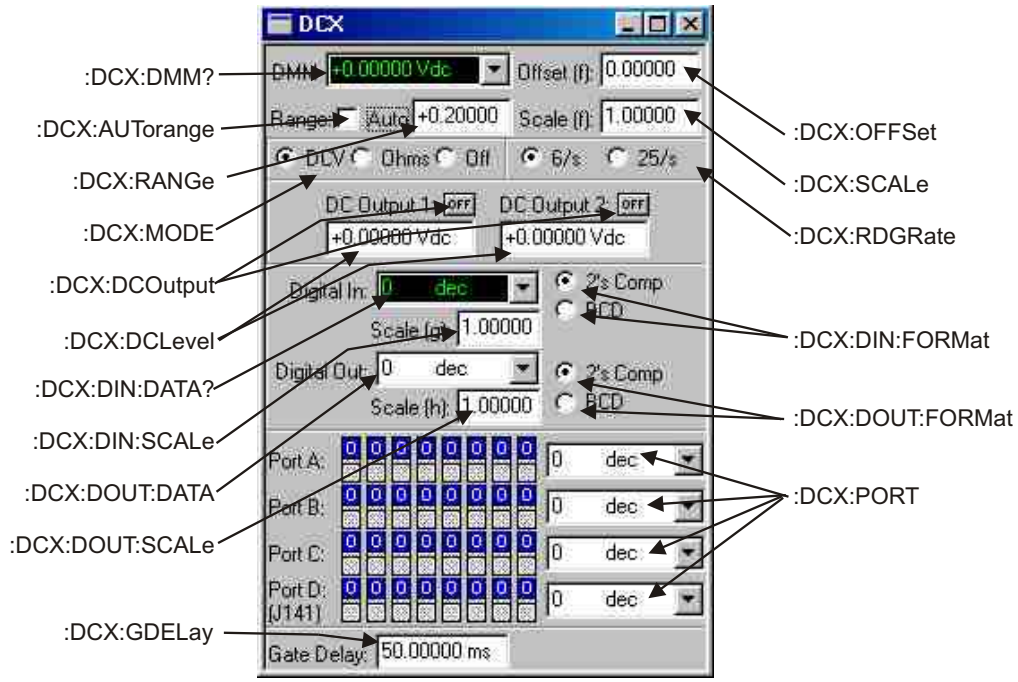


Figure 18-1. ATS software DCX control panel DCX GPIB commands.

: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 DC 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 DC 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

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 BCD binary inputs (those in which the four-bit nybble has a value greater than 9) will cause erroneous values to be read from the port. No error will be generated. The excess digits will overflow into the next higher digit. This following table shows an example of how the values are interpreted for invalid BCD bit patterns:

Binary Input	Interpreted Decimal Number
0000 1010	10
0000 1011	11
0000 1100	12
0000 1101	13
0000 1110	14
0000 1111	15

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 ATS-2 panel control for this command, but there is a comparable AP BASIC 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:**settings**

Example: `:DCX:DIN:SET?`

Response: `:DCX:DIN:FORMAT TWOS;SCALE 1;RATE 32`

:DCX:DMM?

Returns the 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

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 digital output format setting.

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

where entry value is the decimal value entered with the :DCX:DOUT:DATA command and Scale 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:**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). Port D is the DCX-127 rear panel port labeled J141.

The example sets all of the control pins on port A high.

The command arguments are:

- **port** - { A | B | C | D }
- **setting** - <nr1> (range: ≥ 0, ≤ 255)

Default: A, 0; B, 0; C, 0; D,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 | D }

Response argument(s):

- **response_port** - { A | B | C | D }
- **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:**settings**

Example: `:DCX:SET?`

Response:
```
:DCX:DCLEVEL DC1,0;DCLEVEL DC2,0;DCOUTPUT
DC1,OFF;DCOUTPUT DC2,OFF;MODE VOLTS;RANGE
500V;RANGE 2E+06O;AUTORANGE ON;OFFSET
0;SCALE 1;RDGRATE R6;GDELAY 0.1;PORT
A,0;PORT B,0;PORT C,0;PORT
D,0;:DCX:DIN:FORMAT TWOS;SCALE 1;RATE
32;:DCX:DOUT:DATA 0DEC;FORMAT TWOS;SCALE 1
```

Chapter 19

SWR-2122 Switcher Commands

The header-path for SWR-2122 commands is :SWR:.

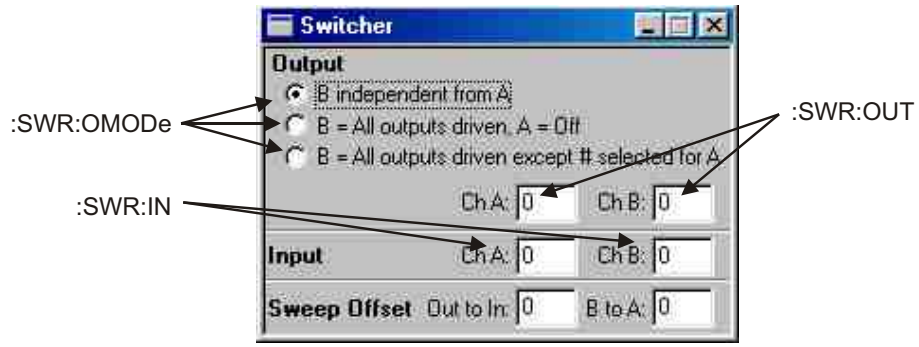


Figure 19-1. ATS software Switcher control panel SWR GPIB commands.

: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 ATS-2. 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:**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
```

Appendix A

ASCII Code Chart & IEEE-488 Codes

HEX	0	1	2	3	4	5	6	7
0	0 NUL 0 0	20 DLE 16 10	40 0 SP 32 20	60 16 0 48 30	100 0 @ 64 40	120 16 P 80 50	140 0 ` 96 60	160 16 p 112 70
1	1 GTL SOH 1 1	21 LLO DC1 17 11	41 1 ! 33 21	61 17 1 49 31	101 1 A 65 41	121 17 Q 81 51	141 1 a 97 61	161 17 q 113 71
2	2 STX 2 2	22 DC2 18 12	42 2 " 34 22	62 18 2 50 32	102 2 B 66 42	122 18 R 82 52	142 2 b 98 62	162 18 r 114 72
3	3 ETX 3 3	23 DC3 19 13	43 3 # 35 23	63 19 3 51 33	103 3 C 67 43	123 19 S 83 53	143 3 c 99 63	163 19 s 115 73
4	4 STC EOT 4 4	24 DCL DC4 20 14	44 4 $ 36 24	64 20 4 52 34	104 4 D 68 44	124 20 T 84 54	144 4 d 100 64	164 20 t 116 74
5	5 PPC ENQ 5 5	25 PPU NAK 21 15	45 5 % 37 25	65 21 5 53 35	105 5 E 69 45	125 21 U 85 55	145 5 e 101 65	165 21 u 117 75
6	6 ACK 6 6	26 SYN 22 16	46 6 & 38 26	66 22 6 54 36	106 6 F 70 46	126 22 V 86 56	146 6 f 102 66	166 22 v 118 76
7	7 BEL 7 7	27 ETB 23 17	47 7 ' 39 27	67 23 7 55 37	107 7 G 71 47	127 23 W 87 57	147 7 g 103 67	167 23 w 119 77
8	10 GET BS 8 8	30 SPE CAN 24 18	50 8 (40 28	70 24 8 56 38	110 8 H 72 48	130 24 X 88 58	150 8 h 104 68	170 24 x 120 78
9	11 TCT HT 9 9	31 SPD EM 25 19	51 9) 41 29	71 25 9 57 39	111 9 I 73 49	131 25 Y 89 59	151 9 i 105 69	171 25 y 121 79
A	12 LF 10 A	32 SUB 26 1A	52 10 * 42 2A	72 26 : 58 3A	112 10 J 74 4A	132 26 Z 90 5A	152 10 j 106 6A	172 26 z 122 7A
B	13 VT 11 B	33 ESC 27 1B	53 11 + 43 2B	73 27 ; 59 3B	113 11 K 75 4B	133 27 [91 5B	153 11 k 107 6B	173 27 { 123 7B
C	14 FF 12 C	34 FS 28 1C	54 12 , 44 2C	74 28 < 60 3C	114 12 L 76 4C	134 28 \ 92 5C	154 12 l 108 6C	174 28 | 124 7C
D	15 CR 13 D	35 GS 29 1D	55 13 - 45 2D	75 29 = 61 3D	115 13 M 77 4D	135 29] 93 5D	155 13 m 109 6D	175 29 } 125 7D
E	16 SO 14 E	36 RS 30 1E	56 14 . 46 2E	76 30 > 62 3E	116 14 N 78 4E	136 30 ^ 94 5E	156 14 n 110 6E	176 30 ~ 126 7E
F	17 SI 15 F	37 US 31 1F	57 15 / 47 2F	77 UNL ? 63 3F	117 15 O 79 4F	137 UNT _ 95 5F	157 15 o 111 6F	177 DEL 127 7F
	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

ATS-2 GPIB 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 **ATS-2 GPIB Default Settings**, following. 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.

Each time a DSP program is loaded, the settings of those programs have their own default values to which they are initialized.

ATS-2 GPIB Default Settings

```
:AGEN:
    REF:
        DBM 600;
        DBR 0.3873V;
        FREQ 1000;
        WATT 8;
:AGEN:
    OUTPUT OFF;
    CONFIG BAL;
    IMPEDANCE Z40;
    AMPL A,1V;
    AMPL B,1V;
    WFM DASINE,SINE;
:AGEN:
    DAIMD:
        SMPTE:
            HIFREQ 3000HZ;
            IMFREQ 80;
:AGEN:
    DASINE:
        FRQ1 1000HZ;
        FRQ2 1000HZ;
        BURINTERVAL 3CYCLES;
        BURLEVEL 10PCT;
        BURTIMEON 1CYCLES;
        RATIO 0.25X_Y;
        VPHASE 0

:AUX:
    DOUT 0;

:ANLG:
    AUTORANGE A,ON;
    AUTORANGE B,ON;
    CONVERTER AD1;
    COUPLING A,AC;
    COUPLING B,AC;
    IMPEDANCE A,ZHI;
    IMPEDANCE B,ZHI;
    SOURCE A,XLR;
    SOURCE B,XLR

: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;
    :DGEN:
        OUTPUT OFF;
        INVERT A,OFF;
        INVERT B,OFF;
        AMPL A,0.999756FFS;
        AMPL B,0.999756FFS;
        BURTIMEON 1CYCLES;
        BURLEVEL 0.25X_Y;
        BURINTERVAL 3CYCLES;
        RATIO 0.25X_Y;
        DITHERTYPE TRI;
        ARBSIZE 8192;
        FRQ1 997.001HZ;
        FRQ2 999.999HZ;
        VPHASE 0;
        OFFSET 0FFS;
        WFM SINE,SINE;
        SAMPLES 1;
        SMPTE:
            HIFREQ 3000HZ;
            IMFREQ 60

    :DIN:
        REF 48000;
        BANDWIDTH 700;
        RESOLUTION 24,BITS;
        DETECTOR AVERAGE;
        MODE ACTIVE;
        PKMODE HALF;
        SCALEFREQBY MEASURED;
        DEEMPHASIS OFF;
        FORMAT XLR;
        IMPEDANCE BNC,Z75;
        IMPEDANCE XLR,Z110;
        CONNECTOR 1

    :DIOSTATUS:
        XMT A,
            "04,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00"
            ;
        XMT B,
            "04,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00"
            ;
```

```
:DOUT:
    AMPL 5;
    FORMAT XLR;
    INVALID 0;
    INVRT OFF;
    JAMPL 0UI;
    JFREQ 998.644;
    JWFM NONE;
    PARITY 0;
    PREEMPHASIS OFF;
    RATE 48000HZ;
    RESOLUTION 24,BITS;
    SCALEFREQBY OUTRATE

:MON:
    SOURCE AINPUT;
    VOLUME 0

:SETTLING:
    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 JITTER,3,3E-09SEC,3,0.1,EXP,0,1;
    DIN RATE,0.5,0.1HZ,3,0.03,FLAT,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 A,AMPA,FRMS,1,1E-06V,1,0.001,FLAT,0,1;
        FUNC B,AMPA,FRMS,1,1E-06V,1,0.001,FLAT,0,1;
        FUNC A,AMPA,NORMAL,1,1E-06V,3,0.03,FLAT,0,1;
        FUNC B,AMPA,NORMAL,1,1E-06V,3,0.03,FLAT,0,1;
        FUNC A,AMPD,FRMS,1,1E-07FFS,1,0.001,FLAT,0,1;
        FUNC B,AMPD,FRMS,1,1E-07FFS,1,0.001,FLAT,0,1;
        FUNC A,AMPD,NORMAL,1,1E-06FFS,3,0.03,FLAT,0,1;
        FUNC B,AMPD,NORMAL,1,1E-06FFS,3,0.03,FLAT,0,1;
        FUNC A,BPA,FRMS,3,1E-08V,2,0.02,EXP,0,1;
        FUNC B,BPA,FRMS,3,1E-08V,2,0.02,EXP,0,1;
        FUNC A,BPA,NORMAL,3,1E-08V,3,0.1,EXP,0,1;
        FUNC B,BPA,NORMAL,3,1E-08V,3,0.1,EXP,0,1;
        FUNC A,BPD,FRMS,3,1E-08FFS,2,0.02,EXP,0,1;
        FUNC B,BPD,FRMS,3,1E-08FFS,2,0.02,EXP,0,1;
        FUNC A,BPD,NORMAL,3,1E-08FFS,3,0.1,EXP,0,1;
        FUNC B,BPD,NORMAL,3,1E-08FFS,3,0.1,EXP,0,1;
        FUNC A,PHASE,NORMAL,0,0.2DEG,2,0.02,FLAT,0,1;
        FUNC A,THDA,FRMS,3,1E-07V,2,0.02,FLAT,0,1;
        FUNC B,THDA,FRMS,3,1E-07V,2,0.02,FLAT,0,1;
        FUNC A,THDA,NORMAL,3,1E-07V,3,0.1,EXP,0,1;
        FUNC B,THDA,NORMAL,3,1E-07V,3,0.1,EXP,0,1;
        FUNC A,THDD,FRMS,3,1E-07FFS,2,0.02,FLAT,0,1;
        FUNC B,THDD,FRMS,3,1E-07FFS,2,0.02,FLAT,0,1;
        FUNC A,THDD,NORMAL,3,1E-07FFS,3,0.1,EXP,0,1;
        FUNC B,THDD,NORMAL,3,1E-07FFS,3,0.1,EXP,0,1;
        FUNC A,THDRATIO,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
        FUNC B,THDRATIO,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
        FUNC A,THDRATIO,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1;
        FUNC B,THDRATIO,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1;
        FUNC A,RATIO,NORMAL,3,0.0001PCT,3,0.03,FLAT,0,1;
        FUNC B,RATIO,NORMAL,3,0.0001PCT,3,0.03,FLAT,0,1;
        FUNC A,RATIO,FRMS,1,0.0001PCT,1,0.001,FLAT,0,1;
        FUNC B,RATIO,FRMS,1,0.0001PCT,1,0.001,FLAT,0,1;
        FUNC A,SMPTE,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
        FUNC B,SMPTE,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
        FUNC A,SMPTE,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1;
        FUNC B,SMPTE,FRMS,3,1E-05PCT,2,0.02,FLAT,0,1;
        FUNC A,XTALK,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
        FUNC B,XTALK,NORMAL,3,1E-05PCT,3,0.1,EXP,0,1;
```

```
            FUNC A,XTALK,FRMS,3,1E-05PCT,2,0.02,EXP,0,1;
            FUNC B,XTALK,FRMS,3,1E-05PCT,2,0.02,EXP,0,1;
            LEVEL CHAD,NORMAL,1,1E-06FFS,3,0.03,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 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,FLAT,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:
        HARMONIC CH1A,FAMPLITUDE,1,1E-06V,3,0.03,FLAT,0,1;
        HARMONIC CH1D,FAMPLITUDE,1,1E-06FFS,3,0.03,FLAT,0,1;
        HARMONIC CH2A,FAMPLITUDE,1,1E-06V,3,0.03,FLAT,0,1;
        HARMONIC CH2D,FAMPLITUDE,1,1E-06FFS,3,0.03,FLAT,0,1;
        HARMONIC CH1A,FFRQ,0.5,0.01HZ,1,0.002,FLAT,0,1;
        HARMONIC CH2A,FFRQ,0.5,0.01HZ,1,0.002,FLAT,0,1;
        HARMONIC CH1A,SUM1,0.5,0.01V,1,0.002,FLAT,0,1;
        HARMONIC CH1D,SUM1,0.5,0.01FFS,1,0.002,FLAT,0,1;
        HARMONIC CH2A,SUM1,0.5,0.01V,1,0.002,FLAT,0,1;
        HARMONIC CH2D,SUM1,0.5,0.01FFS,1,0.002,FLAT,0,1;
        HARMONIC CH1A,SUM1,0.5,0.01X_Y,1,0.002,FLAT,0,1;
        HARMONIC CH2A,SUM1,0.5,0.01X_Y,1,0.002,FLAT,0,1;
        HARMONIC CH1A,SUM2,0.5,0.01V,1,0.002,FLAT,0,1;
        HARMONIC CH1D,SUM2,0.5,0.01FFS,1,0.002,FLAT,0,1;
        HARMONIC CH2A,SUM2,0.5,0.01V,1,0.002,FLAT,0,1;
        HARMONIC CH2D,SUM2,0.5,0.01FFS,1,0.002,FLAT,0,1;
        HARMONIC CH1A,SUM2,0.5,0.01X_Y,1,0.002,FLAT,0,1;
        HARMONIC CH2A,SUM2,0.5,0.01X_Y,1,0.002,FLAT,0,1;
        TIMEOUT 4

    :SWR:
        OMODE INDEPENDENT;
        IN A,0;
        IN B,0;
        OFFSET CHB,0;
        OFFSET OUT,0;
        OUT A,0;
        OUT B,0

    :SYNC:
        ENABLE OFF;
        SRC AESEBU;
        IMPEDANCE ZLO;
        FREQ 48000;
        IFLOCK OFF

    :TRIG:
        ERRTRIG:
            CODING ON;
            CONFIDENCE ON;
            LOCK ON;
            PARITY ON;
    :TRIG:
        SOURCE ARCV;
```

DSP Default Settings

```
    :DSP:
        PROGRAM DANLR;
    :DSP:
        REF:
            DBM 600;
            DBR1 0.1FFS;
            DBR2 0.1FFS;
```

```
        DBRA 0.3873V;
        DBRB 0.3873V;
        FREQ 1000;
        VFS 1;
        WATT 8;
```

DSP Audio Analyzer (DANLr) Default Settings

```
:DSP:DANLR:
    AUTORANGE A,ON
    AUTORANGE B,ON
    COUPLING A,AC
    COUPLING B,AC
    DETECTOR FRMS
    FAUTORANGE A,ON
    FAUTORANGE B,ON
    FILTERFREQ 1000HZ
    HPFILTER F10
    INPUT DIGITAL
    LPFILTER FS_2
    MODE AMPLITUDE
    PRANGE AUTO
    RDGRATE R8
    RESPONSE 20
    TUNINGSRC FIXED
    WTG UNWT
```

Multitone Audio Analyzer (FASTtest) Default Settings

```
:DSP:
    PROGRAM FASTTEST;
    TIMEOUT 10.000000;
    SRCPARAMS FREQ,HZ,20,20000,30,LOG;
:DSP:
    FASTTEST:
        FREQRES 0;
        INPUT ANLG;
        LENGTH AUTO;
        MODE SPECTRUM;
        PMODE INDEPENDENT;
        PROCESS SYNC;
        TRGDELAY 0;
        TRIGGER DGEN;
        PKTRIG 1
```

FFT Spectrum Analyzer (FFT) Default Settings

```
:DSP:
    PROGRAM FFT;
    TIMEOUT 10.000000;
    SRCPARAMS FREQ,HZ,20,20000,30,LOG;
:DSP:
    FFT:
        ACQLENGTH XLENGTH;
        AVGS 1;
        AVGTYPE 1;
        COUPLING DC;
        DELAY 0;
        MODE INTRPOLATE;
        INPUT ANLG;
        START 0;
```

```
        TRIGGER FREE;
        TSLOPE POS;
        WINDOW EQR;
        XLENGTH 8192;
        PKTRIG 1
```

Harmonic Distortion Analyzer (HARMONIC) Default Settings

```
:DSP:
    PROGRAM HARMONIC;
:DSP:
    HARMONIC:
        HAR1 1,0;
        HAR1 2,0;
        HAR2 1,0;
        HAR2 2,0;
        MODE HISPEED;
        INPUT ANLG;
        STEERING:
            FREQUENCY 1000HZ;
            SOURCE CNTR
```

Digital Interface Analyzer (INTervu) Default Settings

```
:DSP:
    PROGRAM INTERVU;
    TIMEOUT 10.000000;
    SRCPARAMS FREQ,HZ,20,20000,30,LOG;
:DSP:
    INTERVU:
        AVGS 1;
        JDETECTION STABLE;
        MODE INTRPOLATE;
        TMODE POSTTRIG;
        TRIGGER ARCV;
        WINDOW BH;
        ERRTRIG:
            CODING ON;
            CONFIDENCE ON;
            LOCK ON;
            PARITY ON
```

Appendix C

ATS-2 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)

```
2   COMMAND NOT FOUND
3   INCOMPLETE COMMAND HEADER
4   TOO MANY COMMAND HEADERS
5   TOO MANY PARAMETERS
6   NOT ENOUGH PARAMETERS -OR- MISSING UNIT SUFFIX
7   ILLEGAL PARAMETER TYPE
8   MISSING TERMINATOR
9   MISSING SUFFIX
10  ILLEGAL COMMAND HEADER
11  INCOMPLETE ARBITRARY BLOCK DATA
12  ABPD TERMINATED WITHOUT EOI
13  SYNTAX ERROR
14  ILLEGAL STRING
15  UNKNOWN PARAMETER
16  MACRO BUFFER FULL
17  MACRO NOT FOUND
18  ILLEGAL MACRO
19  MACRO ALREADY EXISTS
20  MACRO LOAD FAILURE
21  MACRO PARAM SUBSTITUTION FAILURE
22  COMMAND NOT ALLOWED IN MACRO DEFINITION
23  OUT OF MEMORY TO ALLOCATE FOR ARB IN MACROS
24  DDT MACRO TOO BIG (MAX = 1024 BYTES)
25  DDT MACRO TOO SMALL
```

```
26  COMMA MISSING
27  ILLEGAL MACRO LABEL
28  PARAMETER OUT OF RANGE
29  ILLEGAL GET
30  INDEFINITE ARBITRARY BLOCK TERMINATED ILLEGALLY
31  INDEFINITE ARBITRARY BLOCK NOT ALLOWED AS ARGUMENTS TO MACROS
32  FIRMWARE UNABLE TO RECOGNIZE THIS HARDWARE CONFIGURATION
33  FIRMWARE UPDATE FAILED
```

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
8   BURST OPTION NOT INSTALLED
10  NOT IMPLEMENTED
11  BELOW MINIMUM AMPLITUDE
12  ABOVE MAXIMUM AMPLITUDE
13  BELOW MINIMUM FREQUENCY
14  ABOVE MAXIMUM FREQUENCY
17  REQUESTED IMFREQ OUT OF RANGE
19  REQUESTED HIFREQ OUT OF RANGE
20  ILLEGAL BURST LEVEL REQUESTED
22  ILLEGAL BURST INTERVAL OR TIME ON
25  SELECTED IMPEDANCE OPTION NOT INSTALLED
```

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
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
25  ILLEGAL BURST INTERVAL OR TIME ON
```

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
9   UNKNOWN MODE
```

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 UNITS FOR REQUESTED MEASUREMENT
11  TIMEOUT ON ACQX?, XFRM? OR REPR?
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
23  BATCH TABLE QUERY FAILED
24  PROGRAM IS NOT A BATCH PROGRAM
25  DSP IN OPSTATE SETUP
26  DSP IN OPSTATE READ
27  NO ACQUIRED DATA AVAILABLE
```

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
15  ILLEGAL RANGE
16  USER FILTER NOT LOADED
17  INPUT MUST BE DIGITAL
```

Group 512: FFT DSP Program errors (Execution)

```
1   ILLEGAL SOURCE
2   FFT PROGRAM NOT LOADED
3   ILLEGAL TRIGGER DELAY
4   INCOMPATIBLE AVERAGE TYPE AND WINDOWING MODE
5   DSP IN OPSTATE READ
6   INVALID START TIME SPECIFIED
7   ATTEMPT TO SET TRANSFORM LENGTH FAILED
8   ATTEMPT TO SET AVERAGES FAILED
9   INVALID WINDOW TYPE SELECTED
10  INVALID COUPLING OPTION
11  DATA REQUESTED FROM DISABLED CHANNEL
12  INVALID SAMPLE RATE REQUESTED
13  INVALID SENSITIVITY LEVEL REQUESTED
14  INVALID INPUT REQUESTED
15  INVALID MODE REQUESTED
16  INVALID TRIGGER SLOPE REQUESTED
17  INVALID TRIGGER SOURCE REQUESTED
18  DSP OPTION NOT INSTALLED
19  ATTEMPT TO INVOKE DISABLED READING
20  DSP IN OPSTATE SETUP
21  ANALOG ANALYZER OPTION NOT INSTALLED
22  INVALID TRANSFORM LENGTH REQUESTED
23  INVALID TRIGGER LEVEL REQUESTED
24  INVALID TRIGGER LEVEL FOR SOURCE
25  INVALID TRIGGER SENSITIVITY FOR SOURCE
26  INVALID ACQUISITION LENGTH
```

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
13  INCOMPATIBLE TRIGGER, SLOPE SET TO POS
```

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
14 DSP IN OPSTATE SETUP

Group 516: Digital I/O module errors (Execution)

1 ILLEGAL IMPEDANCE
5 ILLEGAL JITTER WAVEFORM
6 ILLEGAL MODE
7 ILLEGAL LENGTH
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
17 INPUT FORMAT IS OPTICAL
18 OUTPUT FORMAT IS OPTICAL

Group 517: SYNC/REF module errors (Execution)

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
12 DCX NOT AVAILABLE

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 HAS NOT BEEN SET. THE FIRST 16K OR 32K SAMPLES OF THE IMPULSE RESPONSE WILL BE USED TO COMPUTE THE FREQUENCY RESPONSE.
5 CHANNEL 2 TIME FRAME HAS NOT BEEN SET. THE FIRST 16K OR 32K SAMPLES OF THE IMPULSE RESPONSE WILL BE USED TO COMPUTE THE FREQUENCY RESPONSE.
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
22 TOO MANY FILTERS SELECTED FOR OPERATION AT 262 KHZ. CONSIDER USING FEWER FILTERS OR REDUCING THE SAMPLE RATE.

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
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.
47 CANNOT RETRANSFORM IF AVERAGING, 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 MOVE TO BIN CENTER PROCESS WAS HALTED. CHOOSING THE SUBTRACT 1/2 PK-PK OR SUBTRACT AVG OPTIONS ON THE PANEL MAY ALLOW YOU TO USE THE MOVE TO BIN CENTER FEATURE.
54 THE DC COMPONENT OF THE CH 2 SIGNAL IS GREATER IN MAGNITUDE THAN ANY AC COMPONENT AND SO THE MOVE TO BIN CENTER PROCESS WAS HALTED. CHOOSING THE SUBTRACT 1/2 PK-PK OR SUBTRACT AVG OPTIONS ON THE PANEL MAY ALLOW YOU TO USE THE MOVE TO BIN CENTER 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 MOVE TO BIN CENTER. USE THE FOLLOWING FORMULA: MINIMUM FREQUENCY = 6 * SAMPLE FREQUENCY IN HZ / FFT LENGTH. INCREASING THE FFT LENGTH MAY SOLVE THE PROBLEM.
57 THE GREATEST MAGNITUDE TONE FOR CH 2 IS TOO LOW IN FREQUENCY TO USE MOVE TO BIN CENTER. USE THE FOLLOWING FORMULA: MINIMUM FREQUENCY = 6 * SAMPLE FREQUENCY IN HZ / FFT LENGTH. INCREASING THE FFT LENGTH MAY SOLVE THE PROBLEM.
58 RESAMPLING PROCESS ABORTED. NO ZERO CROSSINGS FOUND. CHECK LEVEL METERS TO VERIFY SIGNAL IS PRESENT.
59 RESAMPLING PROCESS ABORTED. THE TRIGGER POINT IS FLUCTUATING BY MORE THAN 3 SAMPLES. THIS USUALLY HAPPENS WHEN THE TRIGGER SOURCE IS SET TO FREE RUN.
60 DISPLAY MODE MAY NOT BE 'EYE PATTERN' WHEN SWEEP SOURCE IS 'AMPLITUDE'. CHANGE THE DISPLAY MODE FROM 'EYE PATTERN', OR SELECT THE 'TIME' SWEEP SOURCE.
61 DISPLAY MODE MAY NOT BE 'EYE PATTERN' WHEN SWEEP SOURCE IS 'FREQUENCY'. CHANGE THE DISPLAY MODE FROM 'EYE PATTERN', OR SELECT THE 'TIME' SWEEP SOURCE.
62 DISPLAY MODE MAY NOT BE 'EYE PATTERN' WHEN SWEEP SOURCE IS 'JITTER'. CHANGE THE DISPLAY MODE FROM 'EYE PATTERN', OR SELECT THE 'TIME' SWEEP SOURCE.
63 ALL DATA SOURCES MUST BE 'EYE OPENING' OR 'NONE' WHEN 'EYE PATTERN' DISPLAY MODE IS SELECTED.
65 DATA SOURCE MUST BE 'PROBABILITY' WHEN SWEEP SOURCE IS 'AMPLITUDE'. CHANGE THE DATA SOURCE TO 'PROBABILITY', OR SELECT ANOTHER SWEEP SOURCE.
66 DATA SOURCE MUST BE 'PROBABILITY' WHEN SWEEP SOURCE IS 'JITTER'. CHANGE THE DATA SOURCE TO 'PROBABILITY', OR SELECT ANOTHER SWEEP SOURCE.
70 DATA SOURCES 'PROBABILITY' AND 'AMPLITUDE' MAY NOT BE SELECTED SIMULTANEOUSLY.
71 DATA SOURCES 'JITTER' AND 'AMPLITUDE' MAY NOT BE SELECTED SIMULTANEOUSLY.
72 DATA SOURCES 'JITTER' AND 'PROBABILITY' MAY NOT BE SELECTED SIMULTANEOUSLY.
73 INVALID COMBINATION OF DATA SOURCES.
81 MAXIMUM ZERO CROSSING TIME DIFFERENCE EXCEEDED. CHECK FREQUENCY OF INPUT SOURCE.
82 NO ZERO CROSSINGS FOUND. CHECK INPUT SOURCE.
83 ONLY ONE ZERO CROSSING FOUND. CHECK INPUT SOURCE.
84 MINIMUM SWEEP TIME TOO LOW FOR EYE PATTERN. MUST BE GREATER THAN -1 US.
85 MAXIMUM SWEEP TIME TOO HIGH FOR CURRENT EYE PATTERN.
86 SPECIFIED TIME INTERVAL DOES NOT CONTAIN VALID JITTER DATA. INCREASE THE MAXIMUM SWEEP TIME.
87 SPECIFIED TIME INTERVAL DOES NOT CONTAIN VALID JITTER DATA. DECREASE THE MINIMUM SWEEP TIME.
88 SPECIFIED TIME INTERVAL DOES NOT CONTAIN ACQUISITION DATA. INCREASE THE MAXIMUM SWEEP TIME.
89 SPECIFIED TIME INTERVAL DOES NOT CONTAIN ACQUISITION DATA. DECREASE THE MINIMUM SWEEP TIME.

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 524: HARMONIC errors (Execution)

1 ILLEGAL SOURCE.
2 HARMONIC PROGRAM NOT LOADED.
4 INVALID INPUT REQUESTED.
7 ILLEGAL FREQ.
8 INVALID UNIT FOR INPUT DOMAIN
9 DISTORTION READINGS NOT AVAILABLE IF STEERING SOURCE NOT CNTR

Group 525: ANLG errors (Execution)

1 DSP IN OPSTATE READ
2 HIGH BANDWIDTH OPTION NOT INSTALLED
3 600 OHM IMPEDANCE INPUT OPTION NOT INSTALLED
4 INVALID RANGE VALUE

Appendix D

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. ATS-2 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 $v = (-1)^s \bullet \infty$
(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:

$$
\begin{aligned}
v &= (-1)^s \, 2^{e-127} \, (1.f) \\
&= (-1)^0 \, 2^6 \, (1.75) \\
&= (64) \, (1.75) \\
&= 112
\end{aligned}
$$

Appendix E

ATS-2 GPIB Firmware Information

ATS Version 1.20 Features Not Implemented in GPIB Firmware Version 1.001

The features of ATS version 1.20 listed below are not supported in ATS-2 GPIB firmware version 1.001.

- Analog Generator EQ Sine.
- Analog Generator D/A EQ Sine.
- Digital Generator EQ Sine.
- Digital Interface Jitter Generator EQ Sine.
- Sweeps controlled by the ATS-2 sweep panel.
- Computation functions for sweep data.
- Limit testing of sweep data.
- Regulation Mode.
- The Digital Generator :DGEN:AMPL command does not support output amplitude units of HEX.
- The Digital Generator :DGEN:ARBLoad command does not automatically set the sample rate when an arbitrary waveform is loaded into the digital generator waveform buffers.

GPIB Firmware v1.001—Known Bugs

The following list describes all known problems in ATS-2 GPIB firmware version 1.001 as of October 2002:

1. The :DSP: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. This is an I/O deadlock condition. Two workarounds are suggested:
   Workaround (a): Avoid the problem by reading the entire response.
   Workaround (b): Send a *RST to reset the instrument and clear the I/O queues. You may want to store the current settings first with *SAV and then recall them with *RCL after the reset.

2. Definite Length Arbitrary Block data must have a correct byte count for the data. If the byte count following the # character (example #15*IDN?) is less than the actual number of data bytes the firmware may lock up and require a power off-on cycle. This is due to a bug in the GPIB parser that does not correctly process message units following a <definite length arb block> data element if the next character after the block is not a ";" semicolon character (a proper message unit separator).
   If the byte count is too high, then valid message units following the block will be included in the block and will not be correctly parsed. This error may also result in a firmware lock-up.
   Workaround (a): Check Arbitrary Block data count carefully to make sure it matches that count of actual data bytes.
   Workaround (b): Use indefinite length Arbitrary Block data format (see page 1-11). Indefinite length data cannot be used in compound commands, but should be adequate in most circumstances.
3. Query Errors are not handled according to std IEEE-488.2. The instrument does not issue a query error (by setting bit 2 of the Standard Event Status Register) and does not clear the unread message from the output queue. The consequence is that unread queries will remain in the output queue.
   A query error occurs under two fundamental conditions:
   a) a query response is generated by a valid query terminated with an END message <PMT> but this response is not read from the output queue before another query is sent.
   b) the instrument is talk-addressed but the output queue is empty (talk with nothing to say).

Index

A

B

C

Q

R

S

T

U

V

W

X