About this Manual

We've added this manual to the Agilent website in an effort to help you support your product. This manual is the best copy we could find; it may be incomplete or contain dated information. If we find a more recent copy in the future, we will add it to the Agilent website.

Support for Your Product

Agilent no longer sells or supports this product. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available. You will find any other available product information on the Agilent Test & Measurement website, www.tm.agilent.com.

HP References in this Manual

This manual may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemical analysis businesses are now part of Agilent Technologies. We have made no changes to this manual copy. In other documentation, to reduce potential confusion, the only change to product numbers and names has been in the company name prefix: where a product number/name was HP XXXX the current name/number is now Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

Programmer's Guide

HP 71500 Series

HP Part No. 70820-90053
Printed in USA May 1993

1400 Fountain Grove Parkway, Santa Rosa, CA 95403, USA

Certification

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

Warranty

This Hewlett-Packard instrument product is warranted against defects in material and workmanship for a period of one year from date of shipment. During the warranty period, Hewlett-Packard Company will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Hewlett-Packard. Buyer shall prepay shipping charges to Hewlett-Packard and Hewlett-Packard shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Hewlett-Packard from another country.

Hewlett-Packard warrants that its software and firmware designated by Hewlett-Packard for use with an instrument will execute its programming instructions when properly installed on that instrument. Hewlett-Packard does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error-free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HEWLETT-PACKARD SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

Assistance

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

Safety Symbols

The following safety symbols are used throughout this manual. Familiarize yourself with each of the symbols and its meaning before operating this instrument.

Caution The *caution* sign denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in damage to or destruction of the instrument. Do not proceed beyond a *caution* sign until the indicated conditions are fully understood and met.

Warning **The *warning* sign denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in injury or loss of life. Do not proceed beyond a *warning* sign until the indicated conditions are fully understood and met.**

General Safety Considerations

Warning ***Before this instrument is switched on*, make sure it has been properly grounded through the protective conductor of the ac power cable to a socket outlet provided with protective earth contact.**

Any interruption of the protective (grounding) conductor, inside or outside the instrument, or disconnection of the protective earth terminal can result in personal injury.

Warning **There are many points in the instrument which can, if contacted, cause personal injury. Be extremely careful.**

Any adjustments or service procedures that require operation of the instrument with protective covers removed should be performed only by trained service personnel.

Caution *Before this instrument is switched on,* make sure its primary power circuitry has been adapted to the voltage of the ac power source.

Failure to set the ac power input to the correct voltage could cause damage to the instrument when the ac power cable is plugged in.

1

Programming Fundamentals

This chapter provides the necessary information to operate a microwave transition analyzer system via a computer. The topics described in this chapter are as follows:

- Setup Procedures for Remote Operation
- Address Switches
- Communication With the System
- Status Reporting Structure
- Synchronization of Events and Commands
- Data Transfer

Setup Procedures

Note Refer to the HP 71500A Microwave Transition Analyzer Installation and Verification Manual for more information on installation, configuration, and addressing for microwave transition analyzer systems.

The following procedure describes how to connect your equipment for remote operation specifically for a microwave transition analyzer system.

1. Connect computer, microwave transition analyzer system, and other peripherals with HP-IB cables.

Note When connecting the HP-IB cables to the microwave transition analyzer system, make sure that these connections are made to the HP-IB connector on the mainframe and *not* to the module itself. This ensures access to all equipment via the public HP-IB.

2. After the HP-IB cables are installed, reset all instruments connected to the bus. (If you are not sure how to reset a device, switch its line power off, then on, to reset it.)
3. Check the HP-IB address of the microwave transition analyzer module on the address map. To view the address map, press the DISPLAY key on the display front panel, then press the ADDRESS MAP softkey. Turn the knob on the front panel of the display until the microwave transition analyzer appears in the address map.

Note The microwave transition analyzer module must be located in row 0 for HP-IB access and error-reporting capabilities.

Address Switches

Address switches set the HP-MSIB address of an element (module); the column address switches also set the HP-IB address for masters and independent elements. Some master elements can also have their HP-IB address set through the use of softkeys (that is, soft-set address).

The hard address switches for the microwave transition analyzer are located on the top of the module.

Note HP-IB address 31 is an illegal address and should not be used.

Descriptions of the address switches for the microwave transition analyzer are given below.

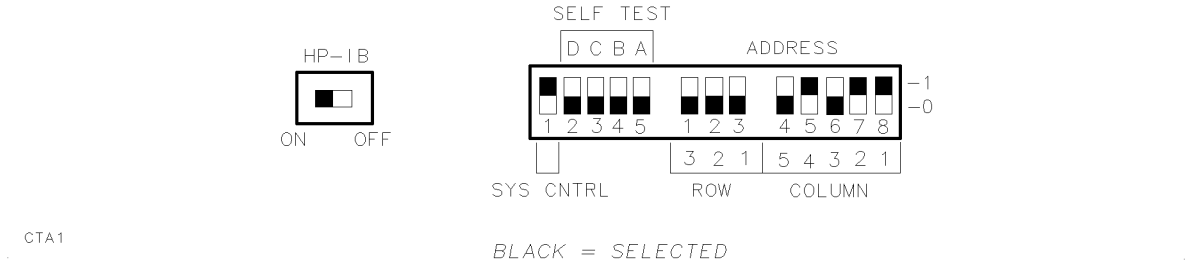


Figure 1-1. HP 71500A Address Switches

HP-IB ON-OFF When this is set to OFF, the microwave transition analyzer is switched off the HP-IB.

Column ADDRESS Switches 1—5 These set the HP-MSIB column address which is also the HP-IB address.

Row ADDRESS Switches 1—3 These set the HP-MSIB row address.

Self-Test Switches

Switch A enables or disables portions of the power-on self test. In the OFF (or 0) position, the full self test is executed. In the ON (or 1) position, various portions of the self test are skipped to keep the test time approximately 10 seconds or less.

Switch B enables or disables the power-on HPMSIB check. In the OFF (or 0) position, the normal power on test is performed. In the ON (or 1) position, the HPMSIB power-up test is skipped.

Caution Disabling this test can prevent correct operation of the system. Switch B is intended for factory or service use only.

Switch C enables or disables looping on RAM test failures. In the OFF (or 0) position, the looping is disabled. In the ON (or 1) position, the RAM data test will loop on failures until the switch is moved to the OFF (or 0) position.

Caution This mode can prevent correct operation of the system. Switch C is intended for factory or service use only.

Switch D enables or disables printing any errors from certain portions of the power-up self test. In the OFF (or 0) position, this mode is disabled. In the ON (or 1) position, the power-up self test will configure the "local" HP-IB as TON (talker only) and certain errors will print out information pertaining to the errors.

Caution This mode can prevent correct operation of the system. Switch D is intended for factory or service use only.

SYS CNTRL (system controller) indicates whether or not the module is the system controller on the "local" HP-IB. If this switch is set in the ON (or 1) position, the module is the system controller and may assert IFC and REN. If this switch is set in the OFF (or 0) position, the module may not assert IFC nor REN. For more details on system controller versus active controller operation refer to Appendix G, "Passing Control".

Soft-Set HP-IB Addresses

The HP-IB address of the microwave transition analyzer master module may be changed from the front panel of the display. The soft-set address remains in effect until power is cycled, or until another soft-set address is entered.

Note Changing the HP-IB address via the display front panel does not affect the position of the modules on the address map.

Use the following procedure to enter a soft-set HP-IB address.

1. Press the (DISPLAY) key on the display front panel.
2. When the display Main Menu appears, press the `ADDRESS MAP` softkey.
3. Use the front-panel knob or step keys to scroll the highlighted rectangle to the box labeled "70820A MTA."
4. When the next menu appears, press `HP-IB ADDRSET`.
5. Enter the new HP-IB address using the numeric keys on the display front panel.
6. Press `ENTER`.

Communication with the System

This section develops some fundamental techniques for controlling the microwave transition analyzer and obtaining sound measurement results. Remote operation of the microwave transition analyzer is controlled with commands that mainly correspond to front-panel softkey functions.

It is important to understand how messages are communicated to the microwave transition analyzer. Therefore, enter and output statements and command syntax will be discussed in this section. It should be noted that HP BASIC is used for all examples in this manual.

Talking to the Instrument

Commands are sent to the instrument with ASCII program messages. These messages are placed on the bus using an output command and passing the device address, program message, and terminator. Passing the device address ensures that the program message is sent to the correct interface and instrument.

Note

The actual `OUTPUT` command you use when programming is dependent on both the controller and programming language that you are using.

Angular brackets "< >" enclose words or characters that symbolize a program code parameter or an HP-IB command.

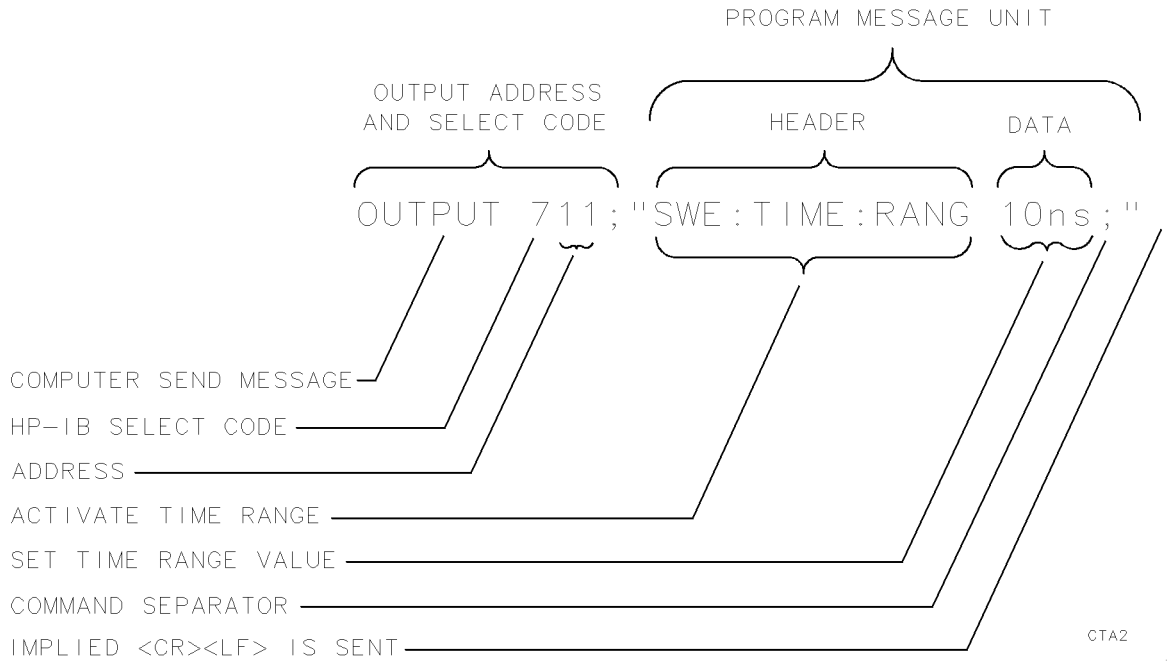
Information that is displayed in quotes represents the information that is sent across the bus. The message terminator is the only additional information that is also sent across the bus.

For HP 9000 Series 200/300 controllers, it is not necessary to type in the actual <terminator> at the end of the program message. These controllers automatically terminate the program message internally.

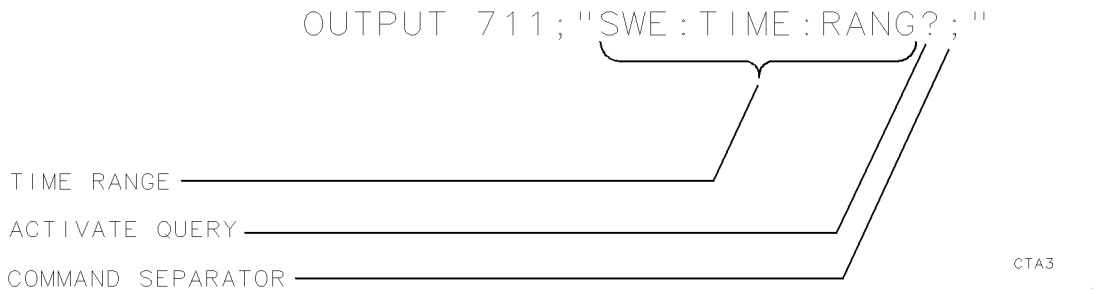
Executing Remote Commands

Microwave transition analyzer programs control the passage of commands and data between the microwave transition analyzer and the computer on the Hewlett-Packard Interface Bus (HP-IB), using HP BASIC **OUTPUT** and **ENTER** statements.

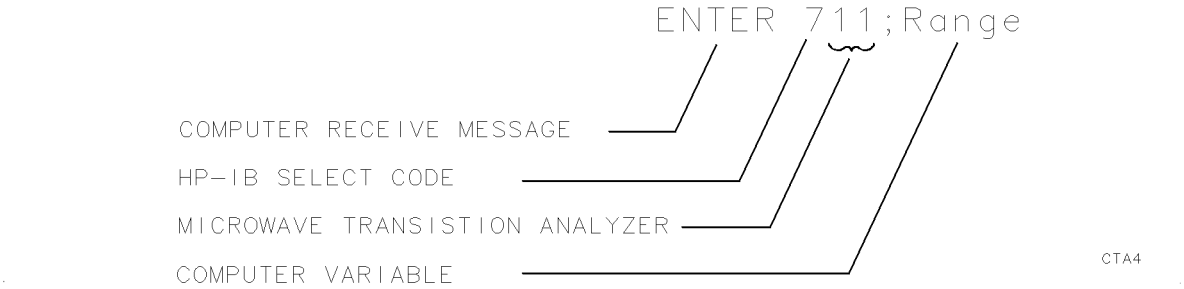
An **OUTPUT** statement tells the computer to send a message to the microwave transition analyzer. For example, executing the output statement below sets the time range to 10 ns.



An **ENTER** statement used in conjunction with a microwave transition analyzer query returns information to the computer. To return the time range value to the computer, first form a query by adding a question mark (?) to the command:



Next, the enter statement can be used to assign the returned value to a variable in the computer:



The value of the time range above is equated to the computer variable "Range". The variable may be printed, stored, or used for any other computer function.

Syntax Requirements

All of the program examples in this manual show the recommended command syntax. All microwave transition analyzer commands must be constructed according to specific syntactical rules which are outlined in Chapter 2, Programming Commands. Chapter 2 lists all of the remote microwave transition analyzer commands in alphabetical order according to each functional group subsystem. A group syntax diagram for each subsystem has been provided in addition to syntax diagrams for each individual command or query.

Longform or Shortform Mnemonic Format

Program commands can be a combination of either uppercase or lowercase ASCII characters, whereas the instrument responses are always returned in uppercase.

Capital letters are used to indicate the shortform (abbreviated spelling) of a command program header that is minimally required. Lowercase letters indicate the longform (complete spelling).

Note When specifying a program command or query, use either the abbreviated spelling indicated by the shortform or the complete spelling of the longform. Any other spelling other than the shortform or longform is not valid.

Both program commands and queries may be used in either longform or in shortform. For example, either of the following program lines changes the timebase to 0.1 seconds per division and delays the on-screen presentation (with respect to the trigger event) by 1 second in time.

```
OUTPUT @Mta; "SWEEP:TIME:RANGE 1;DELAY 1"        ! Longform

OUTPUT @Mta; "SWE:TIME:RANG 1;DEL 1"             ! Shortform
```

Programs written in longform are easily read and are almost self-documenting. The shortform syntax conserves the amount of controller memory needed for program storage and reduces the amount of I/O activity.

Refer to the individual syntax diagrams of each command for both the longform and shortform mnemonic spellings.

Local and Remote Control

The local, remote, and remote with local lockout modes may be used for various degrees of front-panel control while a program is running.

- In local mode, the instrument accepts and executes bus commands and the front panel is entirely active.
- In remote mode, all front-panel input is disabled except for the local key and the line switch.
- In remote with local lockout mode, all controls (except the line switch) are entirely locked out. (Local control can only be restored by the controller.)

Note Cycling the power also restores local control, however, this will also reset certain HP-IB states.

The instrument is placed in remote mode by setting the REN bus control line true, and then addressing the instrument to listen. Sending the local lockout command `LLO` places the instrument in local lockout mode. The instrument can be returned to local mode by either setting the REN line false, or sending the instrument the go-to-local command `GTL`.

Initialization Statements

Programs should begin with an initialization statement which is comprised of a series of HP BASIC and microwave transition analyzer commands to ensure that the bus and all appropriate interfaces are in a known state. The following example shows how to initialize the microwave transition analyzer.

```
10  ASSIGN @Mta to 711
20  CLEAR @Mta
30  OUTPUT @Mta;"*SRE 0;*ESE 0;*RST;"
```

The ASSIGN command is an HP BASIC command that creates an I/O path name and assigns that name to an I/O resource. In the example above, the I/O path name is "@Mta" and is assigned to the device at HP-IB address 11.

Note All program examples in this manual assume that the microwave transition analyzer is addressed at HP-IB address 711.

The ASSIGN command offers several advantages when included in a microwave transition analyzer program. For example, the microwave transition analyzer address is easily changed in the computer program and the program can transfer data to a mass storage unit.

The RESET command, *RST, presets all of the parameters of the microwave transition analyzer and provides a good starting point for all measurement processes. Executing *RST actually is the same as executing a number of microwave transition analyzer commands that set the microwave transition analyzer to a known state.

The CLEAR command is an HP BASIC command that clears the input buffer, the output buffer, and the command parser of the specified instrument. That is, a device on HP-IB is "cleared" so that it is ready for operation. This command may be used to clear devices on the bus singly or in unison. It is often desirable to reset only one instrument so that other instruments on the bus are not affected.

To clear the microwave transition analyzer, the "CLEAR @Mta" statement may be entered into the computer.

To clear all devices at select code 7, the "CLEAR 7" statement may be entered into the computer.

Status Reporting Structure

This section describes and defines the status reporting structure used in the HP 71500A Microwave Transition Analyzer. In general, the status data structure is used to "request service" or indicate a specific condition (for example, operation complete) via SRQ (Service Request). Therefore, this structure may be used to alert the user that certain events have occurred without actually initiating a request for this information. Refer to Figure 1-2 for a model of the status data structure.

Each of the integral parts of the status data structure are described below in more detail.

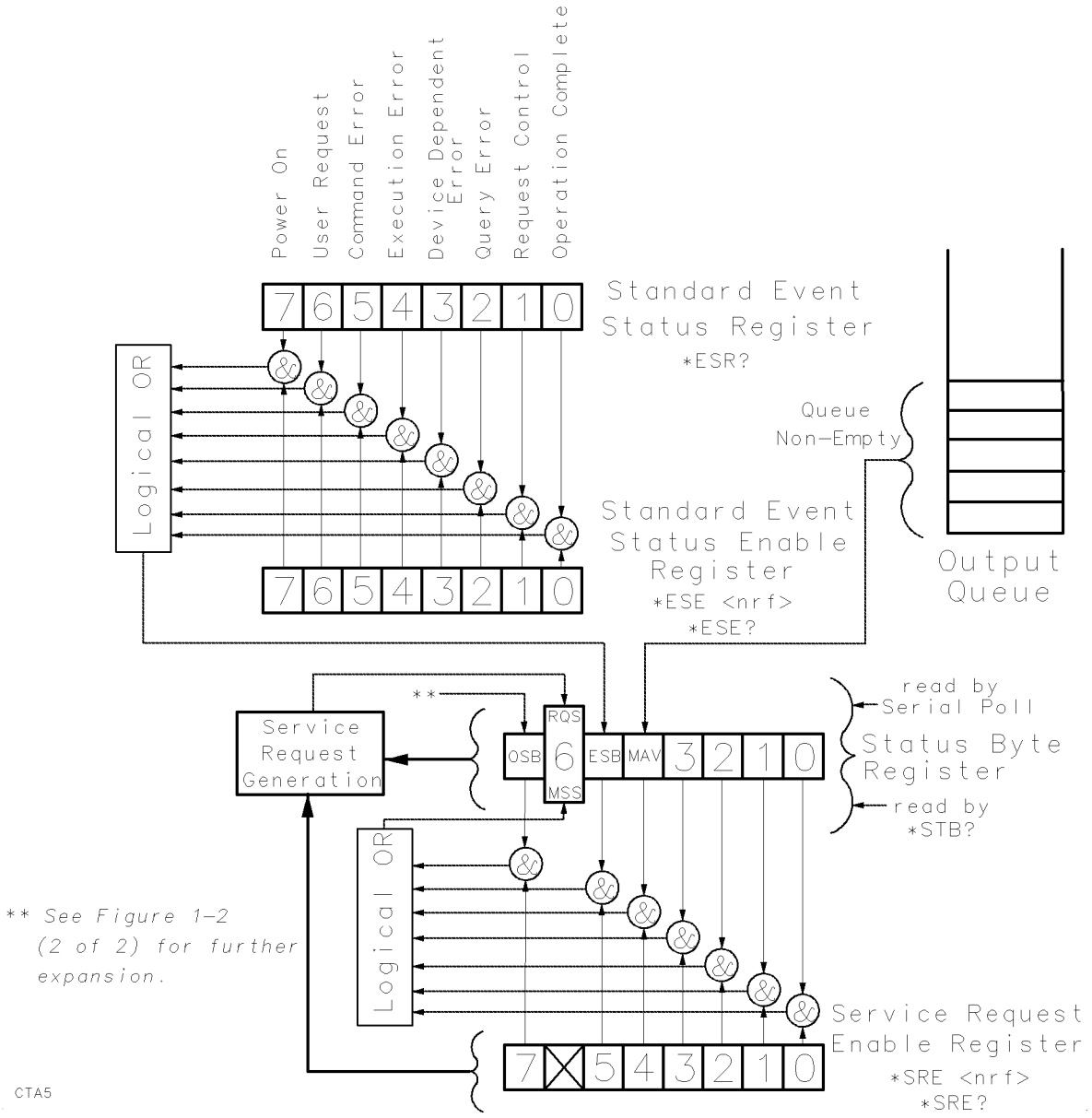


Figure 1-2. Status Data Structure (1 of 2)

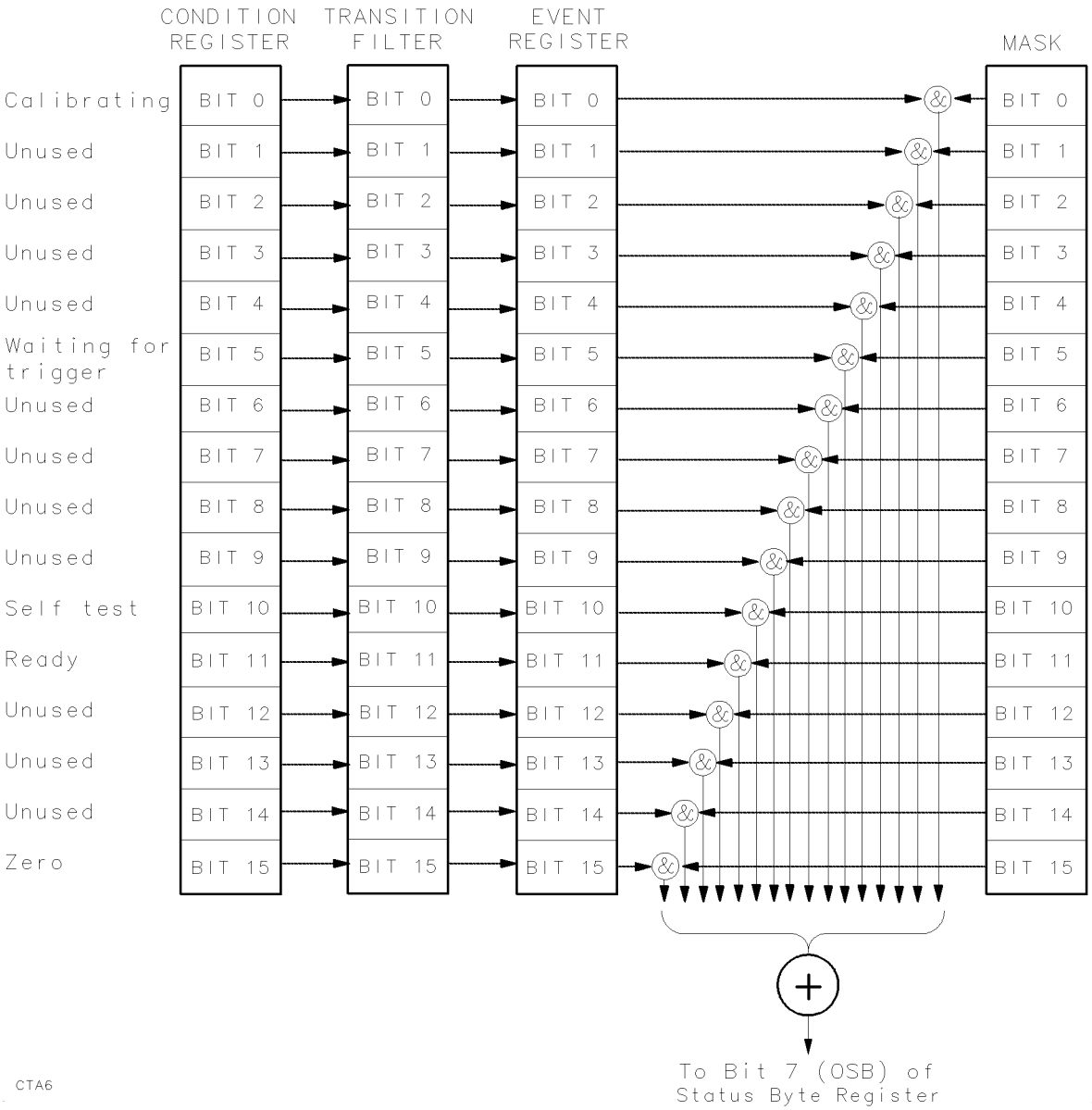


Figure 1-2. Status Data Structure (2 of 2)

Status Byte Register

The Status Byte Register contains the device's Status Byte (STB) summary messages, Request Service (RQS) messages, and Master Summary Status (MSS) messages. See Figure 1-3.

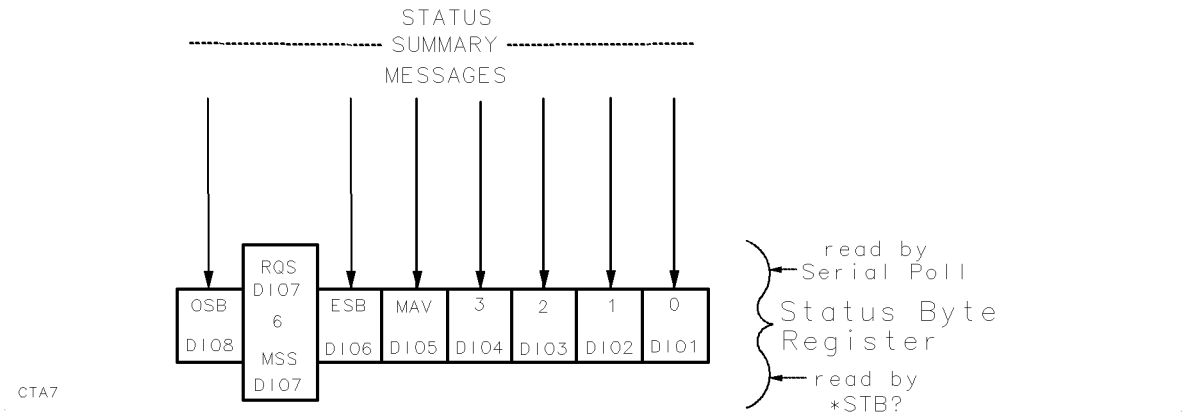


Figure 1-3. Status Byte Register

Note Bits 0 through 3 are not used at this time.

The Status Byte Register can be read with either a serial poll or the READ STATUS BYTE common query (*STB?). Both of these methods read the status byte message identically. However, the value sent for the bit 6 position depends on the method used.

When the Status Byte Register is read with a serial poll "SPOLL (@Mta)", the device returns the 7-bit status byte message plus the single-bit RQS message to the controller as a single data byte.

When the Status Byte Register is read with the *STB? common query, the device returns the 7-bit status byte message plus the single-bit MSS message as a single <NR1 NUMERIC PROGRAM DATA> element. The response to *STB? is identical to the response to a serial poll except that the MSS summary message appears in the bit 6 position in place of the RQS message. The MSS summary message indicates that the device has at least one reason for requesting service.

Serial Poll

When a serial poll of the instrument is requested, the RQS bit is returned on bit 6 of the status byte. An example of how to use the service request by conducting a serial poll of all instruments on the bus is described below.

In this example, assume that there are two instruments on the bus; a microwave transition analyzer at address 11 and a printer at address 1. It is also assumed that we are operating on Interface Select Code 7.

The program command for serial poll using HP BASIC 4.0 is `Stat=SPOLL(711)`. Address 711 is the address of the microwave transition analyzer. The command for checking the printer is `Stat=SPOLL(701)` because the address of the printer is 01 on bus address 7. This command reads the contents of the HP-IB Status Register into the variable called "Stat". At that time, bit 6 of the variable Stat can be tested to see if it is set (that is, bit 6=1).

The serial poll operation can be conducted in the following manner:

1. Enable interrupts on the bus. This allows the controller to "see" the SRQ line.

2. If the SRQ line is high (that is, if some instrument is requesting service), then check the instrument at address 1 to see if bit 6 of its status register is high.
3. Disable interrupts on the bus.
4. To check whether bit 6 of an instrument's status register is high, use the following command line.

```
IF BIT(Stat,6) THEN
```

5. If bit 6 of the instrument at address 1 is not high, then check the instrument at address 11 to see if bit 6 of its status register is high.
6. As soon as the instrument with status bit 6 high is found, check the rest of the status bits to determine what is required.

The SPOLL(711) command causes much more to happen on the bus than simply reading the register. This command clears the bus, automatically addresses the talker and listener, sends SPE (serial poll enable) and SPD (serial poll disable) bus commands, and reads the data.

Standard Event Status Bit (ESB) Summary Message

The ESB summary message is a defined message that appears in bit 5 of the Status Byte Register. Its state indicates whether or not one or more of the enabled events have occurred since the last reading or clearing of the Standard Event Status Register. Refer to Figure 1-2.

The ESB summary message is TRUE when an enabled event in the Standard Event Status Register is set TRUE. Conversely, the ESB summary message is FALSE when no enabled events are TRUE.

Message Available (MAV) Queue Summary Message

The MAV summary message is a defined message that appears in bit 4 of the Status Byte Register. The state of the message indicates whether or not the Output Queue is empty. Whenever the device is ready to accept a request by the controller to output data bytes, the MAV summary message shall be TRUE. The MAV summary message shall be FALSE when the Output Queue is empty. Refer to Figure 1-2.

The MAV summary message is used to synchronize information exchange with the controller. The controller can, for example, send a query command to the device and then wait for MAV to become TRUE. The system bus is available for other use while an application program is waiting for a device to respond. If an application program begins a read operation of the Output Queue without first checking for MAV, all system bus activity is held up until the device responds.

Note Due to the operation of HP-MSIB, the MAV bit is not implemented when using HP-MSIB.

Operation Status Bit (OSB) Summary Message

The OSB summary message is a defined message that appears in bit 7 of the Status Byte Register. Its state indicates whether or not one or more of the enabled events have occurred since the last reading or clearing of the Standard Operation Status Register. Refer to Figure 1-2.

The OSB summary message is TRUE when an enabled event in the Standard Operation Status Register is set TRUE. Conversely, the OSB summary message is FALSE when no enabled events are TRUE.

Clearing the Status Byte Register

The CLEAR STATUS (*CLS) common command causes the Event Registers and Queues of the status data structure to be cleared so that the corresponding summary messages are clear. The Output Queue and its MAV summary message are an exception and are unaffected by *CLS.

Service Request Enable Register

The Service Request Enable Register is an 8-bit register that can be used by the programmer to select which summary messages in the Status Byte Register may cause service requests. The programmer may select reasons for the device to issue a service request by altering the contents of the Service Request Enable Register. Refer to Figure 1-4.

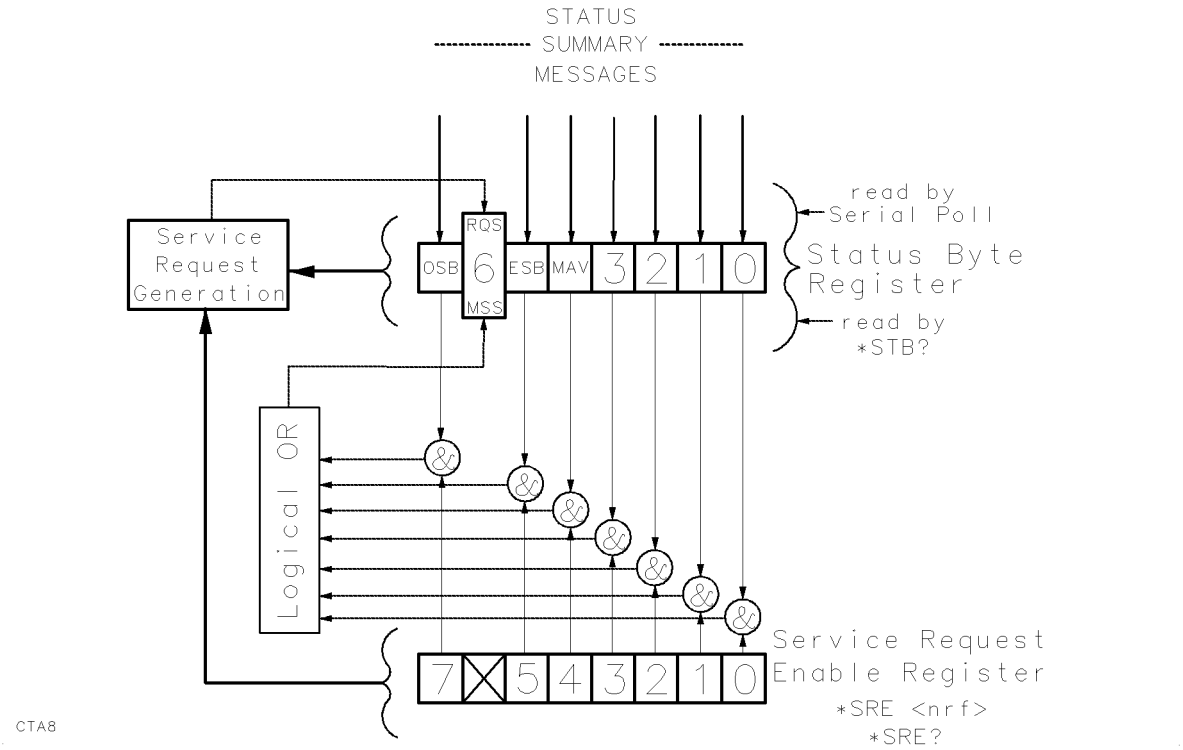


Figure 1-4. Service Request Enable Register

Reading/Writing the Service Request Enable Register

The Service Request Enable Register is read with the SERVICE REQUEST ENABLE (*SRE?) common query. The response message to this query is a <NR1 NUMERIC PROGRAM DATA> element that represents the sum of the binary-weighted values of the Service Request Enable Register (2 raised to the power of the bit number). The value of unused bit 6 is always zero.

The Service Request Enable Register is written to with the SERVICE REQUEST ENABLE (*SRE) common command followed by a <DECIMAL NUMERIC PROGRAM DATA>

element. The <DECIMAL NUMERIC PROGRAM DATA>, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the Service Request Enable Register. A bit value of one indicates an enabled condition. A bit value of zero indicates a disabled condition.

Note *SRE is not affected by *RST.

The device always ignores the value of bit 6.

Service Request Generation

The Service Request function provides the device with the capabilities of requesting service from the controller via the Service Request interface (SRQ, a line on HP-IB) and reporting that it has requested service via the Request Service message (RQS, bit 6 of the Status Byte Register).

The generation of service requests ensures that the device shall:

1. Assert an SRQ when a previously "enabled" condition occurs.
2. Keep SRQ asserted until the controller has recognized the service request and polled the device, or has taken specific action to cancel the request (for example, *CLS command).
3. Release SRQ when polled so that the controller can detect an SRQ from another device.
4. Assert an SRQ again if another condition occurs, whether or not the controller has cleared the first condition. If the previous condition has not been cleared, the next condition must be different than the first for an SRQ to be asserted again.

Whenever the contents of the Status Byte Register or the Service Request Enable Register are changed, the device must determine whether the change affects the service request state of the device. Device status transitions do not affect the state of the SRQ interface directly. Instead, changes to the Status Byte Register and the Service Request Enable Register generate two local messages which either assert (Request Service TRUE) or unassert (Request Service FALSE) the hardware.

The device shall generate a new service request (assert Request Service TRUE) when:

1. A bit in the Status Byte Register changes from FALSE to TRUE while the corresponding bit in the Service Request Enable Register is TRUE.
2. A bit in the Service Request Enable Register changes from FALSE to TRUE while the corresponding bit in the Status Byte Register is TRUE.
3. A bit in the Status Byte Register changes from FALSE to TRUE and the corresponding bit in the Service Request Enable Register changes from FALSE to TRUE simultaneously.

In general, the controller application program must never assume that an SRQ indicates that a new reason for service has occurred, but only that a new reason for service may have occurred, and that the application program should check the device Status Byte Register to determine whether this is indeed the case.

Clearing the Service Request Enable Register

The SERVICE REQUEST ENABLE (*SRE) common command followed with a <DECIMAL NUMERIC PROGRAM DATA> element value of zero clears the Service

Request Enable Register. A cleared register does not allow status information to generate a hardware Request Service message and thus, no service requests are issued.

Standard Event Status Register

The Standard Event Status Register structure has specific defined events assigned to specific bits. Refer to Figure 1-5.

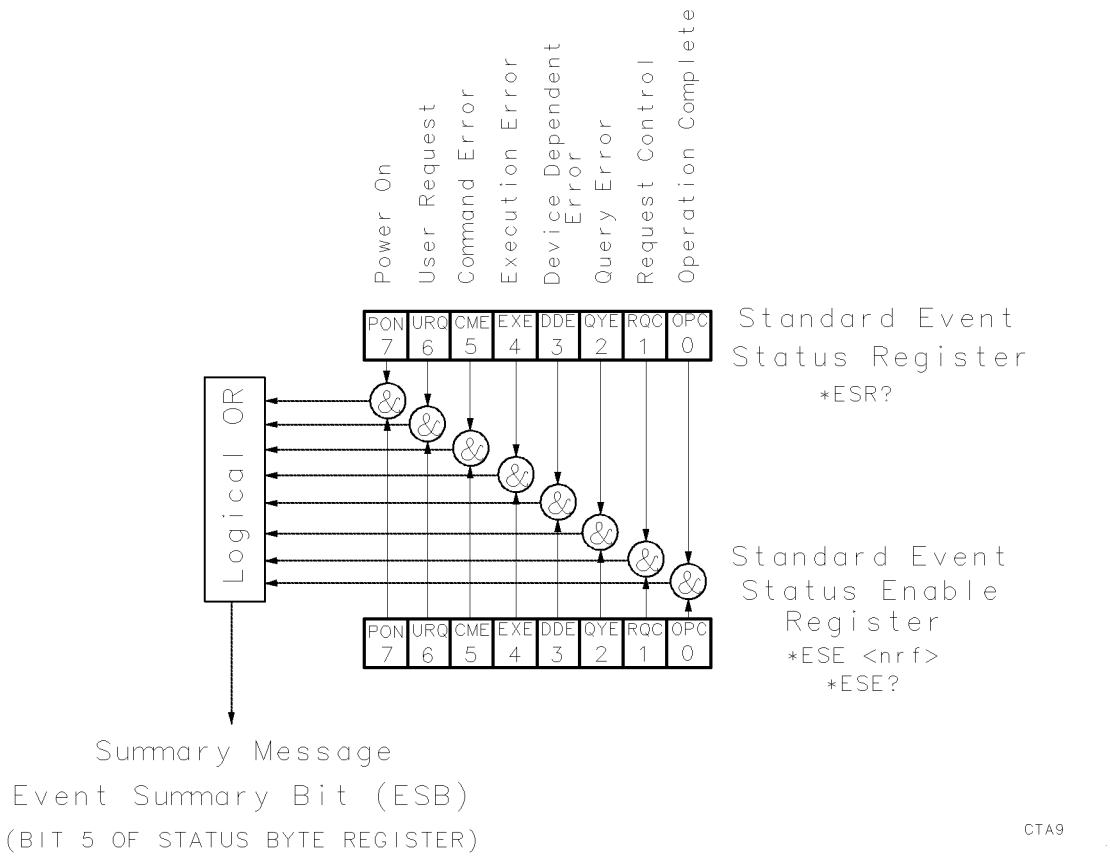


Figure 1-5. Standard Event Status Register

Bit 7 - Power On (PON) This event bit indicates that an off-to-on transition has occurred in the power supply of the device.

Bit 6 - User Request (URQ) This event bit is not used at this time.

Bit 5 - Command Error (CME) This event bit indicates that one of the following events has occurred:

1. A syntax error (controller-to-device message) has been detected. Possible errors include a data element which violates the device listening formats or whose type is unacceptable to the device.
2. A semantic error has occurred indicating that an unrecognized header was received. Unrecognized headers include incorrect device-dependent headers and incorrect or unimplemented common commands.

3. A Group Execute Trigger (GET) was entered into the Input Buffer inside of a <PROGRAM MESSAGE>. A GET message is a controller-to-device message defined as an addressed command.

The Command Error bit is not set to report any other device-dependent condition. Events that are reported as Command Errors also are not reported as Execution Errors, Query Errors, or Device-Dependent Errors. Refer to the appropriate bit definitions for more information.

Bit 4 - Execution Error (EXE)

This event bit indicates that:

1. A <PROGRAM DATA> element following a header was evaluated by the device as outside of its legal input range, or is otherwise inconsistent with the capabilities of the device.
2. A valid program message could not be properly executed due to some device condition.

Following an Execution Error, the device continues to process the input stream.

Execution errors are reported by the device after rounding and expression evaluation operations have taken place. For example, rounding a numeric data element is not reported as an Execution Error.

Events that generate Execution Errors do not generate Command Errors, Query Errors, or Device-Dependent Errors. Refer to the appropriate bit definitions for more information.

Bit 3 - Device-Dependent Error (DDE)

This event bit indicates that an error has occurred which is neither a Command Error, a Query Error, nor an Execution Error.

A Device-Dependent Error is any executed device operation that did not properly complete due to some condition, such as overrange.

Following a Device-Dependent Error, the device continues to process the input stream.

Events that generate Device-Dependent Errors do not generate Command Errors, Query Errors, or Execution Errors. Refer to the appropriate bit definitions for more information.

Bit 2 - Query Error (QYE)

This event bit indicates that:

1. An attempt is being made to read data from the Output Queue when no output is either present or pending.
2. Data in the Output Queue has been lost.

The Query Error bit is not set to report any other condition. Events that generate Query Errors do not generate Execution Errors, Command Errors, or Device-Dependent Errors.

Bit 1 - Request Control (RQC)

This event bit is not used at this time.

Bit 0 - Operation Complete (OPC) This event bit is generated in response to the OPERATION COMPLETE (*OPC) common command. It indicates that the device has completed any pending operations and that the parser is ready to accept more program messages. The parser is the logical portion of the device which takes Data Byte Messages, END messages, and hardware Group Execute Trigger messages from the Input Buffer and analyzes them by separating out the various syntactic elements.

Reading/Writing the Standard Event Status Register

The Standard Event Status Register is destructively read (that is, read and cleared) with the STANDARD EVENT STATUS REGISTER common query (*ESR?).

The Standard Event Status Register cannot be written to remotely except to clear it.

Clearing the Standard Event Status Register

The Standard Event Status Register shall only be cleared by:

- A CLEAR STATUS common command (*CLS).
- A power-on sequence which initially clears the Standard Event Status Register then records any subsequent events during the power-on sequence of the device including setting the PON event bit (Bit 7).

Standard Event Status Enable Register

The Standard Event Status Enable Register allows one or more events in the Standard Event Status Register to be reflected in the ESB summary-message bit (Bit 5 of the Status Byte Register). This register is defined for 8 bits, each corresponding to the bits in the Standard Event Status Register. Refer to Figure 1-5.

Reading/Writing the Standard Event Status Enable Register

The Standard Event Status Enable Register is read with the STANDARD EVENT STATUS ENABLE common query (*ESE?). Data is returned as a binary-weighted <NR1 NUMERIC RESPONSE DATA>.

The Standard Event Status Enable Register is written to by the STANDARD EVENT STATUS ENABLE common command (*ESE). Data is encoded as <DECIMAL NUMERIC PROGRAM DATA>.

Clearing the Standard Event Status Enable Register

The Standard Event Status Enable Register shall be cleared by the following:

- Sending the *ESE common command with a data value of zero.
- A power-on event.

The Standard Event Status Enable Register is specifically not affected by the RESET common command (*RST).

Output Queue

The Output Queue stores response messages until they are read. The availability of the output is summarized by the Message Available (MAV) summary message (bit 4 of the Status Byte Register). The MAV summary message is used to synchronize information exchange with the controller. Refer to Figure 1-6.

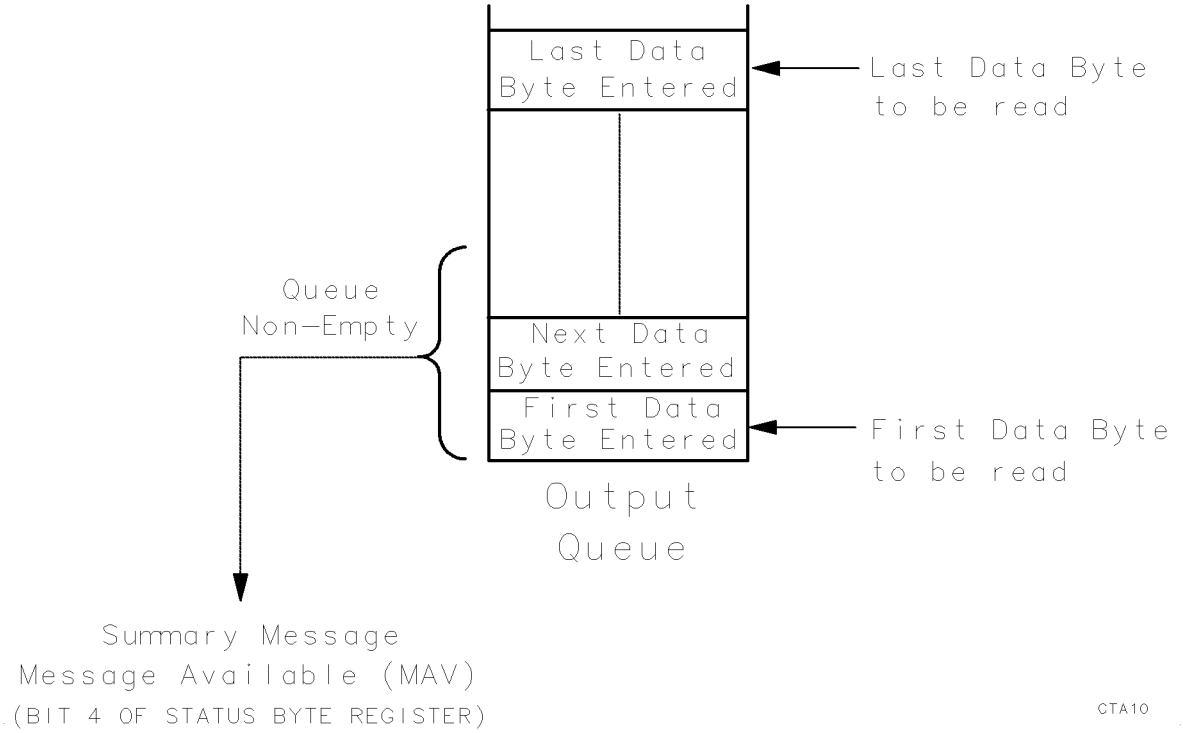


Figure 1-6. Output Queue

The Response Formatter places Data Byte Messages and END messages into the Output Queue in response to query commands. These bytes are removed from the Output Queue as they are read by the controller. As long as the Output Queue contains one or more bytes, MAV is TRUE.

The Output Queue is cleared upon power-on, Device Clear Active State Message (dcas), or the RESET (*RST) common command without causing a Query Error. A Query Error is generated if the contents of the Output Queue are discarded for any other reason.

Standard Operation Status Register

Events in the Standard Operation Status Register may be enabled with the following commands:

- STATus:OPERation:ENABle
- STATus:OPERation:PTRansition
- STATus:OPERation:NTRansition

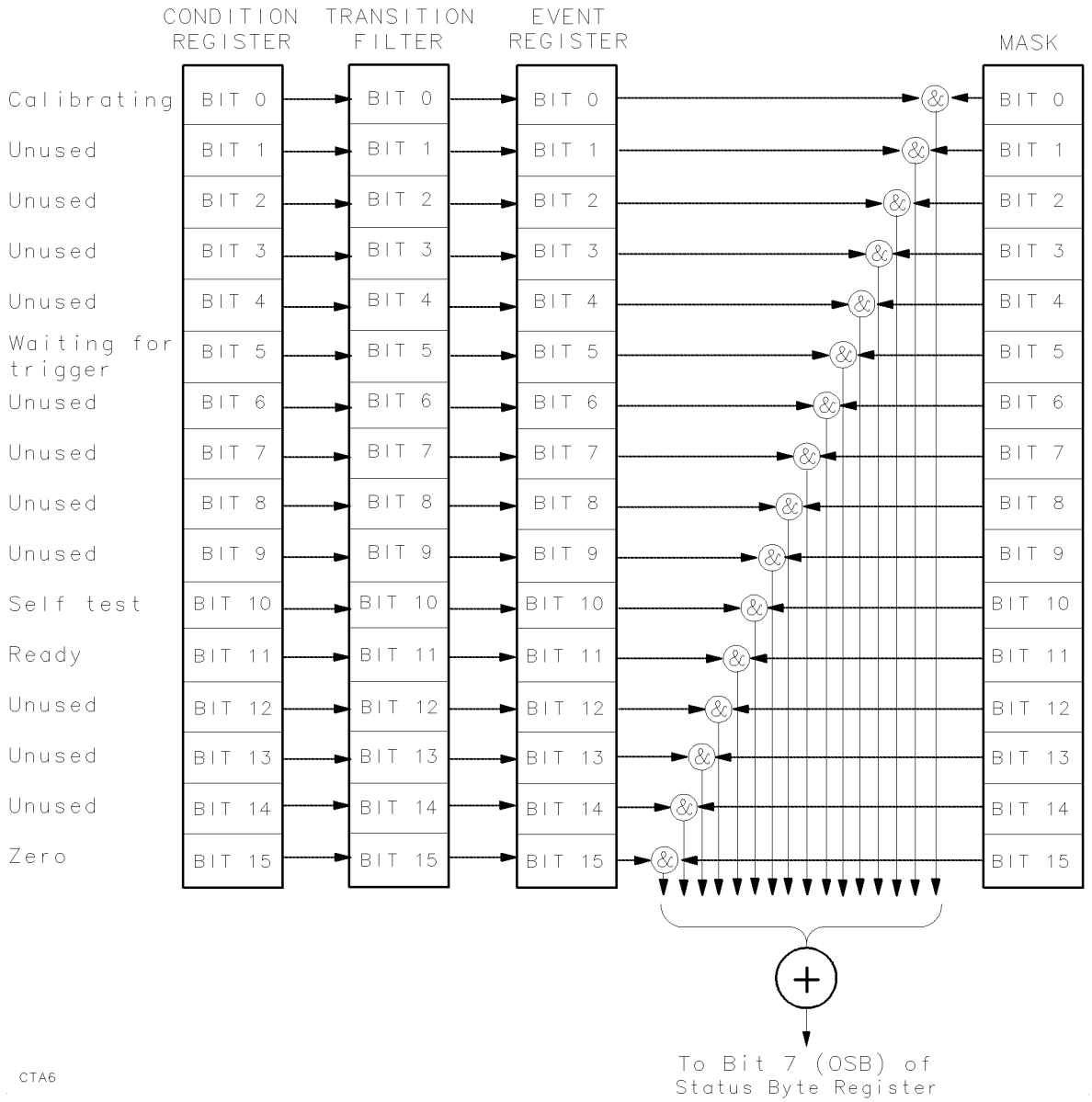


Figure 1-7. Standard Operation Status Structure

The definition of the Operation Status Register bits (see Figure 1-7) is as follows:

- bit 0: calibrating—set at the start of calibration, cleared at the end of calibration
- bit 1: unused (defined as settling)
- bit 2: unused (defined as ranging)
- bit 3: unused (defined as sweeping)
- bit 4: unused (defined as measuring)
- bit 5: waiting for trigger—set when the measurement is waiting for a trigger, cleared when a trigger occurs
- bit 6: unused (defined as waiting for arm)

- bit 7: unused (defined as correcting)
- bit 8: unused (defined as instrument-specific)
- bit 9: unused (defined as instrument-specific)
- bit 10: self test in progress (instrument-specific)—set at the start of self-test, cleared at the end of self-test
- bit 11: ready (instrument-specific)—set by the :READy command, cleared by a program message terminator
- bit 12: unused (defined as instrument-specific)
- bit 13: unused (reserved)
- bit 14: unused (defined as user-defined program running)
- bit 15: always zero

Synchronization of Events and Commands

This section describes techniques which may be used to insure synchronization between events and commands, which in turn insures valid measurements.

A potential problem with commands that take appreciable time to finish is that the application program needs to know when the commands have finished. Therefore, to make measurements remotely, the controller must know when trace data is available to the microwave transition analyzer before making a measurement. This potential problem can be avoided by using synchronization commands that instruct the module to wait until a measurement is finished before making any other measurements.

For example, consider a data-logging device which is commanded to take a measurement with the RUN command and then to make a frequency measurement using MEAS:FREQ?. The RUN command is a command that allows execution of subsequent commands while the device operations initiated by the RUN command are still in progress. The RUN command therefore takes appreciable time to perform. Figure 1-8 shows a timing diagram of this operation without the use of a synchronization command.

It should be noted that, without the use of a synchronization command, the RUN command is still in progress when the FREQUENCY measurement is initiated. If the RUN command has not completed its operation, any data used to make other measurements (for example, FREQUENCY) while the RUN command is still in progress may be invalid. Only after an operation has been completed can the data be assumed to be valid.

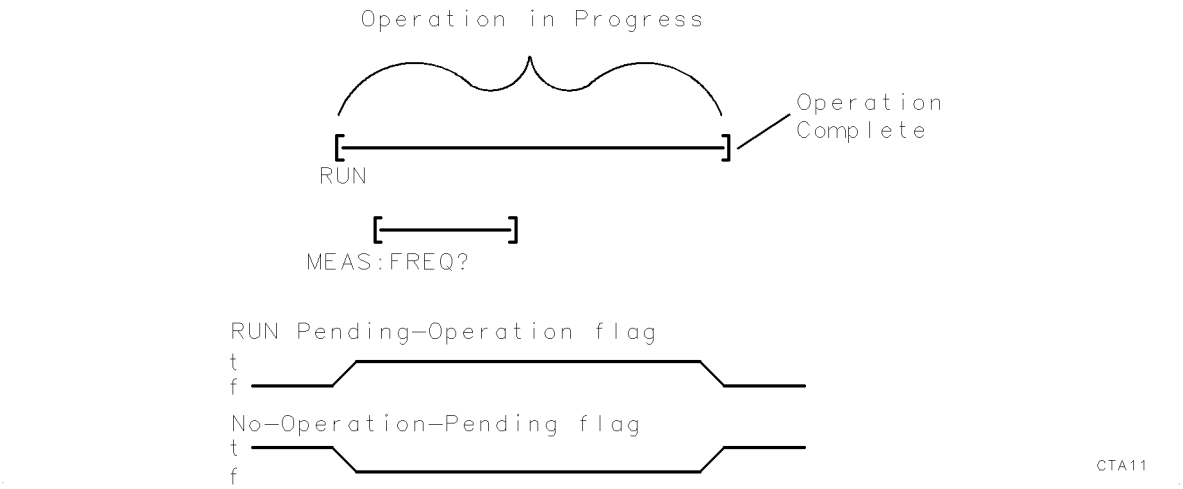


Figure 1-8. Timing Diagram Without A Synchronization Command

Figure 1-9 illustrates the same example as described above except that a synchronization command is used. It should be noted that, with the use of a synchronization command, the RUN command has completed its operation before the FREQUENCY measurement is initiated. Therefore, the data used to make the FREQUENCY measurement is valid.

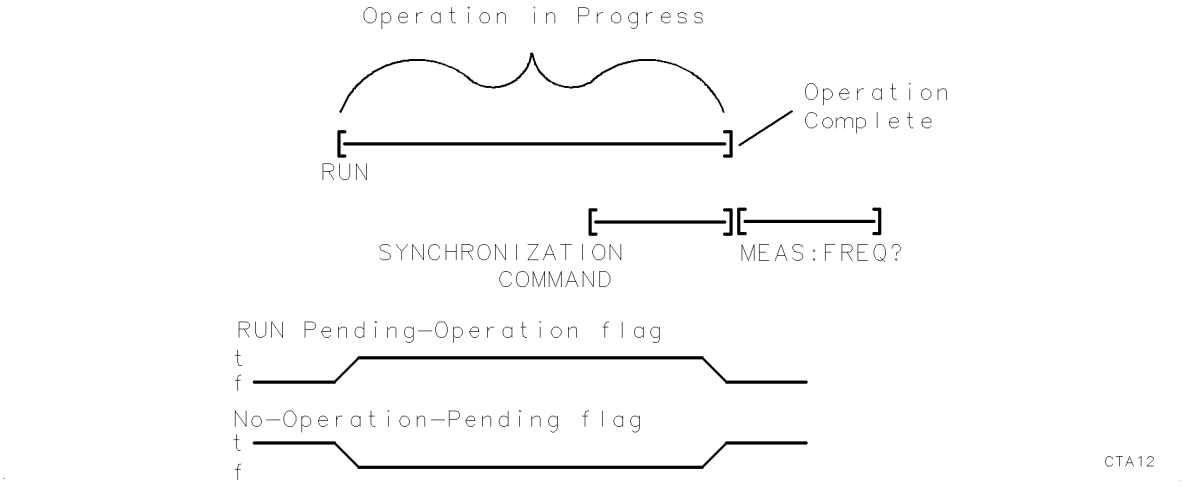


Figure 1-9. Timing Diagram Using A Synchronization Command

General Sequence

The following example HP BASIC statements illustrate the general sequence that should be followed when making measurements via remote control.

1. OUTPUT @Mta;"*RST; :SWE:MODE ASIN; ... "

   RESET the module, put it into SINGLE SWEEP mode, then set up all desired parameters (for example, voltage range, time range, and so on).

2. OUTPUT @Mta;"RUN"

   Invoke a sweep. (In SINGLE SWEEP mode, a single measurement will be taken.)

3. SYNCHRONIZATION COMMAND Use a synchronization command to allow the module to complete one operation before starting another. Refer to Synchronization Commands below for more detailed information on the three forms of synchronization commands that are available.

4.
```
OUTPUT @Mta;"MEAS:SOUR CHAN1;FREQ?"
ENTER @Mta;Frequency
```

Ask for and enter measurement data.

Synchronization Commands

There are three forms of synchronization commands that can be used in the general sequence for remote measurements listed above. The three forms are listed and described below.

- **Wait-To-Continue (*WAI)** This command instructs the module to not use the bus until all operations are complete. If using triggered sweeps, this command may cause the bus to "wait" indefinitely if the trigger does not occur.
- **Operation Complete (*OPC)** This command instructs the module to send a "1" to the Output Queue of the status data structure when all operations are complete. Similar to the *WAI command, the *OPC command may result in the bus waiting indefinitely for an event to occur.
- **Assert SRQ (Service Request)** This form of synchronization uses a sequence of commands indicating that after the *OPC query is received, the module is to assert SRQ when all operations are complete. This method has the advantage of the HP-IB never being in a state where it is waiting indefinitely. The sequence of commands for this synchronization method is:

```
OUTPUT @Mta;"*ESE 1;"                ! This command only needs to be
sent once
OUTPUT @Mta;"*SRE 32;"               ! This command only needs to be
sent once
OUTPUT @Mta;"*CLS;*OPC;"
REPEAT
UNTIL BIT (SPOLL(@Mta),6)=1
```

Note "All operations complete" is defined as 1) there is no trace in progress, and 2) all remote commands have been parsed and processed.

Data Transfer

The microwave transition analyzer represents trace data and non-trace data differently. For non-trace data, the microwave transition analyzer will send information back in ASCII character code as either text or numeric data. As an example of ASCII text, the SWEEP:MODE query (SWE:MODE?) returns ASIN|SING|AUTO|TRIG as a response. As an example of ASCII numeric data, the SWEEP:TIME RANGE query (SWE:TIME:RANG?) returns a value X.XXEXX as a response.

Note Some numeric responses are integers (for example, no decimal point or exponent).

For trace data, the microwave transition analyzer supports 16-bit binary, 32-bit binary, and ASCII transfers (MSByte first, LSByte last). Binary trace data is preceded by a header to indicate that binary data is about to be received. The header is "#0" which indicates indeterminate length binary data. When using the microwave transition analyzer via HP-IB, the last data byte will have the EOI (END OR INTERRUPT) status line asserted with it. Listed below is an HP BASIC example program that reads trace data from the microwave transition analyzer.

```
OUTPUT @Mta;"WAV:DATA?"                    ! Request trace data.
ENTER @Mta USING "#,2A";Header$            ! Read the header.
FOR J=1 TO Points
  ENTER @Mta USING "#,W";Value(J)          ! Read each point.
NEXT J
```

2

Language Reference Information

Introduction

This chapter contains organization and syntax information for the microwave transition analyzer commands. The programming commands are divided into three types: common commands, top-level commands, and subsystem commands. A command syntax summary is shown at the beginning of each subsystem.

The commands or queries of each subsystem are listed in alphabetical order according to their mnemonics. A syntax diagram, parameter information, and functional description is provided for each command; however if more detailed information regarding a command function is necessary, refer to Chapter 3, "Key Definitions" in the HP 71500A/HP 70820A Microwave Transition Analyzer Reference Manual.

Note Either the longform or shortform spelling of the command may be used in programming.

The subsystems in the microwave transition analyzer are:

ACQUIRE	defines measurement conditions.
BASIC CALIBRATION	starts the instrument calibration sequence, retrieves calibration data, or reloads calibration data.
CHANNEL	controls all vertical or Y-axis functions of the input channels.
CONFIGURE	specifies the method for acquiring slave modules at power on.
DAC	controls the 12-bit DAC output.
DIAGNOSTICS DISPLAY	controls the screen format and how waveforms, data, and text are displayed on-screen.
FUNCTION	controls the waveform math functions.
HISTOGRAM	allows various statistics of the input channels or function to be measured.
LIMIT	controls masks for pass/fail testing.
MARKER	controls the two cursor markers.
MASS MEMORY	provides mass storage capabilities for the HP 70820A.
MEASURE	makes various waveform measurements.
MODULATOR SOURCE	controls the internal pulse generator.

PEAK POWER METER	*Available with Option H10 only.* controls the operation of the HP 70820A as a statistical peak power meter.
PROGRAM	controls HP Instrument BASIC.
RF PULSE	controls the various parameters relating to RF pulses.
SECURE	provides a convenient method for destroying or blanking module information.
SOURCE	controls the configuration and settings of an external source.
STATUS	accesses the non-IEEE 488.2 status structures for this instrument.
SWEEP	controls all horizontal or X-axis functions.
SYSTEM	returns the state of the microwave transition analyzer system.
TABLE	controls tabular data measurements.
TRIGGER	defines and controls trigger conditions.
WAVEFORM	provides access to waveform data and determines trace parameters.
WINDOW	specifies the FFT and math windows to be used.
XLINE, YLINE	provides access to line markers.

Notation Conventions

Pictorial

The following general guidelines refer to the pictorial syntax diagrams for each functional subsystem in this manual.

- All items enclosed by a rounded envelope are case insensitive (that is, either uppercase or lowercase letters may be used) and must be entered exactly as shown; however, the minimum number of characters is shown in uppercase.
- Items enclosed by a rectangular box indicate parameters used in the command sequence. A description of each parameter is given in the respective command descriptions.
- Command sequence items are connected by lines. Each line can be followed in only one direction, as indicated by an arrow at the end of each line.
- Any combination of command sequence items that can be generated by following the lines in the proper direction is syntactically correct.
- A command sequence item is optional if there is a valid path around it.

Textual Notation

CAPITAL LETTERS	Capital letters are used to indicate the shortform of a command program header that is minimally required. Lowercase letters indicate the longform (complete spelling).
< >	Angular brackets are used to enclose elements of the language being defined. Refer to "Functional Syntax Conventions" below for an explanation of information contained within these brackets.

[ ] Square brackets are used to enclose optional information not required for execution of the command sequence.

| The vertical bar can be read as "or" and is used to separate alternative parameter options.

{ } Braces (or curly brackets) are used to enclose one or more parameters that may be included zero or more times.

Functional Syntax Conventions

Functional syntax is required to create program messages that are transmitted to a device. Program messages are composed of sequences of program message units, each unit representing a program command or query. Each program command or query is composed of a sequence of functional syntactic elements. Legal program commands and queries are created from functional element sequences generated by using the functional syntax diagrams.

Terminated program messages are complete "controller-to-device" messages. They are sequences of zero or more <program message unit> elements. The <program message unit> element represents a programming command or data sent to the device from the controller. See Figure 2-1.

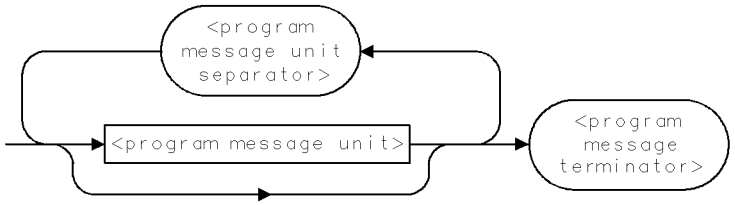


Figure 2-1. Terminated <program message> Functional Element Syntax

Example: terminated <program message>

```
RUN;STOP<LF>
```

A <program message unit> element is defined in more detail as either a <command message unit> or <query message unit>. See Figure 2-2.

Figure 2-2. <program message unit> Functional Element Syntax

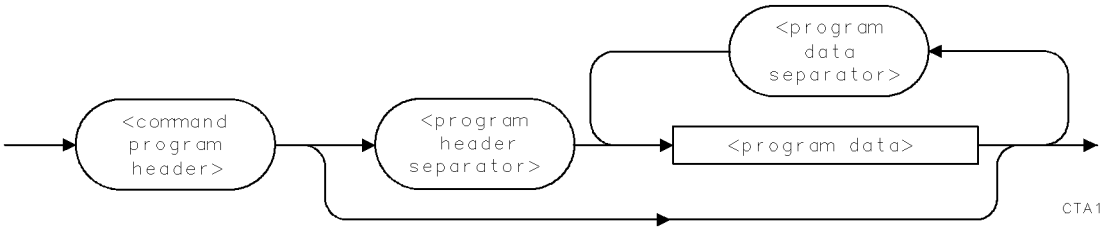


Figure 2-3. <command message unit>

Example: <command message unit>

```
WINDOW:USER 2,.5,.5,-.5
```

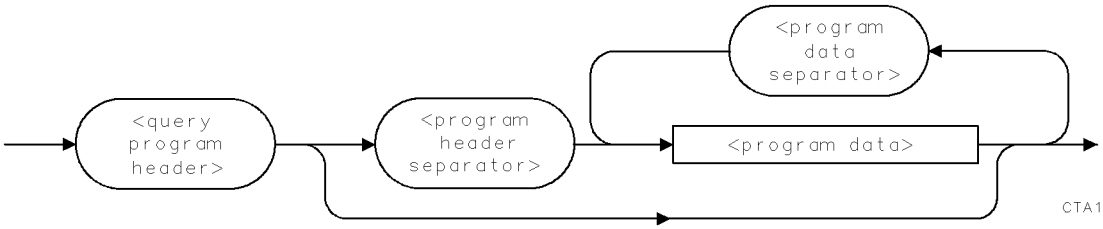
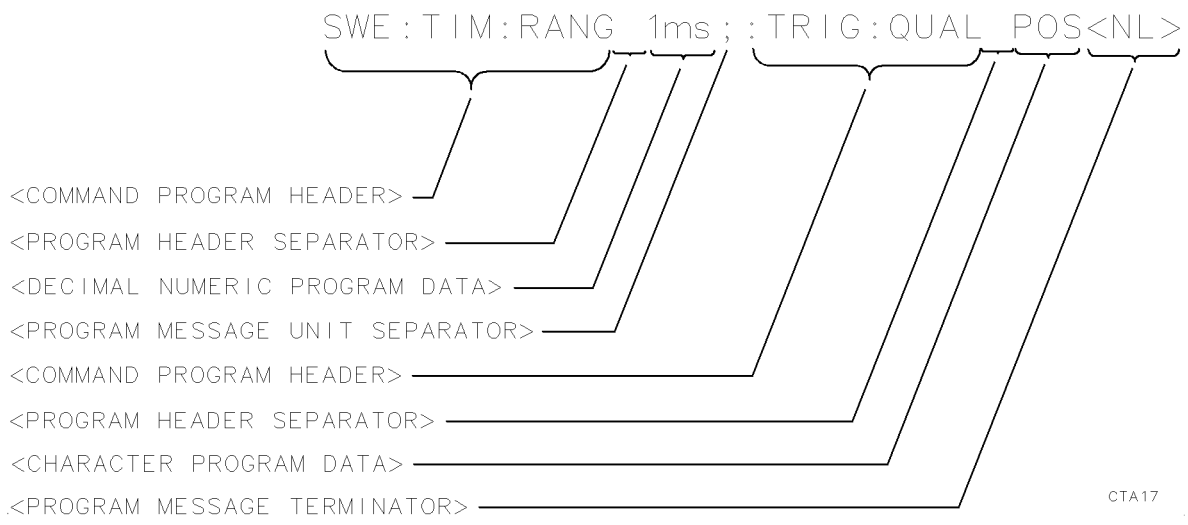


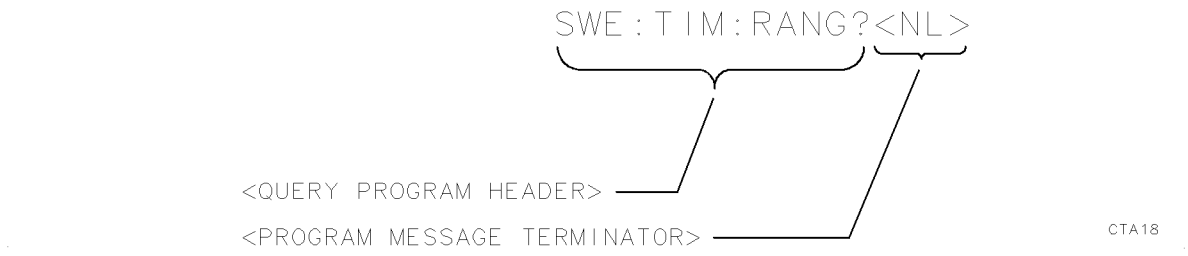
Figure 2-4. <query message unit>

Example: <query message unit>

```
MEASURE:PVOL? .02,3
```

The syntax for typical command and query sequences is shown below:





Separator Functional Elements

The three separator functional elements that are used in the functional syntax of the microwave transition analyzer system are:

- <program message unit separator>
- <program data separator>
- <program header separator>

<program message unit separator>. The <program message unit separator> separates sequential <program message unit> elements from one another within a <program message>. See Figure 2-5.

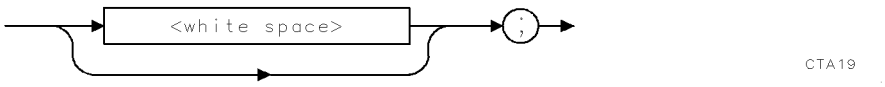


Figure 2-5. <program message unit separator>

where <white space> is defined as

where <white space character> is defined as a single ASCII-encoded byte in the range 00-09, 0B-20 hexadecimal (0-9, 11-32 decimal). This range includes the ASCII-control characters and the space, but excludes the new line (NL).

<program data separator>. The <program data separator> separates sequential <program data> elements from one another after a <command program header> or <query program header>. It is used when a <command program header> or <query program header> has multiple parameters.

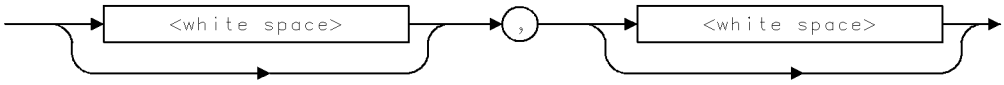


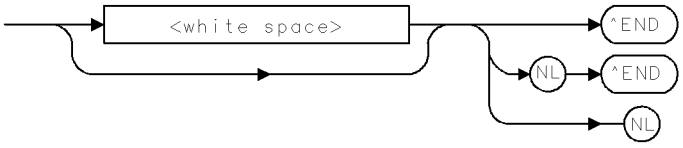
Figure 2-6. <program data separator>

<program header separator>. The <program header separator> separates the <command program header> or <query program header> from the <program data> elements.

Figure 2-7. <program header separator>

Terminator Functional Element

A <program message terminator> terminates a <program message> which is a sequence of one or more definite-length <program message unit> elements. A <program message terminator> is defined as



where ^END is used to indicate concurrent transmission of the END message with the last preceding data byte and,

where NL (new line) is defined as a single ASCII-encoded byte 0A (10 decimal).

Program Header Functional Elements

Program header functional elements represent the operation to be performed in a device. These elements serve as the headers of commands or queries and are defined as either a <command program header> or a <query program header>.

A <command program header> is defined as

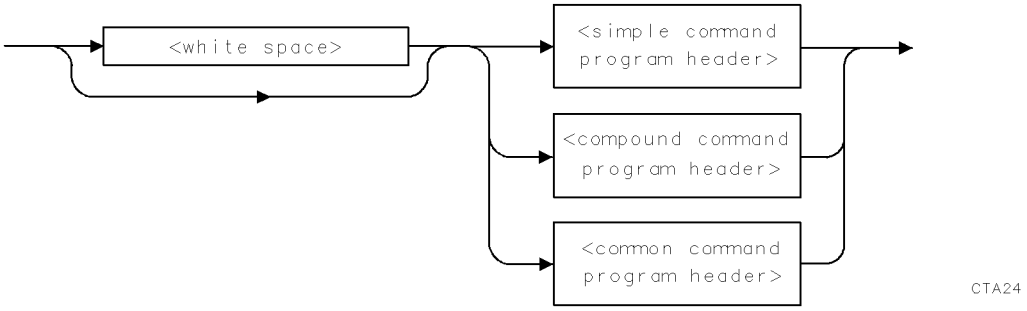
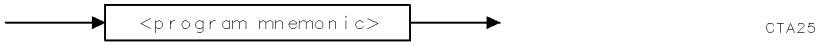


Figure 2-8. <command program header>

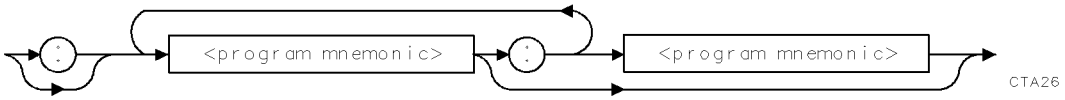
where <simple command program header> is defined as



Example: <simple command program header>

```
RUN
```

where <compound command program header> is defined as



Example: <compound command program header>

```
SWEEP:TIME:RANGE
```

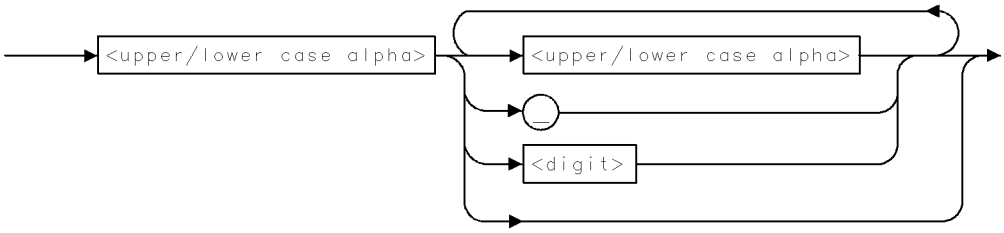
where <common command program header> is defined as

CTA27

Example: <common command program header>

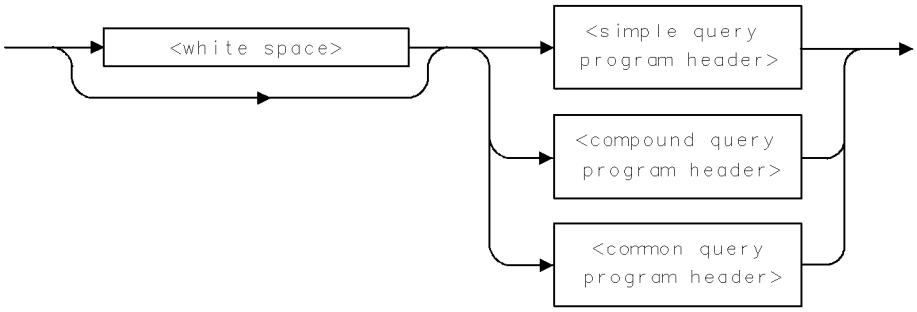
```
*OPC
```

where <program mnemonic> is defined as



CTA28

A <query program header> is defined as



CTA29

Figure 2-9. <query program header>

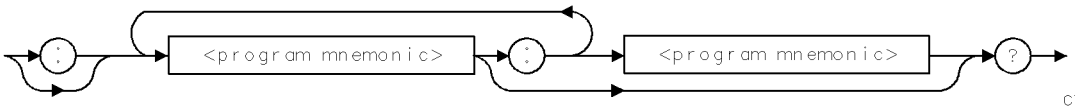
where <simple query program header> is defined as

CTA30

Example: <simple query program header>

```
ERROR?
```

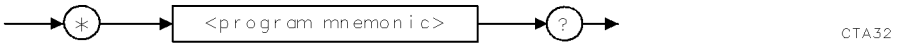
where <compound query program header> is defined as



Example: <compound query program header>

```
SWEEP:TIME:RANGE?
```

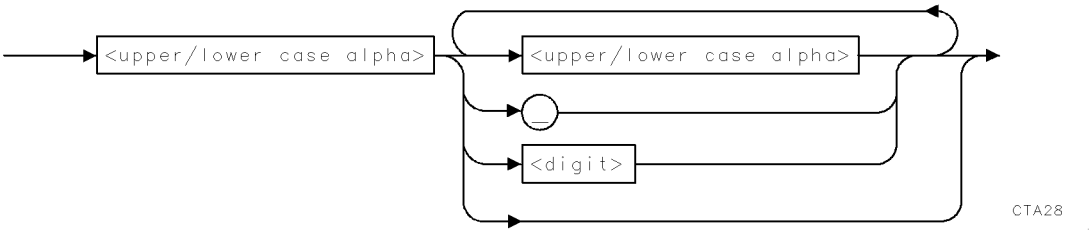
where <common query program header> is defined as



Example: <common query program header>

```
*OPC?
```

where <program mnemonic> is defined as



Program Data Functional Elements

A <program data> functional element represents parameter information sent to the device. The five possible data types are:

- <character program data>
- <decimal numeric program data>
- <string program data>
- <arbitrary block program data>
- <expression program data>

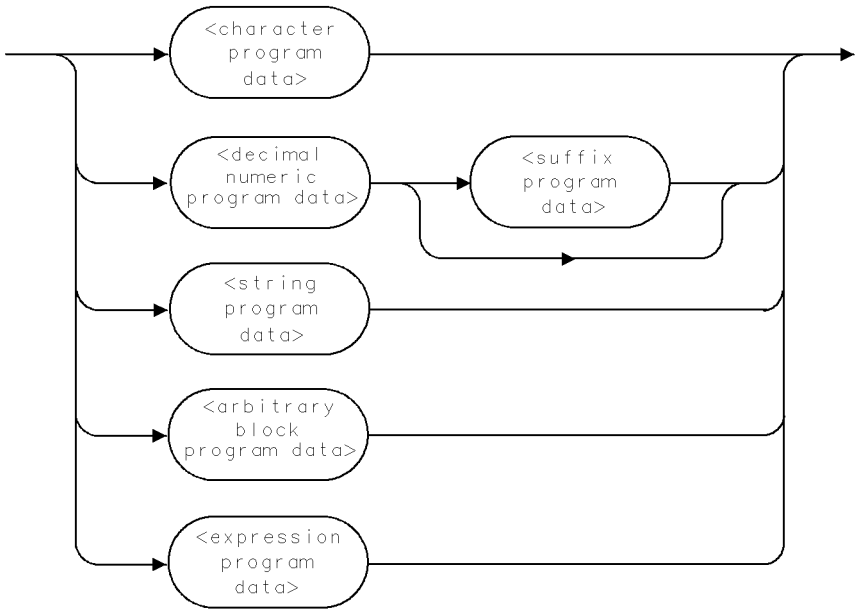
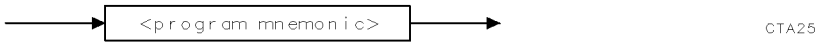


Figure 2-10. <program data> Functional Element Syntax

<character program data>. <character program data> is a data type suitable for sending short mnemonic data, generally where a numeric data type is not suitable.

A <character program data> element is defined as



Example: <character program data>

```
VIEW CHAN1
```

<decimal numeric program data>. <decimal numeric program data> is a data type suitable for sending decimal integers or decimal fractions with or without exponents. <suffix program data> is an optional field that may follow and is used to indicate associated multipliers and units (see Figure 2-11).

A <decimal numeric program data> element is defined as

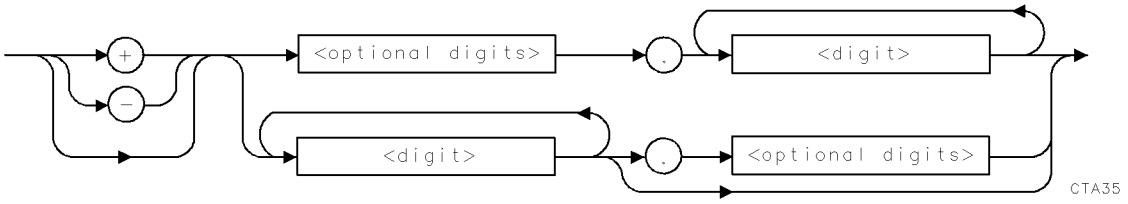
Example: <decimal numeric program data>

```
ACQUIRE:COUNT 16
```

Example: <decimal numeric program data> with optional <suffix program data>

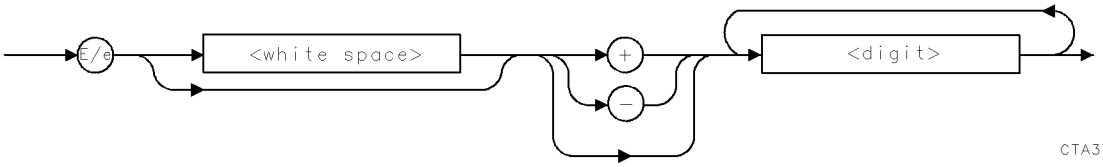
```
SWEEP:TIME:RANGE 1 us
```

where <mantissa> is defined as



where <optional digits> is defined as

where <exponent> is defined as



A <suffix program data> element permits the use of a suffix following the <decimal numeric program data> <nrf>. The suffix expresses associated units and (optional) multipliers that modify how the <decimal numeric program data> <nrf> is interpreted by the device.

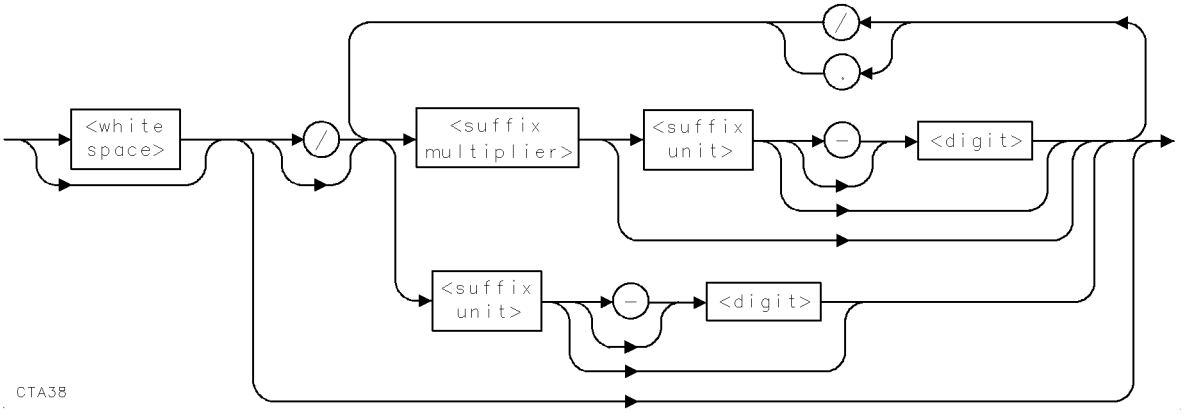


Figure 2-11. <suffix program data>

where

- <suffix multiplier> is defined as a sequence of one, two, or three <upper/lower case alpha> characters.
- <suffix unit> is defined as a sequence of one to four <upper/lower case alpha> characters.

Table 2-1. Allowed <suffix multiplier> Mnemonics

Value	Mnemonic
1E18	EX
1E15	PE
1E12	T
1E9	G
1E6	MA
1E3	K
1E−3	M
1E−6	U
1E−9	N
1E−12	P
1E−15	F
1E−18	A

Table 2-2. Allowed <suffix unit> Mnemonics

Class	Units	Mnemonic
ratio	decibel	DB
power	decibel milliwatt	DBM
frequency	hertz	HZ
angle	degree	DEG
time	seconds	S
amplitude	volts	V
ratio	percent	PCT

Note Because the parser has been defined as case insensitive by IEEE488.2, there is no difference between mUNIT and MUNIT hence the MA multiplier for 1E6. The suffix unit MHZ (1E6 HZ) is a special case, which is recognized by the module, and should not be confused with <suffix multiplier>HZ.

<string program data>. <string program data> is a data type suitable for sending seven-bit ASCII character strings where the content needs to be "hidden" by delimiters. For example, if a string is delimited by single-quote characters, and you need to include a single-quote character in the string, repeat the single-quote character twice. This also holds true for double-quote characters.

Example: <string program data>

DISPLAY:STRING 'This '' is a single-quote character'

DISPLAY:STRING "This ""is a double-quote character"

<arbitrary block program data>. <arbitrary block program data> is a data type suitable for sending blocks of arbitrary eight-bit information. See Figure 2-12.

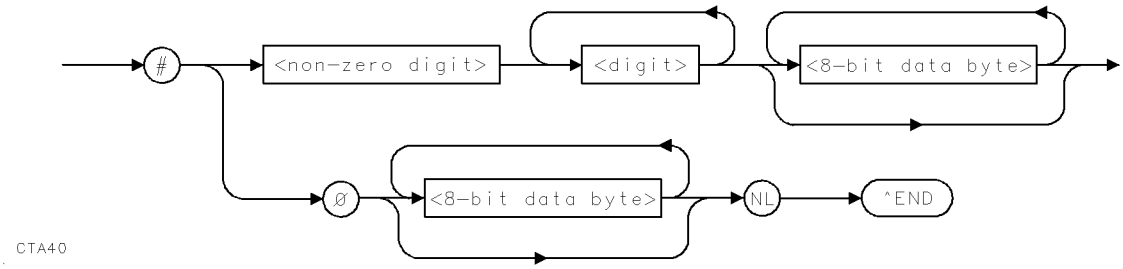


Figure 2-12. <arbitrary block program data>

Example: <arbitrary block program data>

```
WAVEFORM:DATA <data>
```

where

- <non-zero digit> is defined as a single ASCII encoded byte in the range 31—39 (49—57 decimal).
- <8-bit byte> is defined as an 8-bit byte in the range 00—FF (0—255 decimal).

<expression program data>. The <expression program data> element evaluates to a scalar, vector, matrix, or string value. It allows parameters to be manipulated by the instrument.

An <expression program data> element is defined as

Example: <expression program data>

```
FUNCTION:DEF (CHAN1)
```

where <expression> is defined as a sequence:

1. of ASCII-encoded data bytes in the range 20 to 7E (32 to 126 decimal) except the double quote, number sign, single quote, and semicolon (characters 22, 23, 27, and 3B hexadecimal or 34, 35, 39, and 59 decimal respectively), or
2. a device-defined set of the <program data> elements.

Functional Element Summary

This section summarizes all functional elements used in this chapter. Refer to the beginning of this chapter for more detailed information on the functional syntax elements and their definitions. For more information about the syntax elements, refer to the individual command descriptions and syntax diagrams.

Functional Syntax Elements

<program message unit>	Represents a single command, programming data, or query received by the device. For example, `RUN` or `ERROR?`.
<command message unit>	Represents a single command or programming data received by the device. For example, `WINDOW:USER 2,.5,.5,-.5`.
<query message unit>	Represents a single query sent from the controller to the device. For example, `MEASURE:PVOL? .02,3`.
<program message unit separator>	Separates sequential <program message unit> elements from one another within a program message. For example, ";" (semicolon).
<white space>	Represents one or more ASCII characters in the range 00-09, 0B-20 hexadecimal (0-9, 11-32 decimal). This range includes the ASCII-control characters and the space, but excludes the new line (NL).
<program data separator>	Separates sequential <program data> elements that are related to the same header. For example, "," (comma).
<program header separator>	Separates the (command or query) header from any associated <program data> elements. For example, " " (blank space).
<program message terminator>	Terminates a program message. For example, <NL>.
<command program header>	Specifies the function or operation to be performed in the device and may be optionally followed by associated parameters encoded as <program data> elements. A <command program header> is further defined as either a <simple command program header>, a <compound command program header>, or a <common command program header>. For example, `RUN`, `SWEEP:TIME:RANGE`, or `*OPC`.

<query program header> — Similar to <command program header>, except a query indicator (?) shows that a response is expected from the device. A <query program header> is further defined as either a <simple query program header>, a <compound query program header>, or a <common query program header>. For example, `ERROR?`, `SWEEP:TIME:RANGE?`, or `*OPC?`.

<program data> — Represents any of the five different program data types:

- <character program data> is a data type suitable for sending short mnemonic data, generally where a numeric data type is not suitable. For example, ON or OFF.
- <decimal numeric program data> is a data type suitable for sending decimal integers or decimal fractions with or without exponents. <suffix program data> is an optional field that may follow and is used to indicate associated multipliers and units. For example, 1.23E2 or 1.2 KHZ.
- <string program data> is a data type suitable for sending seven-bit ASCII character strings where the content needs to be "hidden" by delimiters. For example, "testing; one, two, three".
- <arbitrary block program data> is a data type suitable for sending blocks of arbitrary eight-bit information.
- <expression program data> is a data type suitable for sending data that is evaluated as one or more non-expression data elements before further parsing. For example, (CH1+CH2).

Syntax Elements

<character> — Represents any symbol, digit, letter, or punctuation mark. Refer to the respective command description for further definition.

<character string> — Represents a list of characters.

<digit> — 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

<channel_number> — Represents the number of the channel. Range: 1 or 2.

<correction_number> — Represents the user correction number. Range: 1 or 2.

<function_number> — Represents the number of the function. Range: 1, 2, 3, or 4.

<marker_number> — Represents the number of the marker. Range: 1 or 2.

<memory_number> — Represents the number of the waveform memory. Range: 1, 2, 3, or 4.

<suffix multiplier> — Represents a sequence of one, two, or three <upper/lower case alpha> characters. For example, M.

<suffix unit> — Represents a sequence of one to four <upper/lower case alpha> characters. For example, HZ.

<nrf>	Integer, fixed point, or floating point number (NR1, NR2, or NR3).
<NR1>	Integer.
<NR2>	Rational number in fixed-point notation (for example, 1.23).
<NR3>	Rational number in exponential notation. The number of significant digits output for NR3 is twelve.
<terminator>	Represents an optional <white space> character with either an END message on last data byte, newline (NL) with END message, or newline that terminates a program message.
<output terminator>	Terminates response data returned by the device in response to <query message unit> elements. For example, LF with END message.

3

Common Command Set

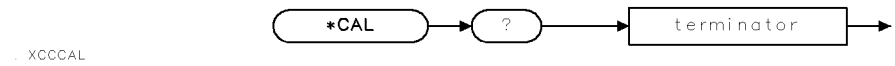
The commands of the Common Command set are available at any time during remote control of an HP 71500A microwave transition analyzer system. See the following list for a command syntax summary of the Common Command set commands.

KEYWORD	PARAMETER FORM	NOTES
CAL?		[query only]
CLS		[no query]
ESE	<nrf>	
ESR?		[query only]
IDN?		[query only]
LRN?		[query only]
OPC		
OPT?		[query only]
RCL	<nrf>	[no query]
RST		[no query]
SAV	<nrf>	[no query]
SRE	<nrf>	
STB?		[query only]
TST?		[query only]
WAI		[no query]

*CAL (Calibrate)

Performs a partial calibration.

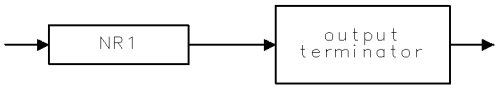
Syntax



Description

The *CAL query causes the module to perform as much of a calibration as is possible without connecting the calibration signal. This partial calibration includes calibrating the internal ADC. This command also returns a value to indicate if the partial calibration was completed without error. For a more complete calibration (which requires connecting the cal signal to the front panel and specifying a channel) use the CAL:ALL command. For more information about any calibration errors, refer to Chapter 3, “Error Messages.”

Query Response



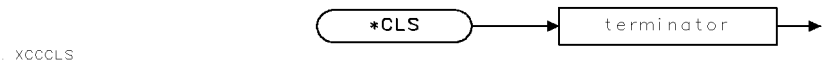
Query Example

```
OUTPUT @Mta;"*CAL?"
ENTER @Mta;Pass_fail
```

*CLS
(Clear Status)

Clears the status data structures and the Request-for-Operation-Complete flag.

Syntax



Command Example

```
OUTPUT @Mta;"*CLS"
```

Description

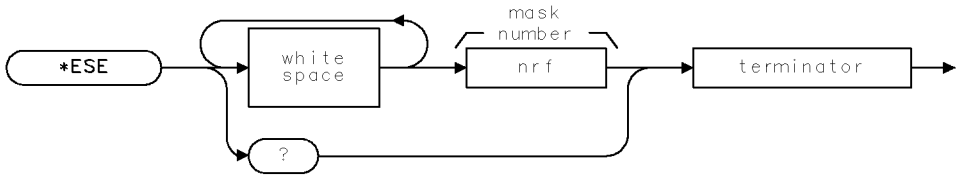
The *CLS command clears status data structures and the Request-for-Operation-Complete (OPC) flag. Refer to the *OPC command in this section for more information.

If the *CLS command immediately follows a program message terminator, the Output Queue and the Message Available (MAV) bit will be cleared. Any new program message after a program message terminator clears the Output Queue.

*ESE
(Standard Event Status Enable)

Sets the bits of the Standard Event Status Enable Register.

Syntax



Power-On State: 0
Preset State: not affected by IP, *RST, or *RCL
Parameter Range: mask number = 0 through 255

Command Example

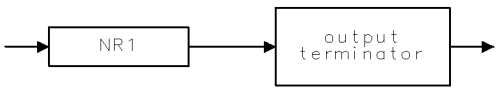
```
OUTPUT @Mta;"*ESE 32"
```

Description

The *ESE command sets the Standard Event Status Enable Register bits.

The query allows the programmer to determine the current contents of the Standard Event Status Enable Register.

Query Response



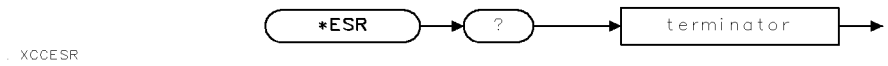
Query Example

```
OUTPUT @Mta;"*ESE?"
ENTER @Mta;Ese_mask
```

*ESR
(Standard Event Status Register)

Returns the current contents of the Standard Event Status Register.

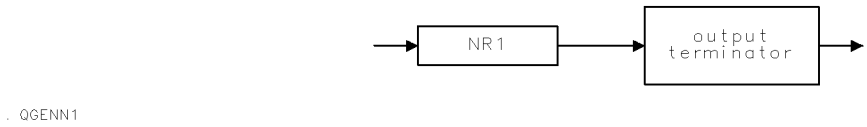
Syntax



Description

The *ESR query allows the programmer to determine the current contents of the Standard Event Status Register. Reading the Standard Event Status Register will clear it.

Query Response



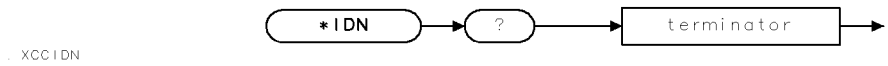
Query Example

```
OUTPUT @Mta;"*ESR?"
ENTER @Mta;Esr_value
```

*IDN (Identification)

Identifies devices over the system interface.

Syntax



Description

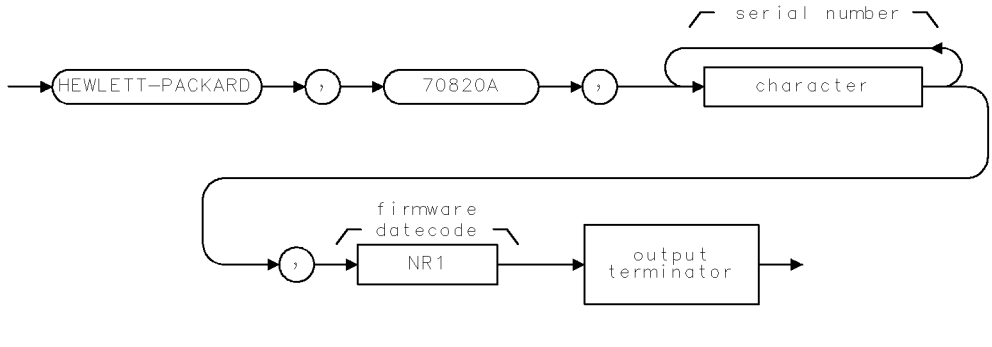
The *IDN query is for identifying devices over the system interface. The response is organized into four fields separated by commas. The field definitions are as follows:

Field 1	Manufacturer
Field 2	Model number
Field 3	Serial number
Field 4	Firmware datecode in year, month, day format

An example of a query response is:

HEWLETT-PACKARD,70820A,2419A00256,910313

Query Response



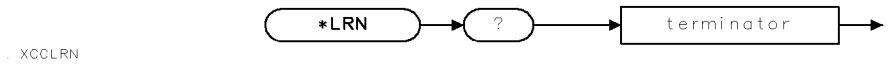
Query Example

```
DIM Id$[100]
OUTPUT @Mta;"*IDN?"
ENTER @Mta;Id$
```

*LRN
(Learn Device Setup Query)

Obtains a programming sequence used to restore the module to the state when the query was made.

Syntax

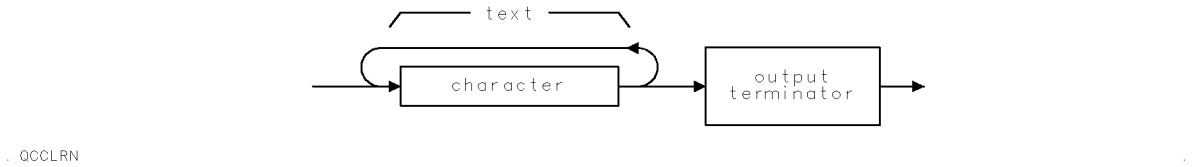


Description

The *LRN query allows the programmer to obtain a programming sequence that may later be used to place the module in the state it was in when the *LRN? common query was made.

Note Due to the complexity of the module state, the amount of returned data is quite large (approximately 3300 bytes).

Query Response



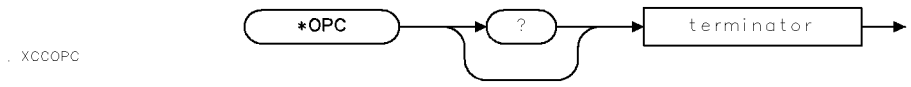
Query Example

```
DIM Learn$[5000]              ! Read the current state.
OUTPUT @Mta;"*LRN?"
ENTER @Mta;Learn$
.
.
.
OUTPUT @Mta;Learn$            ! Restore state.
```

*OPC (Operation Complete)

Sets the request for the Operation Complete flag.

Syntax



Command Example

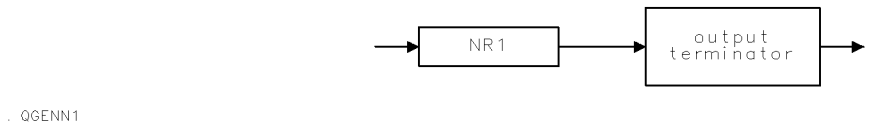
```
OUTPUT @Mta;"*OPC"
```

Description

The *OPC command sets the request for the Operation Complete flag. When all pending device operations have been finished, the OPC bit in the Standard Event Status Register is set.

The query places an ASCII character “1” in the Output Queue when all pending device operations have been finished.

Query Response



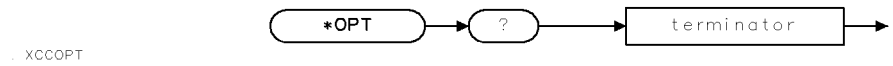
Query Example

```
OUTPUT @Mta;"RUN;*OPC?"
ENTER @Mta;Opc$
```

*OPT
(Option Identification)

Identifies reportable device options over the system interface.

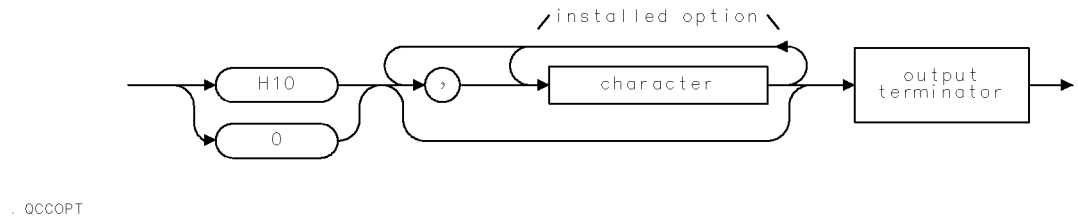
Syntax



Description

The *OPT query is used for identifying reportable device options over the system interface. The response is a comma-delimited list of installed options. If there are no options, a “0” is returned .

Query Response



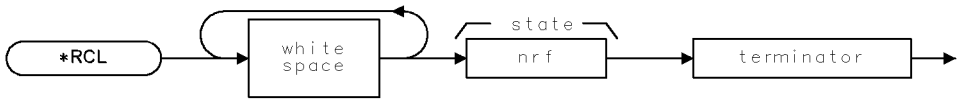
Query Example

```
OUTPUT @Mta;"*OPT?"
ENTER @Mta;Option$
```

*RCL (Recall)

Restores the state of the module from the local battery-backed-up memory.

Syntax



XCCRCL

Parameter Range: state = 1 through 6

Command Example

```
OUTPUT @Mta;"*RCL 1"
```

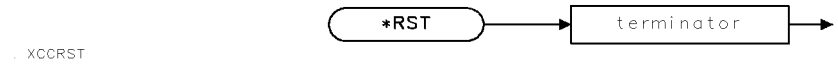
Description

The *RCL command restores the state of the module from the local battery-backed-up memory. The scope of the saved state is identical to the scope of the *RST command (that is, anything affected by *RST is affected by *RCL).

*RST
(Reset)

Performs a device reset.

Syntax



Command Example

```
OUTPUT @Mta;"*RST"
```

Description

The *RST command performs a device reset and the following actions:

- Sets device-dependent functions to a known state independent of the past-use history of the module
- Aborts pending operations
- Clears the Output Queue
- Clears the Request-for-Operation Complete flag

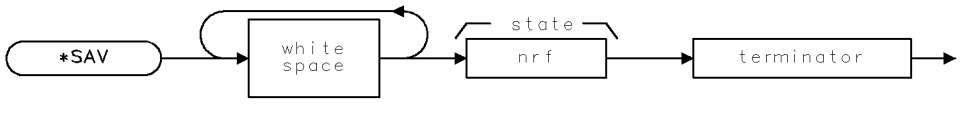
The *RST command will not:

- Affect the state of the HP-IB or HP-MSIB interfaces
- Modify the Standard Status Register Enable setting
- Modify the Standard Event Status Enable setting
- Modify the power-on-clear flag
- Modify calibration data

*SAV (Save)

Stores the current state of the module in local battery-backed-up memory.

Syntax



Parameter Range: state = 1 through 6

Command Example

```
OUTPUT @Mta;"*SAV 1"
```

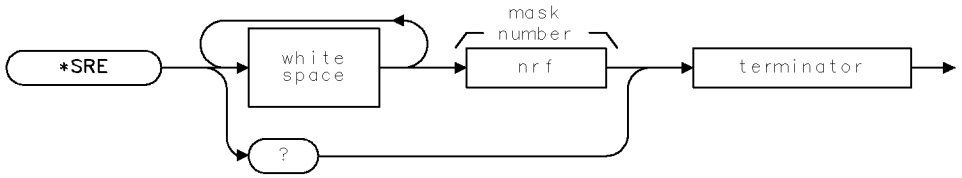
Description

The *SAV command stores the current state of the module in local battery-backed-up memory. The scope of the saved state is identical to the scope of the reset command (that is, anything affected by *RST is saved by *SAV).

*SRE
(Service Request Enable)

Sets the Service Request Enable Register bits.

Syntax



Power-On State: 0
Preset State: not affected by IP, *RST, or *RCL
Parameter Range: mask number = 0 through 255

Command Example

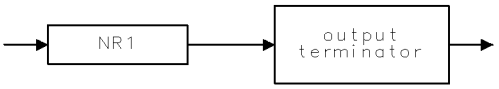
```
OUTPUT @Mta;"*SRE 150"
```

Description

The *SRE command sets the Service Request Enable Register bits.

The query returns the current contents of the Service Request Enable Register. Refer to "Status Reporting Structure" in Chapter 1 for more information.

Query Response



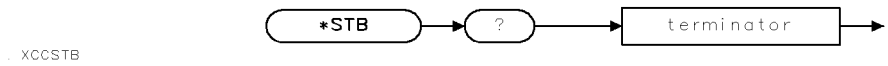
Query Example

```
OUTPUT @Mta;"*SRE?"
ENTER @Mta;Enable_bits
```

*STB
(Read Status Byte)

Reads the Status Byte Register and the Master Summary Status bit.

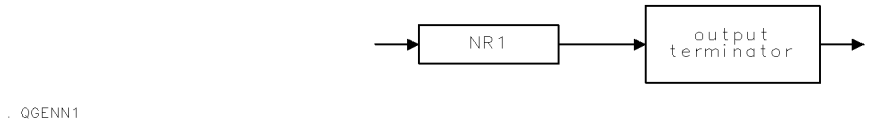
Syntax



Description

The *STB query allows the programmer to read the Status Byte Register and the Master Summary Status bit. Refer to "Status Reporting Structure" in Chapter 1 for more information.

Query Response



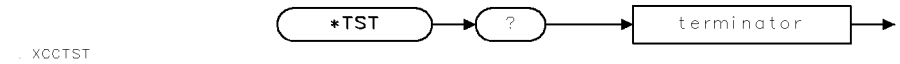
Query Example

```
OUTPUT @Mta;"*STB?"
ENTER @Mta;Status_byte
```

*TST
(Test)

Executes an internal self-test routine.

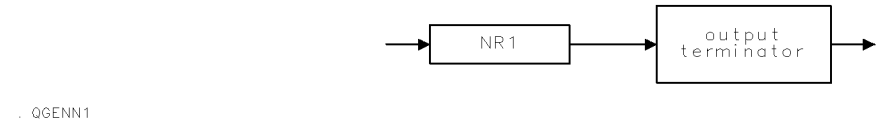
Syntax



Description

The *TST query executes an internal self test and returns a pass/fail code. Pass is equal to 0, fail is equal to 1. Currently, any information regarding a failure is handled via the error reporting.

Query Response



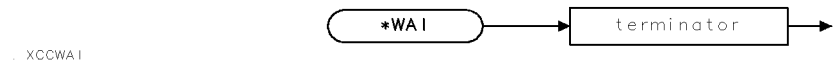
Query Example

```
OUTPUT @Mta;"*TST?"
ENTER @Mta;Pass_or_fail
IF Pass_or_fail = 0 THEN
   PRINT "Self test passed"
ELSE
   PRINT "Self test failed"
END IF
```

*WAI (Wait)

Stops executing any further commands or queries.

Syntax



Command Example

```
OUTPUT @Mta;"RUN;*WAI;"
```

Description

The *WAI command stops the device from executing any further commands or queries until the No-Operation-Pending flag, Power-On flag (PON), or Device Clear Active State Message (DCAS) is true.

Note

This command stops the parsing of commands, not the entry of bus data into an input buffer. To stop the execution of a program at a specific point, use the *OPC command/query.

The *WAI command does not set the operation complete bit in the Standard Event Status Register.

4

Top-Level Command Set

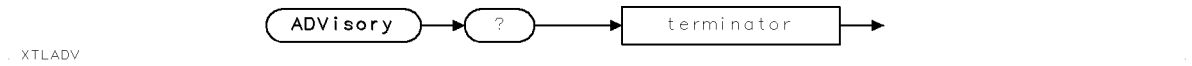
The commands of the Top-Level Command set are general commands that do not reside in any particular subsystem. See the following list for a command syntax summary of the Top-Level Command set commands.

KEYWORD	PARAMETER FORM	NOTES
ADVisory?		[query only]
AMASk	<nrf>	
AUTo		[no query]
BLANk	CHAN<channel_number> |FUNC<function_number> |WMEM<memory_number>	[no query]
DATE	<nrf>[,<nrf>[,<nrf>]]]	
DIGitize		[no query]
ERRor?	[NUMber|STRing]	[query only]
PCB	<nrf>[,<nrf>]	[no query]
READy		[no query]
RUN		[no query]
STOP		[no query]
STORe	CHAN<channel_number> |FUNC<function_number> |WMEM<memory_number> ,WMEM<memory_number> |UCAL<correction_number> [,FORmat|VECtor]	[no query]
TIME	<nrf>[,<nrf>[,<nrf>]]]	
VIEW	CHAN<channel_number> |FUNC<function_number> |WMEM<memory_ number>	[no query]

ADVisory

Returns the current value of the advisory.

Syntax



Related Commands: AMASk
ERRor?

Description

The ADVISORY query returns the current value of the advisory. Each bit of the advisory corresponds to a condition that may be considered an error (for example, oven cold during a measurement).

Each bit corresponds to a possible error report as follows:

BIT 0: error 6622: low IF.

BIT 1: reserved for RF correction resolution check.

BIT 2: reserved for user correction resolution check.

BIT 3: error 6605: correction data clipped

BIT 4: reserved for future use.

BIT 5: error 6208: carrier frequency mixed to poor IF.

BIT 6: error 6209: oven cold during measurement.

BIT 7: error 6201: zoom span too small.

BIT 8: error 6202: time span too small.

BIT 9: error 6200: channel skew too large.

BIT 10: error 6210: math overrange.

BIT 11: reserved for future use.

BIT 12: error 6211: channel1 hardware overrange.

BIT 13: error 6212: channel2 hardware overrange.

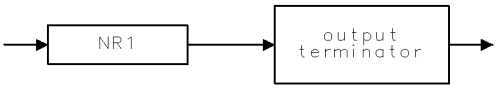
BIT 14: error 231: data questionable.

BIT 15: error 6213: delay too large.

BIT 16 through BIT 31: undefined.

Note Refer to Chapter 3, "Error Messages", for a more complete description of each error.

Query Response



QGENN1

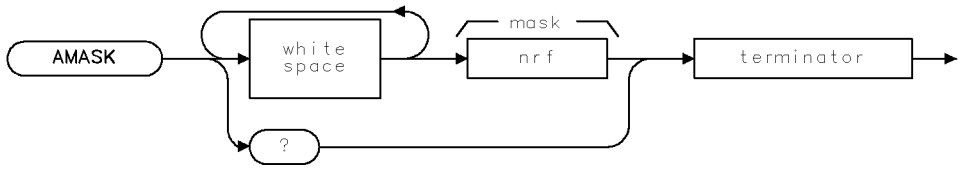
Query Example

```
OUTPUT @Mta;"ADV?"
ENTER @Mta;Advisory
```

AMASk (Advisory Mask)

Sets the mask that determines which advisory conditions are reported as errors.

Syntax



Preset State:	not affected by IP, *RST, or *RCL
Power-On State:	0
Parameter Range:	mask = 0 through $2^{31}-1$
Related Commands:	ADVisory
	ERRor?

Command Example

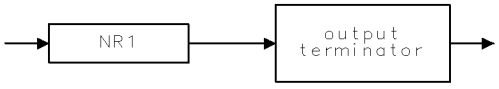
```
OUTPUT @Mta;"AMAS 1"
```

Description

The AMASK command sets the mask that determines which, if any, advisory conditions will be reported as an error. The checking and error reporting will be done at the end of a sweep.

The query returns the current state of AMASK.

Query Response



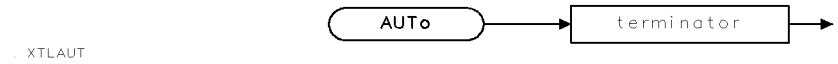
Query Example

```
OUTPUT @Mta;"AMAS?"
ENTER @Mta;Mask
```

AUTo

Performs a measurement function that selects input, vertical sensitivity, vertical offset, trigger level, and sweep speed.

Syntax



Command Example

```
OUTPUT @Mta;"AUT"
OUTPUT @Mta;"*OPC?"
ENTER @Mta;A$
```

Description

The AUTO command performs a measurement function that automatically selects the input, vertical sensitivity, vertical offset, trigger level, and sweep speed for a display of an input signal. In general, it is recommended that the autoscale operation is completed before performing any other action (also see the *OPC and *WAI commands).

The input selection operates as follows:

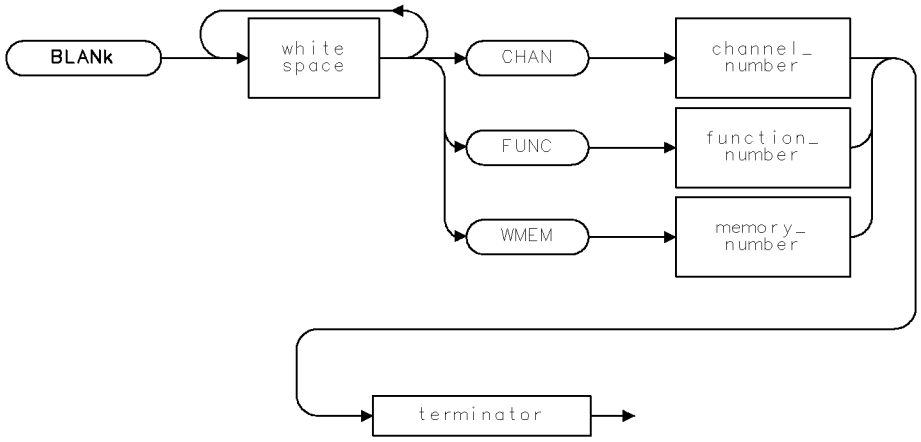
- If a signal is found on both inputs or no signal is found on either input, the input specified by the trigger source will be selected.
- If a signal is found on one input but not the other input, the input with the signal will be selected.

If the autoscale operation fails, an error is declared.

BLANk

Causes the instrument to blank the specified trace.

Syntax



Command Example

```
OUTPUT @Mta;"BLAN WMEM2"
```

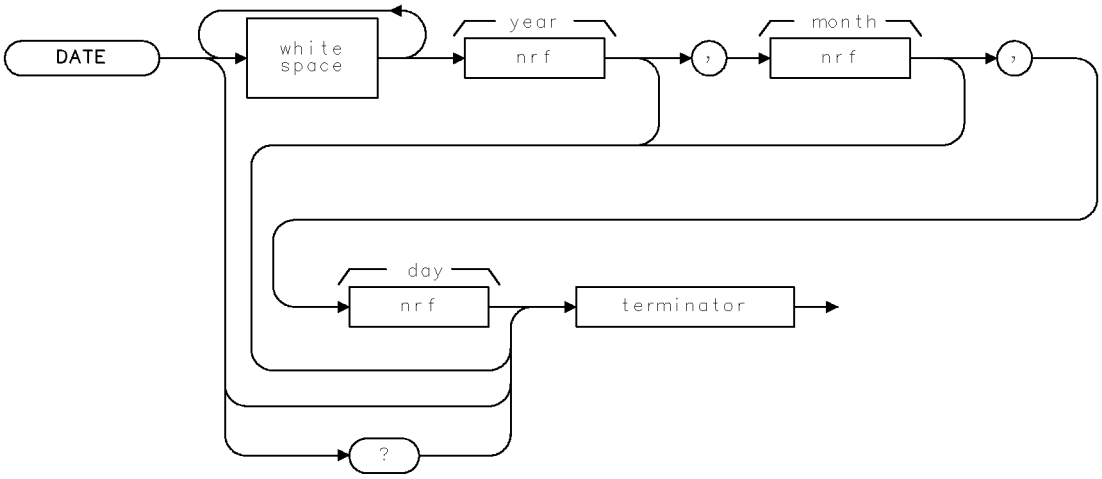
Description

The BLANK command causes the instrument to turn off (that is, to stop displaying on the display screen) the specified channel, function, or waveform memory. Functions are only computed if they are not blanked. Blanking channels or memories has no effect on remote operation, except for execution speed.

DATE

Sets the date.

Syntax



XTLDAT

Parameter Range: 1987 through 2086, where
year = 0 through 99,
month = 1 through 12
day = 1 through 31

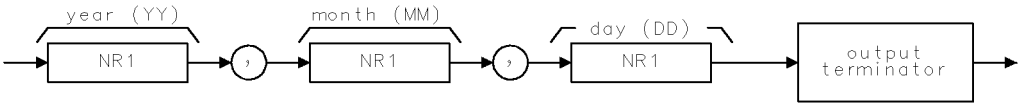
Command Example

```
OUTPUT @Mta;"DATE 89,03,15"
```

Description

The date command allows the user to either set or query the date.

Query Response



QTLDAT

Query Example

```
OUTPUT @Mta;"DATE?"
ENTER @Mta;Year,Month,Day
```

DIGitize

Resets the average and acquires a specified number of sweeps.

Syntax

Related Commands: RUN

Command Example

```
OUTPUT @Mta;"ACQ:COUN 32"     ! Choose 32 average sweeps.
OUTPUT @Mta;"DIG"             ! Issue digitize command.
OUTPUT @Mta;"*OPC?"           ! Wait for operation to complete.
ENTER @Mta;A$
```

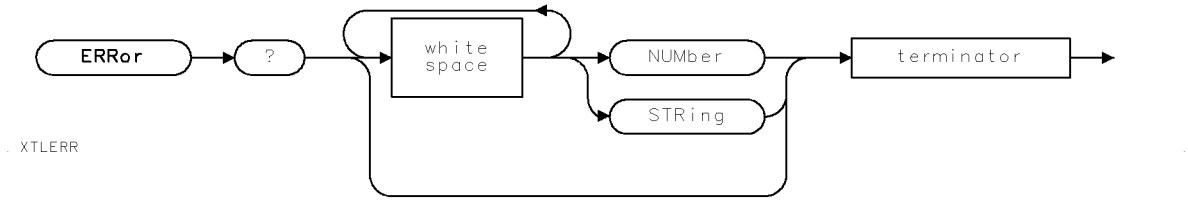
Description

The DIGITIZE command is used as a convenient method of resetting the average count and acquiring a specified number of sweeps. The number of sweeps is determined by the maximum value of the ACQUIRE:COUNT query for any channel (on or off) and any function (on only) that has the value of the TYPE query as AVERAGE, FAVERAGE, HOLD, MAXHOLD, or MINHOLD.

ERRor

Returns the next error in the Error Queue.

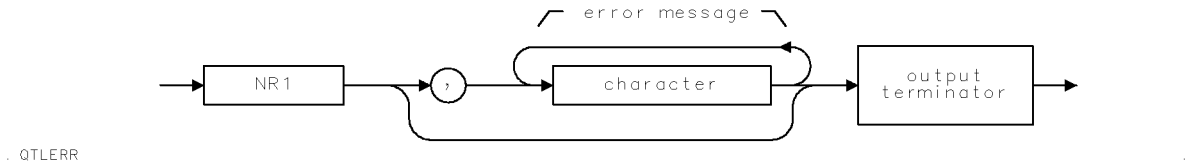
Syntax



Description

The ERROR query returns the next error in the Error Queue. If there are no errors, 0 is returned ("0, no errors" in STRING mode). If there has been an error, the instrument will respond with the first one. Subsequent responses to the ERROR query will continue with the error list until there are no errors remaining at which point 0 is returned. The optional parameters NUMBER and STRING indicate whether a numeric or text description of the error is to be returned.

Query Response



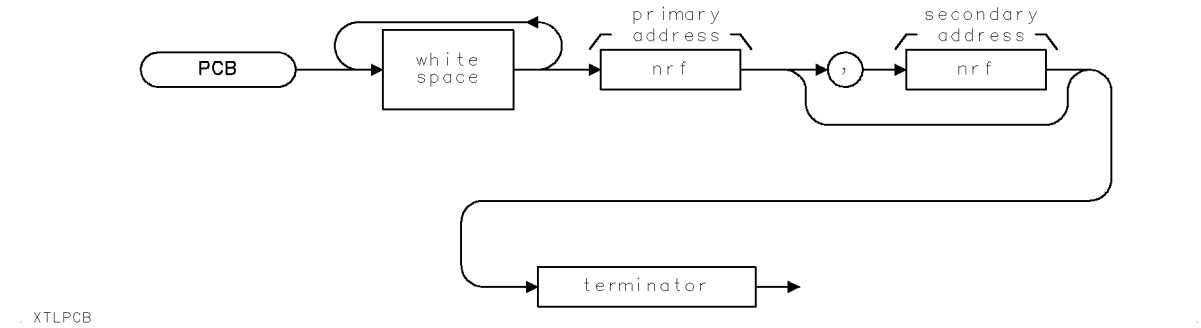
Query Example

```
DIM Error$[100]
REPEAT
  OUTPUT @Mta;"ERR? STR"
  ENTER @Mta;Error$
  PRINT Error$
UNTIL (VAL(Error$)=0)
```

PCB
(Pass Control Back)

Defines which HP-IB address to pass control to the next time the OPC bit in the Standard Event Status Register is set.

Syntax



Parameter Range: primary address = 0 through 30;
secondary address = 0 through 30

Command Example

This example demonstrates passing control back and forth between the private HP-IB and a computer.

```
ASSIGN @Mta TO 710
ASSIGN @Cntl TO 711
OUTPUT @Mta;"PCB 21;*OPC;"    ! Pass control back to address 21.
PASS CONTROL @Cntl            ! Pass control to the HP 70820A.
REPEAT                        ! Wait until the HP 70820A passes
STATUS 7,3;Temp               ! control back.
UNTIL BIT(Temp,6) = 1
```

Description

The Pass Control Back (PCB) command defines the address, on the "private" HP-IB of the microwave transition analyzer, to pass control to the next time that the Operation Complete (OPC) bit in the Standard Event Status Register is set. After PCB is executed, the earliest that the OPC bit can be set is after a program message terminator (that is, line feed) is received.

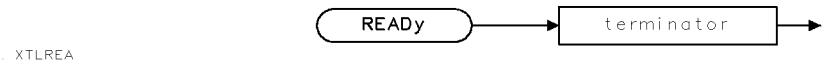
If the secondary address parameter is omitted, the device to pass control to will be addressed with primary addressing.

Refer to Appendix B, "Passing Control," for more information on using the PCB command in conjunction with other commands of the microwave transition analyzer.

READy

Sets the ready bit in the Status Operation Condition Register.

Syntax



Command Example

This example sets up the HP 70820A to assert a service request after the program message terminator when the ready command is parsed.

```
OUTPUT @Mta;"STAT:OPER:NTR 2048;"
OUTPUT @Mta;"STAT:OPER:PTR 0;"
OUTPUT @Mta;"STAT:OPER:ENAB 2048;"
OUTPUT @Mta;"*SRE 128;"
OUTPUT @Mta;"READY"
```

Description

The READY command sets the ready bit in the Status Operation Condition Register.

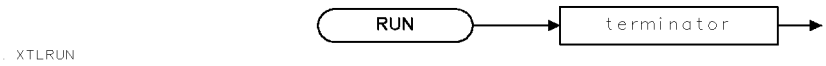
The READY command is similar to the Operation Complete (*OPC) command except that the READY command immediately sets the ready bit whereas the *OPC command waits until all pending operations (for example, triggering or sweep) are complete before the OPC bit is set.

The READY command will normally be used in conjunction with the Status Operation subsystem (STATUS:OPERATION).

RUN

Acquires data for the active waveform display.

Syntax



Related Commands: DIGitize

Command Example

```
OUTPUT @Mta;"RUN"
```

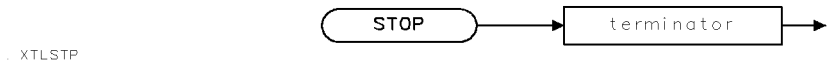
Description

The RUN command instructs the instrument to acquire data for the active waveform display. If the instrument is in single sweep, one trigger is enabled which in turn generates one measurement. If the instrument is in continuous sweep, triggers are enabled repeatedly and the instrument displays the data it acquires continuously.

STOP

Stops acquiring data.

Syntax



Command Example

```
OUTPUT @Mta;"STOP"
```

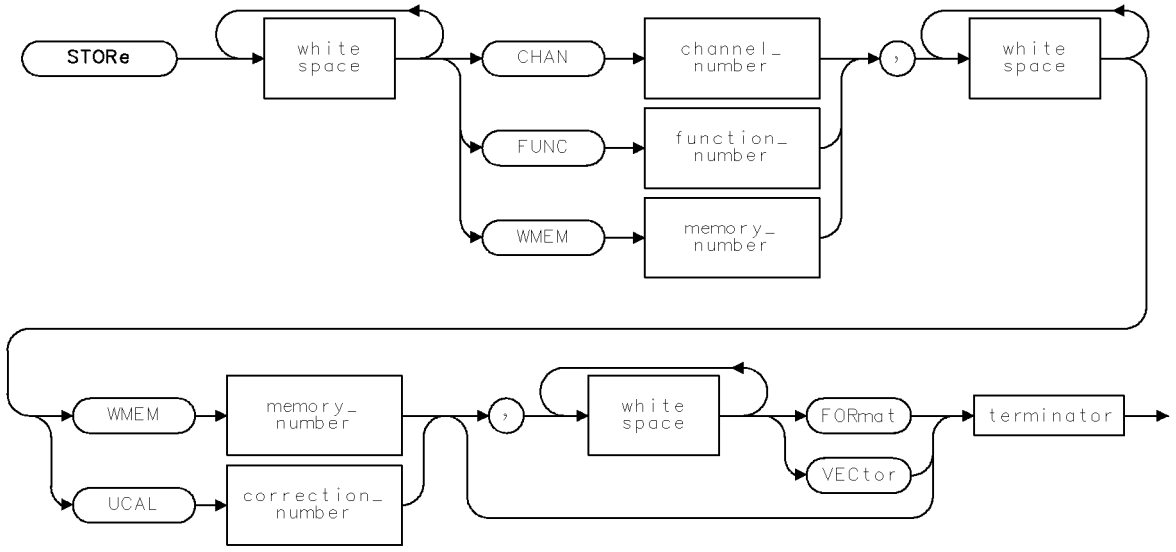
Description

The STOP command instructs the instrument to stop acquiring data for the active waveform display. This does not affect the single sweep or continuous sweep state.

STORe

Moves data in the specified trace to the specified waveform memory.

Syntax



. XTLSTR

Command Example

```
OUTPUT @Mta;"STOR CHAN1, WMEM3"
```

Description

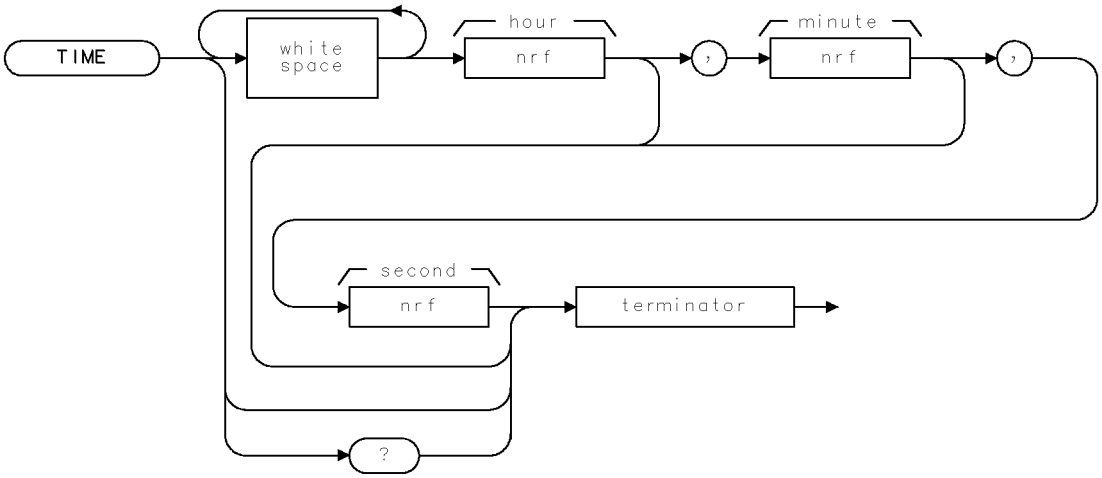
The STORE command instructs the instrument to move the data in the specified trace to the specified waveform memory. All data pertinent to the trace (for example, time range, voltage range, number points, and so on) is stored. The waveform memories are stored in local battery-backed-up RAM.

Note Traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

TIME

Sets the time of day.

Syntax



XTLTIM

Parameter Range: hour = 0 through 23,
minute = 0 through 59,
second = 0 through 59

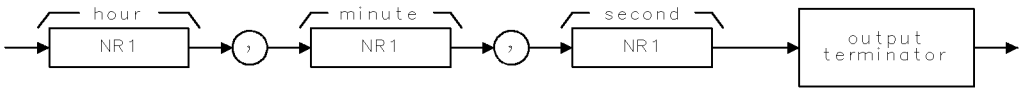
Command Example

```
OUTPUT @Mta;"TIME 15,30"          ! Set time to 3:30 PM
```

Description

The TIME command allows the user to set the time of day or query to return the time of day. The internal clock runs on 24 hour time (that is, at 6:00 PM hour = 18, minute = 0, second = 0).

Query Response



. QTLTIM .

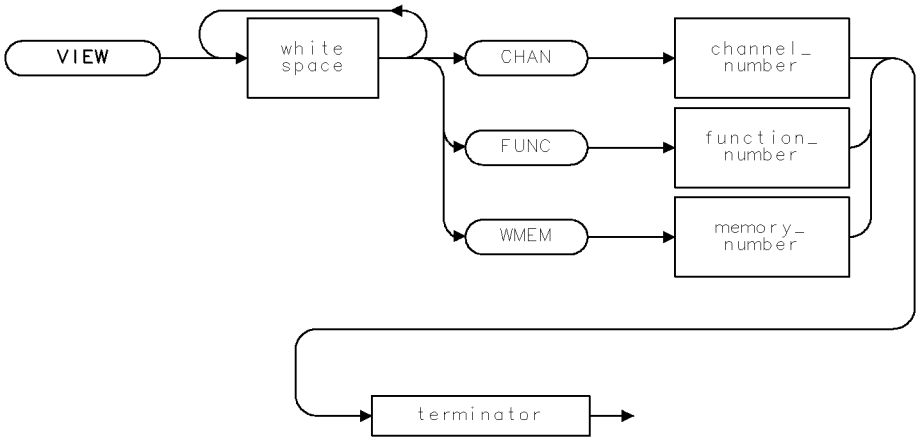
Query Example

```
OUTPUT @Mta;"TIME?"
ENTER @Mta;Time$
PRINT "The time is: "&TIME$
```

VIEW

Displays the specified channel, function, or waveform memory.

Syntax



XTLVIE

Command Example

```
OUTPUT @Mta;"VIEW FUNC1"
```

Description

The VIEW command instructs the instrument to display on screen the specified channel, function, or waveform memory. Functions are only computed if they are not blanked. Blanking channels or memories has no effect on remote operation, except for execution speed.

Note Traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

5

Acquire Subsystem

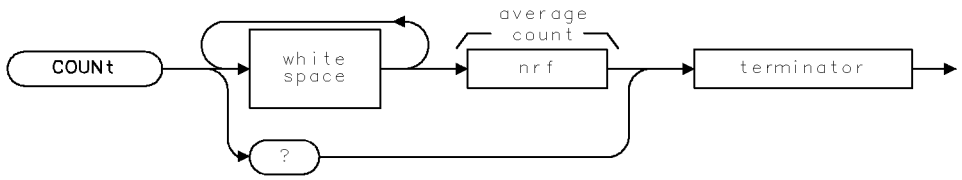
The ACQUIRE subsystem commands are used to set up conditions when a measurement is made. This subsystem is used to select the number of trace averages, the number of points desired, trace length and timebase coupling, and the type of data. See the following list for a command syntax summary of the ACQUIRE subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:ACQuire		
:COUNt	<nrf>	
:DITHer	OFF|ON	
:FORMat	INT16|INT32	
:POINts	<nrf>	
:TYPE	AVERage|FAVerage|HOLD|MAXhold| MINhold|NORMal	

COUNt

Sets the average count for channels.

Syntax



Preset State: 16
Parameter Range: average count = 1 through 1024

Command Example

```
OUTPUT @Mta;"ACQ:COUN 55"
```

Description

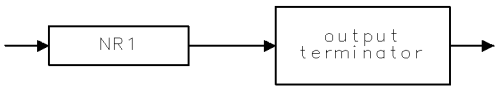
The COUNT command sets the average count for channels. This command does not control the average count for functions (see FUNC:COUNT). Note that each channel can be set to different number of averages using the CHANx:COUNT command.

The average operation is implemented as a moving average, that is:

$$new\ value = \frac{(averages\ - 1) \times old\ value\ + new\ value}{averages}$$

The query returns the last value set for channel 1.

Query Response



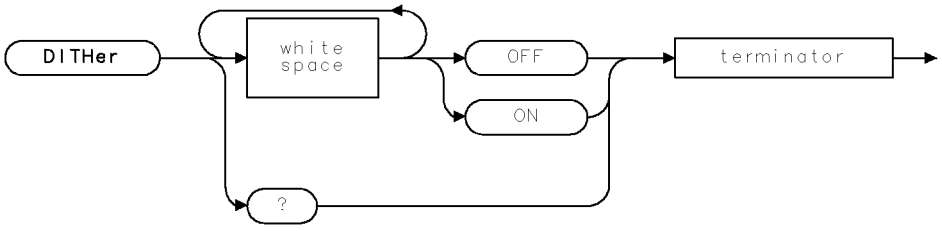
Query Example

```
OUTPUT @Mta;"ACQ:COUN?"
ENTER @Mta;Count$
```

DITHer

Controls dc dithering for data acquisition.

Syntax



Preset State: OFF

Command Example

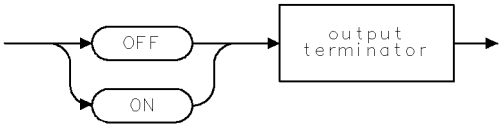
```
OUTPUT @Mta;"ACQ:DITH ON"
```

Description

The DITHER command controls dc dithering for data acquisition. When DITHER is turned ON, a pseudo-random offset is applied to the input DAC each time a measurement is taken. This tends to remove inaccuracies caused by quantization errors in the ADC. Dithering is generally used in conjunction with averaging.

The query returns the current state of the dither flag.

Query Response



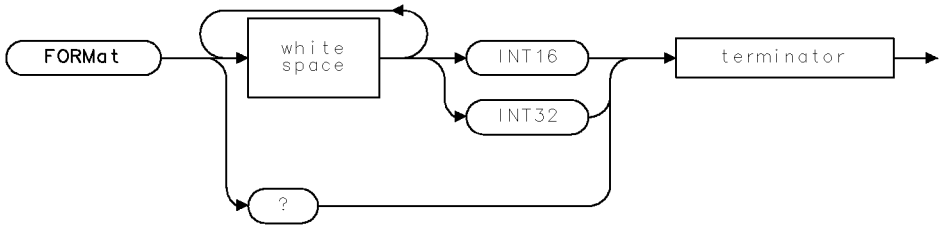
Query Example

```
OUTPUT @Mta;"ACQ:DITH?"
ENTER @Mta;Dither$
```

FORMat

Controls the number of data bits used in acquiring time traces.

Syntax



Preset State: INT16
Related Commands: WAVeform:FORMat

Command Example

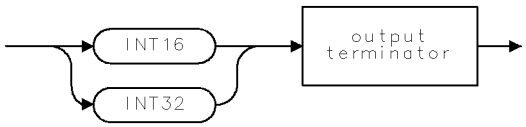
```
OUTPUT @Mta;"ACQ:FORM INT32"
```

Description

The FORMAT command controls the number of data bits used in acquiring time traces. Current formats are integer, either 16 or 32 bits wide.

The query returns the current format for time traces.

Query Response



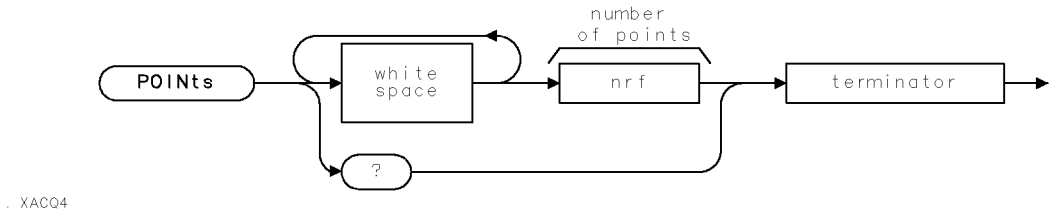
Query Example

```
OUTPUT @Mta;"ACQ:FORM?"
ENTER @Mta;Format$
```

POINts

Sets the number of acquired points in the next measurement.

Syntax



Preset State: 512
Parameter Range: points = 32 through 1024 for time sweeps;
points = 1 through 1024 for frequency or power sweeps
Related Commands: WAVeform:POINts

Command Example

```
OUTPUT @Mta;"ACQ:POIN 500"
```

Description

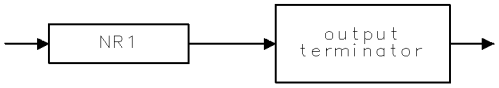
The POINTS command sets the number of points to be acquired in the next measurement (that is, it determines the number of points that will be in a trace after the next measurement is performed).

The query returns the current setting, which is not necessarily the number of points in any existing trace.

Note If the POINTS command is set to 1 when SWEEP:TYPE is set to either FREQUENCY or POWER, the resulting trace length will be 2. A single point will be measured and duplicated. Also, the internal FFT and inverse FFT require a minimum of 32 points. If a trace of fewer than 32 points is used, the result will be 0.

The POINTS command also specifies the number of points acquired during the measurement. Depending on the processing performed, this may not be the same as the number of points in the output trace. For example, if an 800-point time-domain trace is converted to the frequency domain, the trace length is zero padded (that is, the first and last 112 points of the trace are filled with zeroes) to a power of 2 (1024 in this case), the FFT is applied, and the positive frequencies are returned (dc through Fs/2), thus yielding 513 points. Use the WAVEFORM:POINTS query to determine the number of points in a trace.

Query Response



. QGENN1

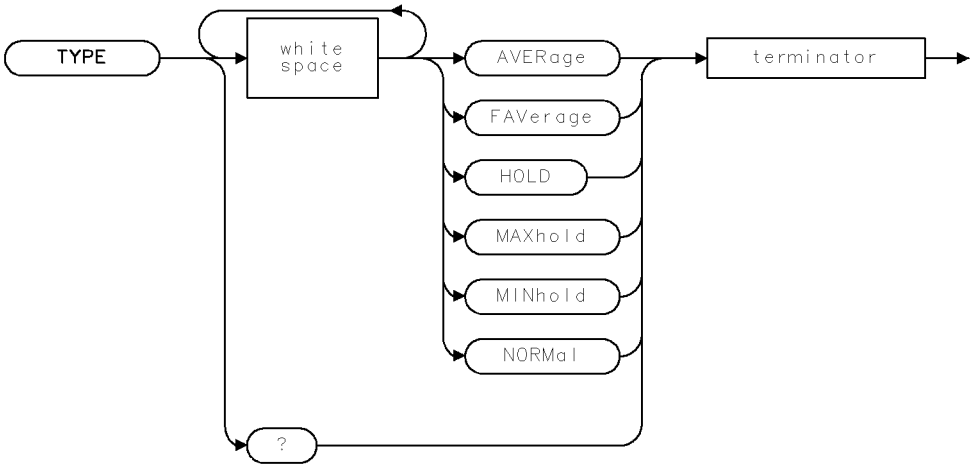
Query Example

```
OUTPUT @Mta;"ACQ:POIN?"
ENTER @Mta;Points$
```

TYPE

Selects a post-processing operation performed on all channels after each measurement.

Syntax



Preset State: NORMal

Command Example

```
OUTPUT @Mta;"ACQ:TYPE AVER"
```

Description

The TYPE command controls the selection of a post-processing operation performed on all channels after each measurement. However, this does not control the post-processing operation for functions (see FUNC:TYPE). The post-processing operation can be set to different values for a specified channel with the CHANx:TYPE command. The query returns the last value set for channel 1.

In all cases, the raw measurement data is evenly spaced and sequentially acquired.

When NORMAL is selected, no post processing is performed.

When AVERAGE is selected, the averaging method described for ACQUIRE:COUNT is applied to the raw (unformatted) data.

When FAVERAGE is selected, the averaging method described for ACQUIRE:COUNT is applied to the formatted data.

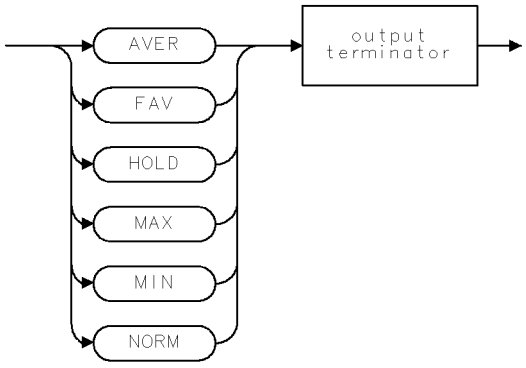
When HOLD is selected, a minimum or maximum hold operation is performed on the formatted data. The odd points of the result (that is, the first, third, and so on) contain the maximum values. The even points of the result (that is, the second, fourth, and so on) contain the minimum values. The trace size is not affected.

When MAXHOLD is selected, a maximum hold operation is performed on the formatted data.

When MINHOLD is selected, a minimum hold operation is performed on the formatted data.

TYPE is not independent of median smoothing. Median smoothing can only be applied if TYPE is NORMAL or AVERAGE.

Query Response



QCHA8

Query Example

```
OUTPUT @Mta;"ACQ:TYPE?"
ENTER @Mta;Type$
```

6

Basic Subsystem

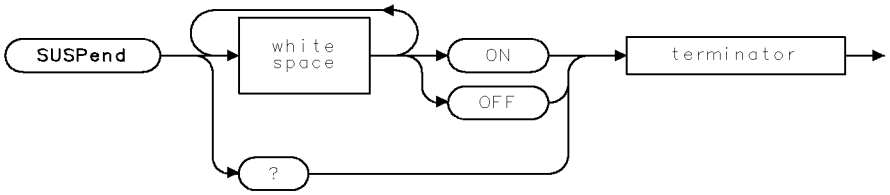
Use the BASIC subsystem commands control operation of Instrument BASIC that are not covered by PROGRAM. See the following list for a command syntax summary of the BASIC subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:BASic		
:SUSPend	OFF|ON	

SUSPend

Changes priority of HP Instrument BASIC.

Syntax



Preset State: ON

Command Example

```
OUTPUT @Mta;"BASIC:SUSP OFF"
```

Description

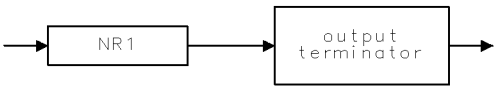
The SUSPEND command enables and disables a pause check at the end of each executed Instrument BASIC statement. When SUSPEND is OFF, no check is done. This results in the fastest execution time of an Instrument BASIC program but has the drawback of possibly preventing the measurement task from running. For example, if a program has the statement "Idle: GOTO Idle" then the 70820A will appear to freeze (i.e. no more measurements will be made) when this statement is executed. This can be solved by writing the loop as:

```
Idle:  WAIT .05
       GOTO Idle
```

The WAIT statement allocates some processor time to the measurement task.

When SUSPEND is ON, after the execution of each Instrument BASIC statement a processor suspend operation is performed which results in approximately half of the processor time allocated to Instrument BASIC and half to the measurement task. Of course this means that programs will run at approximately half the speed as with SUSPEND OFF, but loops such as "Idle: GOTO Idle" will not freeze the system. The query returns the current state of SUSPEND.

Query Response



. QGENN1

Query Example

```
OUTPUT @Mta;"BAS:SUSP?"
ENTER @Mta;Suspend$
```

7

Calibrate Subsystem

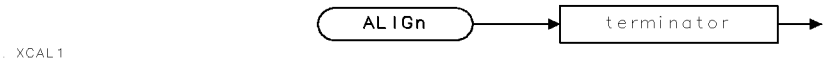
The CALIBRATE subsystem is used to start the calibration sequence in the microwave transition analyzer, to obtain information from the last internal calibration, or to reload old calibration data. Prior to starting the calibration sequence, it is assumed that the user has connected a calibration signal to the channel 1 or channel 2 input. The external calibration signal must be the calibration output signal of the microwave transition analyzer. See the following list for a command syntax summary of the CALIBRATE subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:CALibrate		
:ALIGn		[no query]
:ALL?	CHAN1|CHAN2	[query only]
:CORRect		
:RF	OFF|ON	
:DATA	CHAN1|CHAN2,# <digit>{<digit>}{<character>}	
:DATA?	CHAN1|CHAN2	
:SIGNal		
:AMPLitude	<nrf>	
:FREQuency	<nrf> [[<suffix multiplier>]HZ]	
[:STATe]	OFF|ON	
:SKEW	<nrf> [[<suffix multiplier>]S]	
:USER		
:ADAPtive	OFF|ON	
:DATA	CHAN1|CHAN2,# <digit>{<digit>}{<character>}	
:DATA?	CHAN1|CHAN2	
:MODE	CHAN1|CHAN2,DIVide|MULTiply	
:MODE?	CHAN1|CHAN2	
:STATe	OFF|ON	
:THReshold	<nrf>	

ALIGn

Performs a semi-automatic measurement of the channel-to-channel skew.

Syntax



Parameter Range: defined signal frequency (SWEep:SIGnal) > 20 MHz

Command Example

```
OUTPUT @Mta;"CAL:SKEW 0"
FOR J = 1 TO 26 STEP 5
! Set source frequency here to J × 1 GHz
  OUTPUT @Mta;"CAL:ALIG"
  OUTPUT @Mta;"*OPC?"
  ENTER @Mta;A$
NEXT J
OUTPUT @Mta;"CAL:SKEW?"
ENTER @Mta;Tskew
PRINT "The measured skew value is: ";Tskew
```

Description

The ALIGN command performs a semi-automatic measurement of the channel-to-channel skew. This command measures the difference between the current setting of CAL:SKEW and the actual value and modifies the CAL:SKEW parameter.

The defined signal frequency must be greater than 20 MHz.

The basic procedure for using this command is as follows:

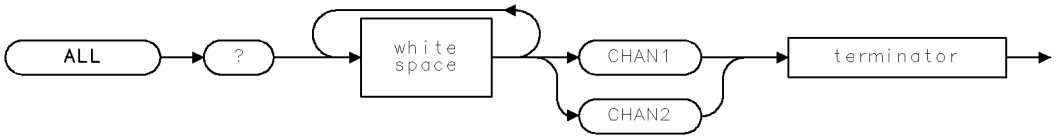
1. Provide an input to both channels.
2. Set the CAL:SKEW parameter to 0.
3. Set the input signal to f (which is the input frequency to the microwave transition analyzer).
4. Report the input signal frequency to the microwave transition analyzer.
5. Execute the CAL:ALIGN command.
6. Repeat steps 3 through 5 with progressively higher values of f until the desired resolution is achieved.

The choice for the first value of f must be low enough that the channel-to-channel skew is unambiguous (In other words, the channel-to-channel skew must be less than one-half the period).

ALL

Selects the specified channel to be calibrated.

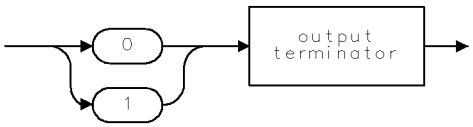
Syntax



Description

The ALL query selects the specified channel to be calibrated and will then return a parameter when complete. The return parameter will be a “0” if the calibration failed and a “1” if the calibration was successful.

Query Response



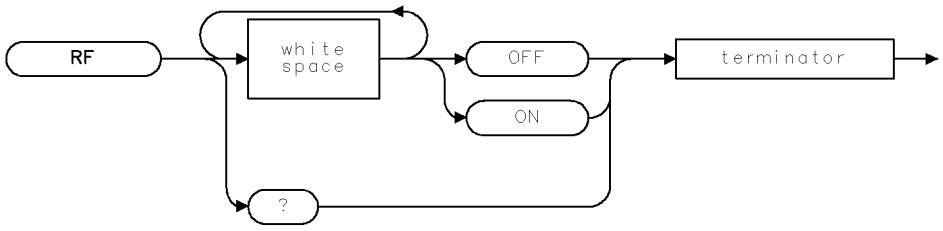
Query Example

```
OUTPUT @Mta;"CAL:ALL? CHAN1"  ! Start calibration of channel 1.
ENTER @Mta;A$
IF A$="1" THEN
PRINT "Calibration passed"
ELSE
PRINT "Calibration failed"
END IF
```

CORRect:RF

Enables and disables RF corrections.

Syntax



Preset State: ON
Related Commands: SWEep:TIME:TRANslate
USER:STATe

Command Example

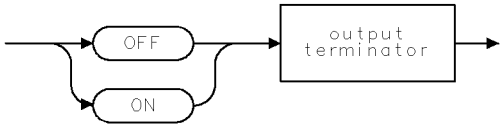
```
OUTPUT @Mta;"CAL:CORR:RF OFF"
```

Description

The CORRECT:RF command enables and disables RF corrections. The query returns the current state of the CORRECT:RF.

When ON, the RF corrections are applied based on the start and stop frequencies for the frequency domain trace (for time domain traces, the start and stop frequencies can be determined by performing an FFT). For example, if the start and stop frequencies are 5 Ghz and 25 GHz, the RF corrections between 5 GHz and 25 GHz are used. Refer to the SWEEP:TIME:TRANSLATE command under the SWEEP subsystem for the various possibilities for determining the start and stop frequencies. Also see the SWEEP:TIME:TRANSLATE and CALIBRATE:USER:STATE command descriptions.

Query Response



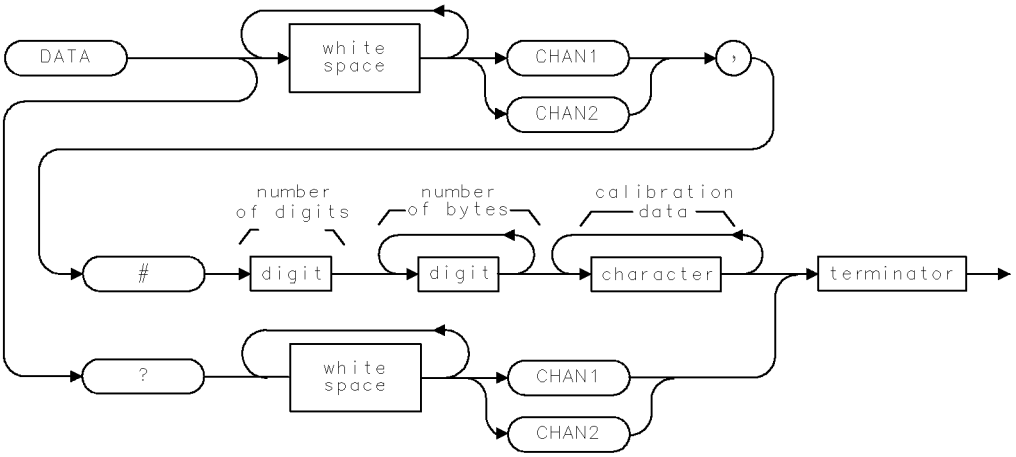
Query Example

```
OUTPUT @Mta;"CAL:CORR:RF?"
ENTER @Mta;Rf_correct$
```

DATA

Uploads and downloads calibration data to and from an external computer.

Syntax



Command Example

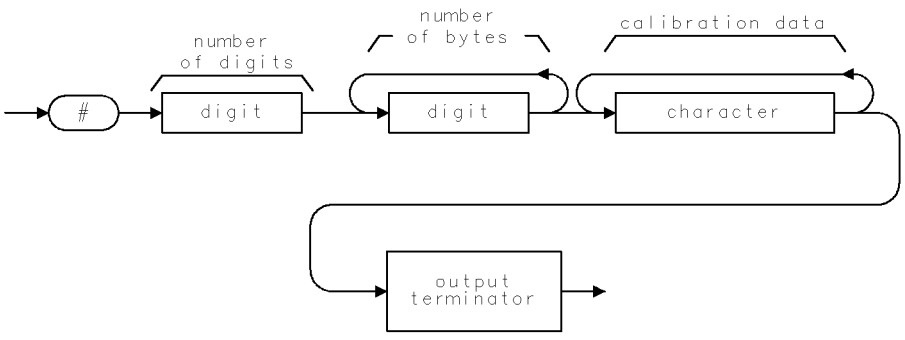
See the query example below.

Description

The DATA command allows the user to upload the calibration data to an external computer as well as to download data that had previously been uploaded. There is approximately 12 KBytes of information per channel.

Note There will always be an even number of bytes of information per channel.

Query Response



The query response is a definite length binary sequence defined as:

```
#<N><length><data><terminator>
```

where

<N>	= ASCII digit indicating number of digits in <length>
<length>	= N ASCII digits indicating the number of bytes in <data>
<data>	= the actual calibration data (in the microwave transition analyzer internal format)
<terminator>	= on HP-IB: line feed character with EOI asserted = on HP-MSIB: END immediate command

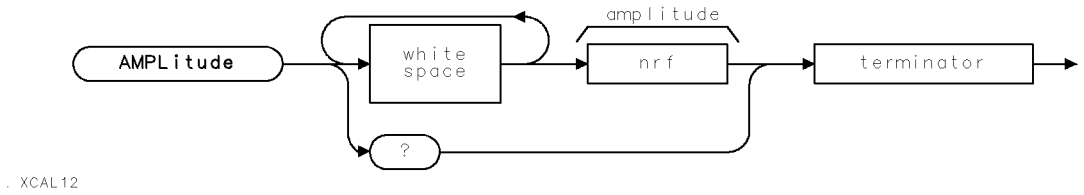
Query Example

```
OUTPUT @Mta;"CAL:DATA? CHAN2"                        ! This portion of code
ENTER @Mta USING "#,A,A";Dummy$,Digit$  ! uploads the channel 2
Format$="#,"&Digit$&"A"                 ! calibration data.
ENTER @Mta USING Format$;Size$
Cal_size=VAL(Size$)/2
ALLOCATE INTEGER cal_data(Cal_size-1)
FOR J=0 To Cal_size-1
ENTER @Mta USING "#,W";Cal_data(J)
NEXT J
ENTER @Mta USING "#,B";Line_feed
.
.
.
Header$="#"                                          ! This section downloads
Header$=Header$&VAL$(INT(LGT(2*Cal_size))+1)  ! the calibration
Header$=Header$&VAL$(2*Cal_size)              ! data into chan 2.
OUTPUT @Mta USING "#,K,K";"CAL:DATA CHAN2,",Header$
FOR J=0 TO Cal_size-1
OUTPUT @Mta USING "#,W";Cal_data(J)
NEXT J
OUTPUT @Mta;Line_feed END                            ! Line_feed = 10
```

SIGNal:AMPLitude

Controls the calibrator signal amplitude.

Syntax



Preset State: 0

Parameter Range: amplitude = 0 through 4

Command Example

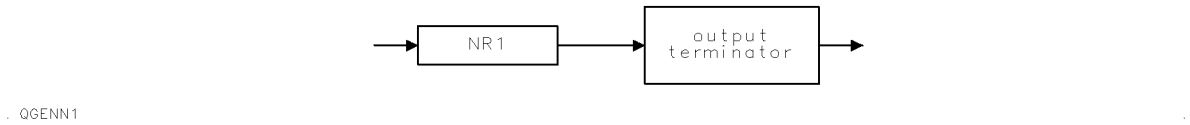
```
OUTPUT @Mta;"CAL:SIGN:AMPL 2"
```

Description

The SIGNAL:AMPLITUDE command controls the amplitude of the microwave transition analyzer calibration signal. The amplitude control is a unitless number that selects an attenuation setting. 0 corresponds to approximately 70 mV; 4 corresponds to approximately 300 mV.

The query returns the current state of SIGNAL:AMPLITUDE.

Query Response



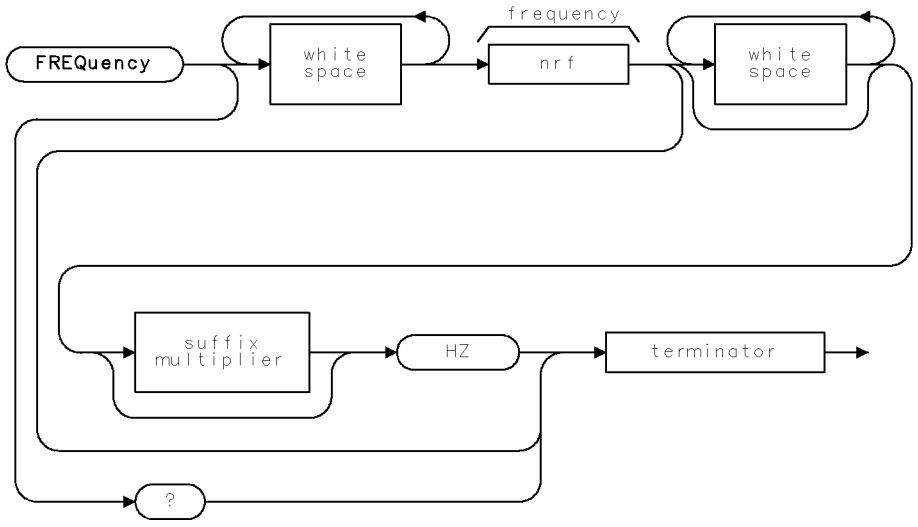
Query Example

```
OUTPUT @Mta;"CAL:SIGN:AMPL?"
ENTER @Mta;Amplitude
```

SIGNal:FREQuency

Controls the calibrator signal frequency.

Syntax



Preset State: 1 MHz

Parameter Range: frequency = ≈152.59 through 5 MHz

Command Example

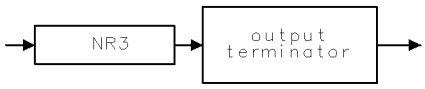
```
OUTPUT @Mta;"CAL:SIGN:FREQ 1.25 MHz"
```

Description

The SIGNAL:FREQUENCY command controls the frequency of the microwave transition analyzer calibration signal. The frequency is derived by dividing a 10 MHz source by an integer from 2 to 65535.

The query returns the current state of SIGNAL:FREQUENCY.

Query Response



QGENNR3

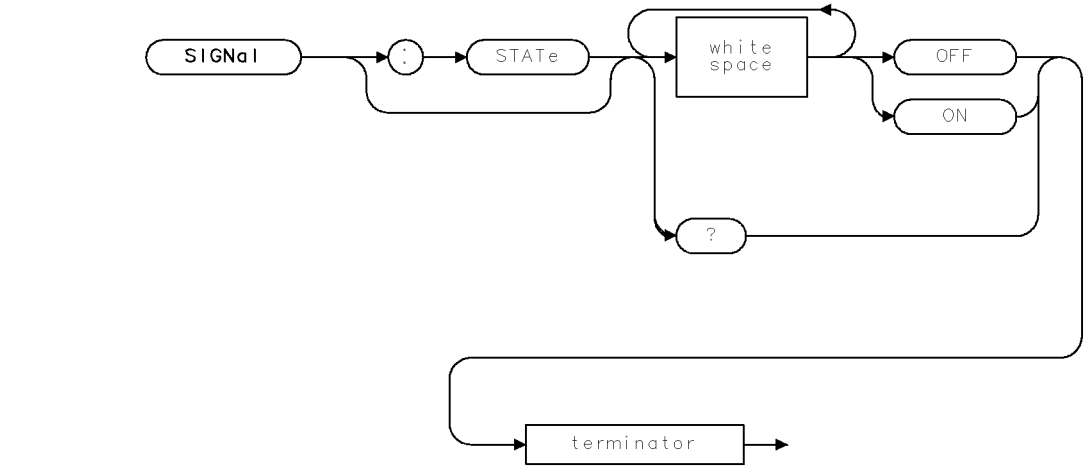
Query Example

```
OUTPUT @Mta;"CAL:SIGN:FREQ?"
ENTER @Mta;Frequency
```

SIGNal:STATe

Turns the calibrator source on or off.

Syntax



Preset State: OFF

Command Example

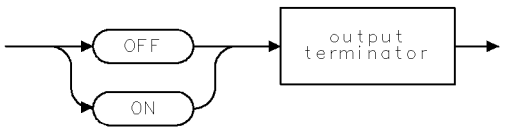
```
OUTPUT @Mta;"CAL:SIGN ON"
```

Description

The SIGNAL:STATE command turns the microwave transition analyzer source on or off.

The query returns the current state of SIGNAL:STATE.

Query Response



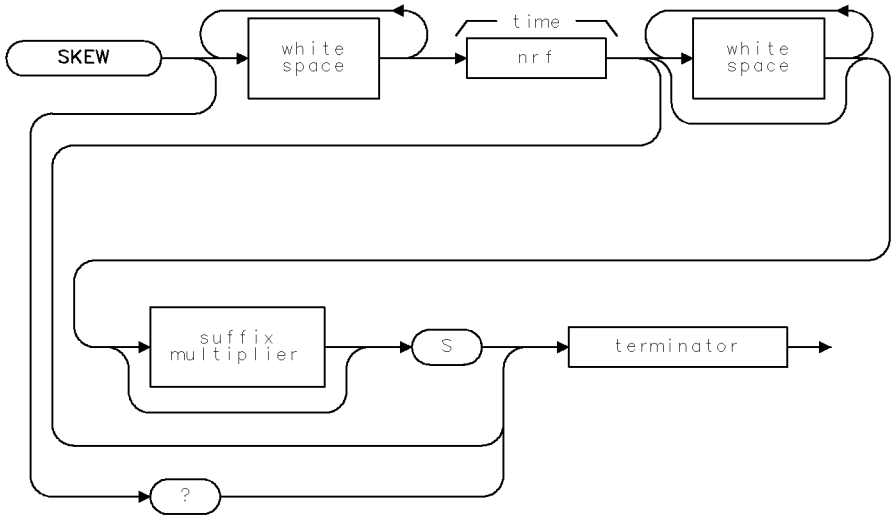
Query Example

```
OUTPUT @Mta;"CAL:SIGN?"
ENTER @Mta;State$
```

SKEW

Specifies the time skew between channels 1 and 2.

Syntax



Parameter Range: time = −1000 to +1000;
Additional limiting will be applied during measurement setup:
time sweep: abs (time/time range) < 4096/trace length
frequency sweep (stepped): no additional limit
Fundamental Unit: seconds

Command Example

```
OUTPUT @Mta;"CAL:SKEW 5 ns"
```

Description

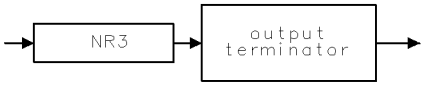
The SKEW command allows the user to specify the time skew between channel 1 and channel 2. In subsequent measurements, the module will attempt to align the channels by shifting the data of the appropriate channel by the specified time.

The value of SKEW can be set semi-automatically with the CALIBRATE:ALIGN command.

Note The ability of the module to align the channels is limited to approximately 20% of the current time range setting. Also, the time skew is not canceled in the following modes:

- When the time range is equal to 0
- When an RF profile is used

Query Response



QGENNR3

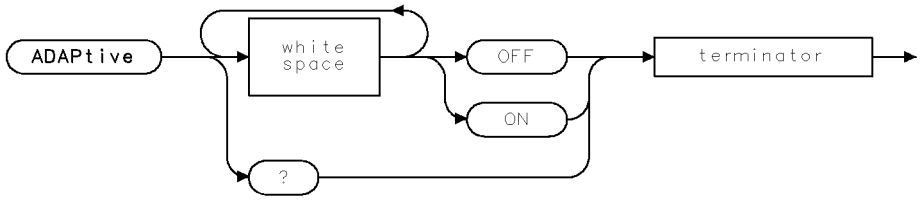
Query Example

```
OUTPUT @Mta;"CAL:SKEW?"
ENTER @Mta;Chan_skew
```

USER:ADAPtive

Enables or disables the adaptive store of trace data into user corrections.

Syntax



Preset State: OFF
Related Commands: STORe
CALibrate:USER:THReshold

Command Example

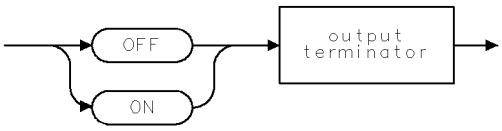
```
OUTPUT @Mta;"CAL:USER:ADAP ON"
```

Description

The USER:ADAPTIVE command enables or disables the adaptive store of trace data into user corrections. The query returns the current setting of USER:ADAPTIVE.

When ON, the store of trace data to user corrections will not perform any trace compression and will store only data points with magnitudes greater than the threshold set by CALIBRATE:USER:THRESHOLD. If there are more points in the trace than can be stored into user corrections, only the first N points found will be stored.

Query Response



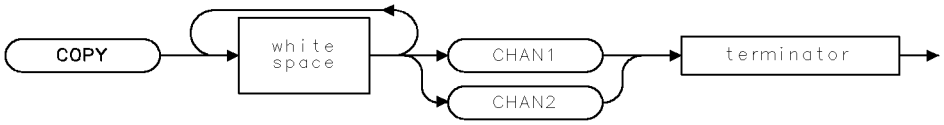
Query Example

```
OUTPUT @Mta;"CAL:USER:ADAP?"
ENTER @Mta;Adaptive$
```

USER:COPY

Copies user calibration data between channels.

Syntax



xcal15

Command Example

```
OUTPUT @Mta;"CAL:USER:COPY CHAN1"
```

Description

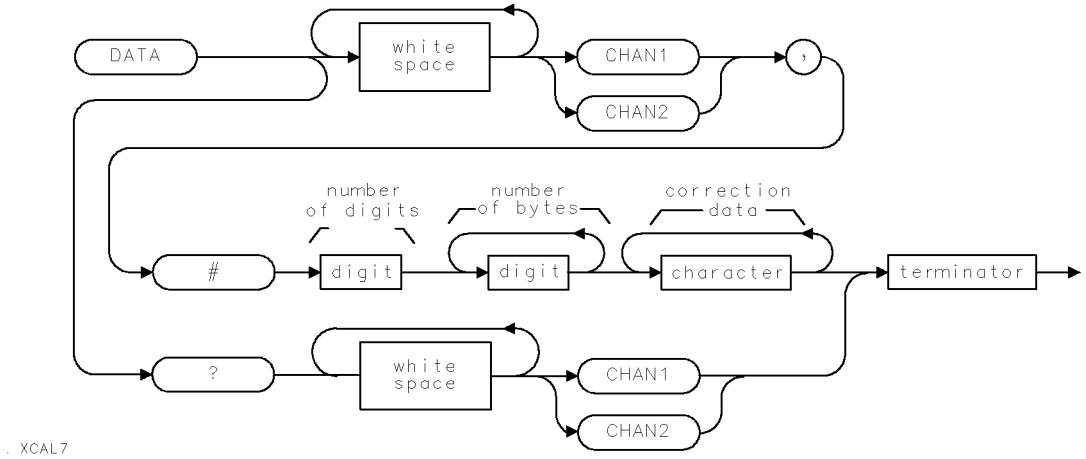
The USER:COPY command will copy user calibration data from the specified channel to the other channel.

There is no query form.

USER:DATA

Sets and reads user correction data.

Syntax



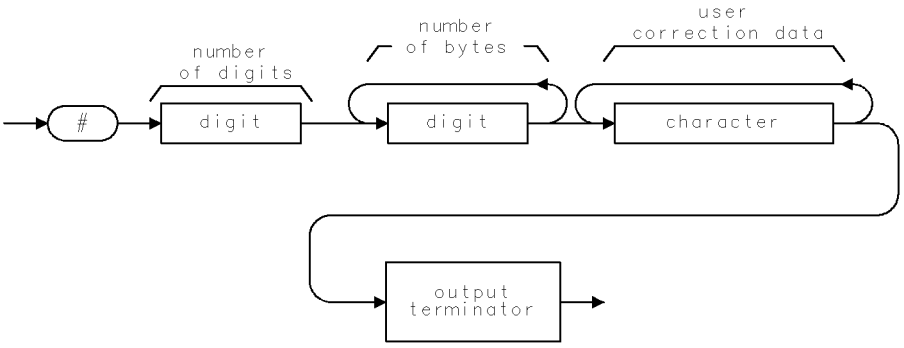
Command Example

See the query example below.

Description

The USER:DATA command allows the user to set and to read the user correction data. Each point in the user correction data is composed of frequency, magnitude, and phase values. The data can be applied as a post-processing operation to the input data (for example, channel 1 and channel 2).

Query Response



QCAL7

The query response is a definite length binary sequence defined as:

```
#<N><length><data><terminator>
```

where

<N>	= ASCII digit indicating number of digits in <length>
<length>	= N ASCII digits indicating the number of bytes in <data>
<data>	= the actual user correction data (in the microwave transition analyzer internal format)
<terminator>	= on HP-IB: line feed character with EOI asserted = on HP-MSIB: END immediate command

The format of the data is four bytes indicating information about the data followed by 80-bit data items. This data is organized as follows:

Variable Name	**Set 8 bits to:**
changed;	1 (this byte should always be set to 1)
sorted;	0 (no, sort the data by frequency)
log_freq;	0 (no, linear interpolation)
points;	number of data points
data[129];	correction data with each data point composed of 80 bits that should be set as follows: bits 79-72 type (0 = “slope”, 1 = “flat”) bits 71-64 higher 8 bits for frequency bits 63-48 lower 16 bits for frequency bits 47-32 lower 16 bits for magnitude bits 31-24 higher 8 bits for magnitude bits 23-16 higher 8 bits for phase bits 15-0 lower 16 bits for phase

The frequency, magnitude, and phase fields are integers with the least significant bit (LSB) of each field representing:

frequency LSB = 10 kHz
magnitude LSB = 0.0001 dB
phase LSB = 0.0001 degree

Query Example

Read data example:

```
10    DIM Type(128),Freq(128),Magn(128),Phase(128)
20    ASSIGN @Mta TO 710
30    OUTPUT @Mta;"CAL:USER:DATA? CHAN1"              ! Ask for data.
40    ENTER @Mta USING "#,A,A";Header$,Digit$         ! Determine size.
50    ENTER @Mta USING "#,"&Digit$&"D";Size
60    Size=(Size-4)/10
70    ! Read "header"
80    ENTER @Mta USING "#,B,B,B,B";Changed,Sorted,Log_freq,Num_points
90    !
100   ! Read the user cal data endpoints
110   !
120   FOR J=0 TO Size-1
130      !
140      ! Read type.... 0 = "slope", 1 = "flat"
150      !
160      ENTER @Mta USING "#,B";Type(J)
170      !
180      ! Read frequency, convert to MHz
190      !
200      ENTER @Mta USING "#,B,W";High8,Low16
210      Freq(J)=.01*FNCombine(High8,Low16)
220      !
230      ! Read magnitude, convert to dB
240      !
250      ENTER @Mta USING "#,W,B";Low16,High8
260      Magn(J)=.0001*FNCombine(High8,Low16)
270      !
280      ! Read phase, convert to degrees
290      !
300      ENTER @Mta USING "#,B,W";High8,Low16
310      Phase(J)=.0001*FNCombine(High8,Low16)
320   NEXT J
330   ENTER @Mta;Lf$
340   END
350   !
360   DEF FNCombine(High8,Low16)
370     IF High8>127 THEN High8=High8-256
380     IF Low16<0 THEN Low16=Low16+65536
390     RETURN (65536*High8+Low16)
400   FNEND
```

Write data example:

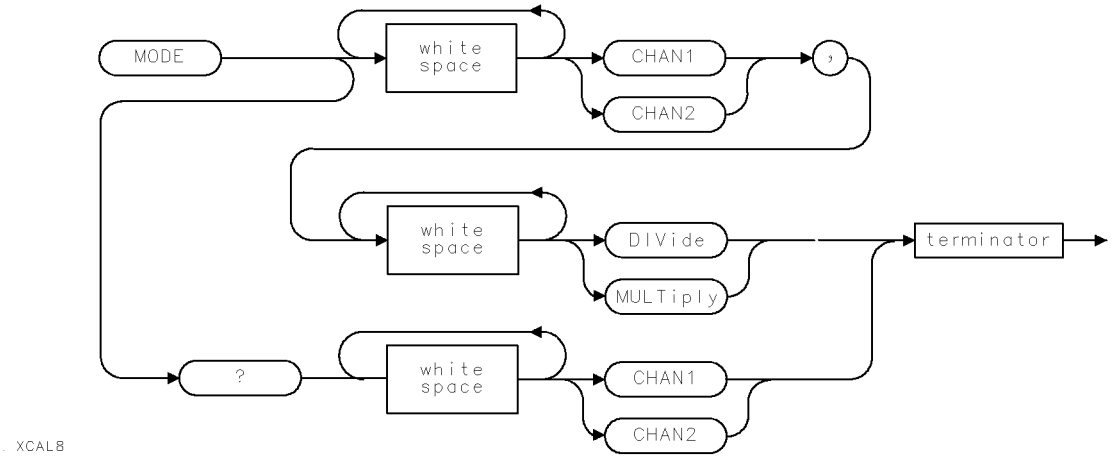
```
10    ASSIGN @Mta TO 710
20    RESTORE User_cal_data
30    READ Num_points                    ! Number of endpoints.
40    !
50    Size$=VAL$(4+10*Num_points)        ! Compute the size of the user
60    Digit$=VAL$(LEN(Size$))            ! calibration data.
70    !
80    ! Send the user calibration data "header"
90    !
100   OUTPUT @Mta USING "#,K";"CAL:USER:DATA CHAN1,#41294"
110   OUTPUT @Mta USING "#,B";1           ! "changed", always set to 1
120   OUTPUT @Mta USING "#,B";0           ! "sorted", always set to 0
130   OUTPUT @Mta USING "#,B";0           ! "log_freq"
140   OUTPUT @Mta USING "#,B";Num_points
150   !
160   ! Send the user calibration data endpoints
170   !
180   FOR J=0 TO Num_points-1
190      READ Type,Freq,Magn,Phase
200      !
210      ! Send type.... 0 = "slope", 1 = "flat"
220      !
230      OUTPUT @Mta USING "#,B";Type
240      !
250      ! Convert frequency to 10 kHz per bit, format, send
260      !
270      Split_value(100*Freq,High8,Low16)
280      OUTPUT @Mta USING "#,B,W";High8,Low16
290      !
300      ! Convert magnitude to .0001 dB per bit, format, send
310      !
320      Split_value(10000*Magn,High8,Low16)
330      OUTPUT @Mta USING "#,W,B";Low16,High8
340      !
350      ! Convert phase to .0001 degree per bit, format, send
360      !
370      Split_value(10000*Phase,High8,Low16)
380      OUTPUT @Mta USING "#,B,W";High8,Low16
390   NEXT J
400   !
410   ! Fill the rest of the data structure with 0
420   !
430   FOR J=Num_points TO 128
440     OUTPUT @Mta USING "#,W,W,W,W,W";0,0,0,0,0
450   NEXT J
```

```
460   OUTPUT @Mta USING "#,B";10 END  ! Send terminating LF.
470   !
480   User_cal_data:   DATA 5
490   !        Type,  Freq (MHz),  Magn (dB),  Phase (deg)
500   DATA     0,      100,        -10,         45
510   DATA     0,      200,         -8,         30
520   DATA     0,      350,         -4,         10
530   DATA     0,      500,          0,        -20
540   DATA     1,     1000,          5,        -50
550   !
560   END
570   !
580   SUB Split_value(Value,High8,Low16)
590     High8=Value DIV 65536
600     Low16=Value MOD 65536
610     IF Low16<0 THEN
620        Low16=Low16+65536
630        High8=High8-1
640     END IF
650     IF Low16>32767 THEN Low16=Low16-65536
660   SUBEND
```

USER:MODE

Determines how user correction data is applied.

Syntax



Preset State: Unaffected by *RST, IP, or power cycle

Command Example

```
OUTPUT @Mta;"CAL:USER:MODE CHAN2, DIV"
```

Description

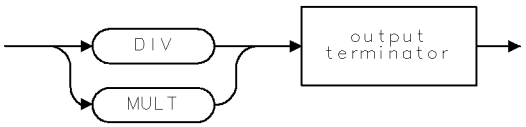
The USER:MODE command determines how user correction data is applied when enabled.

When MULTIPLY is specified, the input data is multiplied by the user correction data. This is typically used to apply filtering operations to the input data.

When DIVIDE is specified, the input data is divided by the user correction data. This is typically used to normalize input data.

The query returns the current state of USER:MODE.

Query Response



. QCAL8 .

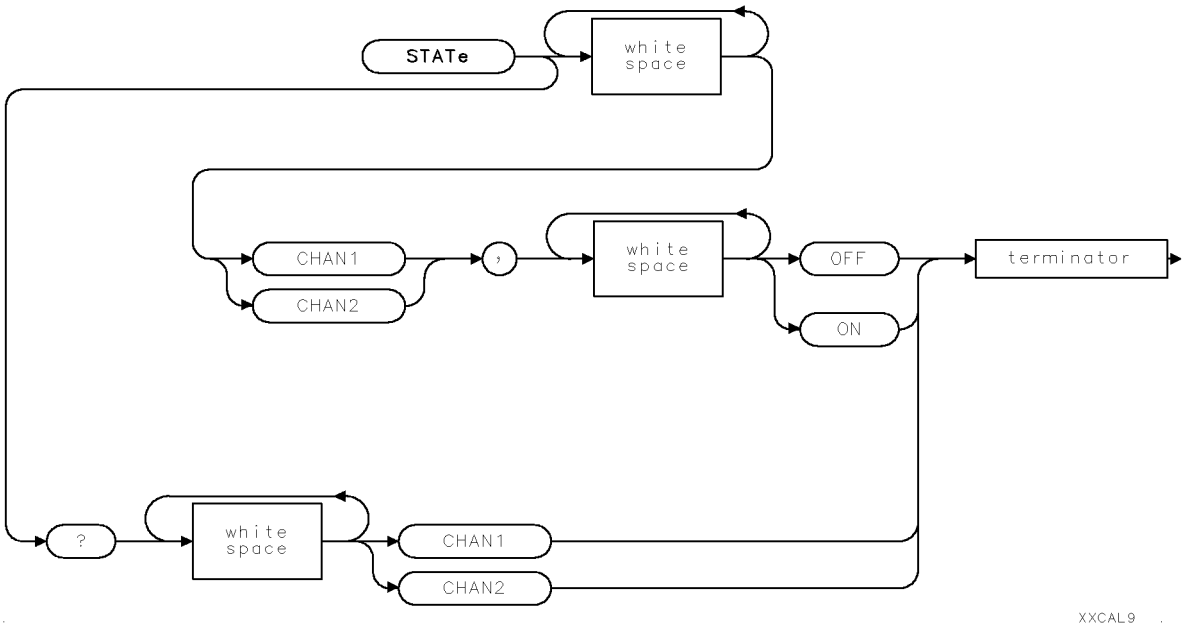
Query Example

```
OUTPUT @Mta;"CAL:USER:MODE? CHAN1"
ENTER @Mta;User_mode$
```

USER:STATe

Enables and disables user correction data.

Syntax



Preset State: OFF
Related Commands: SWEep:TIME:TRANslate
CALibrate:CORRect:RF

Command Example

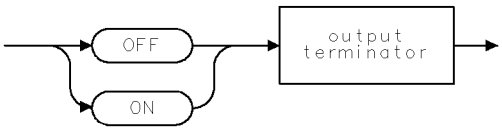
```
OUTPUT @Mta;"CAL:USER:STAT CHAN1,ON"
```

Description

The USER:STATE command enables and disables user corrections. The query returns the current state of USER:STATE.

When turned ON, the USER corrections are applied based on the start and stop frequencies for the frequency domain trace; for time domain traces, the start and stop frequencies can be determined by performing an FFT. For example, if the start and stop frequencies are 5 GHz and 25 GHz the USER corrections between 5 GHz and 25 GHz are used. Refer to the SWEEP:TIME:TRANSLATE command for the various possibilities for determining the start and stop frequencies.

Query Response



QGENON

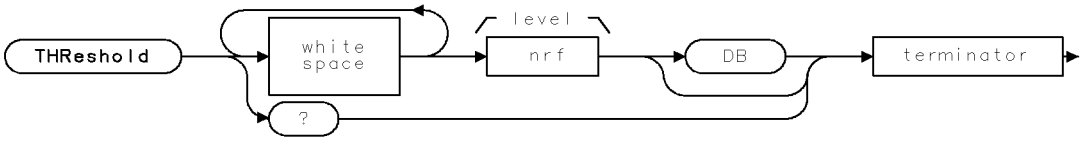
Query Example

```
OUTPUT @Mta;"CAL:USER:STAT?"
ENTER @Mta;User_correct$
```

USER:THReshold

Sets the level for the magnitude compare of the adaptive store of trace data.

Syntax



Preset State: −100 dB
Parameter Range: level = −120 dB through +120 dB
Related Commands: STORe
CALibrate:USER:ADAPtive

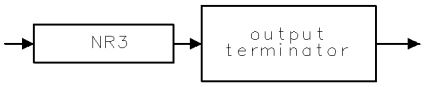
Command Example

```
OUTPUT @Mta;"CAL:USER:THR -50 dB"
```

Description

The USER:THRESHOLD command sets the level for the magnitude compare of the adaptive store of trace data into user corrections. The query returns the current setting of USER:THRESHOLD.

Query Response



Query Example

```
OUTPUT @Mta;"CAL:USER:THR?"
ENTER @Mta;Threshold
```

8

Channel Subsystem

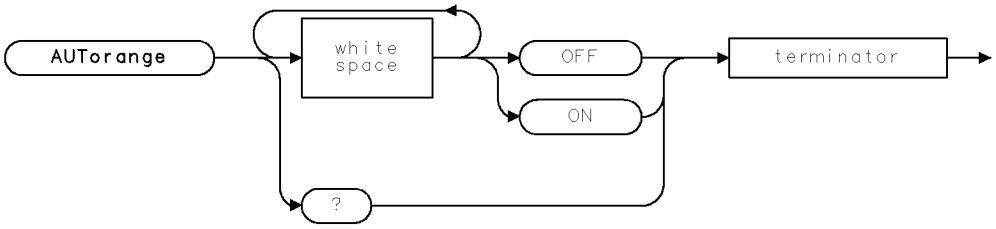
The CHANNEL subsystem allows the user to control all vertical or Y-axis functions of the input channels in the microwave transition analyzer. See the following list for a command syntax summary of the CHANNEL subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:CHANnel<channel_number>		
:AUTorange	OFF|ON	
:COUNt	<nrf>	
:COUPling	AC|DC	
:NULL?	ALL|FILTer|CURRent[,GND|IF]	[query only]
:OFFSet	<nrf> [[<suffix multiplier>]V]	
:PROBe	<nrf> [DB]	
:RANGe	<nrf> [[<suffix multiplier>]V]	
:RLEVel	<nrf> [DBM]	
:SMOoth		
:COUNt	<nrf>	
[:STATe]	OFF|ON	
:TYPE	AVERage|FAVerage|HOLD|MAXhold| MINhold|NORMal	
:ZOOM		
:CENTer	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:FREQ	OFF|ON	
:SPAN	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:STARt	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
[:STATe]	OFF|ON	
:STOP	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:TIME	OFF|ON	

AUTorange

Enables or disables the automatic vertical-sensitivity adjust.

Syntax



Preset State: ON for standard systems;

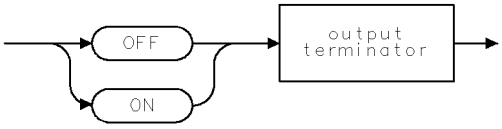
Command Example

```
OUTPUT @Mta;"CHAN1:AUT ON"
```

Description

The AUTORANGE command enables or disables the automatic vertical-sensitivity adjust for the specified channel. This is a hardware adjustment which is independent of the displayed scaling. The query returns the current state of the automatic vertical-sensitivity adjustment.

Query Response



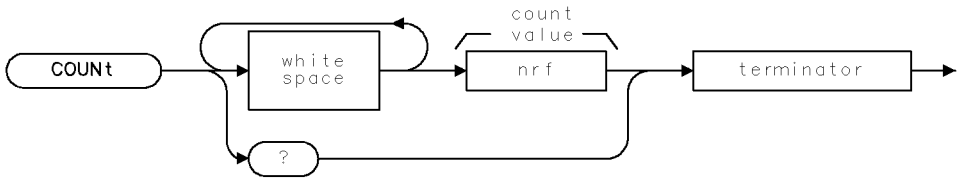
Query Example

```
OUTPUT @Mta;"CHAN2:AUT?"
ENTER @Mta;Autorange$
```

COUNt

Sets the averaging count.

Syntax



Preset State: 16
Parameter Range: count value = 1 through 1024

Command Example

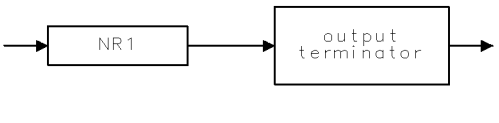
```
OUTPUT @Mta;"CHAN1:COUN 55"
```

Description

The COUNT command is used to set the averaging count value for the specified channel. The query returns the last value set.

Note that if the command program header is CHAN: (as opposed to CHAN1: or CHAN2:) the command applies to all channels and the query applies to channel 1 (which is identical in operation to ACQUIRE:COUNT). Refer to the ACQUIRE:COUNT command for a detailed description of the averaging operation.

Query Response



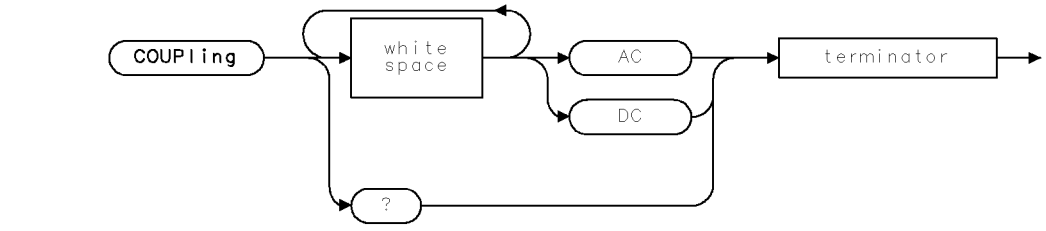
Query Example

```
OUTPUT @Mta;"CHAN1:COUN?"
ENTER @Mta;Count
```

COUPling

Selects either ac or dc IF signal coupling for the HP 70820A.

Syntax



Preset State: DC

Command Example

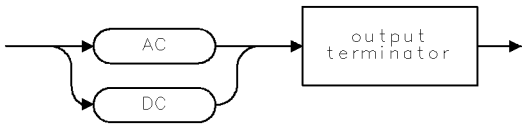
```
OUTPUT @Mta;"CHAN1:COUP AC"
```

Description

The COUPLING command sets the IF signal coupling for the microwave transition analyzer to be either ac or dc coupled. The query returns the current signal coupling.

This command does *not* set the input coupling at the front panel, it sets the coupling between the output of the samplers and the input of the IF.

Query Response



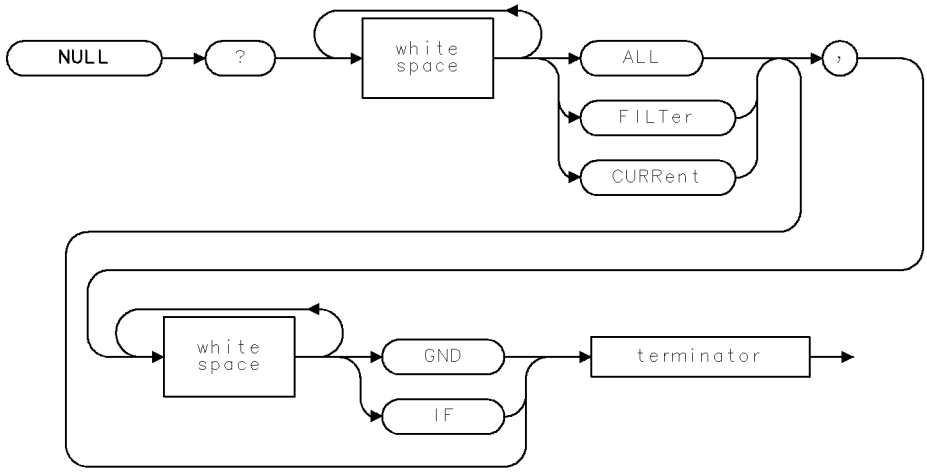
Query Example

```
OUTPUT @Mta;"CHAN1:COUP?"
ENTER @Mta;Coupling$
```

NULL

Performs an update of the dc offset calibration data.

Syntax



Description

The NULL query performs an update of the dc offset calibration data. This query always returns a response that indicates whether or not the dc DAC settings were updated. A response of "1" indicates successful completion, "0" indicates a failure (in which case no calibration data will be changed).

The parameters for the NULL command specify which sections of the data to update and, optionally, which input to use for the measurement.

- Update section parameters:

  ALL This option will update every gain stage for every filter.

  FILTER This option updates all gain stages for the current filter only.

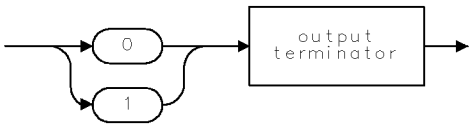
  CURRENT This option updates only the current gain stage for the current filter.

- Input selection parameters (optional):

  GND This is the default input. When GND is selected as the measurement input, the measurement is made with an internal ground reference. This allows updating of the dc calibration data without the need to disconnect the measurement setup from the inputs.

  IF This option selects the I.F. as the measurement input. It is expected that a 50 Ω termination has been placed on the input. Performing this selection will give slightly more accurate results.

Query Response



QCAL3

Query Example

Example 1

```
OUTPUT @Mta;"CHAN1:NULL? ALL"
ENTER @Mta;Null$
```

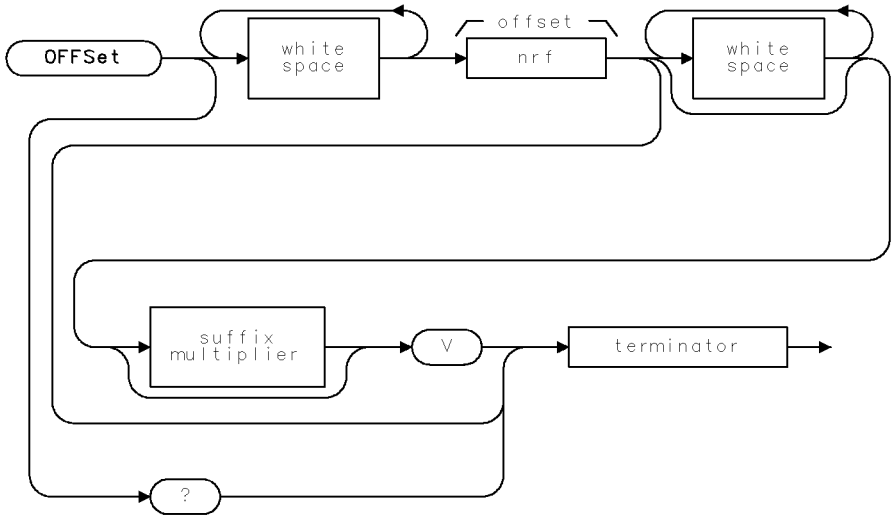
Example 2

```
OUTPUT @Mta;"CHAN1:NULL? ALL, IF"
ENTER @Mta;Null$
```

OFFSet

Sets the amplitude reference of the next measurement.

Syntax



Preset State:	0 V
Parameter Range:	offset = −0.4 to +0.4 volts (when CHANx:PROBe = 0 dB)
Fundamental Unit:	volts
Related Commands:	CHANx:AUTorange
	CHANx:PROBe
	TRIGger:LEVel

Command Example

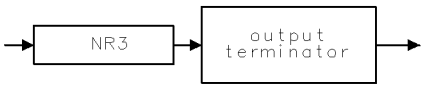
```
OUTPUT @Mta;"CHAN1:OFFS 50 mV"
```

Description

The OFFSET command sets the mid-range voltage level (amplitude reference) of the next measurement for the specified channel (CHANx).

The query returns the currently set value.

Query Response



QGENNR3

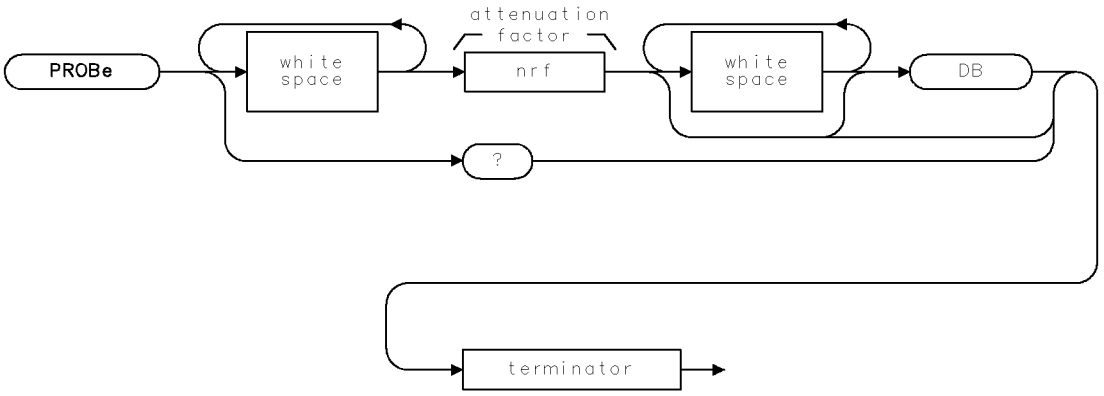
Query Example

```
OUTPUT @Mta;"CHAN1:OFFS?"
ENTER @Mta;Offset
```

PROBe

Sets the value of the probe attenuation factor.

Syntax



XCHA5

Preset State: 0 dB
Parameter Range: attenuation factor = −100 dB to +100 dB
Fundamental Unit: dB

Command Example

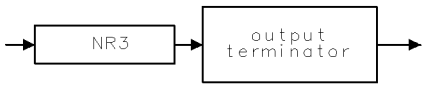
```
OUTPUT @Mta;"CHAN1:PROB 20 dB"
```

Description

The PROBE command sets the value of the probe attenuation factor for the indicated channel (CHANx). When set, the current values for CHANx:RANGE and CHANx:OFFSET are scaled accordingly (that is, if RANGE = 10 mV when PROBE is 0 dB and PROBE is then set to 20 dB, RANGE will become 100 mV).

The query returns the current probe attenuation factor.

Query Response



QGENNR3

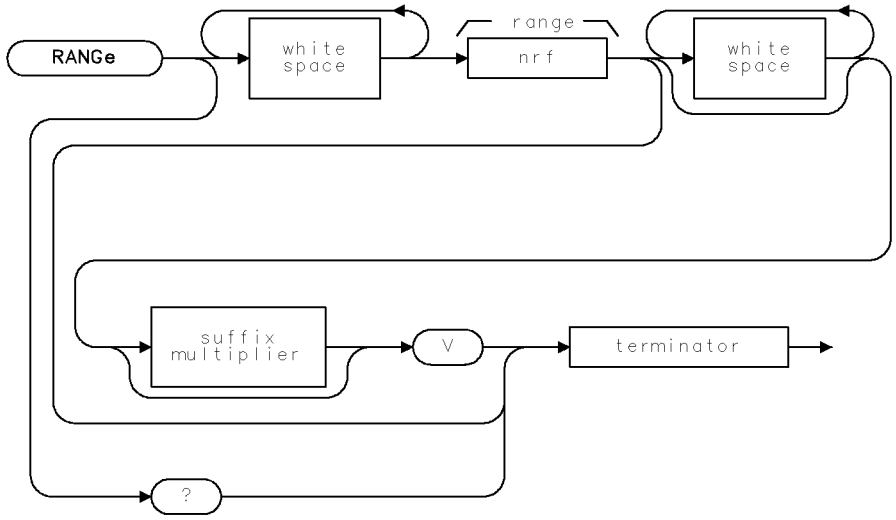
Query Example

```
OUTPUT @Mta;"CHAN1:PROB?"
ENTER @Mta;Probe
```

RANGe

Sets the voltage range of the next measurement.

Syntax



Preset State:	0.2 V
Parameter Range:	range = 200 μV through 640 mV (when CHANx:PROBE = 0 dB)
Fundamental Unit:	volts
Related Commands:	CHANx:AUTorange
	CHANx:PROBe
	TRIGger:LEVel

Command Example

```
OUTPUT @Mta;"CHAN2:RANG 50 mV"
```

Description

The RANGE command sets the voltage-amplitude range (in volts) of the next measurement for the specified channel (CHANx). This command sets the same parameter as the CHANx:RLEVEL command.

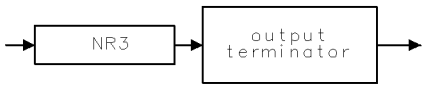
The query returns the current voltage range in volts.

Note The value of CHANx:RANGE is affected by CHANx:AUTORANGE, which defaults to ON.

To find the equivalent setting for CHANx:RLEVEL, the conversion is peak-to-peak volts to dBm:

$$10 \times LOG_{10}((0.707 \times range/2)^2/50/0.001)$$

Query Response



QGENNR3

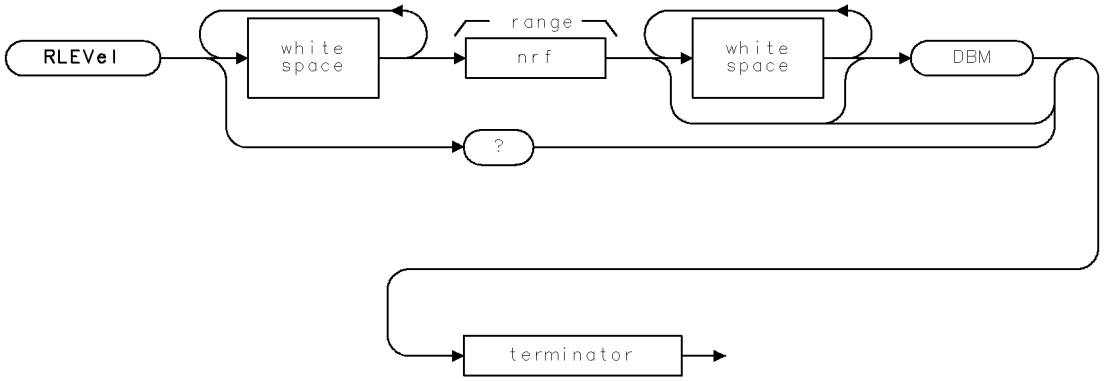
Query Example

```
OUTPUT @Mta;"CHAN1:RANG?"
ENTER @Mta;Range
```

RLEVel

Sets the amplitude of the next measurement.

Syntax



Preset State: −10 dBm
Parameter Range: range = −70 dBm through 0.102 dBm (when PROBE = 0 dB)
Fundamental Unit: dBm

Command Example

```
OUTPUT @Mta;"CHAN1:RLEV -20 dBm"
```

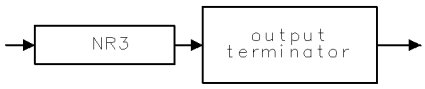
Description

The RLEVEL command sets the voltage-amplitude range (in dBm) of the next measurement for the specified channel (CHANx). This command sets the same parameter as the CHANx:RANGE command. The query returns the current voltage range in dBm.

Note that to find the equivalent setting for CHANx:RANGE, the conversion is dBm to peak-to-peak volts:

$$2 \times sqrt(2 \times 50 \times 0.001 \times 10^{level10})$$

Query Response



QGENNR3

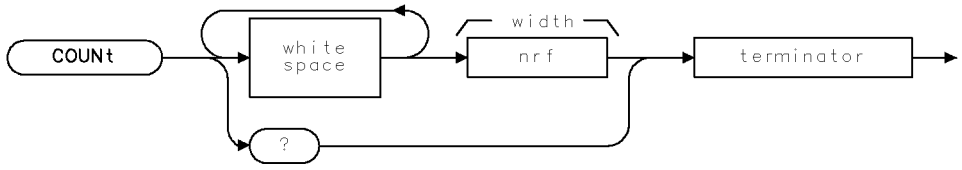
Query Example

```
OUTPUT @Mta;"CHAN1:RLEV?"
ENTER @Mta;Ref_level
```

SMOoth:COUNt

Sets the width of the median filter for the specified channel.

Syntax



Preset State: 16
Parameter Range: width = 1 through 1024
Related Commands: CHANx:SMOoth:COUNt
CHANx:SMOoth:STATe

Command Example

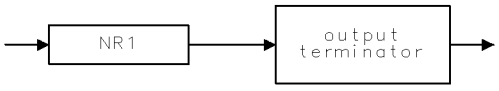
```
OUTPUT @Mta;"CHAN1:SMO:COUN 55"
```

Description

The SMOOTH:COUNT command is used to set the width of the median filter for the specified channel. The query returns the last value set.

See Appendix E, “Median Smoothing,” for a description of median filtering.

Query Response



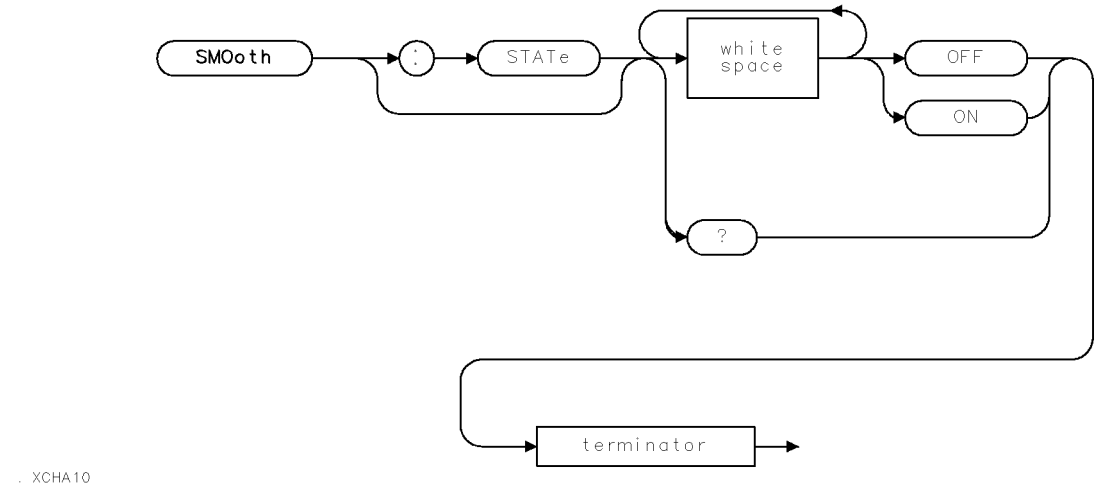
Query Example

```
OUTPUT @Mta;"CHAN2:SMO:COUN?"
ENTER @Mta;Median_width
```

SMOoth:STATe

Enables or disables a median filter process for the specified channel.

Syntax



Preset State: OFF
Related Commands: CHANx:TYPE
CHANx:SMOoth:COUNt

Command Example

```
OUTPUT @Mta;"CHAN1:SMO:STAT ON"
OUTPUT @Mta;"CHAN2:SMO OFF"          ! STATE is optional.
```

Description

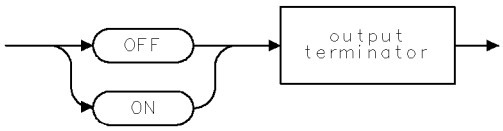
The SMOOTH:STATE command enables or disables a median filter process for the specified channel. The query returns the current setting.

When CHANx:SMOOTH:STATE is set to ON, the value of CHANx:TYPE will be set to NORMAL.

See Appendix E, "Median Smoothing," for a description of median filtering.

SMOOTH:STATE is not independent of CHANx:TYPE. Median smoothing can only be applied if TYPE is NORMAL or AVERAGE.

Query Response



QGENON

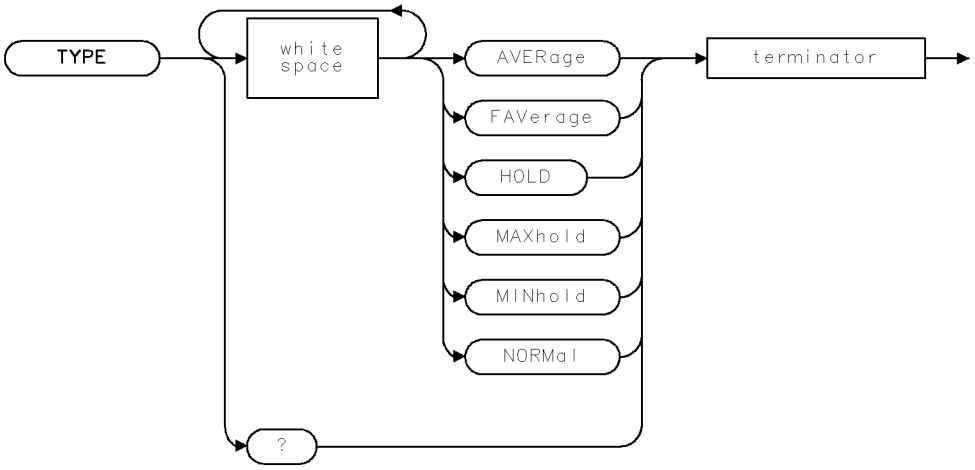
Query Example

```
OUTPUT @Mta;"CHAN1:SMO?"
ENTER @Mta;State$
```

TYPE

Sets the post-processing operation.

Syntax



Preset State: NORMal
Related Commands: ACQuire:TYPE
CHANx:SMOoth:STATe

Command Example

```
OUTPUT @Mta;"CHAN:TYPE AVER"
```

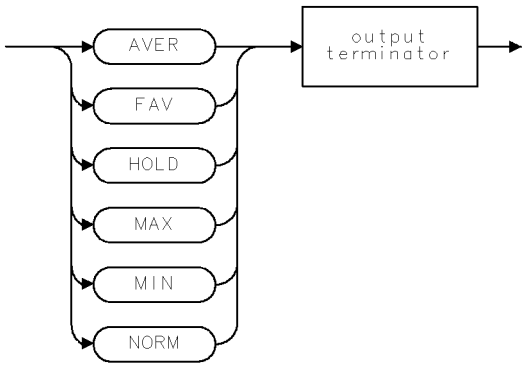
Description

The TYPE command controls the selection of a post-processing operation for the specified channel. However, this does not set the post-processing operation for functions (see FUNCx:TYPE). The query returns the last value set.

It should be noted that if the command program header is CHAN: (as opposed to CHAN1: or CHAN2:) the command applies to all channels and the query (which is identical to ACQUIRE:TYPE) applies to channel 1. Refer to the ACQUIRE:TYPE command for a description of parameters.

TYPE is not independent of median smoothing. Median smoothing can only be applied if TYPE is NORMAL or AVERAGE.

Query Response



QCHA8

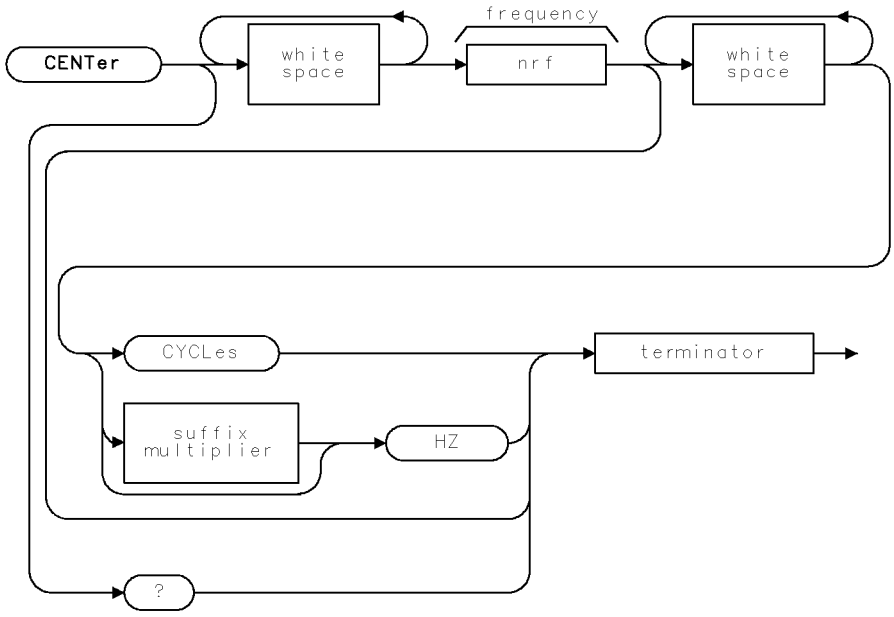
Query Example

```
OUTPUT @Mta;"CHAN1:TYPE?"
ENTER @Mta;Type$
```

ZOOM:CENTer

Specifies the center frequency used for the zoom transform.

Syntax



XCHA11

Preset State: 50 MHz
Parameter Range: frequency = 0 through 100 GHz
Fundamental Unit: hertz

Note A more restrictive range may be imposed by the measurement setup.

Command Example

```
OUTPUT @Mta;"CHAN1:ZOOM:CENT 5GHZ"
```

Description

The ZOOM:CENTER command specifies the center frequency used for the zoom transform. The CENTER command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting a measurement independent of the specific signal frequency.

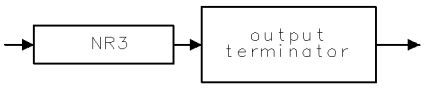
The query always returns the current setting of ZOOM:CENTER in a value of hertz.

The module converts cycles to hertz by using the following equation:

$$hertz = cycles \times signal\ frequency$$

Note Currently this command modifies the same value as the FUNCx:ZOOM:CENTER and SWEEP:TIME:ZOOM:CENTER commands. In future revisions, this value may be independently settable for each channel and function.

Query Response



QGENNR3

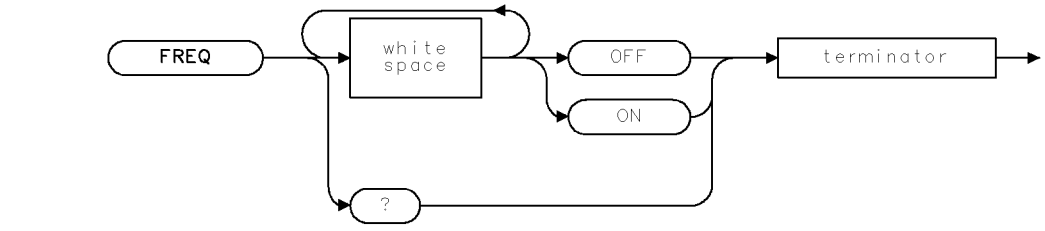
Query Example

```
OUTPUT @Mta;"CHAN1:ZOOM:CENT?"
ENTER @Mta;Center_freq
```

ZOOM:FREQ

Enables or disables the use of zoom transform data when in the frequency domain.

Syntax



Preset State: OFF

Command Example

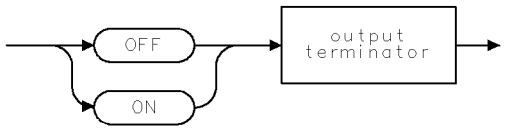
```
OUTPUT @Mta;"CHAN2:ZOOM:FREQ OFF"
```

Description

The ZOOM:FREQ command enables or disables the use of zoom transform data when the channel is in the frequency domain. The query returns the current state.

For example, if ZOOM:FREQ is OFF and the channel is displayed in the frequency domain, the data will be the normal FFT of the time domain data. If ZOOM:FREQ is ON and the channel is displayed in the frequency domain, the data will be from the zoom transform. Note that this command does not enable the zoom transform, it just determines whether or not to use the zoom transform data once the transform is enabled (see CHANx:ZOOM:STATE).

Query Response



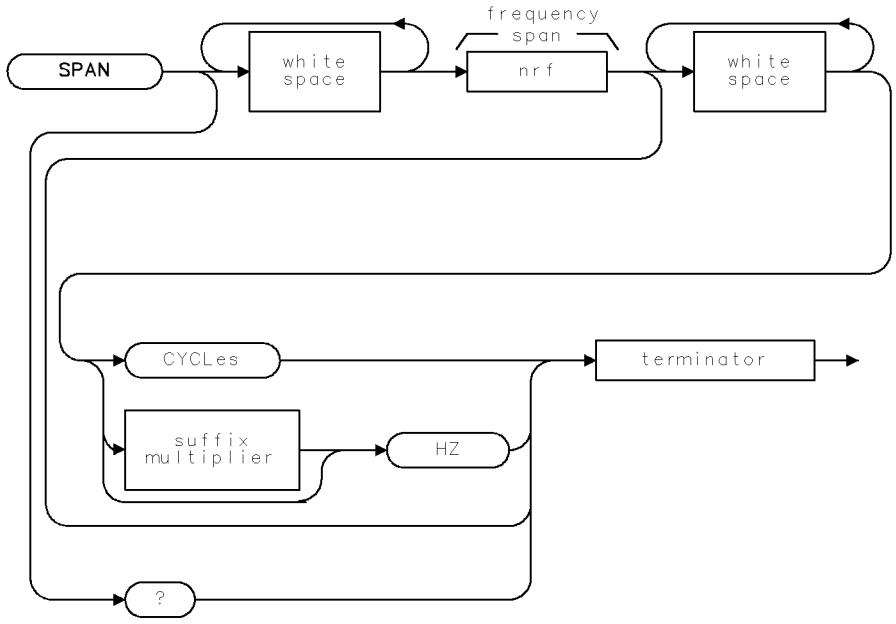
Query Example

```
OUTPUT @Mta;"CHAN1:ZOOM:FREQ?"
ENTER @Mta;Zoom_state
```

ZOOM:SPAN

Specifies the frequency span used for the zoom transform.

Syntax



Preset State: 100 MHz
Parameter Range: frequency span = 0 through 100 GHz
Fundamental Unit: hertz

Note A more restrictive range may be imposed by the measurement setup.

Command Example

```
OUTPUT @Mta;"CHAN1:ZOOM:SPAN 10GHZ"
```

Description

The ZOOM:SPAN command specifies the frequency span used for the zoom transform. The query returns the current setting.

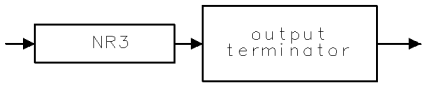
The SPAN command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting a measurement independent of the specific signal frequency. The query always returns the value in hertz.

The module converts cycles to hertz by using the following equation:

$$hertz = cycles \times signal\ frequency$$

Note Currently this command modifies the same value as the FUNCx:ZOOM:SPAN and SWEEP:TIME:ZOOM:SPAN commands. In future revisions this value may be independently settable for each channel and function.

Query Response



QGENNR3

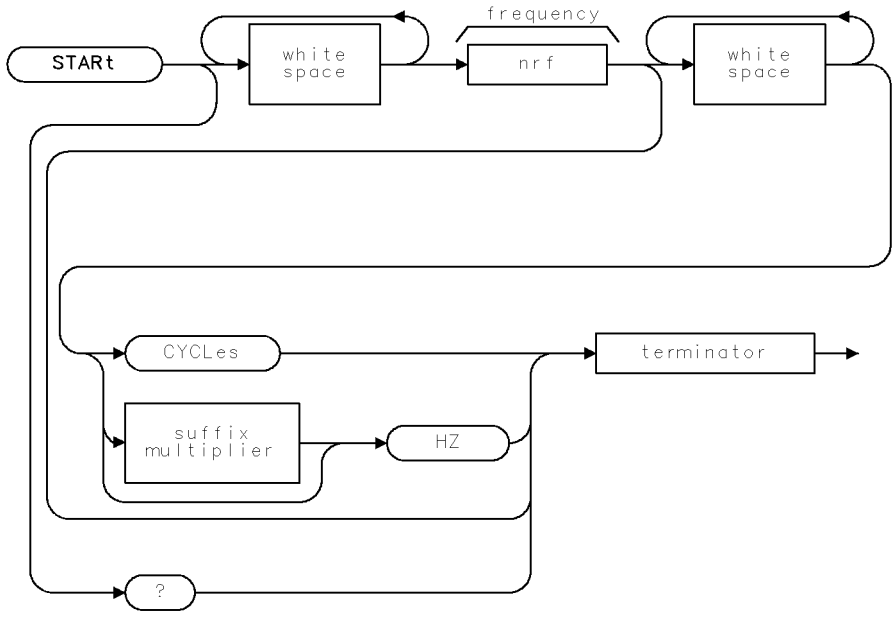
Query Example

```
OUTPUT @Mta;"CHAN1:ZOOM:SPAN?"
ENTER @Mta;Span
```

ZOOM:STARt

Specifies the start frequency used for the zoom transform.

Syntax



XCHA15

Preset State: 0
Parameter Range: frequency = 0 through 100 GHz
Fundamental Unit: hertz

Note A more restrictive range may be imposed by the measurement setup.

Command Example

```
OUTPUT @Mta;"CHAN1:ZOOM:STAR 100MHZ"
```

Description

The ZOOM:START command specifies the start frequency used for the zoom transform. The query returns the current setting of ZOOM:START.

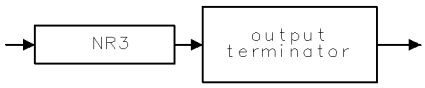
The START command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting a measurement independent of the specific signal frequency. The query always returns the value in hertz.

The module converts cycles to hertz by using the following equation:

$$hertz = cycles \times signal\ frequency$$

Note Currently this command modifies the same value as the FUNCx:ZOOM:STARt and SWEep:TIME:ZOOM:STARt commands. In future revisions this value may be independently settable for each channel and function.

Query Response



QGENNR3

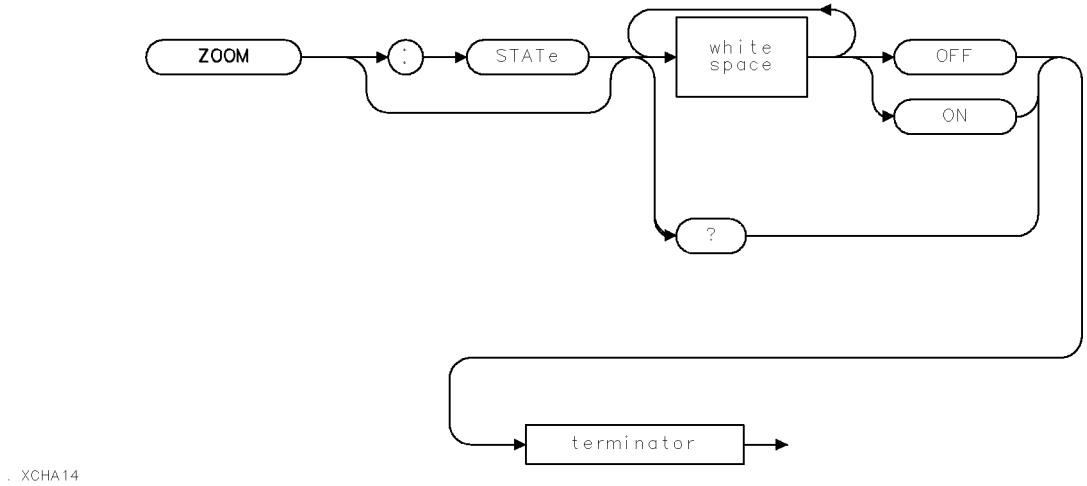
Query Example

```
OUTPUT @Mta;"CHAN1:ZOOM:STAR?"
ENTER @Mta;Start_freq
```

ZOOM:STATe

Determines whether data from the zoom transform is used.

Syntax



Preset State: OFF

Command Example

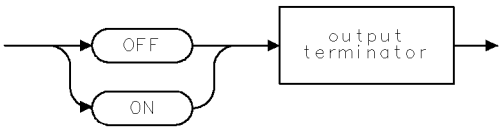
```
OUTPUT @Mta;"CHAN1:ZOOM ON"
```

Description

Note This command is valid only when the channel is formatted in the frequency domain.

The ZOOM:STATE command determines whether data from the zoom transform is used when a channel is formatted in the frequency domain. The query returns the current state of ZOOM:STATE.

Query Response



QGENON

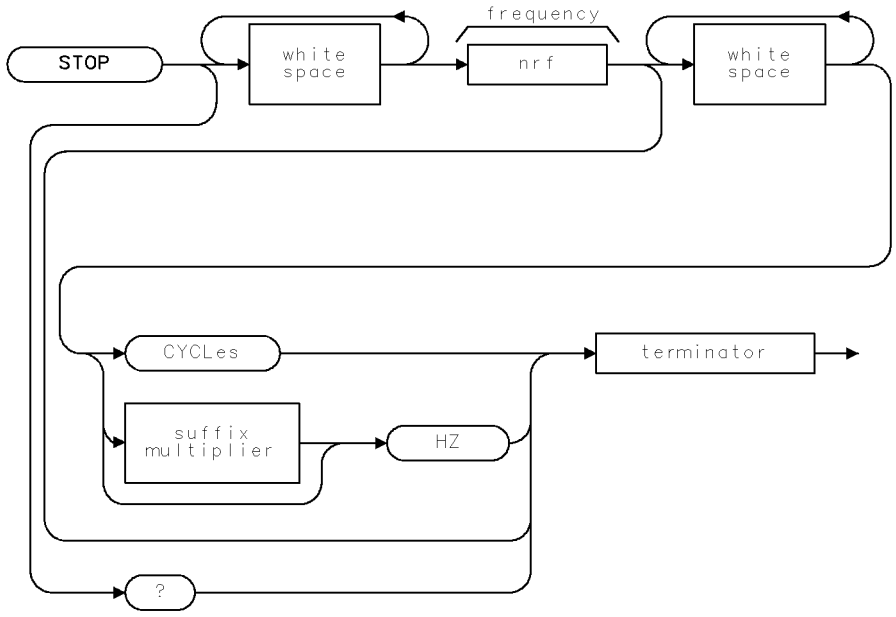
Query Example

```
OUTPUT @Mta;"CHAN1:ZOOM?"
ENTER @Mta;Zoom$
```

ZOOM:STOP

Specifies the stop frequency used for the zoom transform.

Syntax



XCHA16

Preset State: 100 MHz
Parameter Range: frequency = 0 through 100 GHz
Fundamental Unit: hertz

Note A more restrictive range may be imposed by the measurement setup.

Command Example

```
OUTPUT @Mta;"CHAN1:ZOOM:STOP 10GHZ"
```

Description

The ZOOM:STOP command specifies the stop frequency used for the zoom transform. The query returns the current setting.

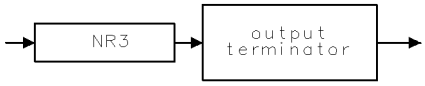
The STOP command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting a measurement independent of the specific signal frequency. The query always returns the value in hertz.

The module converts cycles to hertz by using the following equation:

$$hertz = cycles \times signal\ frequency$$

Note Currently this command modifies the same value as the FUNCx:ZOOM:STOP and SWEEP:TIME:ZOOM:STOP commands. In future revisions this value may be independently settable for each channel and function.

Query Response



QGENNR3

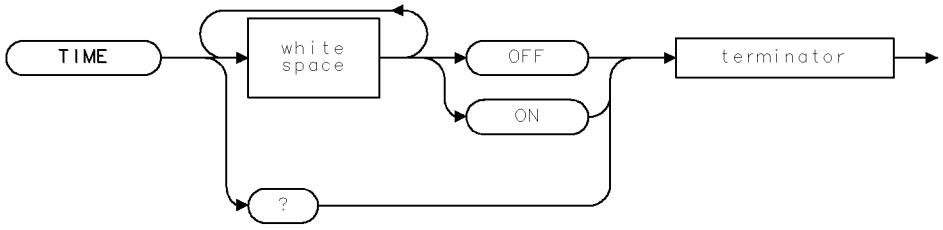
Query Example

```
OUTPUT @Mta;"CHAN1:ZOOM:STOP?"
ENTER @Mta;Stop_freq
```

ZOOM:TIME

Enables or disables the use of zoom transform data.

Syntax



Preset State: OFF

Command Example

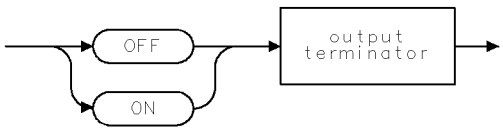
```
OUTPUT @Mta;"CHAN2:ZOOM:TIME OFF"
```

Description

The ZOOM:TIME command enables and disables the use of zoom transform data when the channel is in the time domain. The query returns the current state.

For example, if ZOOM:TIME is OFF and the channel is displayed in the time domain, the data will be the normal time domain data. If ZOOM:TIME is ON and the channel is displayed in the time domain, the data will be the IFFT of the zoom transform data. Note that this command does not enable the zoom transform, it just determines whether or not to use the zoom transform data once the transform is enabled (see CHAN:ZOOM:STATE).

Query Response



Query Example

```
OUTPUT @Mta;"CHAN1:ZOOM:TIME?"
ENTER @Mta;Zoom_state
```

9

Configure Subsystem

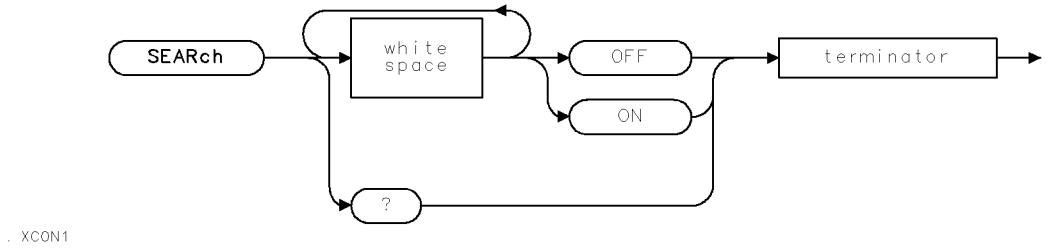
The CONFIGURE subsystem allows the user to specify how the microwave transition analyzer acquires other slave modules into the system. See the following list for a command syntax summary of the CONFIGURE subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:CONFigure		
:SEARch	OFF|ON	

SEARch

Specifies the method which the module acquires slave modules at power on.

Syntax



Preset State: not affected by *RST, IP, or power cycle

Command Example

```
OUTPUT @Mta;"CONF:SEAR ON"
```

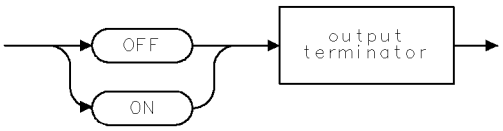
Description

The SEARCH command specifies the method by which the module will attempt to acquire slaves at power on. The query returns the current state of SEARCH.

SEARCH ON The module performs the standard slave search at power up. The standard slave search is a check for modules with a row number greater than and a column number greater than or equal to this module's row and column numbers. If a supported slave is found, it is acquired and used, otherwise no slave is acquired and the slave configuration defined when power was last turned off is discarded with one exception: if no supported slave is found in the slave search and the power-down slave configuration included a slave (or slaves) on HP-IB, the HP-IB slave configuration is retained.

SEARCH OFF The module retains the slave configuration defined when power was last turned off. The standard slave search is not performed. That is, a check is not done for modules with a row number greater than this module and column number greater than or equal to this module. See the HP-MSIB information in the system documentation for a full description of the slave search algorithm.

Query Response



QGENON

Query Example

```
OUTPUT @Mta;"CONF:SEAR?"
ENTER @Mta;Search$
```

10

DAC Subsystem

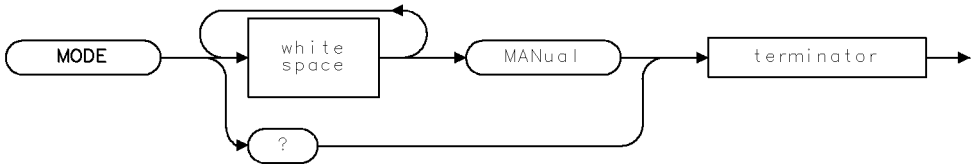
The DAC subsystem is used to control the 12-bit DAC output of the microwave transition analyzer. See the following list for a command syntax summary of the DAC subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:DAC		
:MODE	MANual	
:VALue	<nrf>	

MODE

Determines how the DAC is controlled.

Syntax



Preset State: MANual

Command Example

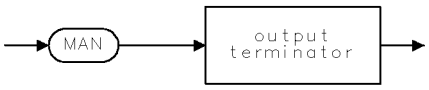
```
OUTPUT @Mta;"DAC:MODE MAN"
```

Description

The MODE command determines how the DAC is controlled. The query returns the current state of MODE.

Currently only MANUAL mode is available. Therefore this command has no effect on this revision of firmware. Future firmware revisions may allow other parameters.

Query Response



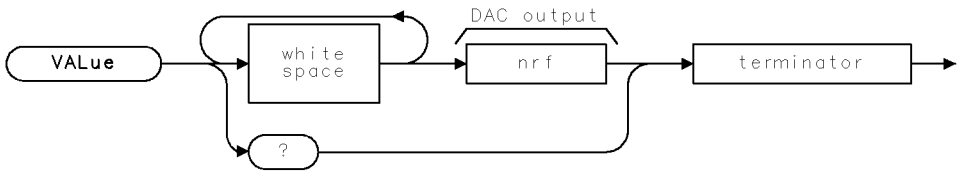
Query Example

```
OUTPUT @Mta;"DAC:MODE?"
ENTER @Mta;Dac$
```

VALue

Sets the 12-bit DAC output of the HP 70820A.

Syntax



Preset State: 0
Parameter Range: DAC output = 0 through 4095

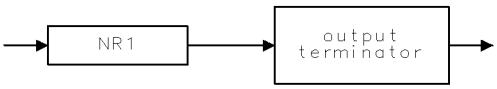
Command Example

```
OUTPUT @Mta;"DAC:VAL 4000"
```

Description

The VALUE command sets the 12-bit DAC output of the microwave transition analyzer. The query returns the current value of VALUE.

Query Response



Query Example

```
OUTPUT @Mta;"DAC:VAL?"
ENTER @Mta;Value
```

11

Diagnostics Subsystem

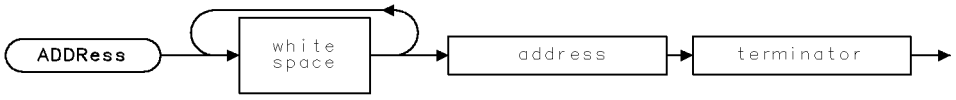
Use the DIAGNOSTICS subsystem commands to control configuration of Instrument BASIC. See the following list for a command syntax summary of the Diagnostics subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:DIAGnostics		
:PASSthru		
:ADDRess	<nrf>	
:STRing	"text"|'text'	
:TIMEout	<nrf> [[<suffix multiplier>]S]	

PASSthru:ADDRess

Sets the passthrough address.

Syntax



xdia1

Power on state: 0
Preset state: Unaffected by *RST, IP
Parameter range: address = 0 to 30

Command Example

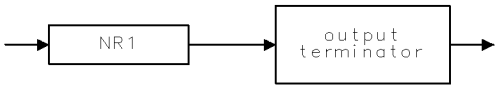
```
OUTPUT @Mta;"DIAG:PASS:ADDR 19"
```

Description

The PASSTHRU:ADDRESS command defines the HP-IB address used by the DIAGNOSTICS:PASSTHRU:STRING command.

The query returns the current value.

Query Response



QGENN1

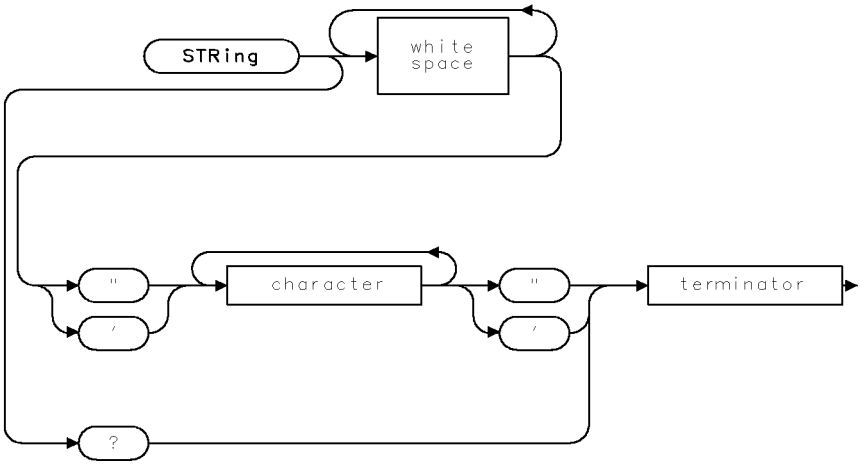
Query Example

```
OUTPUT @Mta;"DIAG:PASS:ADDR?"
ENTER @Mta;Address
```

PASSthru:STRing

Sends text to a device on the private HP-IB.

Syntax



Command Example

```
OUTPUT @Mta;"DIAG:PASS:ADDR 19"
OUTPUT @Mta;"DIAG:PASS:STR 'FREQ:CW 1.1 GHz'"
```

Description

The PASSTHRU:STRING command sends the specified text to a device on the private HP-IB.

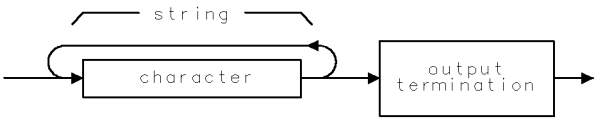
The query sends the specified text to a device on the private HP-IB, gets the response from the device and returns it.

There is a timeout, specified by DIAGNOSTIC:PASSTHRU:TIMEOUT, on both the input and output operations.

Because the HP 70820A considers the response a string, it will always have a linefeed at the end of the response; however, a check is done to prevent adding a linefeed if one already is at the end of the response from the device.

This command can be used even when the HP 70820A is using the HP-IB to control a synthesizer.

Query Response



qdia2

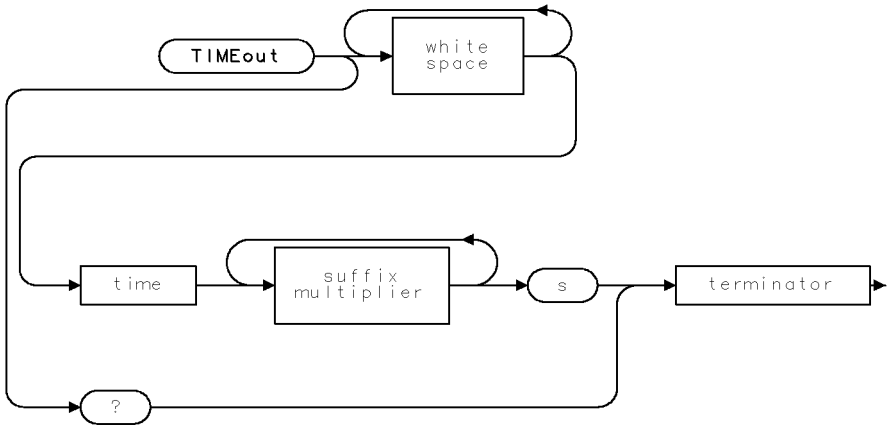
Query Example

```
OUTPUT @Mta;"DIAG:PASS:STR? '*IDN?'"
ENTER @Mta;Response$
OUTPUT @Mta;"DIAG:PASS? '*IDN?'"
ENTER @Mta;Response$
```

PASSthru:TIMEout

Sets the pass-through time out.

Syntax



Power on state: 5 seconds
Preset state: Unaffected by *RST, IP
Parameter range: time = 1 to 32,767 seconds

Command Example

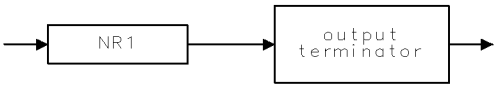
```
OUTPUT @Mta;"DIAG:PASS:TIME 60"
```

Description

The PASSTHRU:TIMEOUT command specifies the timeout for the DIAGNOSTICS:PASSTHRU:STRING.

The query returns the current value.

Query Response



Query Example

```
OUTPUT @Mta;"DIAG:PASS:TIME?"
ENTER @Mta;Timeout
```

12

Display Subsystem

The DISPLAY subsystem is used to control the display of data, text, graticule, and screen format. See the following list for a command syntax summary of the DISPLAY subsystem commands.

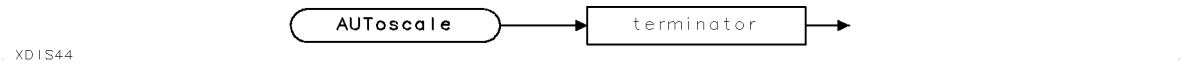
KEYWORD	PARAMETER FORM	NOTES
:DISPlay		
:CHAN<channel_number>|		
:WMEM<memory_ number>|		
:FUNC<function_number>		
:AUToscale		[no query]
:OFFSet	<nrf>[DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ]	
:POSition	BOTTom|TOP	
:RANGe	<nrf>DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ	
:REFerence	<nrf>[PCT][,<nrf>[PCT]]	
:TRACk	CONTinuous|COUPled|OFF	
:CONNect	OFF|ON	
:DATE		
:FORMat	EUROpe|US	
[:STATe]	OFF|ON	
:FORMat	1|2|AUTo	
:GRATicule	OFF| GRID| AXES| FRAMe	
:MENU		
:ABORt	OFF|ON	
:CHANged?	[WAIT]	[query only]
:CLEAr	<nrf>[,<nrf>]|ALL)|ALL	[no query]
:COPY	"{<character>}"|'{<character>}'	
:EXCLude	<nrf>	
:HARDkeys	OFF|ON	
:KEY	<nrf>[,<nrf>]	
:LABel	"{<character>}"|'{<character>}'	
:LIST	"{<character string>}[{,<character string>}]"| '{<character>}[{,character string>}]'	
:MAXimum	<nrf>[DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ|[<suffix multiplier>]S]	
:MINimum	<nrf>[DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ|[<suffix multiplier>]S]	

:PRESet	<nrf>[DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ|[<suffix multiplier>]S]|OFF|ON	
:PROMpt	"{<character>}"|'{<character>}'	
:STATe	"{<character>}"|'{<character>}'	
:STEP	<nrf>[DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ|[<suffix multiplier>]S]|LOGarithmic	
:TERMinator	DB|DBM|DEGrees|HERTz|NONE|PCT|SEConds|VOLTs	
:TEXT	"{<character>"|'{<character>}'	
:TYPE	ALPHa|EXCLude|KNOB|LEFT|LIST|NORMal|TOGGle	
:VALue	<nrf>[DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ|[<suffix multiplier>]S]	
:PERSist	OFF|INFinite|<nrf>	
:PROGram	AUTO|FULL|LOWer|OFF|UPPer	
:REGister		
[:LABel]	<nrf>("{<character>}"|'{<character>}')	
:STATus		
[:STATe]	OFF|ON	
:STRing	"{<character>}"|'{<character>}'	[no query]
:TEXT		
:CLEar		[no query]
:COLor	<nrf>	
:CSIZe ?		[query only]
[:DATA]	"{<character>}"|'{<character>}'	
:PAGE	<nrf>	
:STATe	OFF|ON	
:TIME		
[:STATe]	OFF|ON	
:TITLe		
:STATe	OFF|ON	
[:STRing]	"{<character>}"|'{<character>}'	

xxx:AUToscale

Sets the offset and range values for displaying the trace.

Syntax



Command Example

```
OUTPUT @Mta;"DISP:CHAN2:AUT"
```

Description

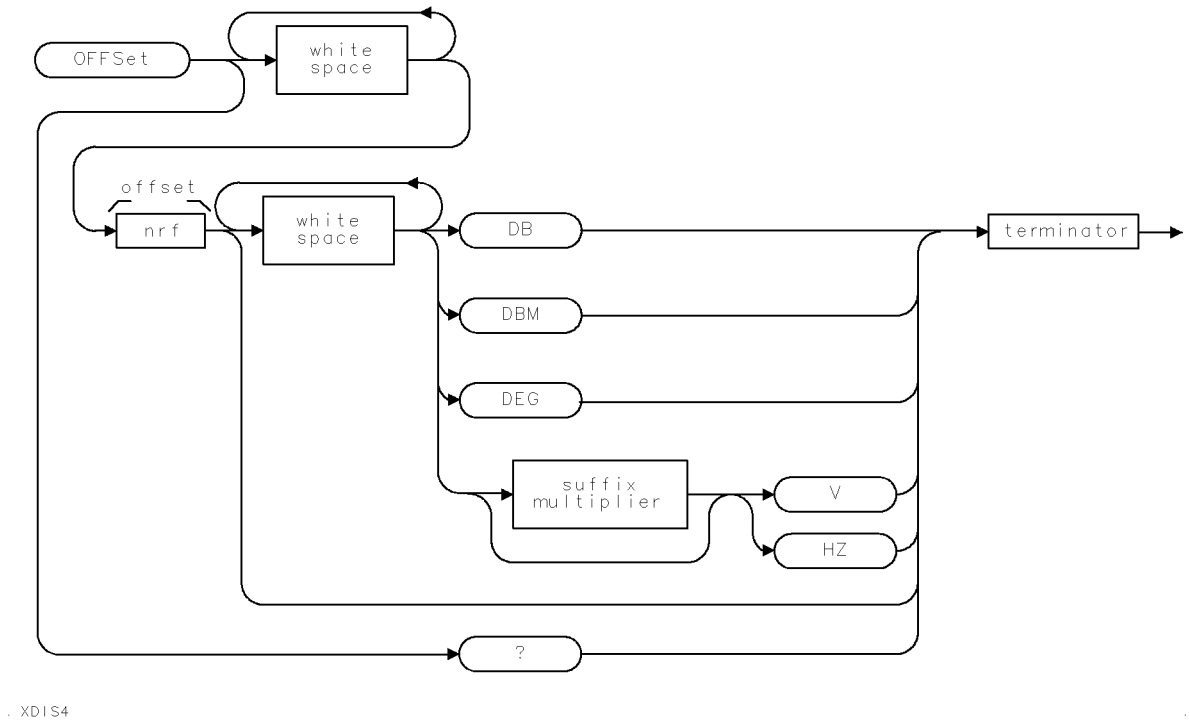
The AUTOSCALE command sets the DISPLAY:xxx:OFFSET and DISPLAY:xxx:RANGE values to reasonable values for displaying the trace.

Note The DISPLAY:xxx:AUTORANGE command sets the DISPLAY:TRACK to OFF.

xxx:OFFSet

Sets the offset of the displayed data.

Syntax



Preset State:	linear = 0
	logged = −10
Parameter Range:	linear: value = (−10000 × trace range) through
	max(10000 × trace range,1.0E20);
	logged: value = (ref level − 80) through (ref level + 80)

Command Example

```
OUTPUT @Mta;"DISP:CHAN2:OFFS 50 mV"
```

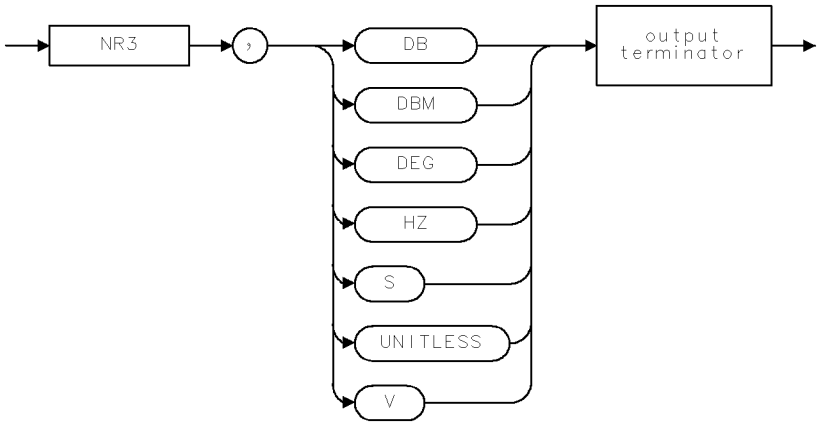
Description

The OFFSET command is used to set the offset of the displayed data. The query returns the current value of OFFSET.

Note The trace range is returned by the query WAVEFORM:VRANGE?.

Note The DISPLAY:xxx:OFFSET command sets the DISPLAY:TRACK to OFF.

Query Response



. QDIS4

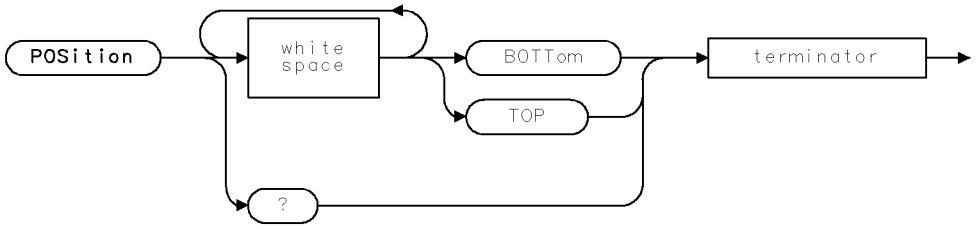
Query Example

```
OUTPUT @Mta;"DISP:CHAN1:OFFS?"
ENTER @Mta;Range,Unit$
```

xxx:POSition

Controls the placement of a trace when the screen format is set to 2.

Syntax



Preset States: TOP for "odd" traces (that is, CHAN1, FUNC1, and so on);
BOTTOM for "even" traces (that is, CHAN2, FUNC2 and so on)

Command Example

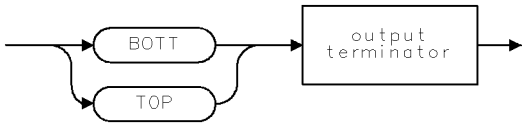
```
OUTPUT @Mta;"DISP:FUNC2:POS TOP"
```

Description

The POSITION command controls the placement of a trace when DISPLAY:FORMAT is 2 (or AUTO and 2 is currently selected). The query returns the current state of POSITION.

Note Traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



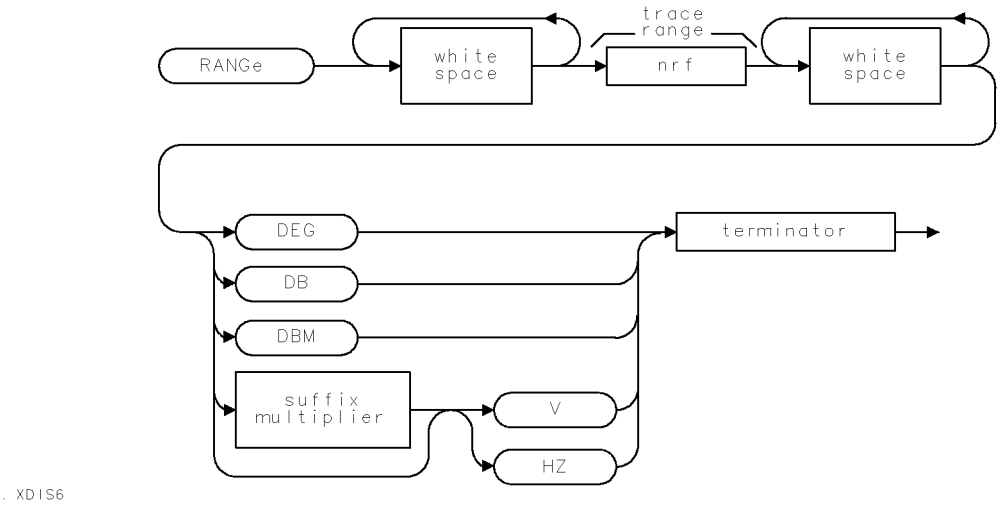
Query Example

```
OUTPUT @Mta;"DISP:WMEM1:POS?"
ENTER @Mta;Position$
```

xxx:RANGe

Sets the range of the displayed data.

Syntax



Preset States: 0.2 for linear data;
80 dB for log data;
360 degrees for phase data

Parameter Range: trace range = (1.0E−6 × trace range) through
max(1.0E4 × trace range,2.0E20)

Command Example

```
OUTPUT @Mta;"DISP:CHAN2:RANG 50 mV"
```

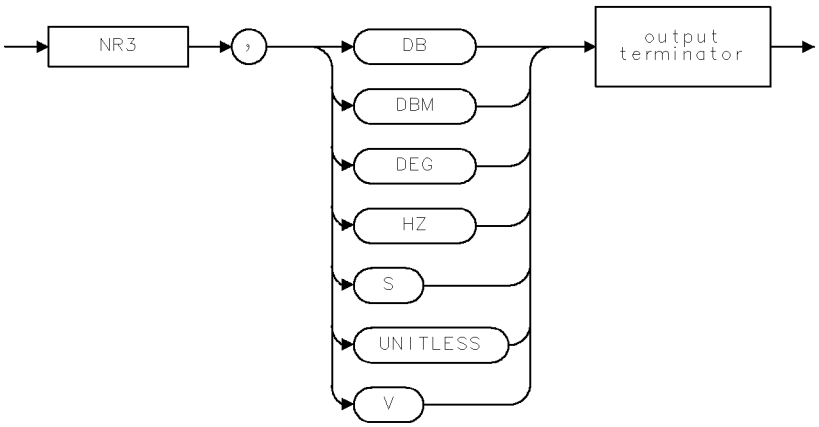
Description

The RANGE command is used to set the range of the displayed data. The query returns the current value of RANGE.

Note The trace range is returned by the query WAVEFORM:VRANGE?.

Note The DISPLAY:xxx:RANGE command sets DISPLAY:TRACK to OFF.

Query Response



. QDIS4

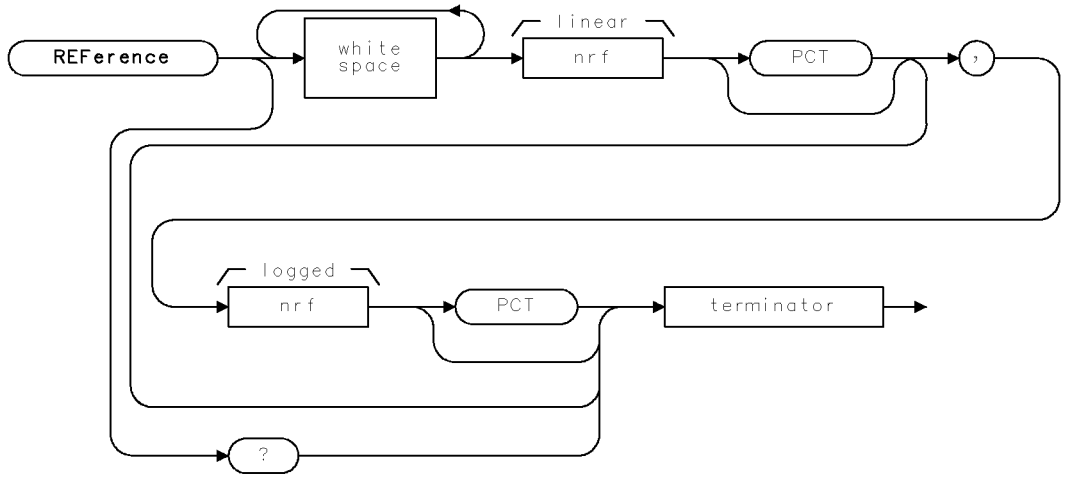
Query Example

```
OUTPUT @Mta;"DISP:CHAN1:RANG?"
ENTER @Mta;Range,Unit$
```

xxx:REFerence

Sets the reference position on the display.

Syntax



XDIS7

Preset States: 50%, 100%
Parameter Range: 0 to 100

Command Example

```
OUTPUT @Mta;"DISP:FUNC1:REF 75"
OUTPUT @Mta;"DISP:FUNC3:REF 25%,75%"
```

Description

The REFERENCE command sets the reference position on the display. This is the position that a value corresponding to DISPLAY:xxx:OFFSET will be plotted. The value of the reference position indicates a position relative to the graticule where 0% is the bottom and 100% is the top. The first parameter for REFERENCE sets the reference position for linear data, the second parameter sets the reference position for logged data. The query returns the current settings of the reference position.

Query Response

qgenn5

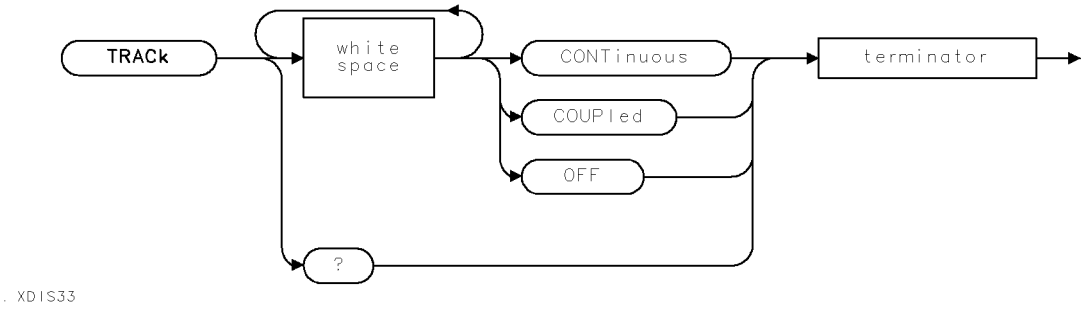
Query Example

```
OUTPUT @Mta;"DISP:CHAN1:REF?"
ENTER @Mta;Linear,Logged
```

xxx:TRACk

Specifies either the autoscaling or hardware-scale tracking mode.

Syntax



Preset State: OFF

Command Example

```
OUTPUT @Mta;"DISP:CHAN1:TRAC CONT"
```

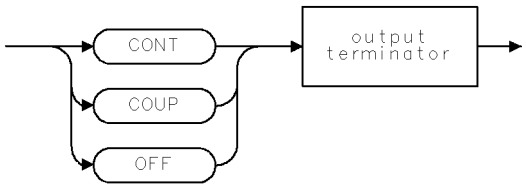
Description

The TRACK command specifies the mode in which autoscaling and hardware-scale tracking operates. The query returns the current setting of the TRACK mode.

The parameters for TRACK are:

CONTINUOUS	This parameter sets the continuous update mode. Traces are automatically scaled after each measurement.
COUPLED	This parameter sets the hardware-coupled mode. Changes to the hardware range setting are coupled to the scale setting.
OFF	This parameter turns off autoscale and tracking.

Query Response



. QDIS33 .

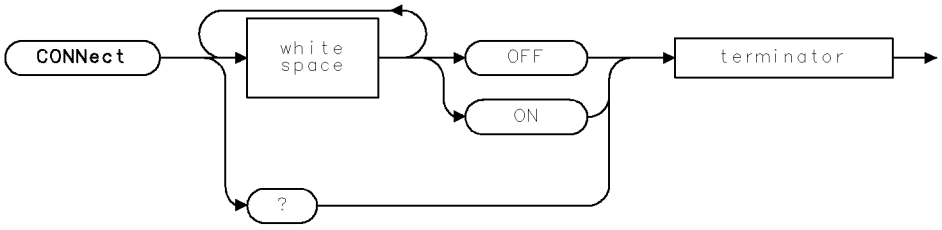
Query Example

```
OUTPUT @Mta;"DISP:CHAN1:TRAC?"
ENTER @Mta;Track_mode$
```

CONNect

Turns the connect-the-dots mode on or off.

Syntax



Preset State: ON

Command Example

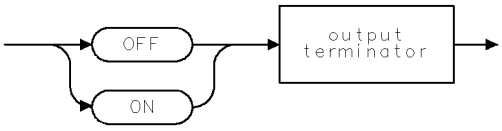
```
OUTPUT @Mta;"DISP:CONN OFF"
```

Description

The CONNECT command turns the connect-the-dots mode on or off for the input channels in the time domain (CHAN1 or CHAN2). When ON is selected, the waveform is displayed as a continuous solid trace. When OFF is selected, only the data points are drawn (for example, the display does not connect the dots). The query returns the current connect mode.

Note Display Dot Generator release 3.2 or later must be installed in the HP 70205 and HP 70206 displays instrument when using the CONNECT OFF command.

Query Response



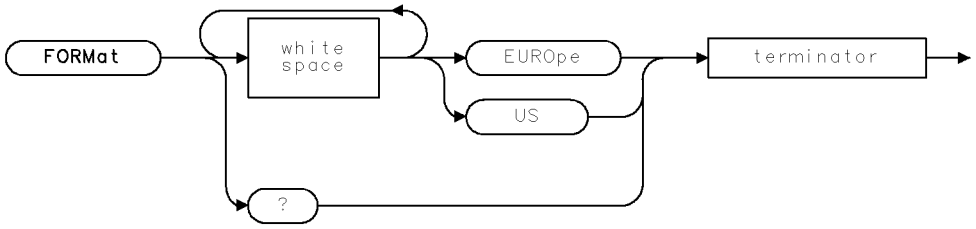
Query Example

```
OUTPUT @Mta;"DISP:CONN?"
ENTER @Mta;Connect$
```

DATE:FORMat

Controls the display format of the date.

Syntax



Preset State: not affected by power cycle, IP, or *RST

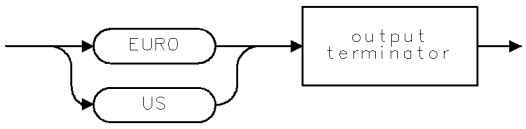
Command Example

```
OUTPUT @Mta;"DISP:DATE:FORM EURO"
```

Description

The DATE:FORMAT command controls the display format of the date. Allowable formats are European (dd.mm.yyyy) or United States (mmm dd, yyyy).

Query Response



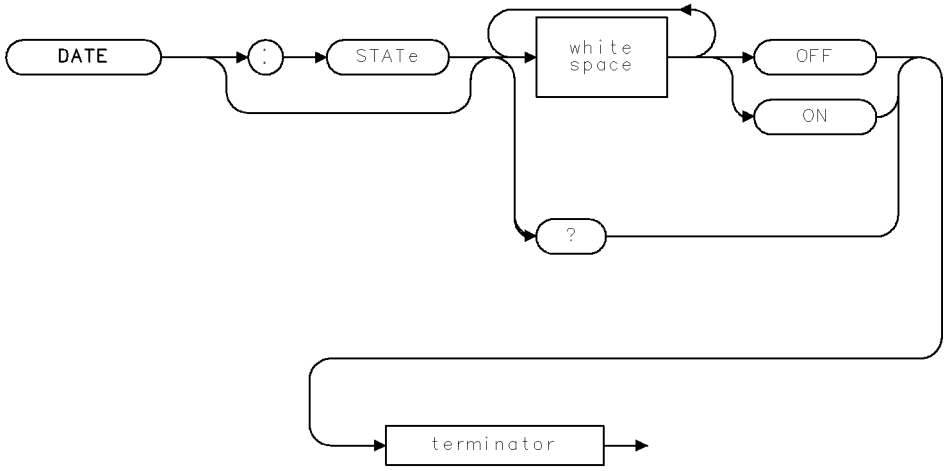
Query Example

```
OUTPUT @Mta;"DISP:DATE:FORM?"
ENTER @Mta;Date_format$
```

DATE:STATe

Turns the display of the date on or off.

Syntax



Preset State: not affected by IP, *RST, or power cycle

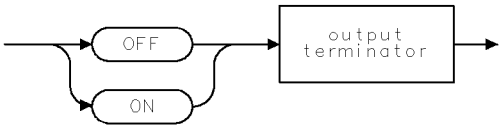
Command Example

```
OUTPUT @Mta;"DISP:DATE ON"
```

Description

The DATE:STATE command turns the display of the date on or off. The query returns the current state of DATE:STATE.

Query Response



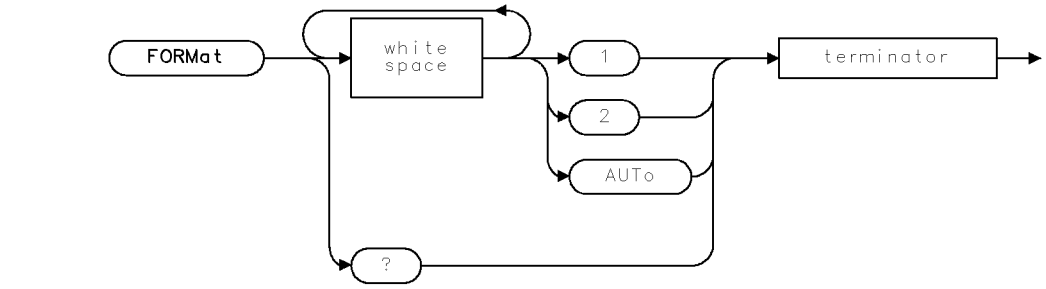
Query Example

```
OUTPUT @Mta;"DISP:DATE?"
ENTER @Mta;Date_state$
```

FORMat

Sets the screen format.

Syntax



Preset State: AUTo

Command Example

```
OUTPUT @Mta;"DISP:FORM 2"
```

Description

The FORMAT command sets the screen format. The query returns the current screen format. The parameter values are:

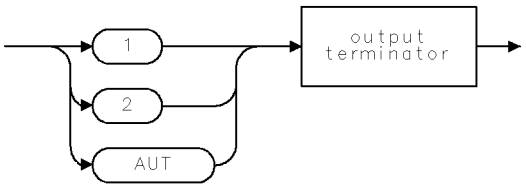
1 Each trace displayed uses the full graphics area where only one graticule, grid, or axis is displayed at a time.

2 Each trace displayed uses one-half of the graphics area where two graticules, grids, or axes are displayed at a time. The traces are positioned with the DISPLAY:POSITION command.

AUTo If there are one or more viewed traces in both the top and bottom positions then 2 is selected, otherwise 1 is selected.

The traces are only displayed if they have been previously turned on by the Top-Level subsystem VIEW command.

Query Response



. QDIS2

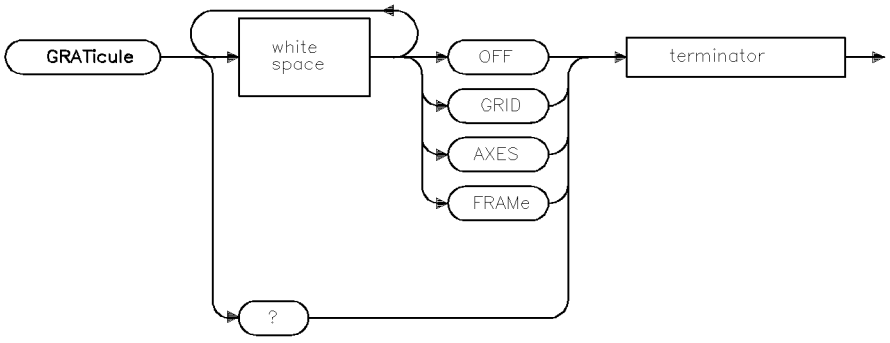
Query Example

```
OUTPUT @Mta;"DISP:FORM?"
ENTER @Mta;Format$
```

GRATicule

Sets the display graticule that formats the display screen.

Syntax



Preset State: GRID

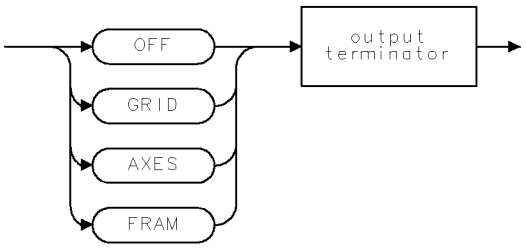
Command Example

```
OUTPUT @Mta;"DISP:GRAT GRID"
```

Description

The GRATICULE command sets the display graticule with one of three sets of vertical and horizontal lines that format the display screen. AXES superimposes one set of vertical and horizontal lines on the display screen. FRAME superimposes lines that border the edges of the display screen. GRID superimposes evenly-spaced vertical and horizontal lines on the display screen. The OFF parameter blanks all graticule format. The query returns the current display graticule format.

Query Response



. QDIS3

Query Example

```
OUTPUT @Mta;"DISP:GRAT?"
ENTER @Mta;Graticule$
```

DISPlay:MENU

The DISPLAY:MENU commands are used to provide the user with the ability to design a custom user interface under computer control (including HP Instrument BASIC). The basic idea is for the user to define the menu key labels and the type of function associated with each key (for example, a KNOB function such as trigger level or a TOGGLE function such as noise filter ON,OFF). The HP 70820A will handle the loading of menus and manipulation of values. The user program will be notified when a user-defined key has changed in value and the program can take the appropriate action.

Command Example

```
10 ASSIGN @Mta to 711
20 OUTPUT @Mta;"DISP:MENU:KEY 1;LABEL 'TEST ';TYPE TOGGLE"
30 LOOP
40   OUTPUT @Mta;"DISP:MENU:CHAN? WAIT"
50   ENTER @Mta;Menu_id,Key_id
60   IF Menu_id=0 AND Key_id=1 THEN
70     OUTPUT @Mta;"DISP:MENU:KEY 1;VALUE?"
80     ENTER @Mta;Value
90     IF Value=1 THEN
100      PRINT "TEST has been set to ON"
110    ELSE
120      PRINT "TEST has been set to OFF"
130    END IF
140  END IF
150 END LOOP
160 END
```

Line 20 defines key 1 as a TOGGLE key with a label “TEST”. Since it is a toggle key, ON/OFF will be appended. Line 40 asks for any changes in state. When one occurs, line 70 determines the new value and the program informs the computer operator.

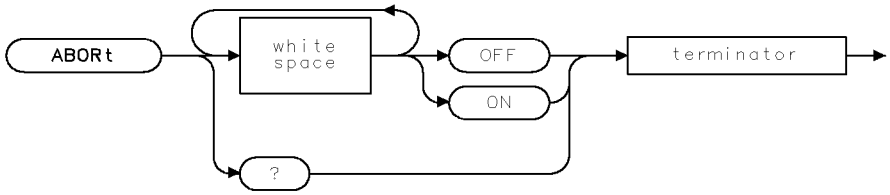
To get started in this subsystem, please consider the following commands first:

DISPlay:MENU:KEY
DISPlay:MENU:LABel
DISPlay:MENU:TYPE
DISPLay:MENU:CHANged
DISPlay:MENU:CLEAr

MENU:ABORt

Enables an abort menu and disables the normal processing of all keys.

Syntax



Preset State: OFF

Command Example

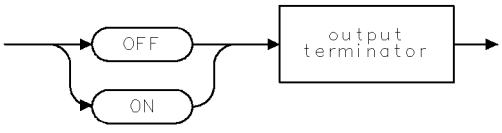
```
OUTPUT @Mta;"DISP:MENU:ABOR ON"
```

Description

The ABORT command enables an abort menu and disables the normal processing of all keys. The abort menu contains a single key labeled "ABORT". When ABORT is set to ON, the current menu is changed to USER (see DISPlay:MENU:STATe), the abort menu is displayed and key presses, other than the "ABORT" key, are ignored. Either pressing the "ABORT" key or setting ABORT to OFF will blank the abort menu and resume normal processing of the keys. The query returns the current state of ABORT.

The intent of this function is to provide the user with some indication that a non-negligible amount of time will elapse before the current operation is completed. If the program polls the state of ABORT, it will serve as a method to cancel the current operation.

Query Response



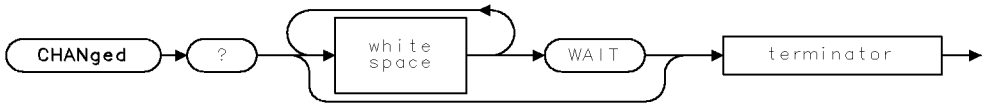
Query Example

```
OUTPUT @Mta;"DISP:MENU:ABOR?"
ENTER @Mta;Abort$
```

MENU:CHANged

Returns the key identifier of a menu key that has changed.

Syntax



Description

The CHANGED query returns the key identifier of a menu key that has changed in value. If there have been no changes: 0,0 is returned. If the WAIT parameter is sent, a value will not be returned until something has changed. The interpretation of the various values is:

−2,−2: IP (instrument preset) or recall state has occurred.

−1,−1: The USER hardkey has been pressed.

0,0: No key has changed value.

0,key: The key is one of the “top” level keys from either a flat menu structure or the left key of the menu tree system.

left,right: The key is the ’right’ menu key in the submenu associated with the LEFT key ’left’.

Query Response

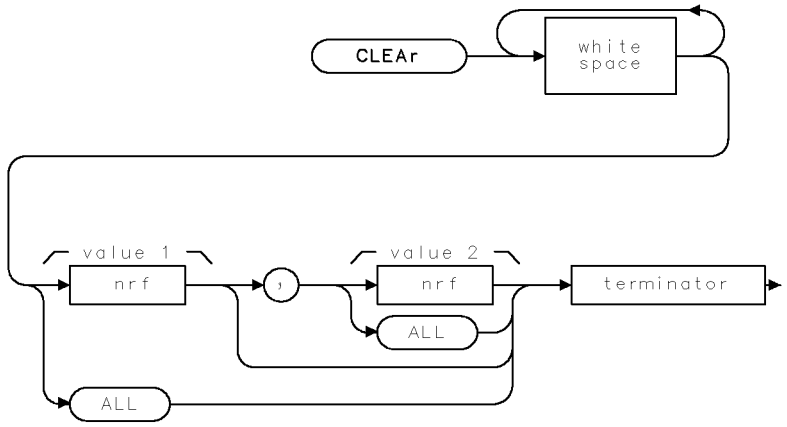
Query Example

```
OUTPUT @Mta;"DISP:MENU:CHAN? WAIT"
ENTER @Mta;Menu_id,Key_id
```

MENU:CLEAr

Undefines the specified key or keys.

Syntax



Parameter Range: value 1 = 1 through 14, if value 2 is omitted
value 1 = 8 through 14, if value 2 is present
1 ≤ value 2 ≤ 7

Command Example

```
OUTPUT @Mta;"DISP:MENU:CLEAR 9"
OUTPUT @Mta;"DISP:MENU:CLEAR 8,ALL"
OUTPUT @Mta;"DISP:MENU:CLEAR ALL"
```

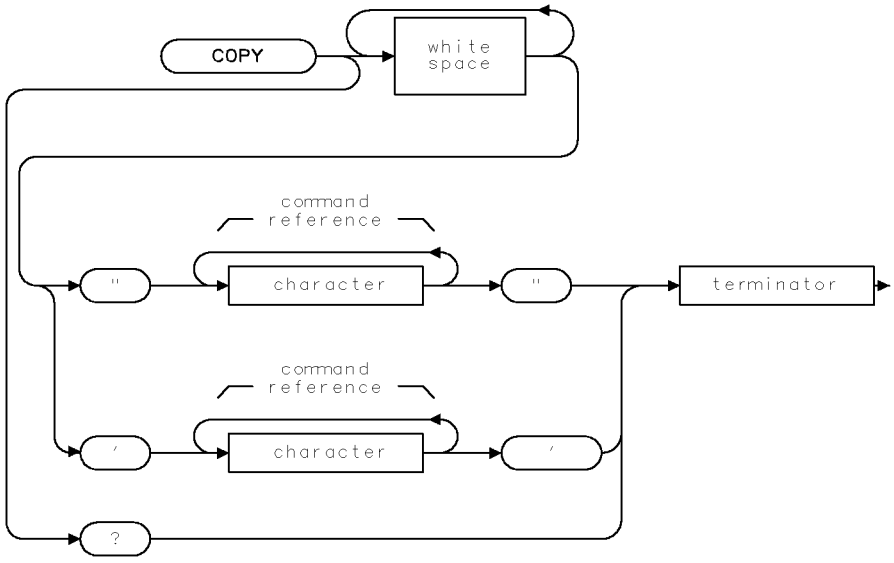
Description

The CLEAR command undefines the specified key or keys.

MENU:COPY

Duplicates a MENU softkey into a USER softkey.

Syntax



Command Example

```
! Copy [Main] [sweep:] to key 1
OUTPUT @Mta;"DISP:MENU:KEY 1"
OUTPUT @Mta;"DISP:MENU:COPY 'SWE:TYPE'"
! Copy [Config] [persist] to key 2
OUTPUT @Mta;"DISP:MENU:KEY 2"
OUTPUT @Mta;"DISP:MENU:COPY 'DISP:PERS'"
```

Description

The COPY command duplicates a softkey from the (MENU) menus into a softkey of the (USER) menu specified by the DISPLAY:MENU:KEY command. The source key is specified by text which, in general, matches the short form of the remote command, without parameters, that performs the same function as that key. For example, SWE:TYPE indicates the `sweep:` softkey in the `Main` menu. The query returns the current definition of a copied key. When using COPY, there are a few points to keep in mind:

- **Many menu keys are context sensitive.** For example, the `SCALE` softkey in the `Scale` menu adjusts the vertical scaling for the trace indicated by the `trace:` softkey. If the `SCALE` softkey is copied without copying the `trace:` softkey, the operator is very likely to become confused.
- **Key definitions may change based on the state of the instrument.** In most cases, the "entire" key is copied, including the variable definition. For example, the `SIGNAL FREQ`

softkey in the `Main` menu changes into `RF src: FRQ/PWR` when a synthesizer is configured. Although this may be confusing, it is intended to prevent problems such as adjusting the synthesizer frequency when there is no synthesizer configured.

- **Only end-function keys may be copied.** If you want to recreate an entire menu, you must copy each key individually.
- **Key functionality is copied, not menu functionality.** Operations that are dependent on being in a menu as opposed to the operation of any one key may not be duplicated. For example, when entering the `mass storage` menu, the screen is split and the catalog of the disk is displayed. This is dependent on being in the `mass storage` menu, not the operation of any key. The display of the catalog information will not occur in the `USER` menu regardless of which keys are copied from the `mass storage` menu.
- **"Copied"** keys 1 through 14 are nonvolatile. You can mix keys defined with COPY with keys defined with the other DISPLAY:MENU commands, however this arrangement will generally require a external computer program (or HP Instrument BASIC program) to manage the menu. Because of this, only the configurations that can be supported without the external computer are saved in nonvolatile memory.

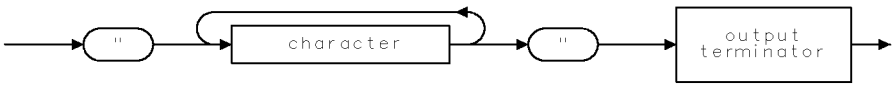
  If you want a user defined menu that will survive a power cycle, use only DISPLAY:MENU:KEY 1 to 14 and DISPLAY:MENU:COPY. Any other configuration will require the original program to restore the keys.

  Also, only "copied" keys 1 through 14 may be stored to a file. Again, this is due to the need for a program to manage any other configuration.
- **The key identifier may not be a valid remote command.** Whenever possible, the remote command syntax was used as the key identifier. Thus, by using the `preview` function in the `Config` menu (which displays the remote command for the operation performed), most of the keys can be identified without having to refer to the manual. The exceptions are:

  Keys that do not have remote commands (for example, the `SCROLL TABLE` softkey in the `Table` menu or the `CH1|CH2` softkey in the `Scale hardwre` menu).

  Cases where the same command is used for multiple keys and a parameter is required to distinguish them (for example, `CH1 RF|VID` and `CH2 RF|VID` softkeys in the `Main`, `sweep options` menu).

Query Response



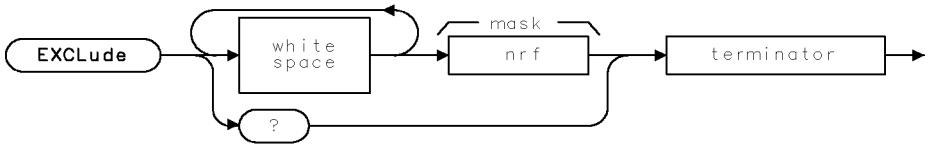
Query Example

```
OUTPUT @Mta;"DISP:MENU:KEY 1"
OUTPUT @Mta;"DISP:MENU:COPY?"
ENTER @Mta;Command$
```

MENU:EXCLude

Defines a set of keys that are a mutually excluded group.

Syntax



Parameter Range: mask = 0 through 32,766
Preset State: 0

Command Example

```
OUTPUT @Mta;"DISP:MENU:EXCL 10"
```

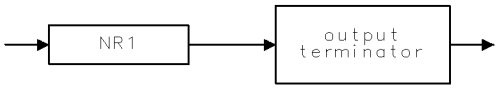
Description

The EXCLUDE command specifies the set of keys to be considered part of a mutually excluding group. The query returns the current value of EXCLUDE.

The bits of the parameter for the EXCLUDE command map to the keys. Bit 1 represents key 1, bit 2 represents key 2, and so on. For example, the decimal value 10 has bits 1 and 3 set which indicate keys 1 and 3.

When a group of keys are defined with DISPLAY:MENU:TYPE EXCLUDE and are part of a group of keys defined by the EXCLUDE command, whenever one of these keys is activated, all of the other keys in the group will be deactivated. An activated key will be displayed with an underline and the response to DISPLAY:MENU:VALUE is 1. A deactivated key will be displayed without an underline and the response to DISPLAY:MENU:VALUE is 0.

Query Response



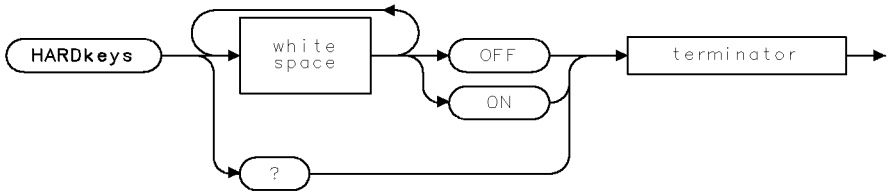
Query Example

```
OUTPUT @Mta;"DISP:MENU:EXCL?"
ENTER @Mta;Mask
```

MENU:HARDkeys

Enables or disables the hardkey panel.

Syntax



Preset State: OFF

Command Example

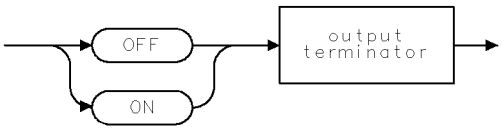
```
OUTPUT @Mta;"DISP:MENU:HARD ON"
```

Description

The HARDKEYS command enables or disables an intercept of the hardkey panel key presses. When ON, the hardkey presses will be reported to the DISPLAY:MENU system (see DISPLAY:MENU:CHANGED). When OFF, the hardkey presses will be processed normally (that is, according to the factory-defined state).

The query returns the current state of HARDKEYS.

Query Response



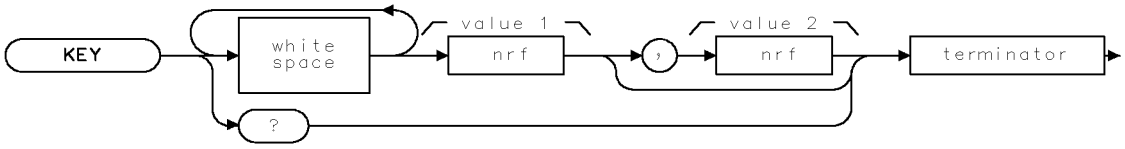
Query Example

```
OUTPUT @Mta;"DISP:MENU:HARD?"
ENTER @Mta;Abort$
```

MENU:KEY

Selects a menu to manipulate.

Syntax



Preset State: 0, 0
Parameter Range: value 1 = 1 through 14, if value 2 is omitted
value 1 = 8 through 14, if value 2 is present
value 2 = 1 through 7

Command Example

```
OUTPUT @Mta;"DISP:MENU:KEY 1;"
```

Description

The KEY command selects a menu key to manipulate. The query returns the currently selected key.

There are two formats for this command. When there is a single numeric parameter, a flat menu of 14 keys is assumed. The query for this format will return 0,keynum. Note that the left-hand menu keys can only be accessed with this format.

When there are two numeric parameters, a menu tree with seven branches (corresponding to the seven, left menu keys) is assumed. Each branch of the menu tree supports seven keys (or leaves). This tree structure is very similar to the menu structure of the factory-defined menus.

Query Response

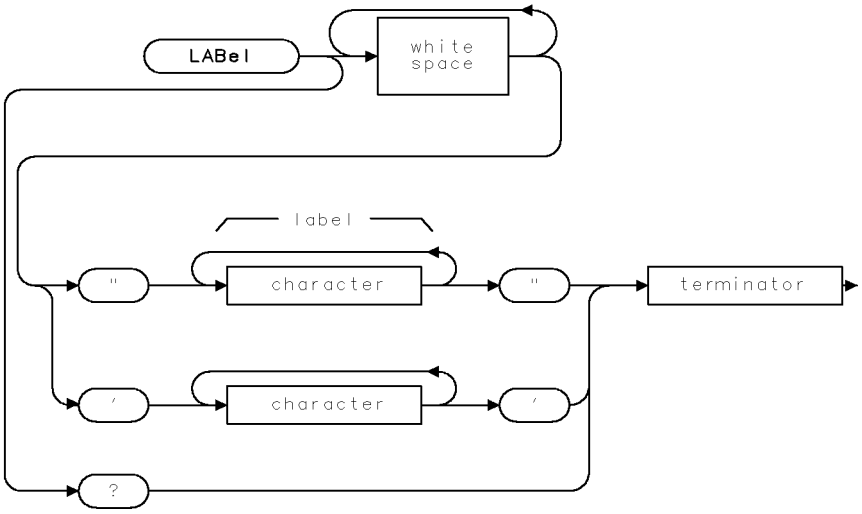
Query Example

```
OUTPUT @Mta;"DISP:MENU:KEY?"
ENTER @Mta;Left,Right
```

MENU:LABel

Defines a menu key label.

Syntax



Preset State: ""

Command Example

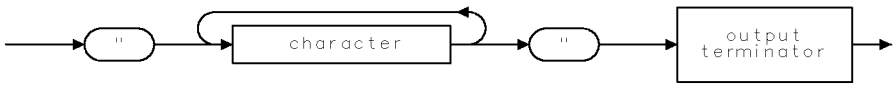
```
OUTPUT @Mta;"DISP:MENU:KEY 1;LAB 'TEST'"
```

Description

The LABEL command defines the menu key label for the currently selected key. The query returns the current label of the key.

Each menu key consists of 2 lines of 7 characters. For certain keys types additional text will be appended to the end of the label before it is displayed. For example, if a key is labeled "TEST " and the key type is TOGGLE then ON/OFF will be appended to make the displayed key: "TEST ON/OFF".

Query Response



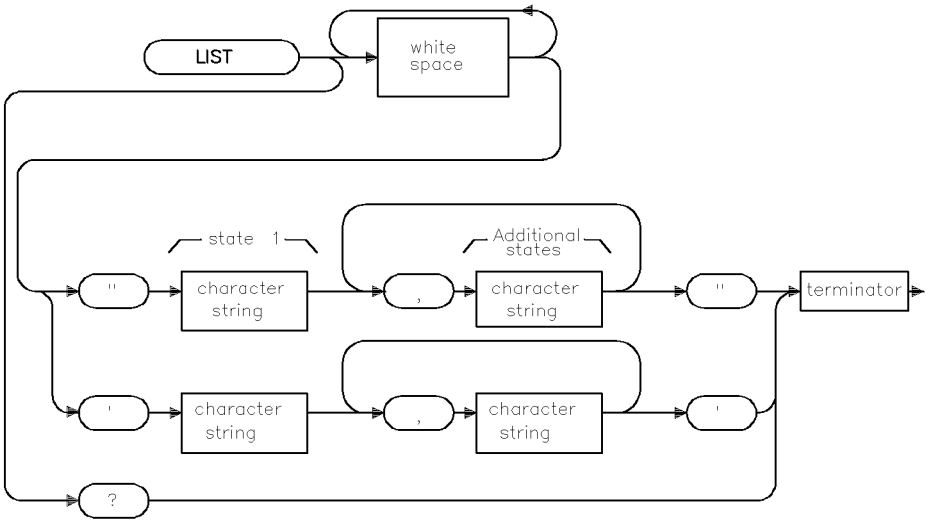
Query Example

```
OUTPUT @Mta;"DISP:MENU:KEY 1;LAB?"
ENTER @Mta;Label$
```

MENU:LIST

Defines the key label for each of the possible values of a key.

Syntax



Preset State: ""

Command Example

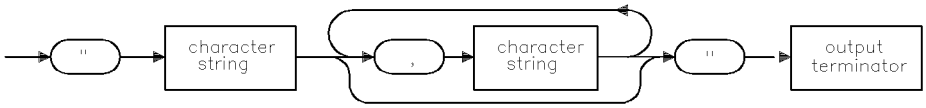
```
OUTPUT @Mta;"DISP:MENU:LIST 'ONE,TWO,THREE'"
```

Description

The LIST command, defines the key label for each of the possible values of a key, defined with DISPLAY:MENU:TYPE LIST. The query returns the current state of list.

The parameter for LIST is a comma separated list of text that will be appended to the key label. A LIST key can have 1 of N values. The value returned by DISPLAY:MENU:VALUE will be 0 through N-1. The key label displayed will be a concatenation of the label defined with DISPLAY:MENU: LABEL and the appropriate entry from the LIST command.

Query Response



qdis56

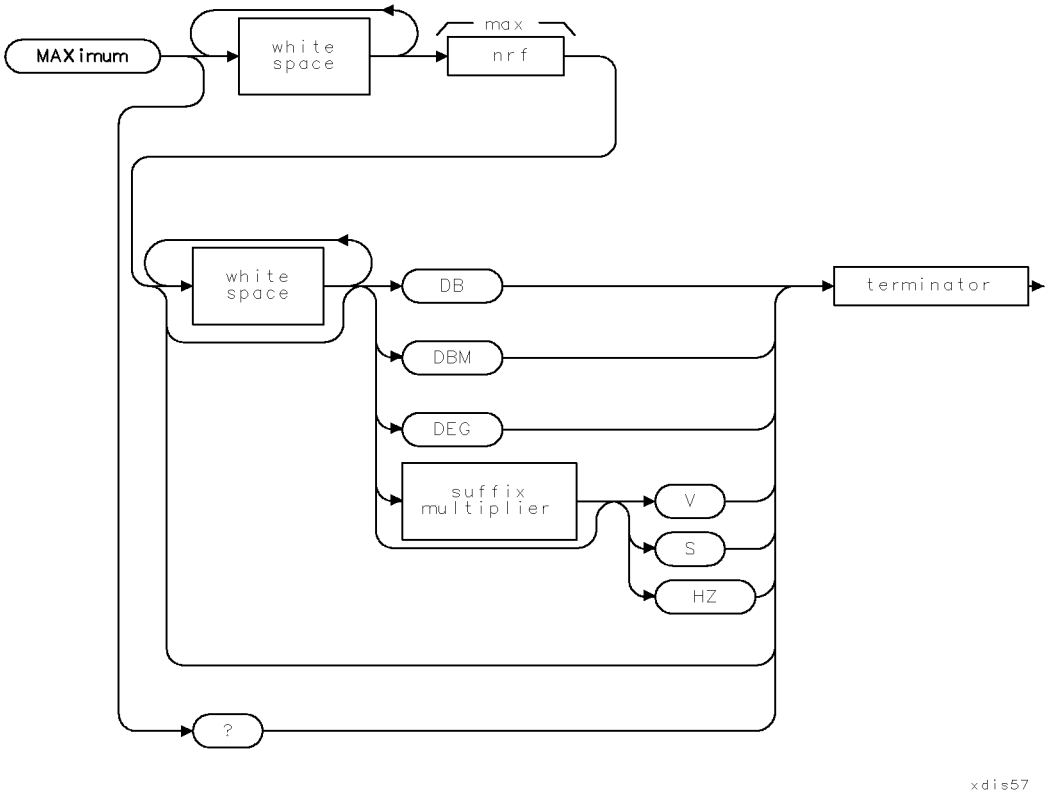
Query Example

```
OUTPUT @Mta;"DISP:MENU:LIST?"
ENTER @Mta;List$
```

MENU:MAXimum

Sets the maximum allowable value.

Syntax



Parameter Range: no range checking is performed
Fundamental Unit: seconds, volts, hertz, degrees, dB, or dBm

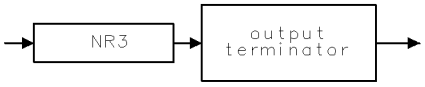
Command Example

```
OUTPUT @Mta;"DISP:MENU:MAX 1GHZ"
```

Description

The MAXIMUM command defines the maximum value for the active parameter of the currently selected key. The query returns the current value of MAXIMUM.

Query Response



QGENNR3

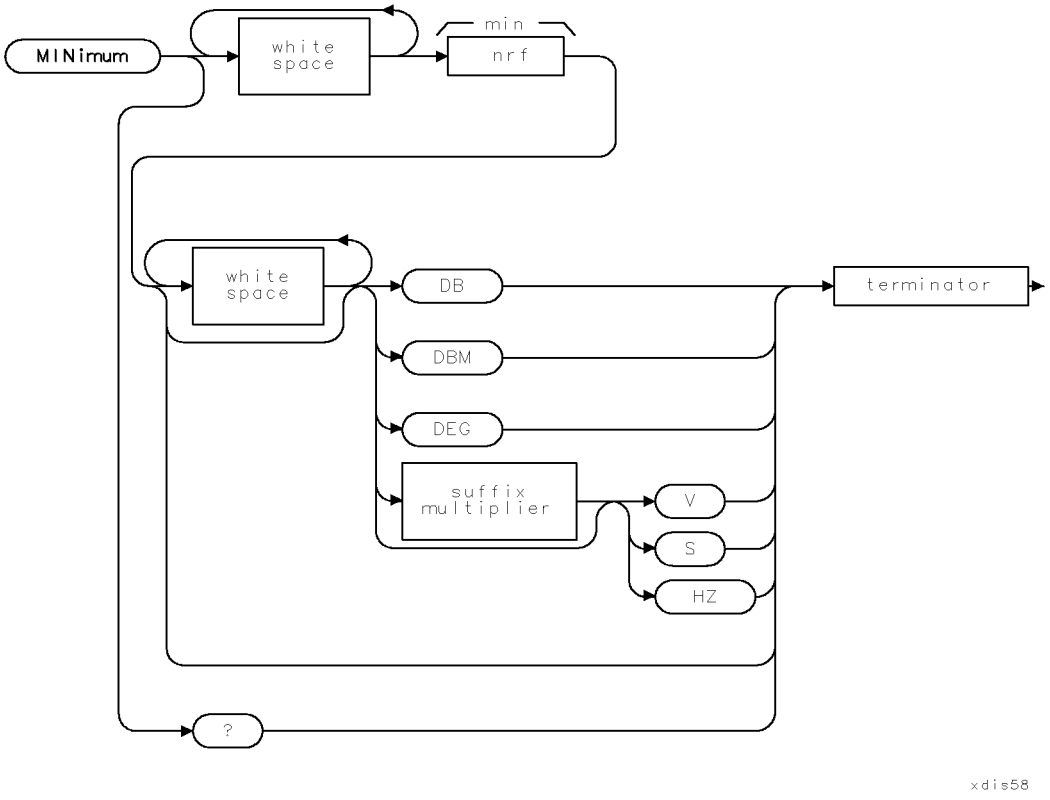
Query Example

```
OUTPUT @Mta;"DISP:MENU:MAX?"
ENTER @Mta;Maximum
```

MENU:MINimum

Sets the minimum allowable value.

Syntax



Parameter Range: no range checking is performed
Fundamental Units: seconds, volts, hertz, degrees, dB, or dBm

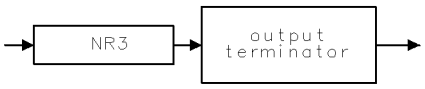
Command Example

```
OUTPUT @Mta;"DISP:MENU:MIN 1MHZ"
```

Description

The MINIMUM command defines the minimum value for the active parameter of the currently selected key. The query returns the current value of MINIMUM.

Query Response



QGENNR3

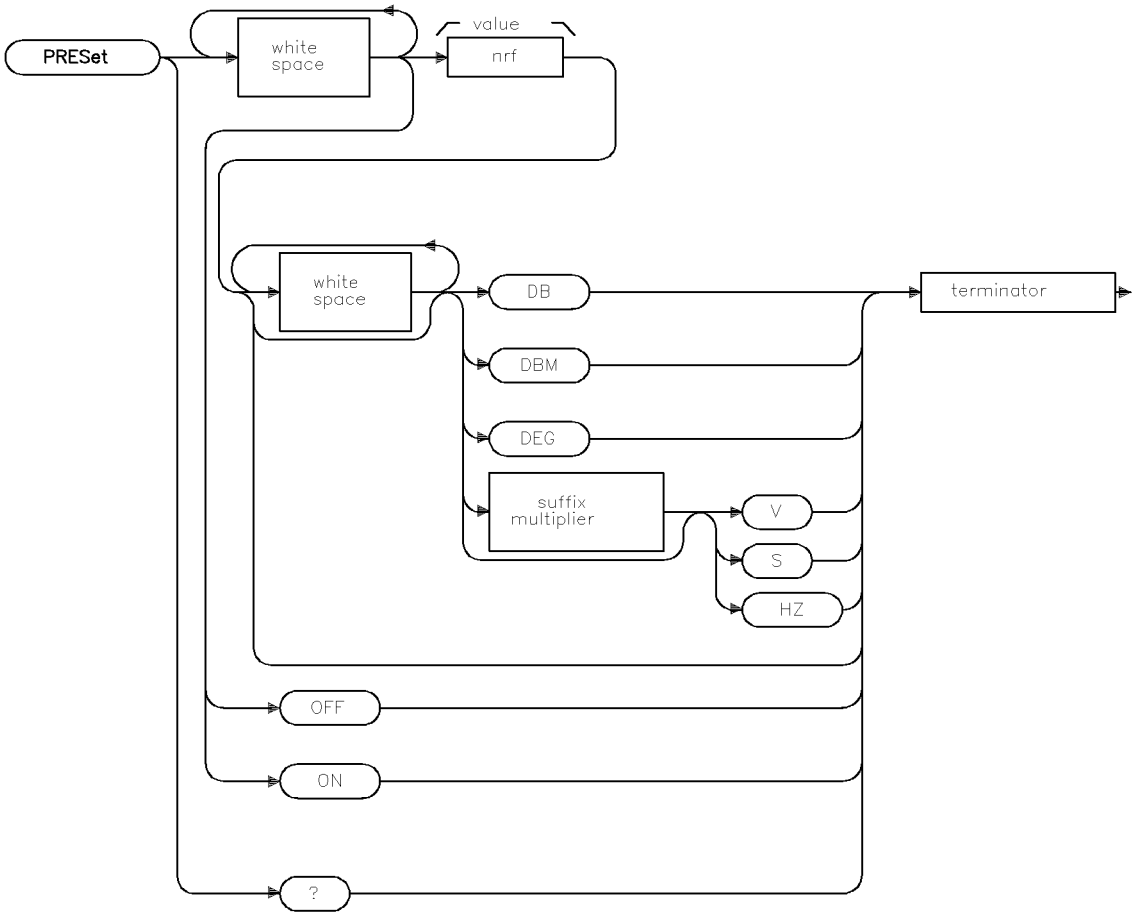
Query Example

```
OUTPUT @Mta;"DISP:MENU:MIN?"
ENTER @Mta;Minimum
```

MENU:PRESet

Defines the preset state of the currently selected key.

Syntax



Parameter Range: no checking is done on parameter range
Fundamental Unit: seconds, volts, hertz, degrees, dB, or dBm

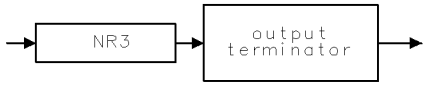
Command Example

```
OUTPUT @Mta;"DISP:MENU:PRES 1 ns"
```

Description

The PRESET command defines the instrument preset state of the currently selected key. The query returns the current value of PRESET.

Query Response



QGENNR3

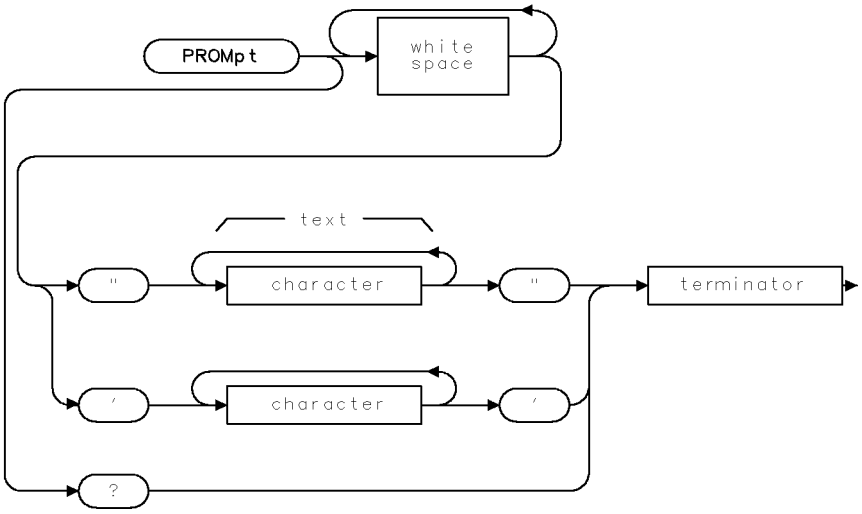
Query Example

```
OUTPUT @Mta;"DISP:MENU:PRES?"
ENTER @Mta;Preset
```

MENU:PROMpt

Defines the textual prompt for the currently selected key.

Syntax



xdis60

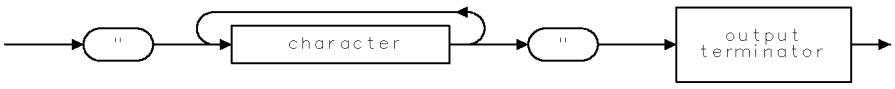
Command Example

```
OUTPUT @Mta;"DISP:MENU:PROM 'center freq:'"
```

Description

The PROMPT command defines the textual prompt for the currently selected key. The prompt, used for keys with DISPLAY:MENU:TYPE set to KNOB, is placed in the active parameter area with the key is active. The query returns the current value of PROMPT.

Query Response



qgenn9

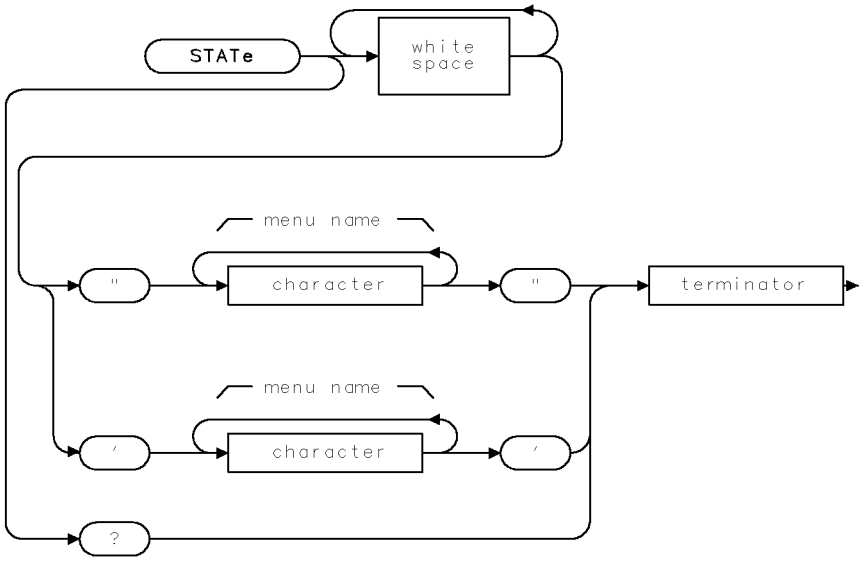
Query Example

```
OUTPUT @Mta;"DISP:MENU:PROM?"
ENTER @Mta;Prompt$
```

MENU:STATe

Selects the menu position.

Syntax



Command Example

```
OUTPUT @Mta;"DISP:MENU:STAT 'USER'"
```

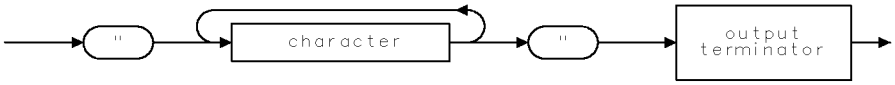
Description

The STATE command allows you to select the menu position corresponding to pressing the (USER) or (MENU) hardkeys. The accepted values are as follows:

- **MENU** activates the menus accessed via the (MENU) key. The "page x of 2" setting is unaffected.
- **MENU1** activates the menus accessed via the (MENU) key and sets the "page x of 2" key is set to "page 1 of 2".
- **MENU2** activates the menus accessed via the (MENU) key and sets the "page x of 2" key is set to "page 2 of 2".
- **USER** activates the user defined menu. Note that if there are no keys defined, all the keys will be blank.
- **IBASIC** activates the HP Instrument BASIC menus if the HP Instrument BASIC option is installed.

The query returns the current position of the menus.

Query Response



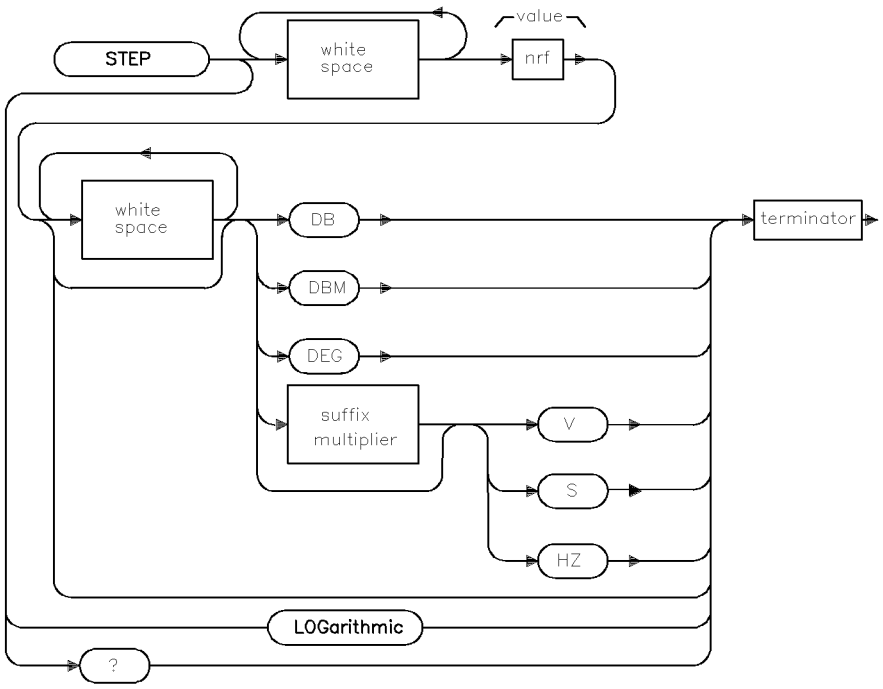
Query Example

```
OUTPUT @Mta;"DISP:MENU:STAT?"
ENTER @Mta;Menu_name$
```

MENU:STEP

Defines the step size of the step keys and knob for the currently selected key.

Syntax



Preset State: 0
Parameter Range: no checking is done on parameter range
Fundamental Unit: seconds, volts, hertz, degrees, dB, or dBm

Command Example

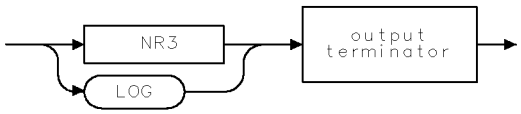
```
OUTPUT @Mta;"DISP:MENU:STEP 1 ns"
```

Description

The STEP command defines the step size of the up and down arrow keys and the resolution of a single knob count for the currently selected key. The query returns the current value of STEP.

The up arrow will cause the active parameter of a key defined as DISPLAY:MENU:TYPE KNOB to increase in value by STEP. The down arrow will cause the active parameter to decrease in value by STEP. The resolution of the knob is ten percent of STEP. If the parameter is LOGARITHMIC instead of a numeric value, the up and down arrow keys will adjust the active parameter in a 1,2,5 sequence and the knob will adjust the active parameter in approximately five percent steps.

Query Response



qdis62

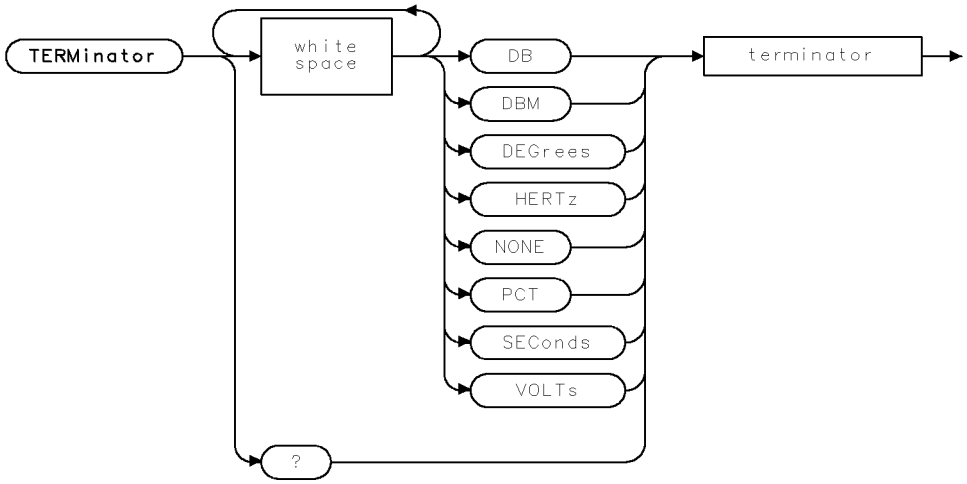
Query Example

```
OUTPUT @Mta;"DISP:MENU:STEP?"
ENTER @Mta;Step$
IF Step$="LOG" THEN
PRINT "LOG"
ELSE
PRINT Val(Step$)
END IF
```

MENU:TERMinator

Defines the termination menu for numeric entry keys.

Syntax



Command Example

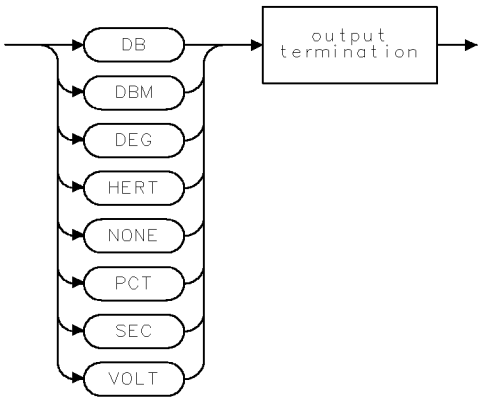
```
OUTPUT @Mta;"DISP:MENU:TERM HERTZ"
```

Description

The TERMINATOR command defines the termination menu used for numeric entry for the currently selected key. The termination menu is available for key defined with DISPLAY:MENU:TYPE set to KNOB.

The query returns the current value of TERMINATOR.

Query Response



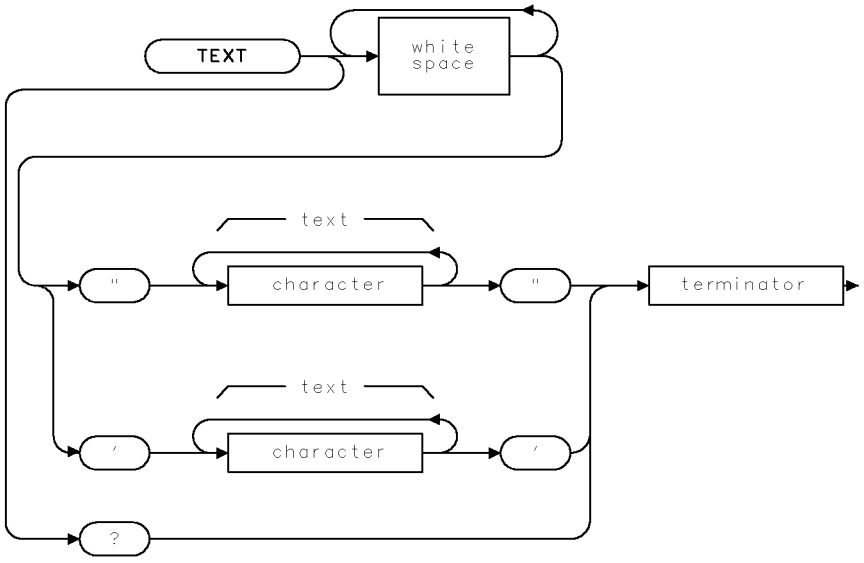
Query Example

```
OUTPUT @Mta;"DISP:MENU:TERM?"
ENTER @Mta;Terminator$
```

MENU:TEXT

Sets the value of the text variable of the currently selected key.

Syntax



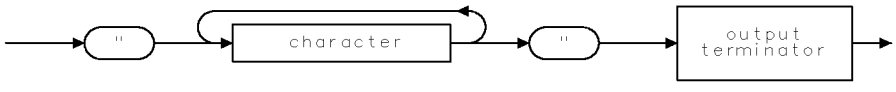
Command Example

```
OUTPUT @Mta;"DISP:MENU:TEXT 'filename'"
```

Description

The TEXT command sets the value of the text variable of the currently selected key. The text variable is used for keys defined with DISPLAY:MENU:TYPE as ALPHA. The query returns the current value of the text variable.

Query Response



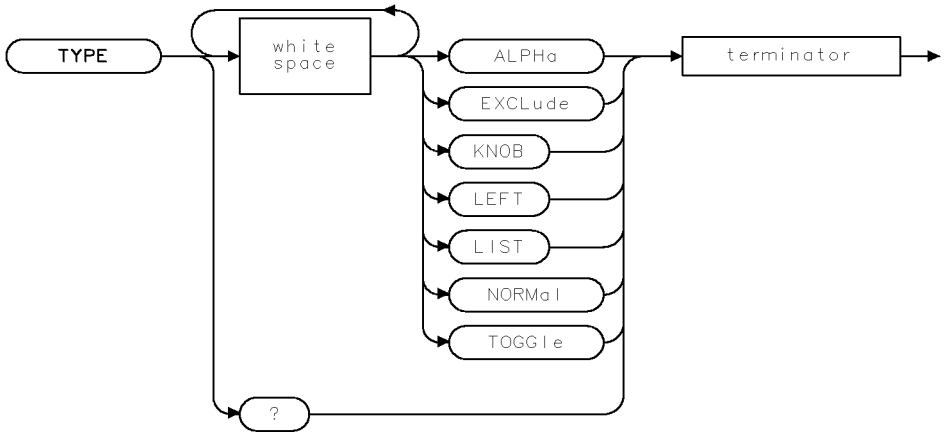
Query Example

```
OUTPUT @Mta;"DISP:MENU:TEXT?"
ENTER @Mta;Text$
```

MENU:TYPE

Defines the key type.

Syntax



Preset State: NORMal

Command Example

```
OUTPUT @Mta;"DISP:MENU:KEY 1;TYPE TOGGLE"
```

Description

The TYPE command defines the key type for the currently selected key. The query returns the current value of TYPE. The following lists the various key types.

ALPHA — The ALPHA key manipulates an alpha-numeric parameter. When the user activates an ALPHA key, a "dymo-label" menu will be enabled to allow alpha-numeric entry. Entry is also supported via a keyboard connected to the display module. Refer to DISPLAY:MENU:PRESET and DISPLAY:MENU:TEXT.

COPY — The COPY key is a duplication of a key that exists in the normal HP 70820A menus. Note that the TYPE command cannot be used to create a COPY key (for example, DISPLAY:MENU:TYPE COPY is not valid). Refer to DISPLAY:MENU:COPY.

EXCLUDE — The EXCLUDE key works in conjunction with other EXCLUDE keys to provide a 1 of N selection. When an EXCLUDE key is selected, it will be underlined and have a value of 1. All other EXCLUDE keys in that group (as defined by the EXCLUDE mask) will be set to 0. Refer to DISPLAY:MENU:EXCLUDE, DISPLAY:MENU:PRESET, and DISPLAY:MENU:VALUE.

KNOB The KNOB key manipulates a numeric value via either the front panel knob or numeric entry. Refer to DISPLAY:MENU:MAXIMUM, DISPLAY:MENU:MINIMUM, DISPLAY:MENU:PRESET, DISPLAY:MENU:PROMPT, DISPLAY:MENU:STEP, DISPLAY:MENU:TERMINATOR, and DISPLAY:MENU:VALUE.

LEFT The LEFT key selects a right hand menu associated with that key. Only 1 LEFT key is active at a time. Each sub-menu may have up to seven keys defined. Refer to DISPLAY:MENU:PRESET and DISPLAY:MENU:VALUE.

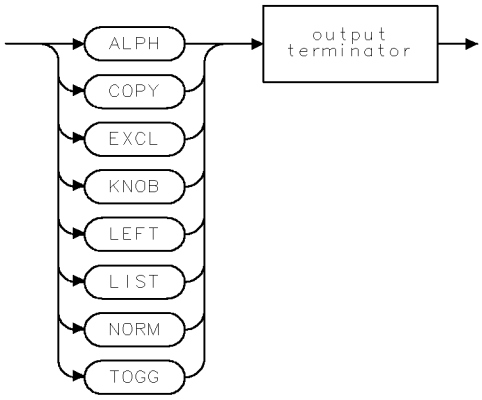
LIST The LIST key provides for a 1 of N selection on a single key. The user provides a list of state labels that the key can assume where a value of 0 corresponds to the first state label, 1 to the 2nd, and so on. The label for the current state will be concatenated to the key label defined with LABEL before being displayed. Refer to DISPLAY:MENU:LIST, DISPLAY:MENU:PRESET, and DISPLAY:MENU:VALUE.

NORMAL The NORMAL key informs the user when it is pressed.

TOGGLE The TOGGLE key is a two-state key. An ON and OFF indication will be concatenated to the key label defined with LABEL before being displayed. The values of 0 and 1 correspond to OFF and ON respectively. Refer to DISPLAY:MENU:PRESET and DISPLAY:MENU:VALUE.

For all key types, with the exception of NORMAL and COPY, the user is notified only of changes in value via the DISPLAY:MENU:CHANGED query.

Query Response



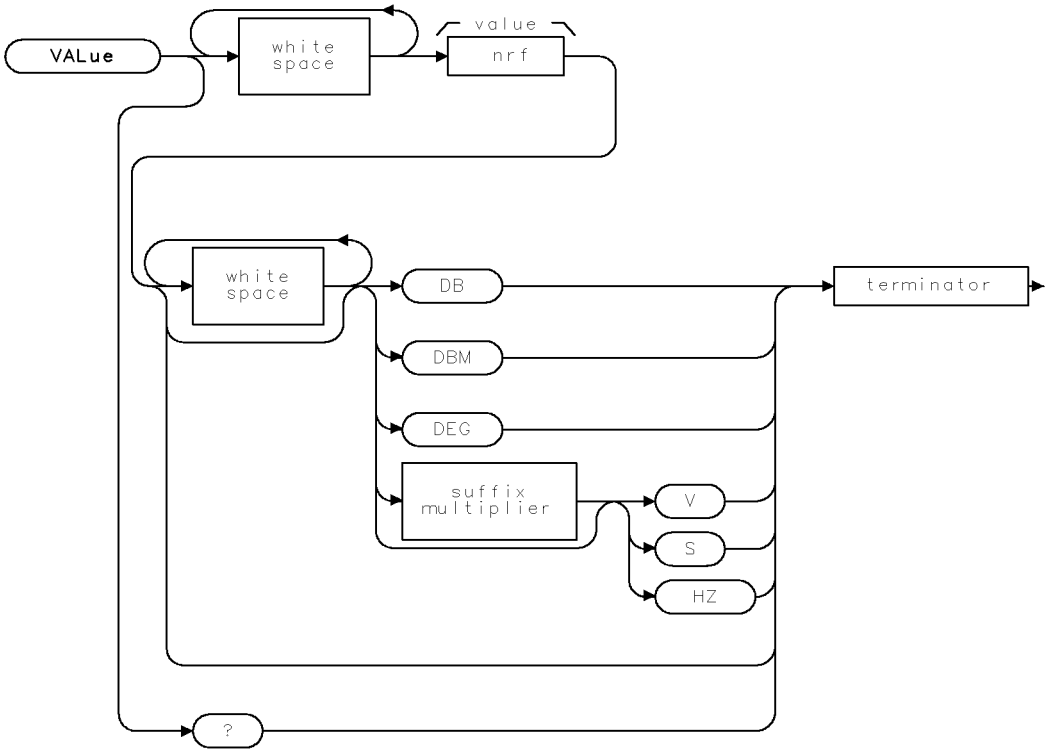
Query Example

```
OUTPUT @Mta;"DISP:MENU:KEY 1;TYPE?"
ENTER @Mta;Type$
```

MENU:VALue

Sets the numeric value of a key.

Syntax



Parameter Range no range checking is performed
Fundamental Units: seconds, volts, hertz, degrees, dB, or dBm

Command Example

```
OUTPUT @Mta;"DISP:MENU:VAL 1GHZ"
```

Description

The VALUE command sets the numeric variable for the currently selected key. The query returns the current value of the numeric variable. The result returned by the VALUE query needs to be interpreted according to the value of DISPLAY:MENU:TYPE for the currently selected key.

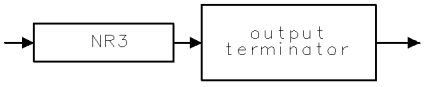
ALPHA Not applicable

EXCLUDE 0 = not selected
1 = selected

KNOB Value of the active parameter

LEFT	0 = not selected 1 = selected
LIST	0 = first state set by DISPLAY:MENU:LIST 1 = second state set by DISPLAY:MENU:LIST 2 = third state set by DISPLAY:MENU:LIST
NORMAL	Not applicable
TOGGLE	0 = off 1 = on

Query Response



QGENNR3

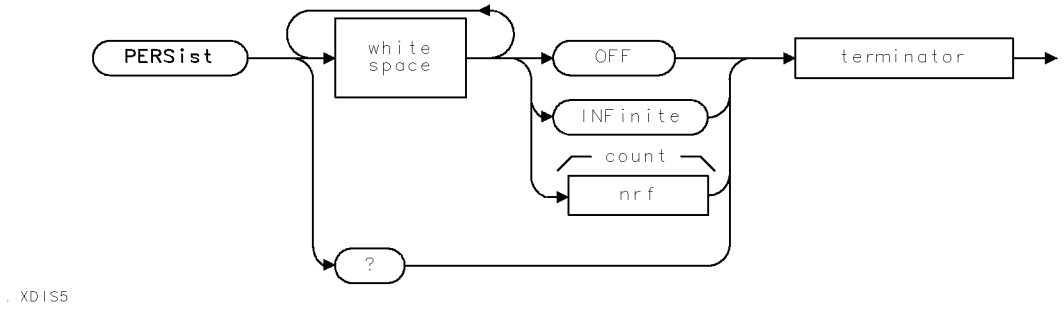
Query Example

```
OUTPUT @Mta;"DISP:MENU:VAL?"
ENTER @Mta;Value
```

PERSist

Specifies infinite or variable persistence display mode.

Syntax



Preset State: OFF
Parameter Range: count = 1 to 1024; where 1024 is the same as INFINITE

Command Example

```
OUTPUT @Mta;"DISP:PERS INF"
```

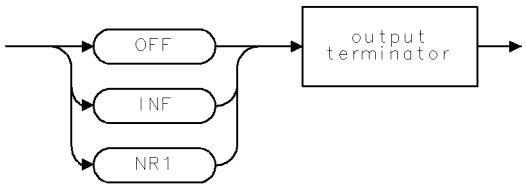
Description

The PERSIST command sets the display mode to infinite or variable persistence. This mode is only available if the graphics are being drawn on an HP 70004A graphics display. This is a display mode only and does not affect any measurements or parameters that can be made remotely.

In the INFINITE mode, the area within the graticule, with the exception of the active parameter area, is never erased.

In the non-infinite mode, the area within the graticule, with the exception of the active parameter area, is erased only after the specified number of sweeps have been taken.

Query Response



. QDIS5

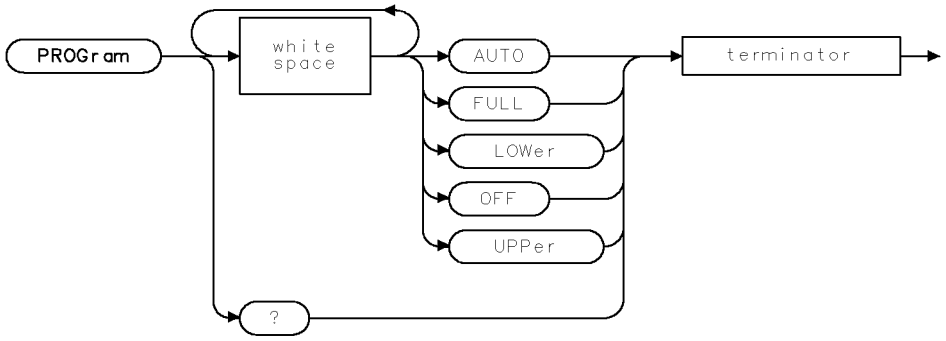
Query Example

```
OUTPUT @Mta;"DISP:PERS?"
ENTER @Mta;Persist$
```

PROGram

Selects the screen formatting.

Syntax



Preset State: AUTO

Command Example

```
OUTPUT @Mta;"DISP:PROG OFF"
```

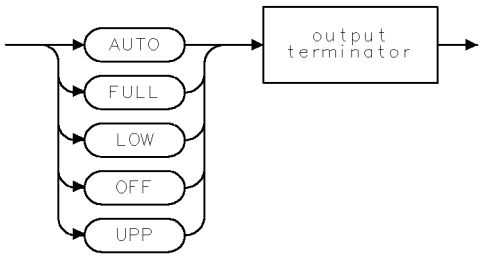
Description

The PROGRAM command determines how the text and graphics on the display will be arranged when the HP Instrument BASIC environment is active.

- FULL mode the entire screen will be a text-area for the use of IBASIC.
- LOWER mode the screen will be split into 2 sections. The upper section will contain the normal graphics and the lower section will be a text area for the use of IBASIC.
- OFF mode the screen will contain the normal, non-IBASIC, display. Any PRINT or DISP statements will write to the area that errors are normally printed.
- AUTO mode the screen format will depend on where in the menus the system is. If the system is in one of the IBASIC menus, the format will be the same as “2”, otherwise the format will be the same as “OFF”.

  The query returns the current state of PROGRAM.

Query Response



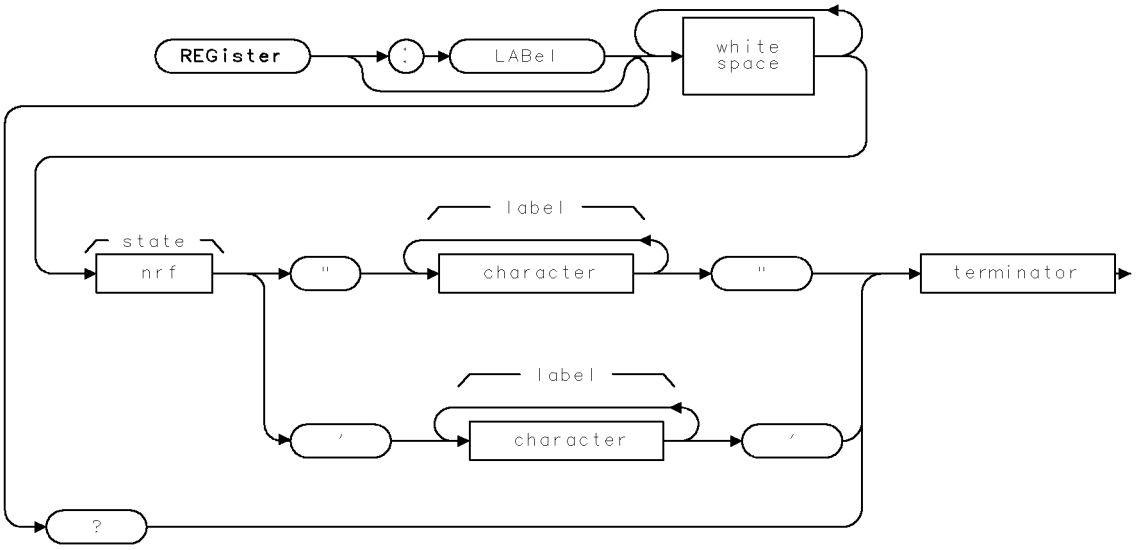
Query Example

```
OUTPUT @Mta;"DISP:PROG?"
ENTER @Mta;Format$
```

REGister:LABel

Sets menu key label for recall register.

Syntax



Preset State: Unaffected by PON, IP, or *RST
Parameter Range: state = 1 through 6
length of label is ≤ 7 characters

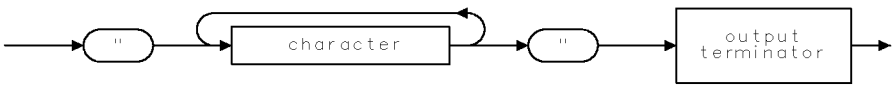
Command Example

```
OUTPUT @Mta;"DISP:REG:LAB 1,'DEMO 1'"
```

Description

The REGISTER:LABEL command allows the user to set the menu key label for the specified recall register. The query returns the current value of REGISTER:LABEL.

Query Response



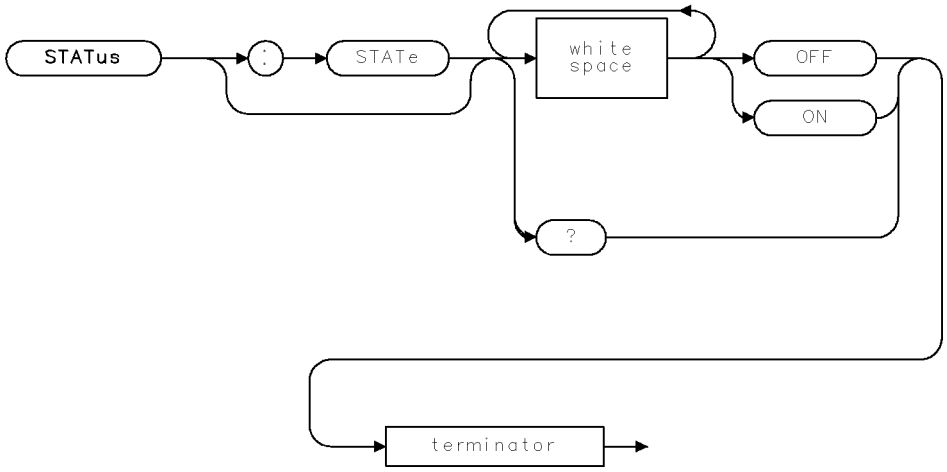
Query Example

```
OUTPUT @Mta;"DISP:REG:LAB? 1"
ENTER @Mta;Label$
```

STATus:STATe

Turns the display of a status block on or off.

Syntax



XDIS43

Preset State: not affected by IP, *RST, or power cycle

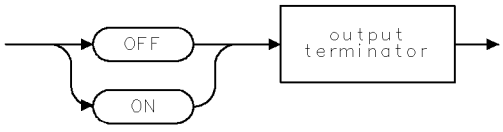
Command Example

```
OUTPUT @Mta;"DISP:STAT:STAT ON"
```

Description

The STATUS:STATE command turns the display of a status block on or off. The query returns the current state of the STATUS:STATE.

Query Response



QGENON

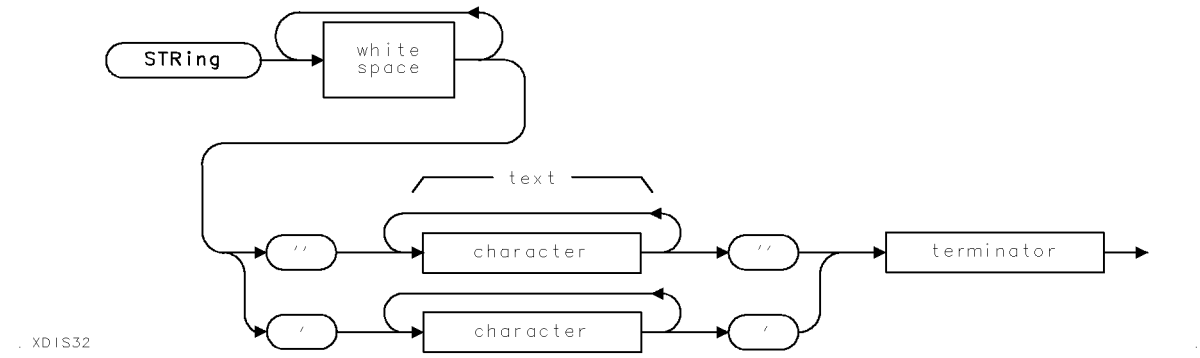
Query Example

```
OUTPUT @Mta;"DISP:STAT:STAT?"
ENTER @Mta;Status_state$
```

STRing

Displays the input string parameter on the display screen.

Syntax



Command Example

Example 1

```
OUTPUT @Mta;"DISP:STR ""Label"""
```

Example 2

```
OUTPUT @Mta;"DISP:STR 'Label'"
```

Description

The STRING command displays the input string parameter on the display screen in the same area as the error messages.

TEXT:CLEar

Clears the user text.

Syntax

xdiss2

Command Example

```
OUTPUT @Mta;"DISP:TEXT:PAGE 1;CLEAR"
```

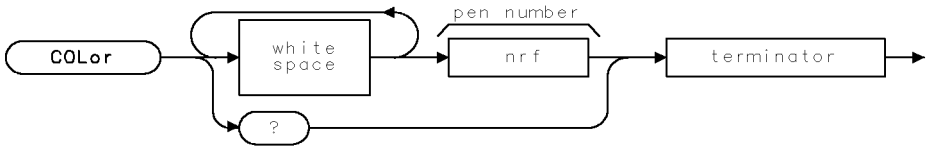
Description

The CLEAR command clears the user text-area specified by PAGE.

TEXT:COLor

Selects the user text-area color.

Syntax



Parameter Range: pen number = 0 through 15
Preset State: 6

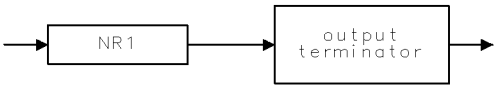
Command Example

```
OUTPUT @Mta;"DISP:TEXT:PAGE 1;COLOR 5"
```

Description

The COLOR command selects the color (that is, pen number) for the user text-area specified by PAGE. The query returns the current value of COLOR.

Query Response



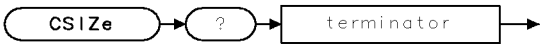
Query Example

```
OUTPUT @Mta;"DISP:TEXT:COLOR?"
ENTER @Mta;Value
```

TEXT:CSIZe

Returns the size of the user text-area.

Syntax



xdis69

Description

The CSIZE query returns the height and width, in characters, of the user text-area specified by PAGE.

Query Response

qgenn8

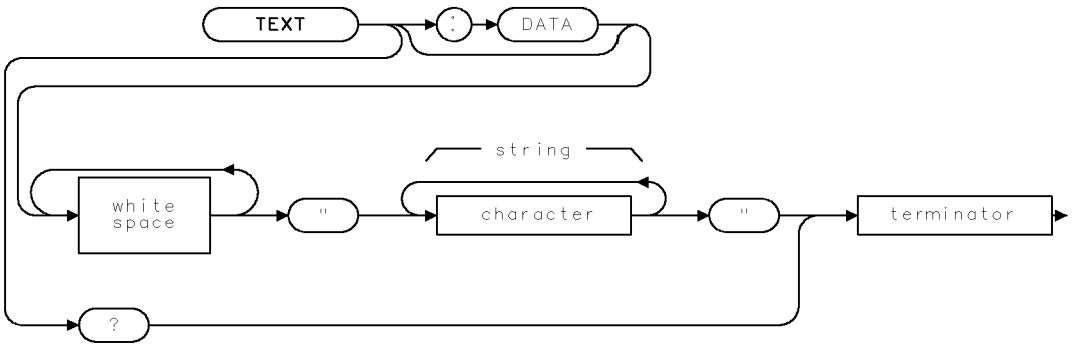
Query Example

```
OUTPUT @Mta;"DISP:TEXT:CSIZ?"
ENTER @Mta;Rows,Columns
```

TEXT:DATA

Defines the contents the user text-area.

Syntax



Parameter Range: maximum of 127 characters
Preset State: ""

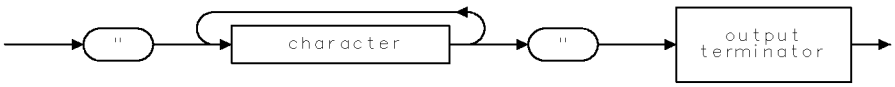
Command Example

```
OUTPUT @Mta;"DISP:TEXT 'Hello world.'"
```

Description

The DATA command defines the contents of the user text-area specified by PAGE. The query returns the current value of DATA.

Query Response



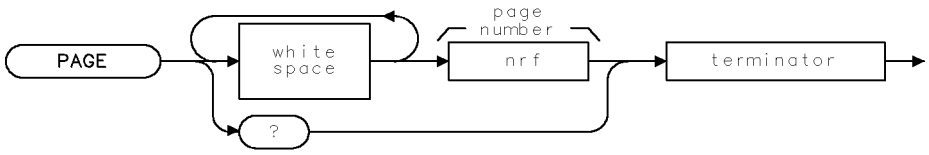
Query Example

```
OUTPUT @Mta;"DISP:TEXT?"
ENTER @Mta;String$
```

TEXT:PAGE

Selects the user text page.

Syntax



Parameter Range: page number = 1; only 1 page number is available at this time
Preset State: 1

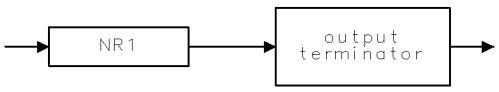
Command Example

```
OUTPUT @Mta;"DISP:TEXT:PAGE 1"
```

Description

The PAGE command selects a user text-area for the rest of the DISPLAY:TEXT commands and queries. The query returns the current value of PAGE.

Query Response



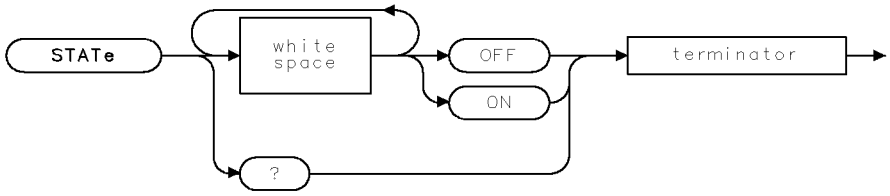
Query Example

```
OUTPUT @Mta;"DISP:TEXT:PAGE?"
ENTER @Mta;Value
```

TEXT:STATe

Enables or disables the user text-area.

Syntax



Preset State: OFF

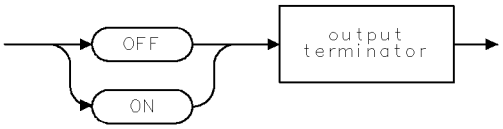
Command Example

```
OUTPUT @Mta;"DISP:TEXT:PAGE 1:STAT ON"
```

Description

The STATE command enables or disables the user text-area specified by PAGE. Note that when STATE is ON, the user text is displayed and some portion of the standard annotation is blanked. The query returns the current value of STATE.

Query Response



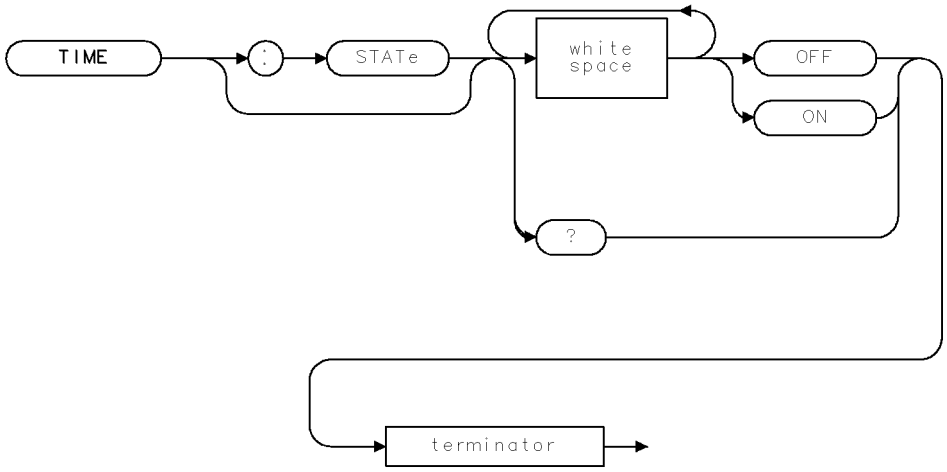
Query Example

```
OUTPUT @Mta;"DISP:TEXT:STAT?"
ENTER @Mta;State$
```

TIME:STATe

Turns the display of the time on or off.

Syntax



XDIS42

Preset State: not affected by IP, *RST, or power cycle

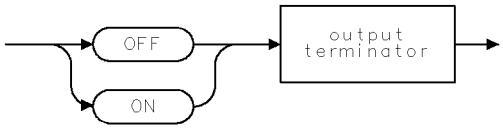
Command Example

```
OUTPUT @Mta;"DISP:TIME ON"
```

Description

The TIME:STATE command turns the display of the time on or off. The query returns the current state of the TIME:STATE.

Query Response



QGENON

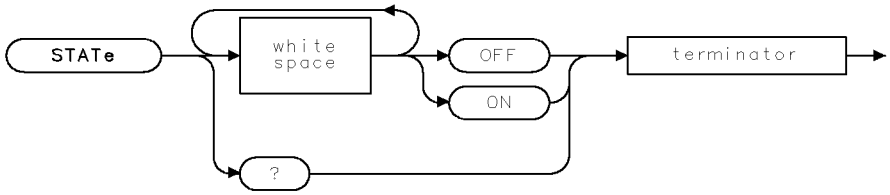
Query Example

```
OUTPUT @Mta;"DISP:TIME?"
ENTER @Mta;Time_state$
```

TITLe:STATe

Turns the display of the screen title on or off.

Syntax



xdis66

Preset State: OFF

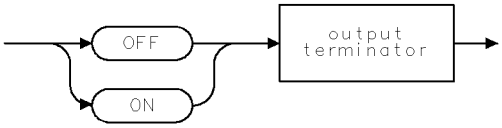
Command Example

```
OUTPUT @Mta;"DISP:TITL:STAT ON"
```

Description

The TITLE:STATE command turns the display of the screen title ON or OFF. The query returns the current state of TITLE:STATE.

Query Response



QGENON

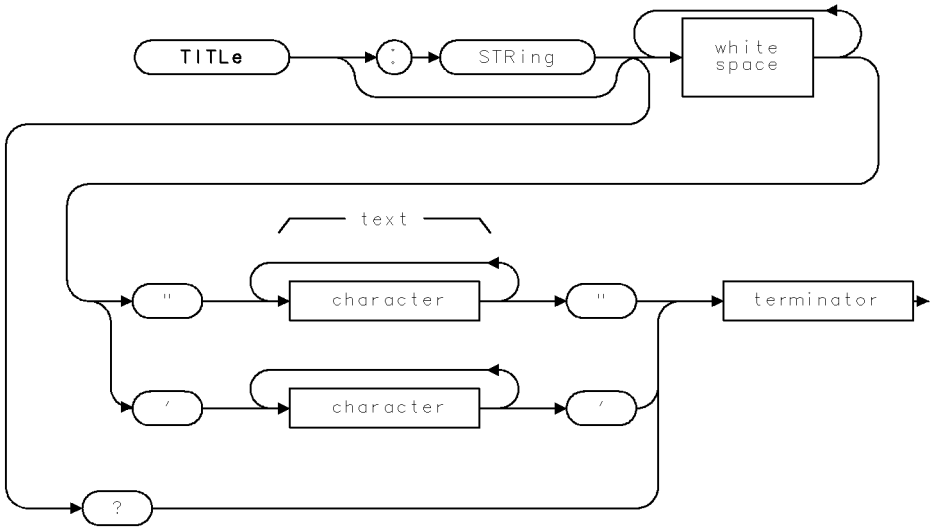
Query Example

```
OUTPUT @Mta;"DISP:TITL:STAT?"
ENTER @Mta;Title_state$
```

TITLe:STRing

Defines the ASCII text for the screen title.

Syntax



Power-On State: ""
Preset State: Unaffected by IP or *RST

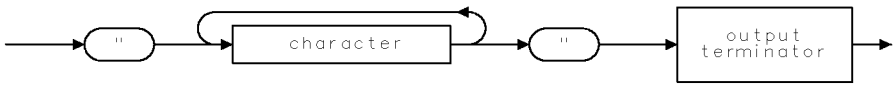
Command Example

```
OUTPUT @Mta;"DISP:TITL 'Hello world'"
```

Description

The TITLE:STRING command defines the ASCII text for the screen title. The query returns the current value of TITLE:STRING.

Query Response



Query Example

```
OUTPUT @Mta;"DISP:TITL?"
ENTER @Mta;Title$
```

13

Function Subsystem

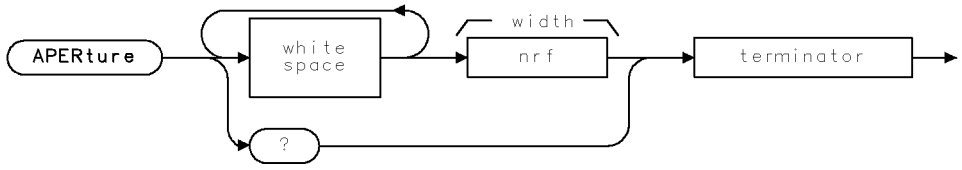
The FUNCTION subsystem allows a trace math operation to be performed using the available channels and waveform memories as operands. The input traces to the FUNCTION subsystem may be real or complex and in the time or frequency domain. See the following list for a command syntax summary of the FUNCTION subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:FUNCtion[<function_number>]		
:APERture	<nrf>	
:COUNt	<nrf>	
:FREeze	OFF|ON	
:RECursion	OFF|ON	
:TD	NORMal|SMOOth	
:TYPE	AVERage|FAVerage|HOLD|MAXhold |MINhold|NORMal	
:XSCale	OFF|ON	
:FUNction <function_number>		
:AUTorange	OFF|ON	
:DEFine	([{<character>}])	
:LABel	'{<character>}'|"{<character>}"	
:OFFSet	<nrf>DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ	
:RANGe	<nrf>DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ	
:RLEVel	<nrf>[DB|DBM]	
:SMOoth		
:COUNt	<nrf>	
[:STATe]	OFF|ON	
:ZOOM		
:CENTer	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:FREQ	OFF|ON	
:SPAN	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:STARt	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
[:STATe]	OFF|ON	
:STOP	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:TIME	OFF|ON	
:FUNCtion		
:XSCale	OFF|ON	

APERture

Sets the aperture width for the DELTA, FM, and TD operations.

Syntax



Preset State: 1
Parameter Range: width = 1 through 1024
Related Command: FUNCx:DEFINE

Command Example

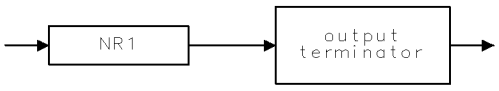
```
OUTPUT @Mta;"FUNC1:APER 5"
```

Description

The APERTURE command sets the aperture width for the DELTA, FM, and TDoperations of the FUNCTIONx:DEFine command. If the command header is FUNC:, as opposed to FUNC1:, FUNC2:, FUNC3:, or FUNC4:, the aperture width is set for all functions.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



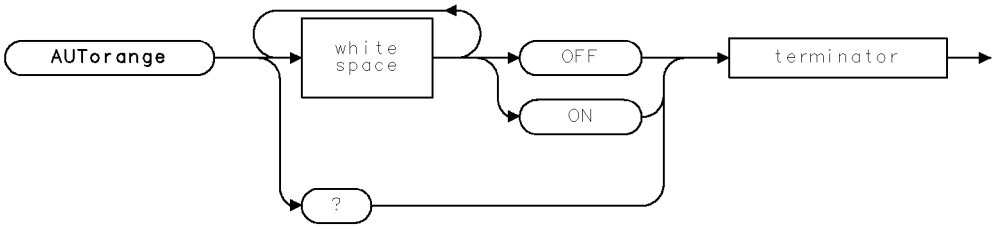
Query Example

```
OUTPUT @Mta;"FUNC2:APER?"
ENTER @Mta;Aperture_width
```

AUTorange

Enables or disables the automatic vertical-sensitivity adjust for the specified function.

Syntax



Preset State: ON

Command Example

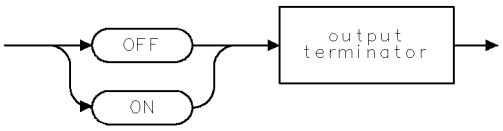
```
OUTPUT @Mta;"FUNC1:AUT ON"
```

Description

The AUTORANGE command enables or disables the automatic vertical-sensitivity adjust for the specified function. The query returns the current state of AUTORANGE.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



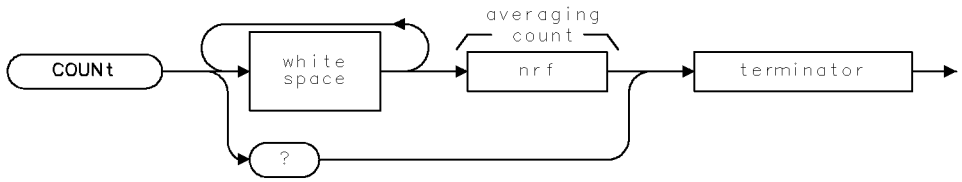
Query Example

```
OUTPUT @Mta;"FUNC2:AUT?"
ENTER @Mta;Autorange$
```

COUNt

Sets the averaging count for the specified function.

Syntax



Preset State: 16
Parameter Range: averaging count = 1 through 1024

Command Example

```
OUTPUT @Mta;"FUNC1:COUN 55"
```

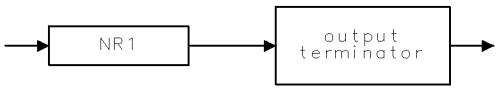
Description

The COUNT command is used to set the averaging count for the specified function. This does not control the averaging count for channels (see CHANx:COUNT or ACQUIRE:COUNT). The query returns the last value set.

Note that the header is FUNC:, as opposed to FUNC1: or FUNC2:. The command applies to all functions. See the ACQUIRE:COUNT command for a detailed description of the averaging operation.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



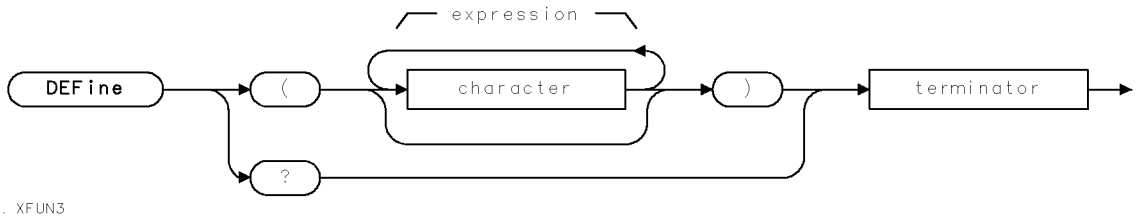
Query Example

```
OUTPUT @Mta;"FUNC1:COUN?"
ENTER @Mta;Count
```

DEFine

Sets or reads an algebraic expression that describes a function.

Syntax



Preset State: FUNC1 = (CHAN1)
FUNC2 = (CHAN2)
FUNC3 = (CHAN1 / CHAN2)
FUNC4 = (CHAN1 / CHAN2)

Related Commands: BLANk (Top-Level)
VIEW (Top-Level)

Command Example

```
OUTPUT @Mta;"FUNC1:DEF ((CHAN1-WMEM1)/WMEM2)"
```

Description

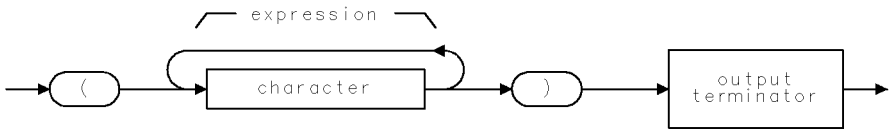
The DEFINE command allows the user to set or read an algebraic expression that describes a function.

When the DEFINE command is sent, the function is turned on (that is, an implied VIEW FUNCx is executed). A function is only evaluated if it is on or if it is required by another function.

Refer to Appendix D, "Math Algorithms," for a more complete description of each operation.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QFUN3

where expression is an algebraic equation.

Note The outer parenthesis are part of the parser syntax, not part of the expression. Also, expression is optional (that is, "FUNCx:DEFINE ();" is legal and undefines the function).

Query Example

```
OUTPUT @Mta;"FUNC2:DEF?"
ENTER @Mta;Func2$
```

The following symbols in the tables below are recognized:

Table 13-3. Operands

Symbol	Description
CHAN1	channel 1
CHAN2	channel 2
FUNC1	function 1
FUNC2	function 2
FUNC3	function 3
FUNC4	function 4
WMEM1	memory 1
WMEM2	memory 2
WMEM3	memory 3
WMEM4	memory 4
number	constants
FSIG	defines a real constant equal to SWEep:SIGnal
FCAR	defines a real constant equal to PULse:CARRier

Table 13-4. Operations

Symbol	Precedence	Description
VS	0	a versus b
+	5	addition
–	5	subtract (two operands)
*	10	multiplication
/	10	division
CHOP	15	Force small magnitudes to 0
CONV	15	cyclic convolution
CORR	15	cyclic correlation
MOD	15	A modulor B
CONST	20	Change domain to “constant”
–	20	negate (single operand)
AC	20	Remove DC portion of signal
AM	20	AM demodulation
ANALY	20	analytic signal
ATAN	20	phase = arctan(I/R)
D/DX	20	derivative
DB	20	20 x log(x)
DC	20	Remove AC portion of signal
DEG	20	unwrapped arctan()
DELTA	20	difference operator
DFT	20	Discrete Fourier Transform
DIFF	20	derivative
EXPJ	20	exp(2 x PI x j x real)
FFT	20	Fast Fourier Transform
FM	20	delta phase / delta t
IDFT	20	Inverse DFT
IFFT	20	Inverse FFT
IMAG	20	imag of complex
INTEG	20	integration
MAGN	20	SQR(R x R+I x I)
PM	20	Phase demodulation
REAL	20	real of complex
SHIFT	20	applies a phase shift to the data. For example, this can be used to remove a linear phase slope.
SIGN	20	Polarity of signal
SQRT	20	square root
SUM	20	summation
TD	20	delta phase / delta f
[n]	30	defines a constant equal to the nth point of the source trace
[n,m]	30	defines a trace equal to the nth through mth elements of the source trace
(none)	50	copy (that is, output = input)

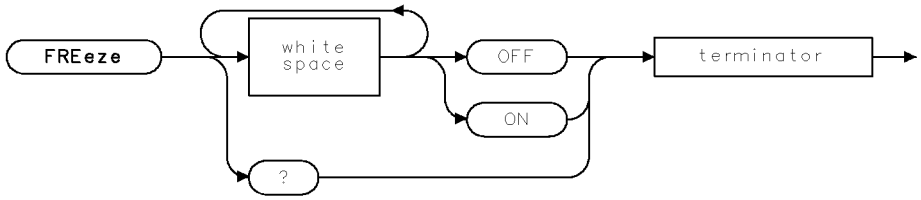
Table 13-5. Miscellaneous

Symbol	Description
(	left parenthesis
)	right parenthesis
	space (ignored)

FREeze

Enables or disables the evaluation of the specified function.

Syntax



Preset State: OFF
Related Commands: FUNCx:DEFine
FUNCx:RECursion

Command Example

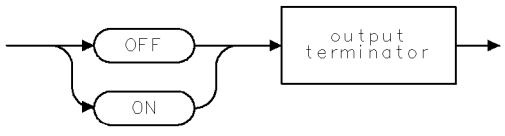
```
OUTPUT @Mta;"FUNC1:DEF (WMEM1-WMEM2)"
OUTPUT @Mta;"FUNC2:DEF (WMEM1+WMEM2)"
OUTPUT @Mta;"FUNC:FRE ON"
OUTPUT @Mta;"STORE FUNC1,WMEM1"
OUTPUT @Mta;"STORE FUNC2,WMEM2"
OUTPUT @Mta;"FUNC:FRE OFF"
```

Description

The FREEZE command enables or disables the evaluation of the specified function. When FREEZE is ON, the only time a function is evaluated is when a new equation is entered. The function will be evaluated when FREEZE is turned OFF. If the header is FUNC:, as opposed to FUNC1:, FUNC2:, FUNC3:, or FUNC4:, then all functions will be affected.

The query returns the current state of FREEZE.

Query Response



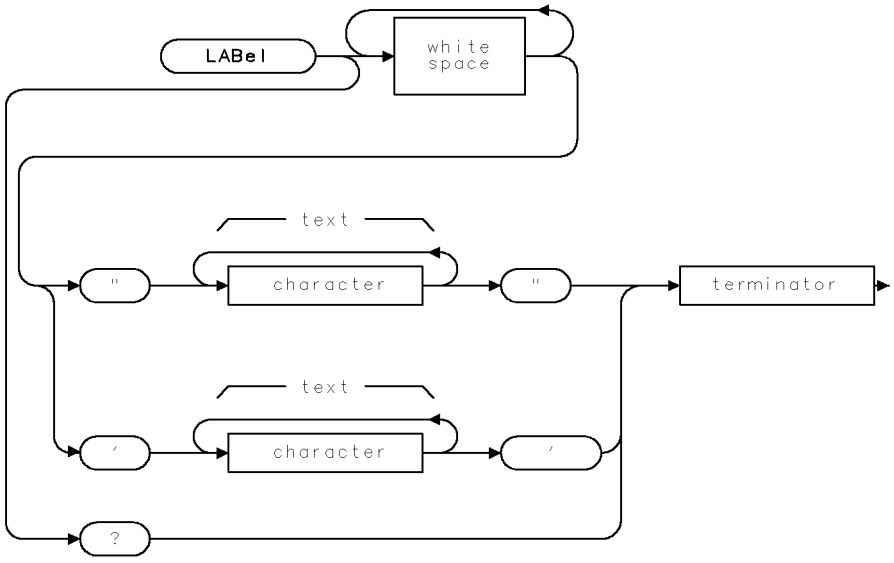
Query Example

```
OUTPUT @Mta;"FUNC1:FRE?"
ENTER @Mta;Freeze$
```

LABel

Defines the screen annotation for the specified function.

Syntax



Preset State: FUNC1: Ch1
FUNC2: Ch2
FUNC3: Ch1/Ch2
FUNC4: Ch1/Ch2

Command Example

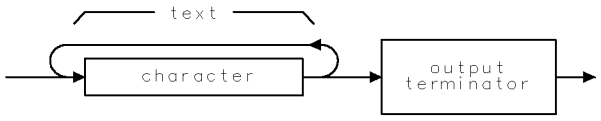
```
OUTPUT @Mta;"FUNC1:LAB 'S11'"
```

Description

The LABEL command defines the screen annotation of a specified function. The string is limited to 7 characters.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



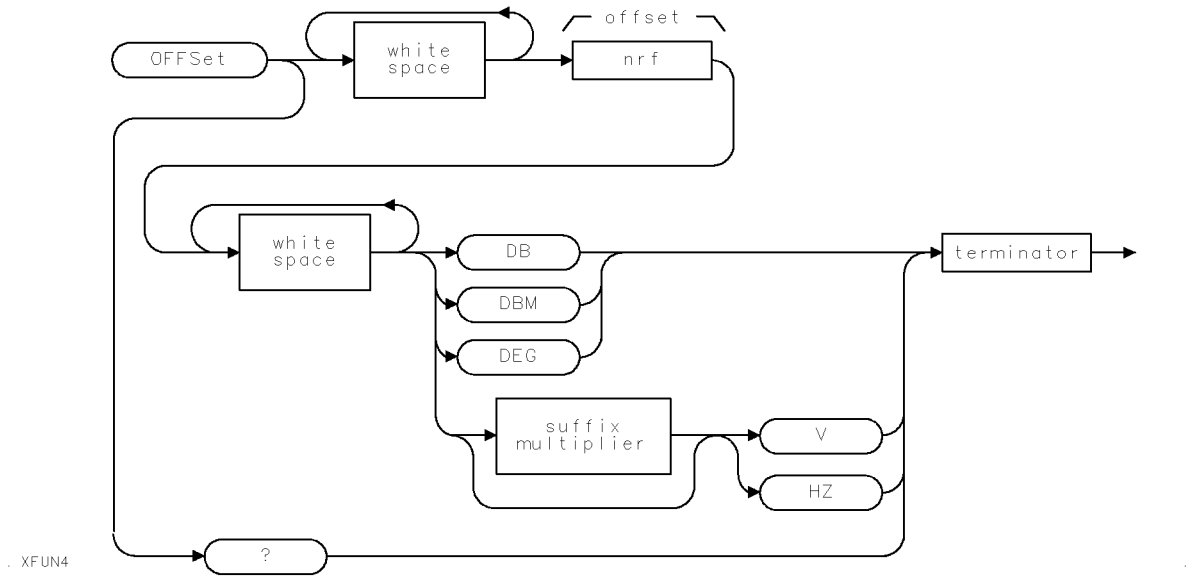
Query Example

```
OUTPUT @Mta;"FUNC1:LAB?"
ENTER @Mta;Label1$
```

OFFSet

Sets the offset for the internal scaling of the function.

Syntax



Preset State:	0.0 (FUNCx:AUTorange is ON)
Parameter Range:	FUNCx:RANGe ≤ offset ≤ FUNCx:RANGe
Fundamental Unit:	volts
Related Commands:	DISPlay:FUNCx:OFFSet
	DISPlay:FUNCx:RANGe
	FUNCx:AUTorange
	FUNCx:RANGe
	FUNCx:RLEVel

Command Example

```
OUTPUT @Mta;"FUNC1:OFFS 2V"
```

Description

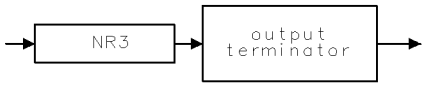
The OFFSET command sets the offset for the internal scaling of the function. The query returns the current value of OFFSET.

When the OFFSET command is sent, the displayed offset value (DISPLAY:FUNCx:OFFSET) is set to the same value.

Note This value is equivalent to the hardware voltage offset for the channels that will autorange unless the OFFSET command is used. Generally it is not wise to set this value unless you know the range and offset of the function for all possible input. Instead, use the DISPLAY:FUNCx:OFFSET command.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENNR3

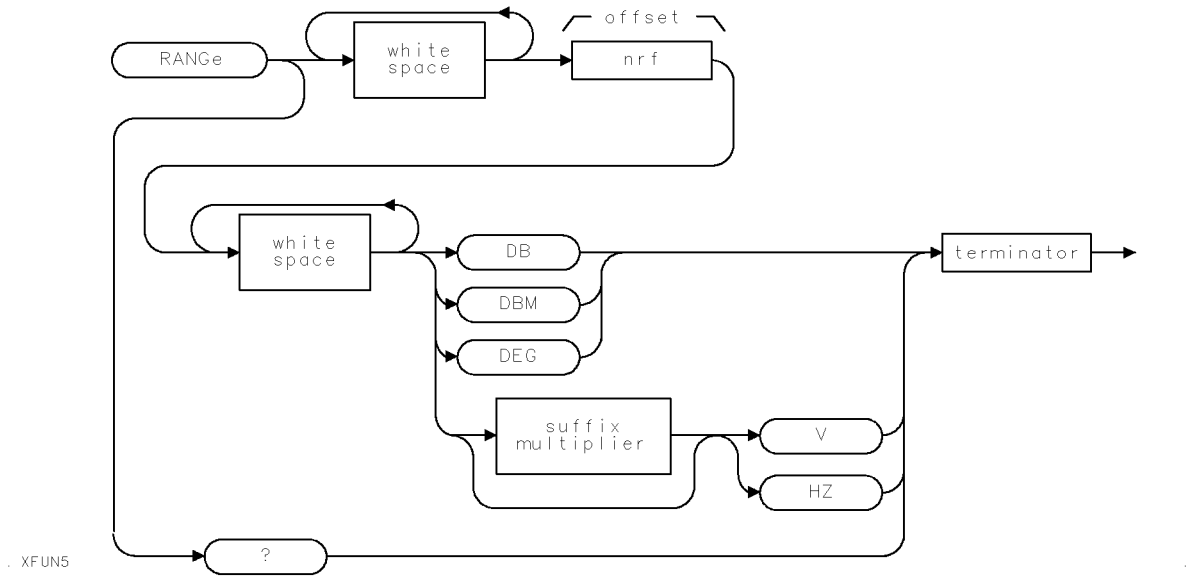
Query Example

```
OUTPUT @Mta;"FUNC1:OFFS?"
ENTER @Mta;Offset
```

RANGe

Sets the range for the internal scaling of the function using a linear input.

Syntax



Preset State: 0.2 (FUNCx:AUTorange is ON)
Parameter Range: 2.0E–20 ≤ linear range ≤ 2.0E20 V
−400 dB ≤ logged reference level ≤ 400 dB
−390 dBm ≤ logged reference level ≤ 410 dBm
Fundamental Unit: volts
Related Commands: DISPlay:FUNCx:OFFSet
DISPlay:FUNCx:RANGe
FUNCx:AUTorange
FUNCx:OFFSet
FUNCx:RLEVel

Command Example

```
OUTPUT @Mta;"FUNC1:RANG 20 mV"
```

Description

The RANGE command sets the range for the internal scaling of the function. The query returns the current value of RANGE.

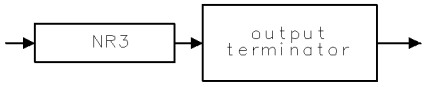
The RANGE command sets the same parameter as FUNCx:RLEVEL. The RANGE command takes linear input whereas the RLEVEL command takes logarithmic input.

When the RANGE command is sent, the displayed range value (DISPLAY:FUNCx:RANGE) is set to the same value.

Note This value is equivalent to the hardware voltage range for the channels that will autorange unless the RANGE command is used. Generally it is not wise to set this value unless you know the range and offset of the function for all possible input. Instead, use the DISPLAY:FUNCx:RANGE command.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENNR3

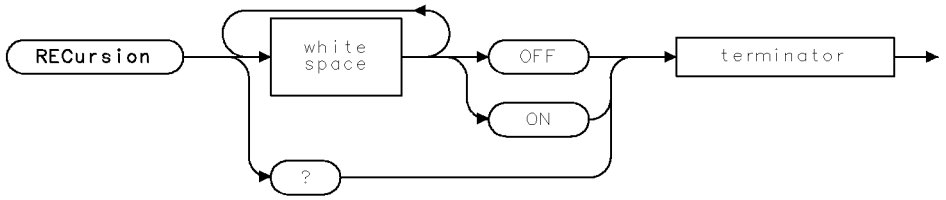
Query Example

```
OUTPUT @Mta;"FUNC1:RANG?"
ENTER @Mta;Range
```

RECursion

Enables or disables the checks for recursive functions.

Syntax



Preset State: OFF
Related Commands: FUNCx:DEFine
FUNCx:FREeze

Command Example

```
OUTPUT @Mta;"FUNC1:REC ON"
OUTPUT @Mta;"FUNC1:FREEZE ON"
OUTPUT @Mta;"FUNC1:DEF (CHAN1)"
OUTPUT @Mta;"FUNC1:DEF (.1*CHAN1+.9*FUNC1)"
```

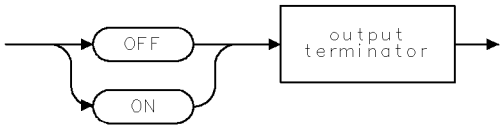
Description

The RECURSION command enables or disables the checks for recursive functions. When RECURSION is ON, a function may be defined in terms of itself (for example *FUNC1:DEF(.1×CHAN1+.9×FUNC1)*). When RECURSION is OFF, this type of equation generates an error. If the header is FUNC:, as opposed to FUNC1:, FUNC2:, FUNC3:, or FUNC4:, then all functions will be affected.

The query returns the current state of RECURSION.

Note It is strongly recommended to have FUNCx:FREEZE set to ON whenever RECURSION is ON since FREEZE gives the user control over when and how often the function will be reevaluated.

Query Response



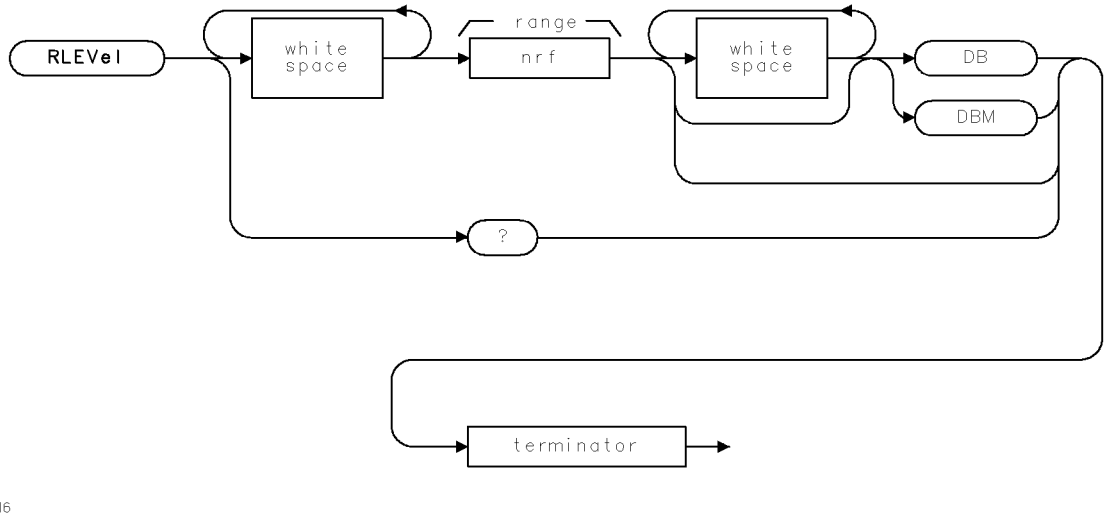
Query Example

```
OUTPUT @Mta;"FUNC1:REC?"
ENTER @Mta;Recursion$
```

RLEVel

Sets the range for the internal scaling of the function using a logarithmic input.

Syntax



Preset State:	−10 dBm (FUNCx:AUTorange is ON)
Parameter Range:	2.0E–20 ≤ linear range ≤ 2.0E20 V
	−400 dB ≤ logged reference level ≤ 400 dB
	−390 dBm ≤ logged reference level ≤ 410 dBm
Fundamental Unit:	dBm
Related Commands:	DISPlay:FUNCx:OFFSet
	DISPlay:FUNCx:RANGe
	FUNCx:AUTorange
	FUNCx:OFFSet
	FUNCx:RLEVel

Command Example

```
OUTPUT @Mta;"FUNC1:RLEV -20 dBm"
```

Description

The RLEVEL command sets the range for the internal scaling of the function. The query returns the current value of RANGE.

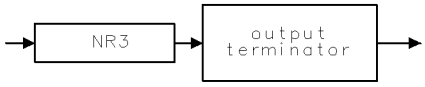
RLEVEL sets the same parameter as FUNCx:RANGE. The RANGE command takes linear input whereas the RLEVEL command takes logarithmic input.

When the RANGE command is sent, the displayed range value (DISPLAY:FUNCx:RANGE) is set to the same value.

Note This value is equivalent to the hardware voltage range for the channels that will autorange unless the RLEVEL command is used. Generally it is not wise to set this value unless you know the range and offset of the function for all possible input. Instead, use the DISPLAY:FUNCx:RANGE command.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENNR3

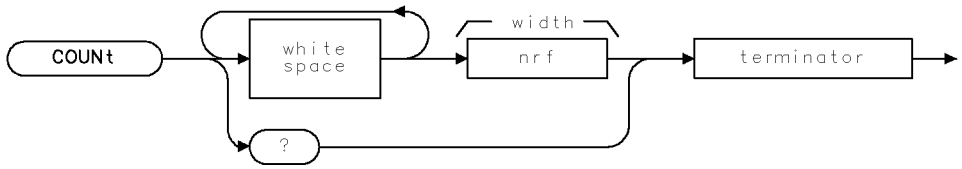
Query Example

```
OUTPUT @Mta;"FUNC1:RLEV?"
ENTER @Mta;Rlevel
```

SMOoth:COUNt

Sets the width of the median filter.

Syntax



Preset State: 16
Parameter Range: width = 1 through 1024

Command Example

```
OUTPUT @Mta;"FUNC1:SMO:COUN 55"
```

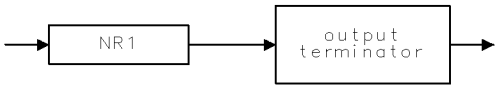
Description

The SMOOTH:COUNT command is used to set the width of the median filter for the specified function. The query returns the last value set.

See Appendix E, “Median Smoothing”, for a description of median filtering.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



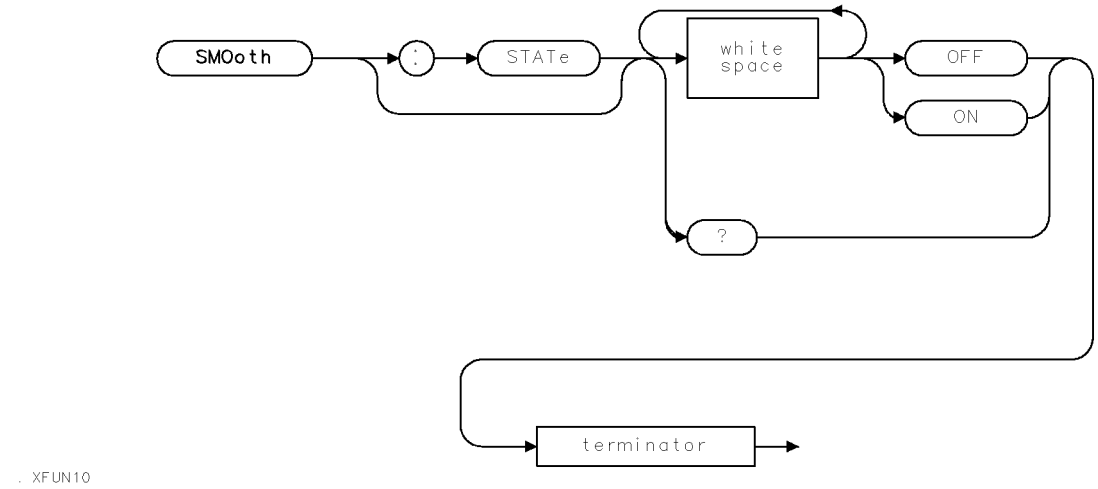
Query Example

```
OUTPUT @Mta;"FUNC2:SMO:COUN?"
ENTER @Mta;Median_width
```

SMOoth:STATe

Enables or disables a median filter process.

Syntax



Preset State: OFF
Related Commands: FUNCx:TYPE
FUNCx:SMOoth:COUNt

Command Example

```
OUTPUT @Mta;"FUNC1:SMO:STAT ON"
OUTPUT @Mta;"FUNC2:SMO OFF"            ! STATE is optional.
```

Description

The SMOOTH:STATE command enables or disables a median filter process for the specified function. The query returns the current setting.

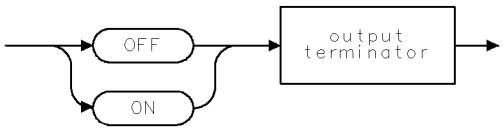
When FUNCx:SMOOTH:STATE is set to ON, the value of FUNCx:TYPE will be set to NORMAL.

See Appendix E, "Median Smoothing", for a description of median filtering.

SMOOTH:STATE is not independent of FUNCx:TYPE. Median smoothing can only be applied if TYPE is NORMAL or AVERAGE.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENON

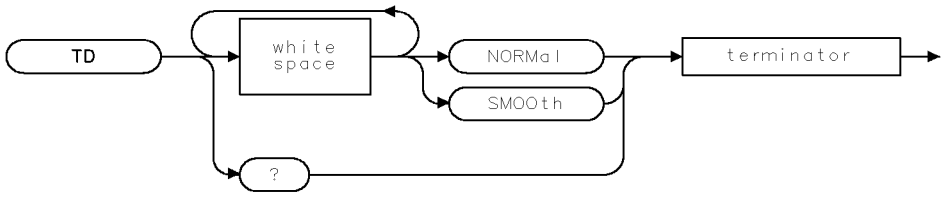
Query Example

```
OUTPUT @Mta;"FUNC1:SMO?"
ENTER @Mta;State$
```

TD

Enables or disables an intermediate smoothing operation for the TD () math operation.

Syntax



xfun34

Preset State: SMOOth

Command Example

```
OUTPUT @Mta;"FUNC:TD NORM"
```

Description

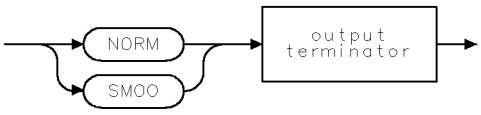
The TD command enables or disables an intermediate smoothing operation for the TD() math operation. The query returns the current state of TD.

When TD is set to SMOOTH, the TD() math operation computes the delta with a 1-point aperture, smooths the result with a 3-point median filter, and then applies a boxcar smoothing with a width specified by FUNCx:APERTURE.

For compatibility with revisions 1.0 and 1.01, set TD to NORMAL.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



qfun34

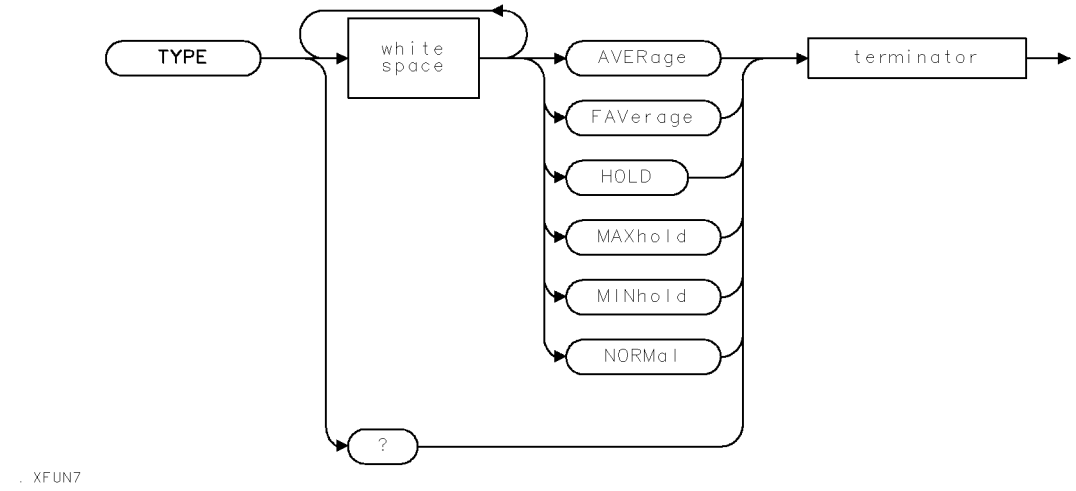
Query Example

```
OUTPUT @Mta;"FUNC1:TD?"
ENTER @Mta;Td$
```

TYPE

Controls selection of a post-processing operation for the specified function.

Syntax



Preset State: NORMal
Related Commands: FUNCx:SMOoth:STATe

Command Example

```
OUTPUT @Mta;"FUNC:TYPE AVER"
```

Description

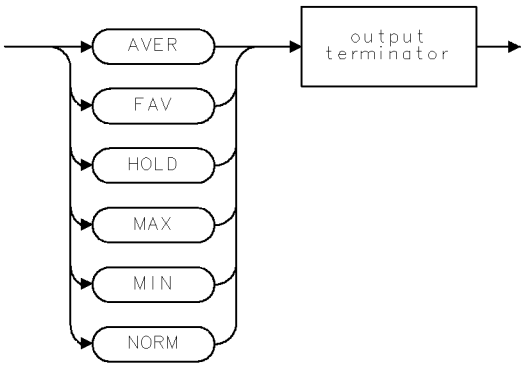
The TYPE command controls the selection of a post-processing operation for the specified function. This does not set the post-processing operation for channels (see ACQUIRE:TYPE or CHANx:TYPE). The query returns the last value set.

Note that if the header is FUNC:, as opposed to FUNC1: or FUNC2:, the command applies to all functions and the query applies to function 1. See the ACQUIRE:TYPE command for a description of the parameters.

TYPE is not independent of median smoothing. Median smoothing can only be applied if TYPE is NORMAL or AVERAGE.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QFUN7

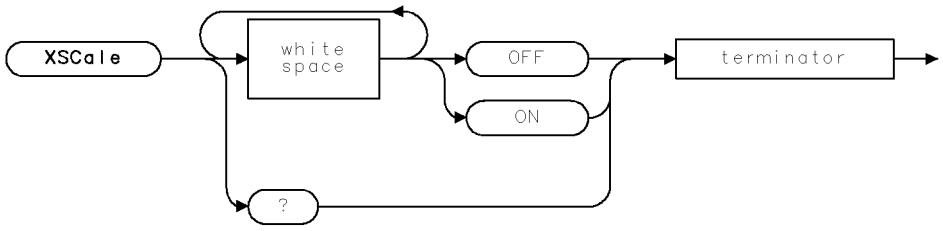
Query Example

```
OUTPUT @Mta;"FUNC1:TYPE?"
ENTER @Mta;Type$
```

XSCale (Horizontal Scaling)

Turns the horizontal scaling mode for the math functions on or off.

Syntax



Preset State: OFF

Command Example

```
OUTPUT @Mta;"FUNC:XSC OFF"
```

Description

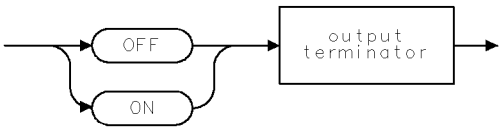
The XSCALE command turns the horizontal scaling mode for the math functions on or off. The horizontal scaling mode, which is a global parameter, applies to all functions so the command header does not have a function selector (that is, use FUNC not FUNC1 or FUNC2). The query returns the current state of XSCALE.

When XSCALE is off, the operands of all math operations will ignore any difference in horizontal range and horizontal offset. An error is declared if the trace lengths are different, but the expression will still be evaluated using the smaller size and only the first N points of the larger trace is used (where N is the size of the smaller trace).

When XSCALE is on and the inputs are in the time domain, the math operations will use linear interpolation to match the size, horizontal range, and horizontal offset. If the traces do not have the same horizontal range, linear interpolation will be used to get evenly spaced data over the smaller horizontal range. If the traces are not the same length, linear interpolation will be used to get the smaller number of points over the appropriate horizontal range. If the traces do not have the same horizontal offset, a "bucket" shift will be done to start both traces at the larger of the two offsets. If there is not enough data (that is, the horizontal range of one trace does not lie completely within the horizontal range of the other trace), an error is declared but the expression will still be evaluated with zeros filled in at the appropriate parts of the trace.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENON

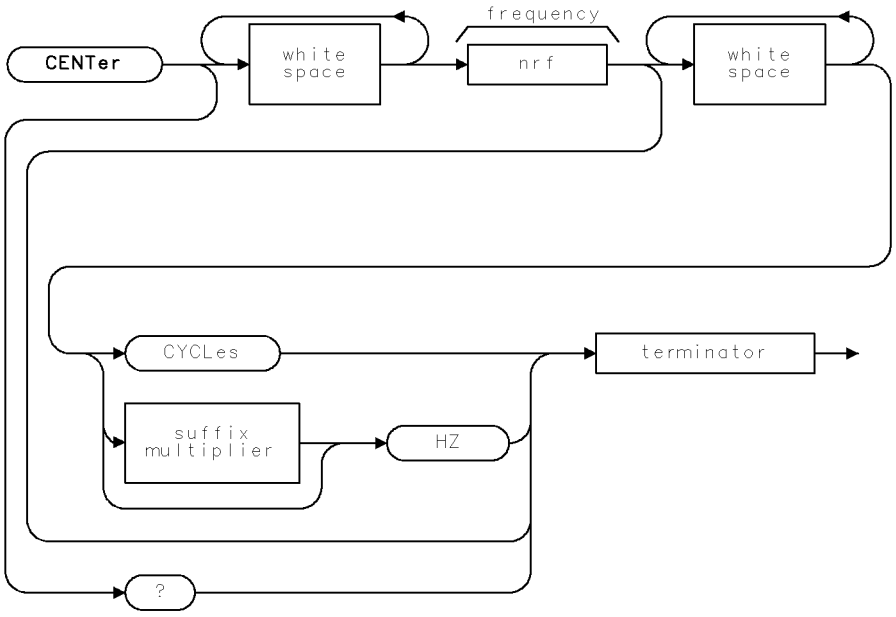
Query Example

```
OUTPUT @Mta;"FUNC:XSC?"
ENTER @Mta;A$
```

ZOOM:CENTer

Specifies the center frequency used for the zoom transform.

Syntax



XFUN11

Preset State: 50 MHz
Parameter Range: frequency = 0 through 100 GHz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"FUNC1:ZOOM:CENT 5 GHz"
```

Description

The ZOOM:CENTER command specifies the center frequency used for the zoom transform. The query returns the current setting.

The CENTER command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting a measurement independent of the specific signal frequency. The query always returns the value in hertz.

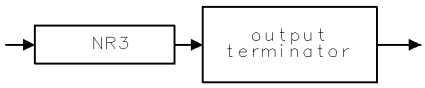
The module converts cycles to hertz by using the following equation:

$$hertz = cycles \times signal\ frequency$$

Note This command modifies the same value as the CHANx:ZOOM:CENTER and SWEEP:TIME:ZOOM:CENTER commands.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENNR3

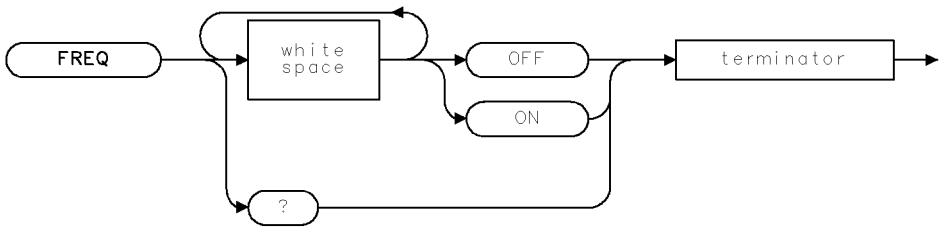
Query Example

```
OUTPUT @Mta;"FUNC1:ZOOM:CENT?"
ENTER @Mta;Center_freq
```

ZOOM:FREQ

Enables or disables the use of zoom transform data when in the frequency domain.

Syntax



Preset State: OFF

Command Example

```
OUTPUT @Mta;"FUNC2:ZOOM:FREQ OFF"
```

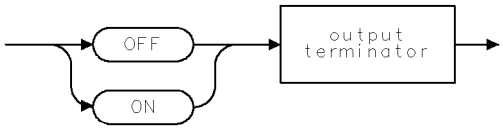
Description

The ZOOM:FREQ command enables and disables the use of zoom transform data when the function is in the frequency domain. The query returns the current state.

For example, if ZOOM:FREQ is OFF and the function is displayed in the frequency domain, the data will be the normal FFT of the time domain data. If ZOOM:FREQ is ON and the function is displayed in the frequency domain, the data will be from the zoom transform. Note that this command does not enable the zoom transform, it just determines whether or not to use the zoom transform data once the transform is enabled (see FUNCx:ZOOM:STATE).

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



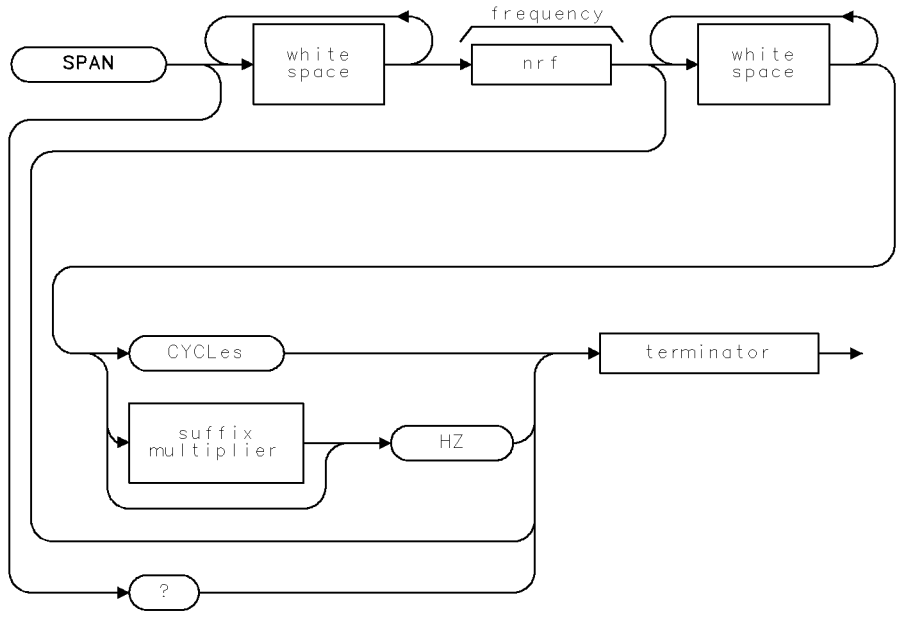
Query Example

```
OUTPUT @Mta;"FUNC1:ZOOM:FREQ?"
ENTER @Mta;Zoom_state
```

ZOOM:SPAN

Specifies the frequency span used for the zoom transform.

Syntax



XFUN13

Preset State: 100 MHz
Parameter Range: frequency = 0 through 100 GHz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"FUNC1:ZOOM:SPAN 10 GHz"
```

Description

The ZOOM:SPAN command specifies the frequency span used for the zoom transform. The query returns the current setting.

The SPAN command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting a measurement independent of the specific signal frequency.

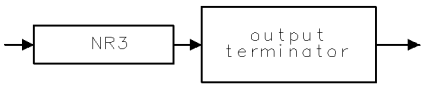
The module converts cycles to hertz by using the following equation:

$$hertz = cycles \times signal\ frequency$$

Note This command modifies the same value as the CHANx:ZOOM:SPAN and SWEEP:TIME:ZOOM:SPAN commands.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENNR3

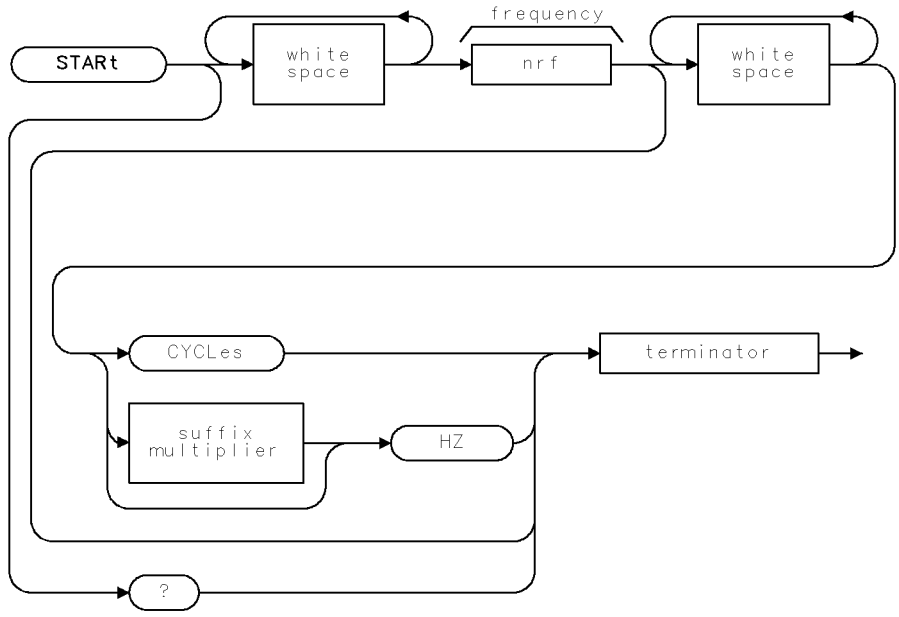
Query Example

```
OUTPUT @Mta;"FUNC1:ZOOM:SPAN?"
ENTER @Mta;Span
```

ZOOM:STARt

Specifies the start frequency used for the zoom transform.

Syntax



XFUN15

Preset State: 0
Parameter Range: frequency = 0 through 100 GHz
Fundamental Unit: hertz
Related Commands: CHANx:ZOOM:STARt
SWEep:TIME:ZOOM:STARt

Command Example

```
OUTPUT @Mta;"FUNC1:ZOOM:STAR 100 MHz"
```

Description

The ZOOM:START command specifies the start frequency used for the zoom transform. The query returns the current setting.

The START command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting a measurement independent of the specific signal frequency. The query always returns the value in hertz.

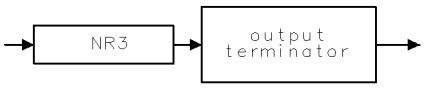
The module converts cycles to hertz by using the following equation:

$$hertz = cycles \times signal\ frequency$$

Note This command modifies the same value as the CHANx:ZOOM:STARt and SWEEP:TIME:ZOOM:START commands.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENNR3

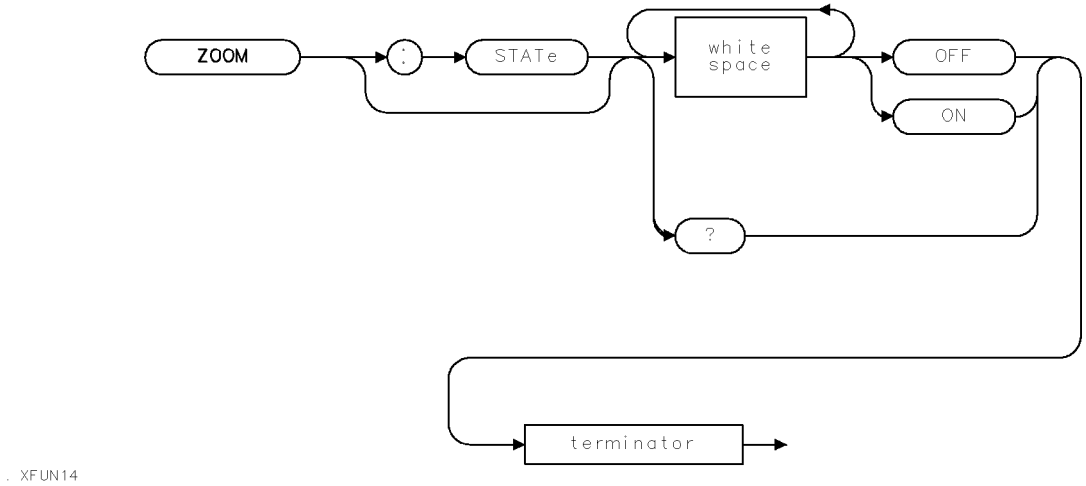
Query Example

```
OUTPUT @Mta;"FUNC1:ZOOM:STAR?"
ENTER @Mta;Start_freq
```

ZOOM:STATe

Determines whether data from the zoom transform is used.

Syntax



Preset State: OFF

Command Example

```
OUTPUT @Mta;"FUNC1:ZOOM ON"
```

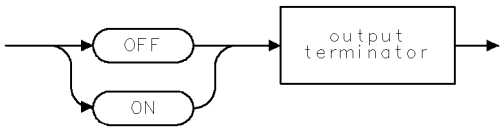
Description

The ZOOM:STATE command determines whether data from the zoom transform is used whenever a channel is referenced as a source operand. The query returns the current state of ZOOM:STATE.

Note Unlike similar commands in the CHANNEL subsystem, the formatting of this function does not affect its operation.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENON

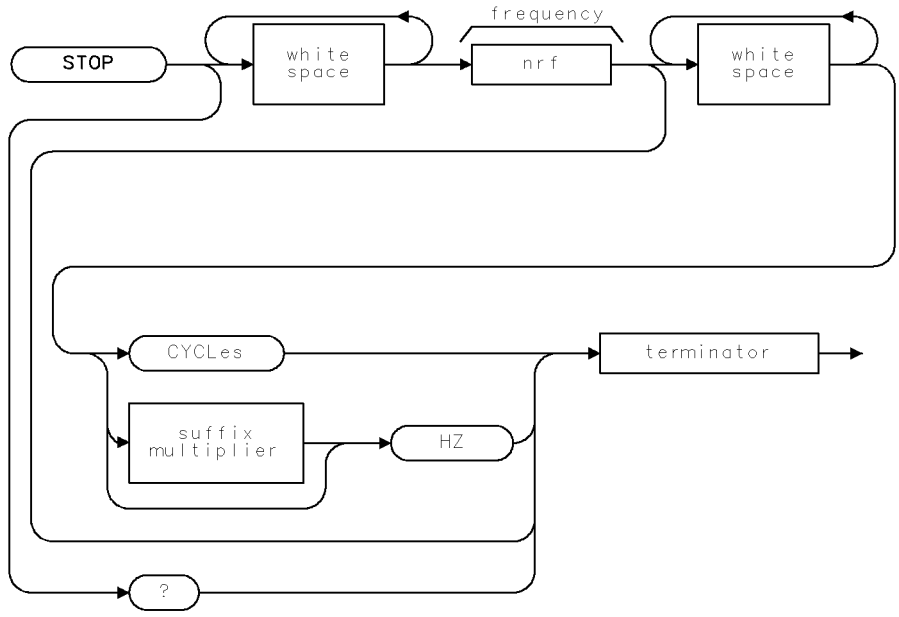
Query Example

```
OUTPUT @Mta;"FUNC1:ZOOM?"
ENTER @Mta;Zoom$
```

ZOOM:STOP

Specifies the stop frequency used for the zoom transform.

Syntax



XFUN16

Preset State:	100 MHz
Parameter Range:	frequency = 0 through 100 GHz
Fundamental Unit:	hertz
Related Commands:	CHANx:ZOOM:STOP SWEep:TIME:ZOOM:STOP

Command Example

```
OUTPUT @Mta;"FUNC1:ZOOM:STOP 10 GHz"
```

Description

The ZOOM:STOP command specifies the stop frequency used for the zoom transform. The query returns the current setting.

The STOP command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting a measurement independent of the specific signal frequency. The query always returns the value in hertz.

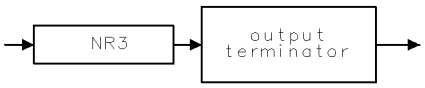
The module converts cycles to hertz by using the following equation:

$$hertz = cycles \times signal\ frequency$$

Note This command modifies the same value as the CHANx:ZOOM:STOP and SWEEP:TIME:ZOOM:STOP commands.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



QGENNR3

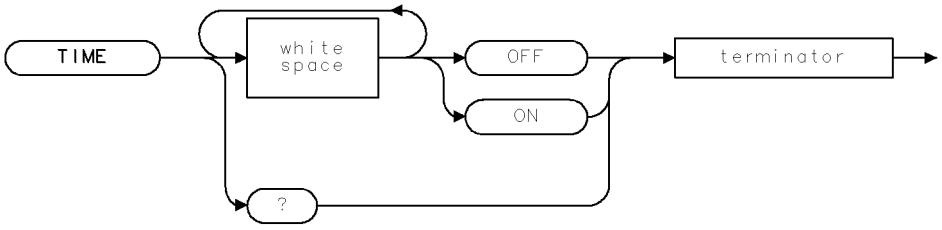
Query Example

```
OUTPUT @Mta;"FUNC1:ZOOM:STOP?"
ENTER @Mta;Stop_freq
```

ZOOM:TIME

Enables or disables the use of zoom transform data.

Syntax



Preset State: OFF

Command Example

```
OUTPUT @Mta;"FUNC2:ZOOM:TIME OFF"
```

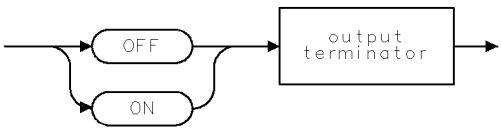
Description

The ZOOM:TIME command enables and disables the use of zoom transform data when the function is in the time domain. The query returns the current state.

For example, if ZOOM:TIME is OFF and the function is displayed in the time domain, the data will be the normal time domain data. If ZOOM:TIME is ON and the function is displayed in the time domain, the data will be the IFFT of the zoom transform data. Note that this command does not enable the zoom transform, it just determines whether or not to use the zoom transform data once the transform is enabled (see FUNCx:ZOOM:STATE).

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



Query Example

```
OUTPUT @Mta;"FUNC1:ZOOM:TIME?"
ENTER @Mta;Zoom_state
```

14

Histogram Subsystem

The HISTOGRAM subsystem can be used to measure various statistics of either the input channels or functions. The process of constructing the histogram

consists of:

- Initializing the histogram by dividing up the horizontal (or vertical) range into N buckets and setting each to 0.
- After each measurement, adding 1 to the corresponding bucket in the histogram for each point in the measurement.

The range and offset of the histogram is determined by the display scaling commands (see DISPLAY:xxx:RANGE and DISPLAY:xxx:OFFSET). For example, if DISPLAY:CHAN1:RANGE is set to 200 mV and DISPLAY:CHAN1:OFFSET is set to 0 volts, bucket 0 of the histogram would correspond to −100 mV.

The measurement data may be further qualified with a window (not to be confused with the window operation used for the FFT). This window specifies a portion of the measurement data to use. For example, when a voltage histogram is constructed, instead of using the entire measurement, a window can be defined about an event of interest (such as a positive edge), then the data outside of the window will be ignored.

After a histogram is constructed, the mean (average) and sigma (standard deviation) can be measured over any portion of the histogram.

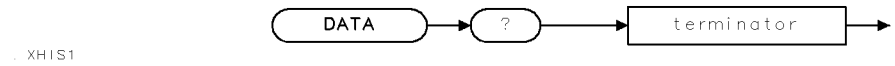
See the following list for a command syntax summary of the HISTOGRAM subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:HISTogram		
:DATA?		[query only]
:EXTRemes?		[query only]
:LLIMit	<nrf>[DEG|DB|DBM|[<suffix multiplier>]V| [<suffix multiplier>]S|[<suffix multiplier>]HZ]	
:MEAN?		[query only]
:PEAK?		[query only]
:PLL?		[query only]
:PUL?		[query only]
:REFerence	<nrf>[DEG|DB|DBM|[<suffix multiplier>]V| [<suffix multiplier>]S|[<suffix multiplier>]HZ], <nrf>[DEG|DB|DBM)|[<suffix multiplier>]V| [<suffix multiplier>]S|[<suffix multiplier>]HZ]	
:SAMPles	<nrf>	
:SIGMa?		[query only]
:SINGle		[no query]
:SOURce	CHAN<channel_number>|FUNC<function_number>	
:STARt		[no query]
:STATe	OFF|ON	
:STOP		[no query]
:ULIMit	<nrf>[DEG|DB|DBM|[<suffix multiplier>]V| [<suffix multiplier>]S|[<suffix multiplier>]HZ]	
:WINDow	X|Y,<nrf>DEG|DB|DBM|[<suffix multiplier>]V| [<suffix multiplier>]S|[<suffix multiplier>]HZ, <nrf>DEG|DB|DBM|[<suffix multiplier>]V| [<suffix multiplier>]S|[<suffix multiplier>]HZ	

DATA

Returns the normalized histogram data.

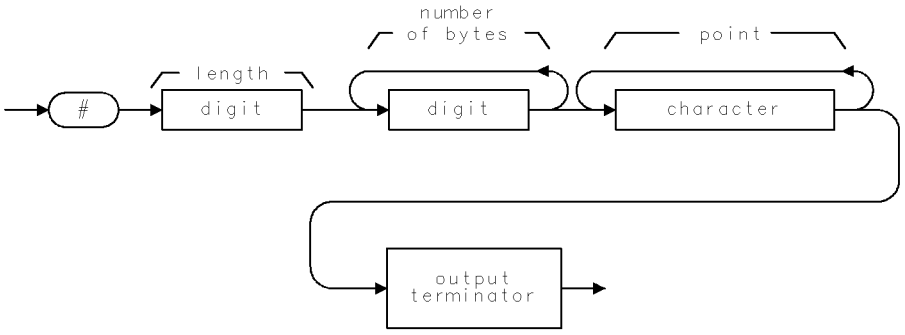
Syntax



Description

The DATA query returns the normalized histogram data. The data consists of interleaved X-Y values of 16 bits per data point. The X-Y values are normalized such that plotting the results into 4096 by 4096 space will match the display of histogram data by the microwave transition analyzer on the modular displays.

Query Response



The query response is defined as:

```
<header><point1>...<pointn><term>
```

<header>	= #<length><number of bytes>
<length>	= an ASCII digit indicating how many digits are in <N>
<number of bytes>	= an ASCII number indicating the number of bytes (8 bit) values of data following
<point>	INT16 (16 bit binary value): HP-IB: MSB first, LSB last MSIB: sent in one 16 bit packet
<term>	HP-IB: line feed with EOI asserted MSIB: END immediate command

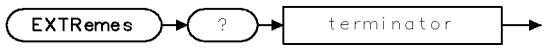
Query Example

```
DIM X(4096),Y(4096)
OUTPUT @Mta;"HIST:DATA?"
ENTER @Mta USING "#,A,A";Header$,Digit$
ENTER @Mta USING "#,"&Digit$&"D";Byte_size
Length=Byte_size/4
FOR J=0 TO Length-1
  ENTER @Mta USING "#,W,W";X(J),Y(J)
NEXT J
ENTER @Mta;Terminator$
```

EXTRemes

Returns the values of histogram buckets.

Syntax



xhis16

Description

The EXTREMES query returns the values of the first and last histogram buckets which have non-zero counts.

Query Response

qgenn5

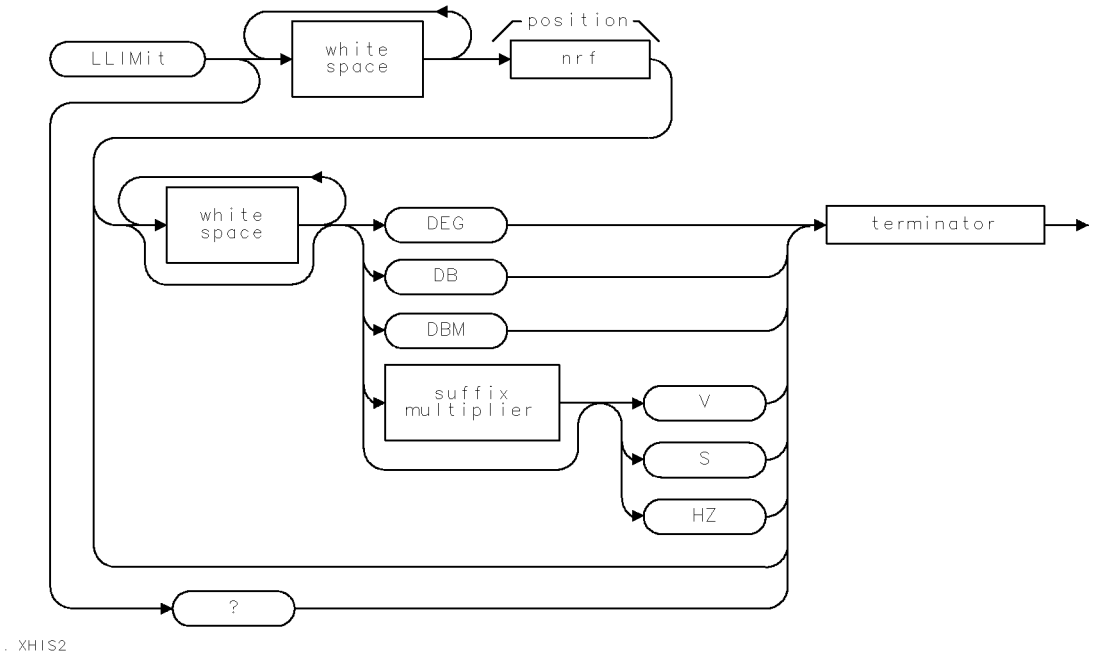
Query Example

```
OUTPUT @Mta;"HIST:EXTRemes?"
ENTER @Mta;Hist_start,Hist_end
```

LLIMit
(Lower Limit)

Sets a line marker to the specified position on a histogram.

Syntax



Preset State:	start (bottom) of range
Parameter Range:	same as measurement range (for example, full scale)
Fundamental Units:	seconds, volts, hertz, degrees, dB, or dBm

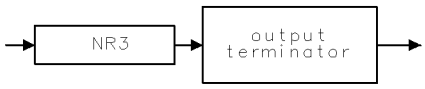
Command Example

```
OUTPUT @Mta;"HIST:LLIM .1 V"
```

Description

The LLIMIT command sets a line marker to the specified X or Y position on a histogram. The query returns the current position of the line marker.

Query Response



QGENNR3

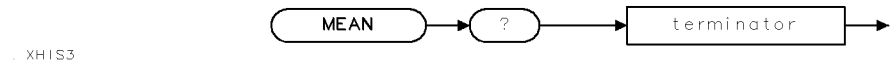
Query Example

```
OUTPUT @Mta;"HIST:LLIM?"
ENTER @Mta;Llimit
```

MEAN

Computes the mean of the histogram.

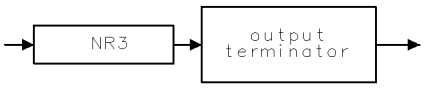
Syntax



Description

The MEAN query computes and returns the mean of the portion of the histogram defined by the HISTOGRAM:REFERENCE command.

Query Response



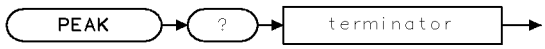
Query Example

```
OUTPUT @Mta;"HIST:MEAN?"
ENTER @Mta;Mean
```

PEAK

Returns the highest peak and lowest valley in the histogram data.

Syntax



xhis15

Description

The PEAK query returns the highest peak and lowest valley in the histogram data. The search limits are set by the histogram upper and lower math limits. This a query only.

Query Response

qgenn5

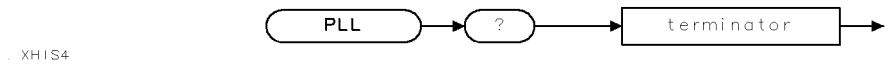
Query Example

```
OUTPUT @Mta;"HIST:PEAK?"
ENTER @Mta;Hist_highest,Hist_lowest
```

PLL
(Probability Lower Limit)

Returns the percentage of the histogram that is below the lower limit setting.

Syntax

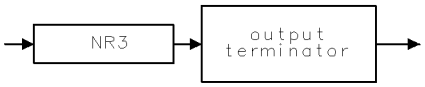


Related Commands: HISTogram:LLIMit
HISTogram:REFerence

Description

The PLL query returns the cumulative probability between the 0 percent reference and the current lower limit setting. The 0 percent and 100 percent levels of the histogram are defined by the HISTOGRAM:REFERENCE command.

Query Response



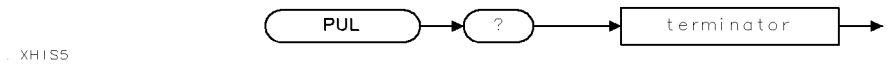
Query Example

```
OUTPUT @Mta;"HIST:PLL?"
ENTER @Mta;Lower_prob
```

PUL
(Probability Upper Limit)

Returns the percentage of the histogram that is above the upper limit setting.

Syntax

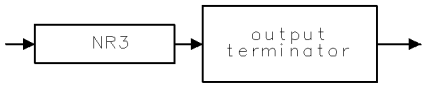


Related Commands: HISTogram:ULIMit
HISTogram:REFerence

Description

The PUL query returns the cumulative probability between 0 percent reference and the current upper limit setting. The 0 percent and 100 percent levels of the histogram are defined by the HISTOGRAM:REFERENCE command.

Query Response



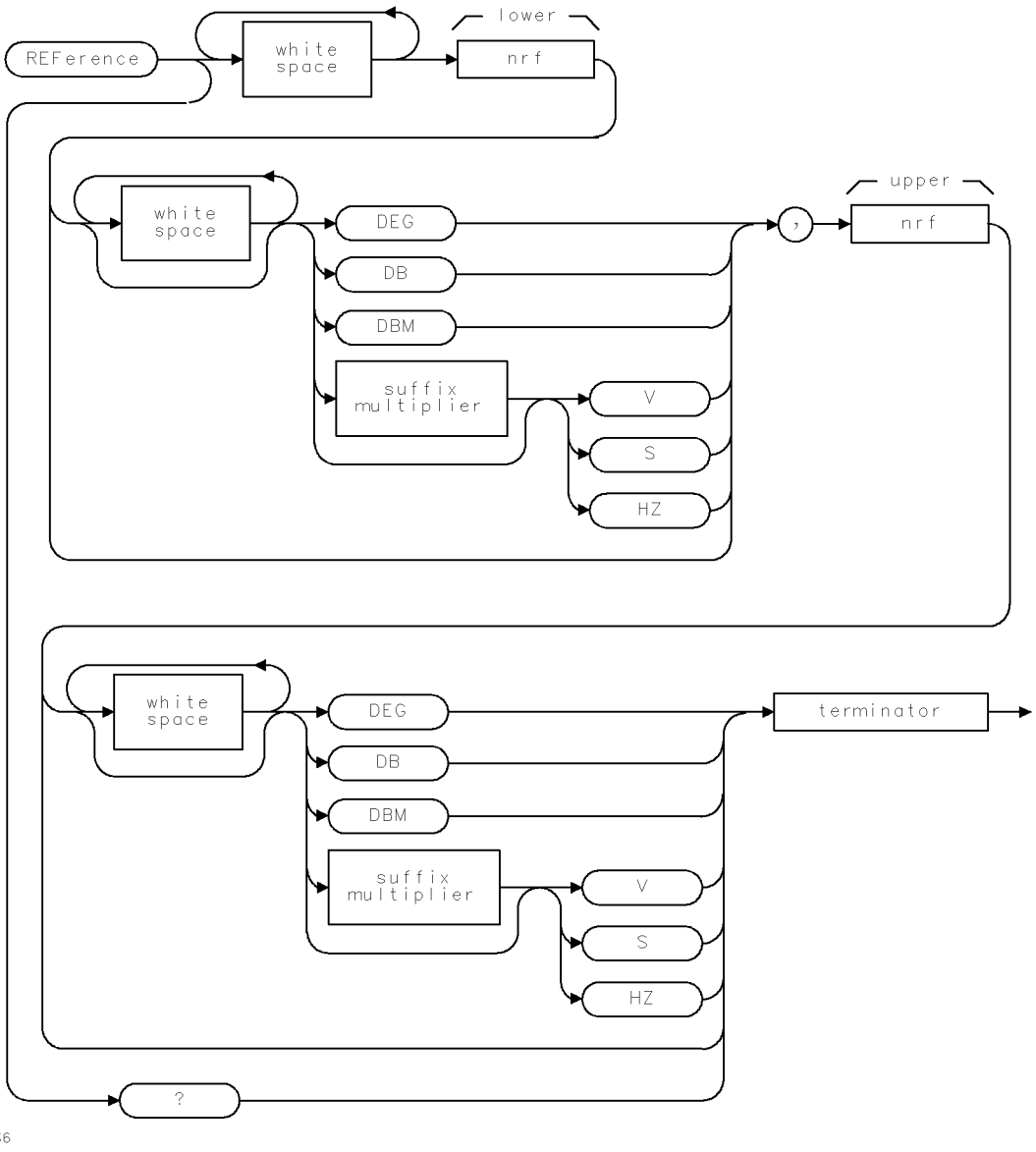
Query Example

```
OUTPUT @Mta;"HIST:PUL?"
ENTER @Mta;Upper_prob
```

REFerence

Defines the portion of the histogram used for all other histogram measurements.

Syntax



Preset State: full scale
Parameter Range: same as measurement range (for example, full scale)
Fundamental Units: seconds, volts, hertz, degrees, dB, or dBm

Command Example

```
OUTPUT @Mta;"HIST:REF 0 V,10 mV"
```

Description

The REFERENCE command defines the portion of the histogram that is to be considered for the rest of the histogram measurement operations (that is, this command defines the X or Y range that is 0 percent to 100 percent). The query returns the current state of REFERENCE.

Query Response

qgenn5

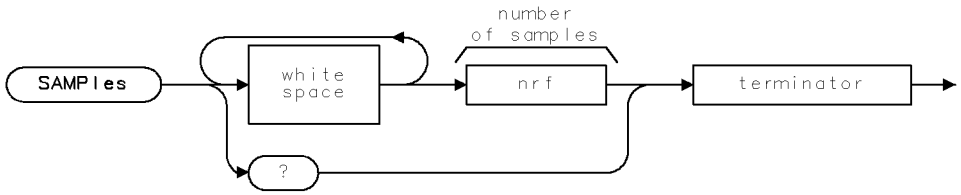
Query Example

```
OUTPUT @Mta;"HIST:REF?"
ENTER @Mta;Lower_ref,Upper_ref
```

SAMPles

Defines the minimum number of data samples required to complete a histogram.

Syntax



Preset State: 1000
Parameter Range: samples = 1 through 500E6

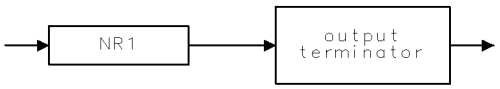
Command Example

```
OUTPUT @Mta;"HIST:SAMP 5000"
```

Description

The SAMPLES command defines the minimum number of data samples that are required to complete a histogram. The query returns the current setting of SAMPLES.

Query Response



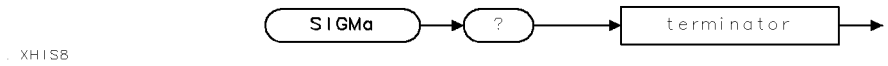
Query Example

```
OUTPUT @Mta;"HIST:SAMP?"
ENTER @Mta;Samples
```

SIGMa

Computes the standard deviation of the histogram.

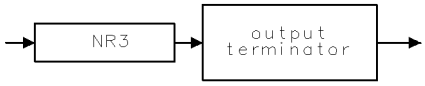
Syntax



Description

The SIGMA query computes and returns the sigma (standard deviation) of the portion of the histogram defined by the HISTOGRAM:REFERENCE command.

Query Response



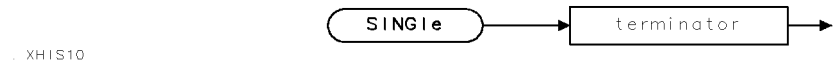
Query Example

```
OUTPUT @Mta;"HIST:SIGM?"
ENTER @Mta;Sigma
```

SINGle

Starts the accumulation for a single histogram measurement.

Syntax



Command Example

```
OUTPUT @Mta;"HIST:SING"
```

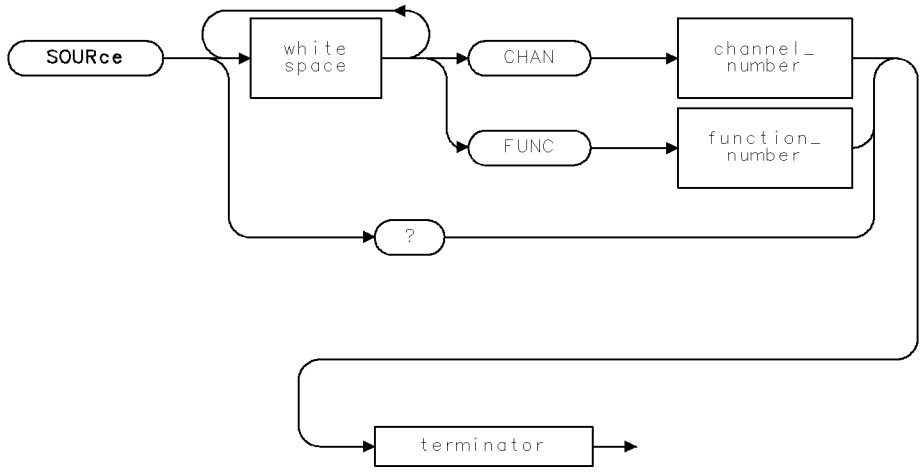
Description

The SINGLE command starts the accumulation of data into the histogram. Unlike HISTOGRAM:START, Once enough data has been accumulated to satisfy the histogram, the accumulation of data will stop, regardless of the state of SWEEP:MODE.

SOURce

Defines the histogram data input.

Syntax



. XHIS9 .

Preset State: FUNC1;

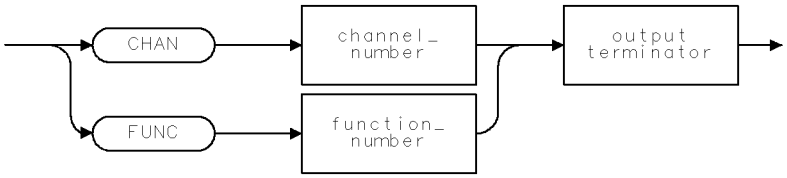
Command Example

```
OUTPUT @Mta;"HIST:SOUR CHAN2"
```

Description

The SOURCE command defines the data input to the histogram. The query returns the current state of SOURCE.

Query Response



. QHIS9 .

Query Example

```
OUTPUT @Mta;"HIST:SOUR?"
ENTER @Mta;Source$
```

STARt

Starts the accumulation of data into the histogram.

Syntax

Command Example

```
OUTPUT @Mta;"HIST:STAR"
```

Description

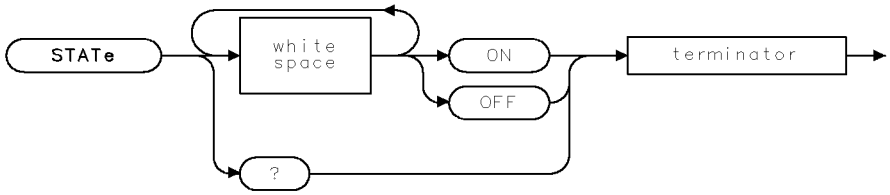
The START command starts the accumulation of data into the histogram. If the microwave transition analyzer is in single sweep mode (SWEEP:MODE = single), enough measurements will be made to satisfy the histogram. If the microwave transition analyzer is in continuous sweep mode (SWEEP:MODE = continuous), the histogram will be continuously updated.

Note For a single update of the histogram, regardless of the setting of SWEEP:MODE, use HISTOGRAM:SINGLE.

STATe

Turns histogram on or off.

Syntax



xhis17

Preset State: OFF

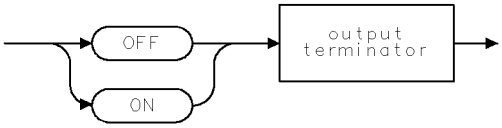
Command Example

```
OUTPUT @Mta;"HIST:STAT ON"
```

Description

The STATE command turns the histogram display on or off. There must be valid histogram data to successfully turn the histogram ON. The query returns the current value of STATE.

Query Response



QGENON

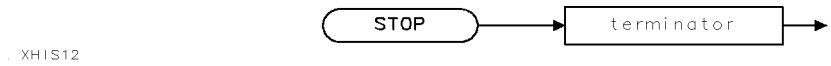
Query Example

```
OUTPUT @Mta;"HIST:STAT?"
ENTER @Mta;State$
```

STOP

Stops the accumulation of data into the histogram.

Syntax



Command Example

```
OUTPUT @Mta;"HIST:STOP"
```

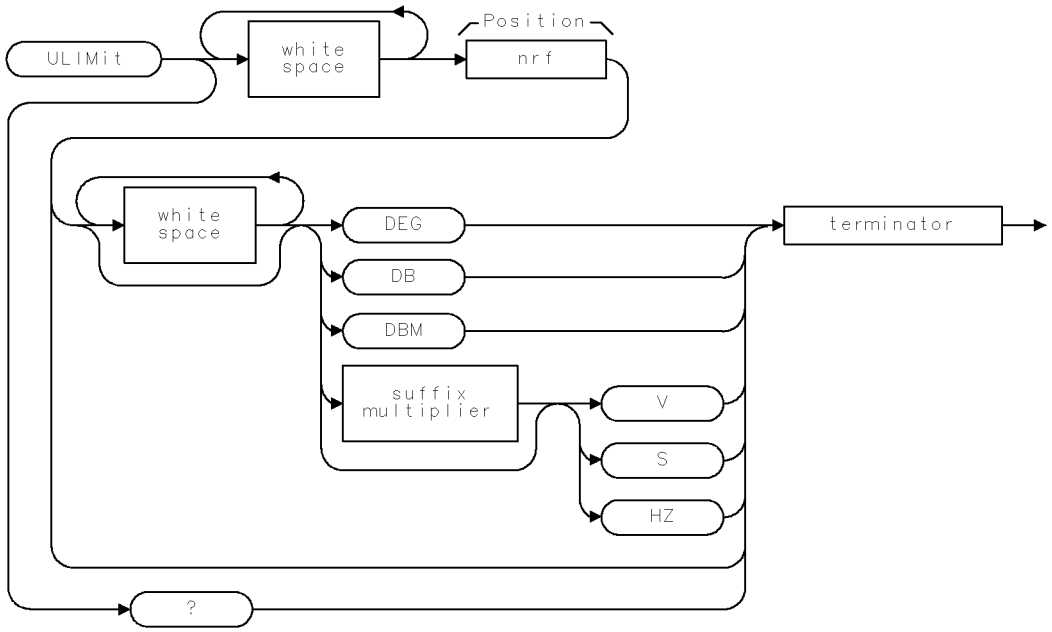
Description

The STOP command stops the accumulation of data into the histogram. If there has been any data accumulated before the STOP command was received, the histogram will be updated.

ULIMit
(Upper Limit)

Sets a line marker to a specified position on a histogram.

Syntax



. XHIS13

Preset State: start (bottom) of range
Parameter Range: same as measurement range (for example, full scale)
Fundamental Units: seconds, volts, hertz, degrees, dB, or dBm

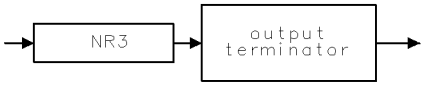
Command Example

```
OUTPUT @Mta;"HIST:ULIM .1 V"
```

Description

The ULIMIT command sets a line marker to the specified X or Y position on a histogram. The query returns the current position of the line marker.

Query Response



QGENNR3

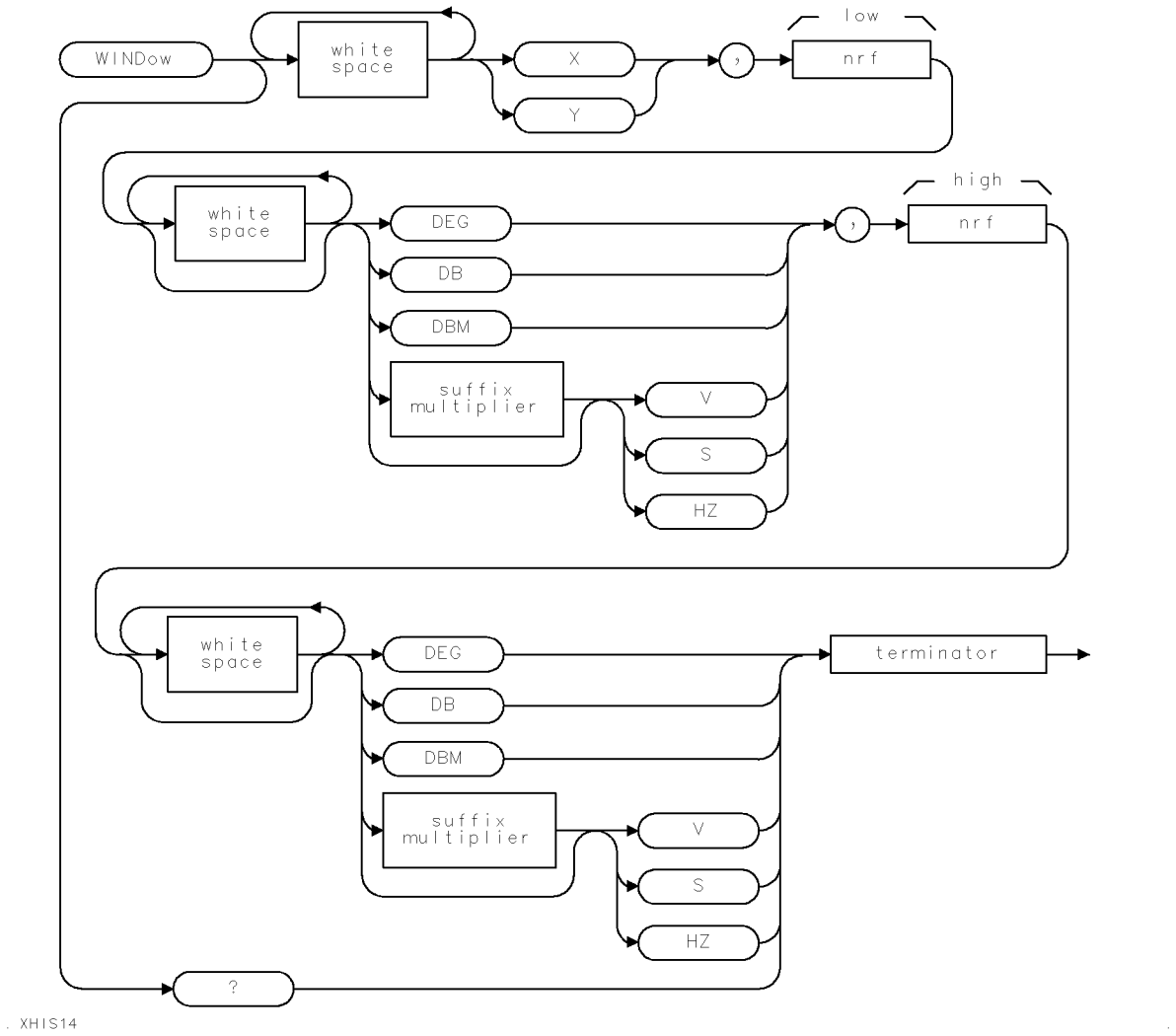
Query Example

```
OUTPUT @Mta;"HIST:ULIM?"
ENTER @Mta;Ulimit
```

WINDow

Defines the type of histogram and measurement portion that contributes to the histogram.

Syntax



Preset State:	horizontal range if type = Y; vertical range if type = X
Parameter Range:	type = Y; <low>, <high> full scale
Fundamental Units:	seconds, volts, hertz, degrees, dB, or dBm

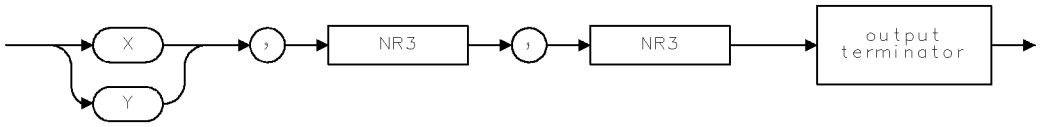
Command Example

```
OUTPUT @Mta;"HIST:WIND Y,-10 ns, 10 ns"
```

Description

The WINDOW command defines the type of histogram (that is, X or Y position) and the portion of the measurement that will contribute to the histogram. The query returns the current setting WINDOW.

Query Response



. QHIS14

Query Example

```
OUTPUT @Mta;"HIST:WIND?"
ENTER @Mta;Type$,Start,Finish
```

15

LIMIT Subsystem

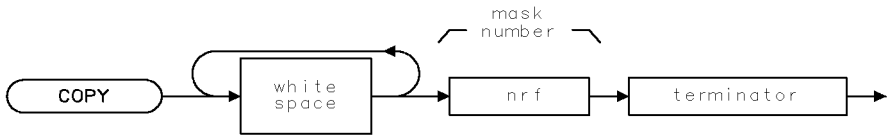
The LIMit command controls the limit lines and masks that are displayed on the screen. See the following list for a command syntax summary of the LIMIT subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:LIMit		
:COPY	<nrf>	
:COUNt	DOTs|TRACe	
:DATA		
:ASCii	{X,Y,}	
:DEFault	HEXagon|ITRIangle|SQUARe|TRIangle	[no query]
:DELete	ALL|<nrf>	[no query]
:ERRor		
:COUNt	<nrf>	
:STOP	OFF|ON	
:TRACe	FUNCx|OFF	
:FAIL?	(ANY|<nrf>)	[query only]
:FCOunt?	<nrf>	[query only]
:RESCale	<nrf>,<nrf>,<nrf>,<nrf>,<nrf>,<nrf>	[no query]
:SELect	<nrf>	
:SHOW	OFF|ON	
:SOURce	CHANx|FUNCx|WMEMx	
:TEST	OFF|ON	
:TRACe		
:COUNt	<nrf>	
:STOP	OFF|ON	
:TYPE	LOWer|POLYgon|UPper	
:X		
:OFFSet	<nrf>	
:SCALe	<nrf>	
:XY		
:SCALe	<nrf>	
:Y		
:OFFSet	<nrf>	
:SCALe	<nrf>	

COPY

Copies the mask.

Syntax



xlim19

Preset State: none

Command Example

```
OUTPUT @Mta;"LIMit:COPY 3"
```

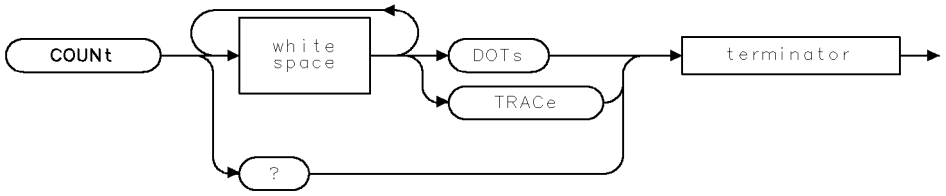
Description

The COPY command copies the mask specified by the input parameter to the currently selected mask.

COUNt

Sets count mode.

Syntax



Preset State: TRACe

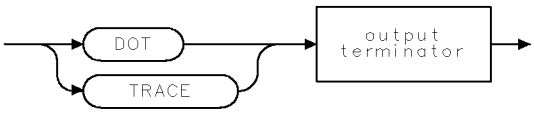
Command Example

```
OUTPUT @Mta;"LIMit:COUN DOT"
```

Description

The COUNT command sets the mask acquisition count mode. In DOTS mode, the individual trace points which violate the mask are counted separately for the mask hit count. In TRACE mode, the hit count is the number of measurement traces which have violated the mask. The query returns the current COUNT mode.

Query Response



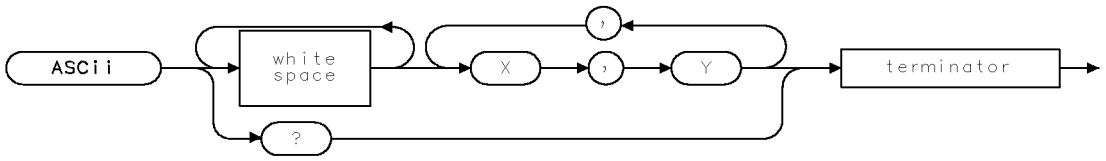
Query Example

```
OUTPUT @Mta;"LIMit:COUN?"
ENTER @Mta;Mask_count$
```

DATA:ASCii

Downloads and uploads mask data points.

Syntax



xlim17

Preset State: none

Command Example

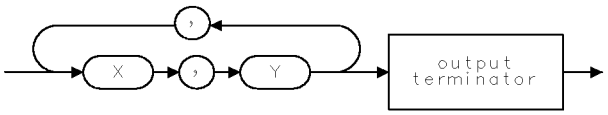
```
OUTPUT @Mta;"LIMit:DATA:ASC 0.0,-0.5,0.5,-0.25,0.5,0.25,0.0,0.5,-0.5,0.25,
-0.5,-0.25"
```

Description

The DATA:ASCII command is used for downloading and uploading mask data points. The format for downloading points is a series of X,Y coordinate pairs for each vertex. The initial scaling is assumed to be normalized such that 1.0 corresponds to full screen width/height.

The query returns the currently selected mask's vertecies in the same format as for downloading.

Query Response



qlim17

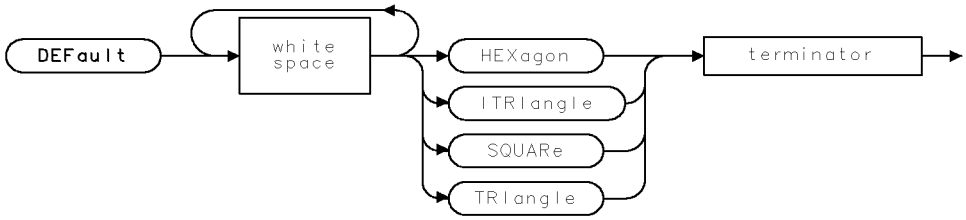
Query Example

```
OUTPUT @Mta;"LIMit:DATA:ASC?"
ENTER @Mta;Mask_data$
```

DEFault

Selects mask type.

Syntax



Preset State: none

Command Example

```
OUTPUT @Mta;"LIMit:DEF HEX"
```

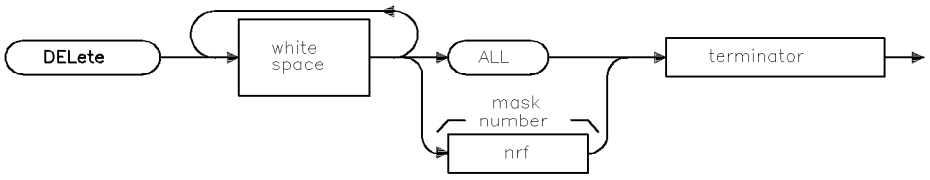
Description

The DEFAULT command selects one of the default masks and loads it into the currently selected mask.

DELete

Clears one or all of the mask definitions.

Syntax



Preset State: none

Command Example

```
OUTPUT @Mta;"LIMit:DEL 3"
```

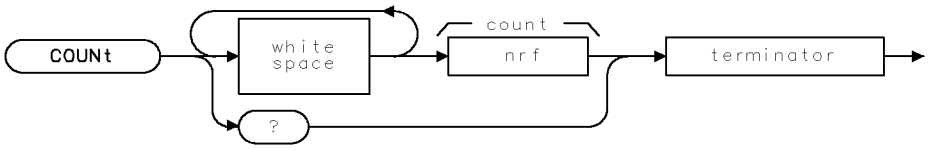
Description

The DELETE command clears one or all of the mask definitions depending upon the parameter given.

ERRor:COUNt

Sets violation number.

Syntax



xlim15

Preset State: 10000
Parameter Range: 0 through 10000

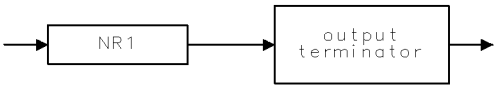
Command Example

```
OUTPUT @Mta;"LIMit:ERR:COUN 124"
```

Description

The ERROR:COUNT command sets the number of mask violations which are allowed to occur before acquisition is stopped. A count value of 10000 will disable stopping (it is the same as an infinite count). The query returns the current setting of the ERROR:STOP count.

Query Response



QGENN1

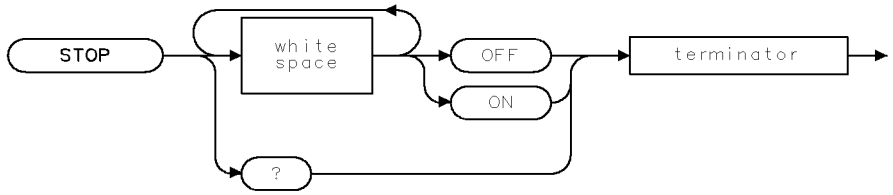
Query Example

```
OUTPUT @Mta;"LIMit:ERR:COUN?"
ENTER @Mta;Mask_ercnt$
```

ERRor:STOP

Controls acquisition stop.

Syntax



Preset State: OFF

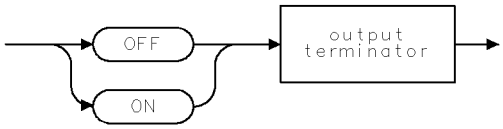
Command Example

```
OUTPUT @Mta;"LIMit:ERR:STOP ON"
```

Description

The ERROR:STOP command controls acquisition stop on error count. When ERROR:STOP is turned ON, measurements stop when the number of mask violations for any mask exceeds the count. The query returns the current state of the ERROR:STOP flag.

Query Response



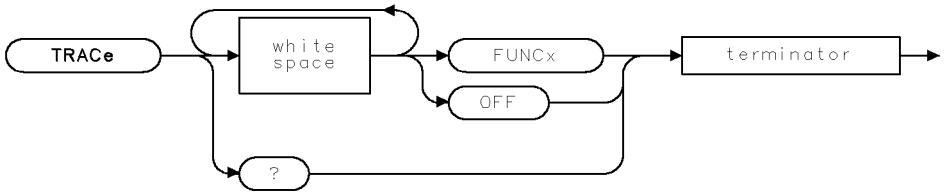
Query Example

```
OUTPUT @Mta;"LIMit:ERR:STOP?"
ENTER @Mta;Mask_estop$
```

ERRor:TRACe

Controls trace display.

Syntax



Preset State: OFF

Command Example

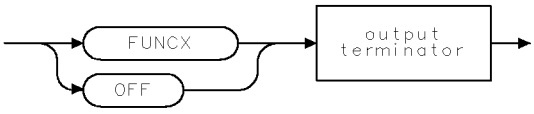
```
OUTPUT @Mta;"LIMit:ERR:TRAC FUNC2"
```

Description

The ERROR:TRACE command controls the display of the error capture trace.

When ERROR:TRACE is assigned to a FUNCTION, any measurement trace which violates a limit line will be copied to a second trace window for easier evaluation. The query returns the current state of the ERROR:TRACE assignment.

Query Response



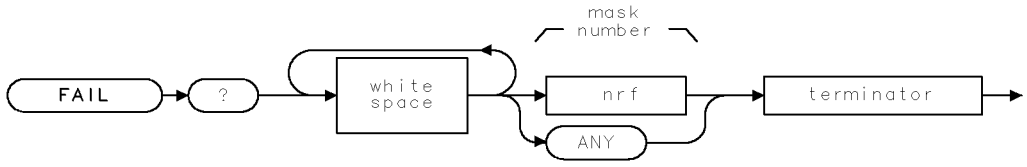
Query Example

```
OUTPUT @Mta;"LIMit:ERR:TRAC?"
ENTER @Mta;Mask_etrac$
```

FAIL

Returns violation status.

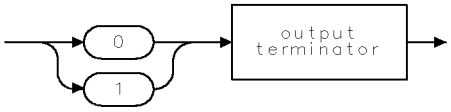
Syntax



Description

The FAIL query returns the status of mask violations. Violations are returned for the mask number given as an argument or for any mask if ANY is specified as an argument. A return value of 1 indicates that the requested mask had at least one violation.

Query Response



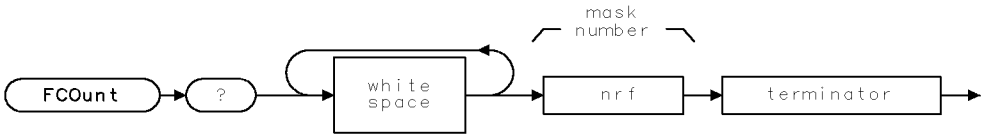
Query Example

```
OUTPUT @Mta;"LIMit:FAIL? ANY"
ENTER @Mta;Mask_fail$
```

FCOunt

Returns the number of violations.

Syntax

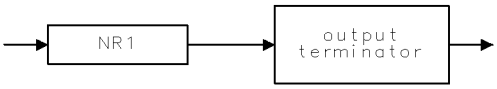


xlim11

Description

The FCOUNT query returns the number of violations for the mask given in the argument.

Query Response



QGENN1

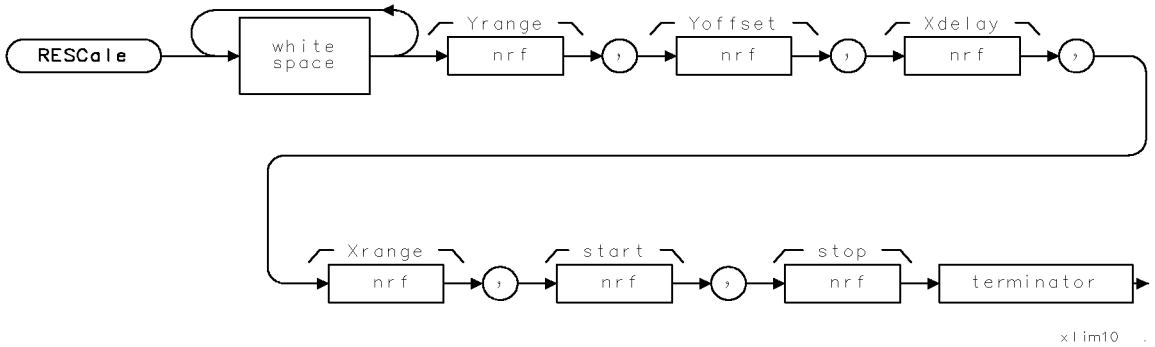
Query Example

```
OUTPUT @Mta;"LIMit:FCO? 6"
ENTER @Mta;Mask_fcount
```

RESCale

Changes limit scaling.

Syntax



Preset State: none

Command Example

```
OUTPUT @Mta;"LIMit:RESCALE 13.6E-3,1.5E-3,257.2E-9,100.0E-9,-150.0E-9,150.0E-9"
```

Description

The RESCALE command changes scaling of the currently selected mask. The definition of input parameters are as follows:

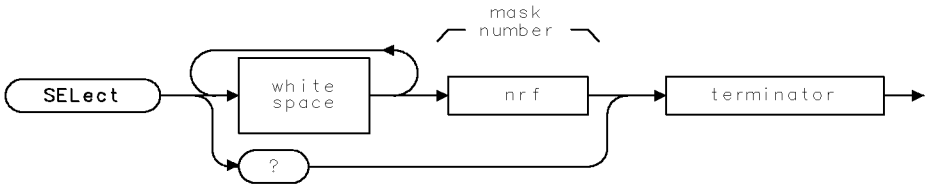
- Yrange is the new Y range for the mask data. The normalized mask data values will be rescaled such that Yrange represents 1.0.
- Yoffset is the mask Y offset that will be set to this value.
- Xdelay is the mask X offset will be set to delay - (Xrange/2). This is useful when trying to match the mask to a periodic signal.
- Xrange would be set to the signal period, delay would be set to a zero crossing value. The mask will then be centered on one signal period. Xrange is the new X range for the mask data. The normalized mask data values will be rescaled such that Xrange represents 1.0. The start parameter is the starting limit for X (usually X axis screen edge). The stop parameter is the end limit for X (usually X axis screen edge).
- Start and stop are used in conjunction with Xrange to try to keep the mask on screen by adding or subtracting a period from the delay if the mask exceeds the limits.

The mask scale factors are set to 1.0 so that further scaling will be with respect to the values given to the RESCALE command.

SELect

Sets the current mask.

Syntax



Preset State: 1
Parmeter Range: 1 through 8

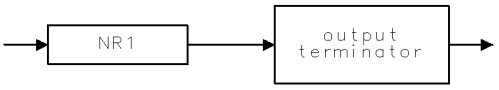
Command Example

```
OUTPUT @Mta;"LIMit:SEL 2"
```

Description

The SELECT command sets the current mask on which operations are to be performed. The query returns the currently selected mask number.

Query Response



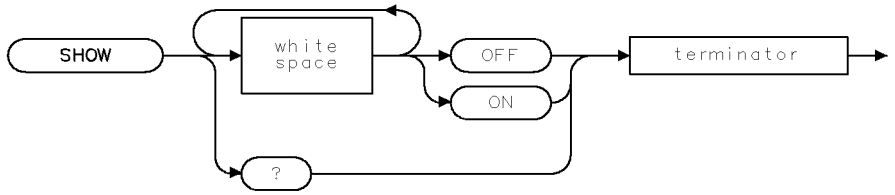
Query Example

```
OUTPUT @Mta;"LIMit:SEL?"
ENTER @Mta;Mask_current$
```

SHOW

Turns the mask display on or off.

Syntax



Preset State: OFF

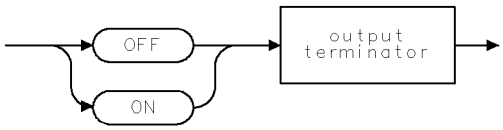
Command Example

```
OUTPUT @Mta;"LIMit:SHOW ON"
```

Description

The SHOW command turns ON and OFF the mask display. The query returns the current state of the SHOW flag.

Query Response



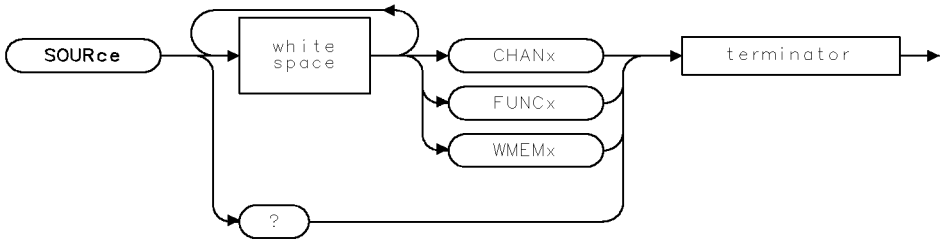
Query Example

```
OUTPUT @Mta;"LIMit:SHOW?"
ENTER @Mta;Mask_show$
```

SOURce

Sets source trace for comparisons.

Syntax



Preset State: FUNC1

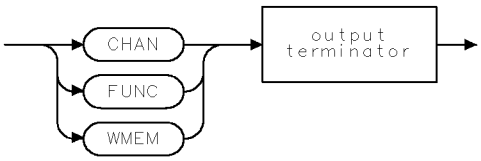
Command Example

```
OUTPUT @Mta;"LIMit:SOUR FUNC2"
```

Description

The SOURCE command sets source trace for mask comparisons. The query returns the current mask source trace name.

Query Response



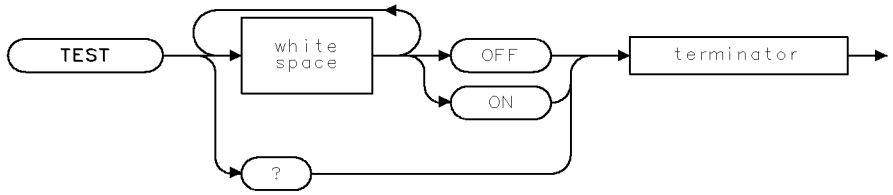
Query Example

```
OUTPUT @Mta;"LIMit:SOUR?"
ENTER @Mta;Mask_source$
```

TEST

Controls mask test acquisition.

Syntax



Preset State: OFF

Command Example

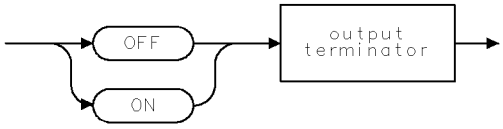
```
OUTPUT @Mta;"LIMit:TEST ON"
```

Description

The TEST command controls mask test acquisition. When TEST is turned ON, each measurement trace is compared to the mask to determine if the mask has been violated.

The query returns the current state of the TEST flag.

Query Response



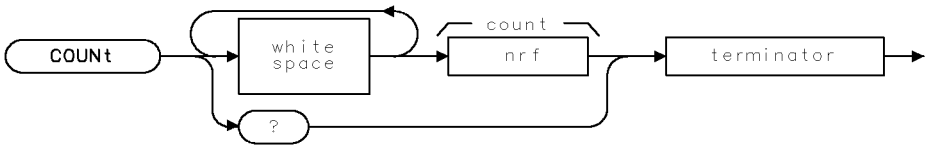
Query Example

```
OUTPUT @Mta;"LIMit:TEST?"
ENTER @Mta;Mask_tst$
```

TRACe:COUNt

Sets the number of measurement traces.

Syntax



Preset State: infinite (10000)
Parameter Range: $1 \leq$ count
Values greater than 9999 select infinite

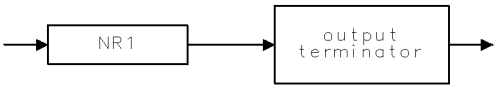
Command Example

OUTPUT @Mta;“LIMit:TRAC:COUN 124”

Description

The TRACE:COUNT command sets the number of measurement traces which will be taken before acquisition is stopped. A count value of 10000 will disable stopping (it is the same as an infinite count). The query returns the current setting of the TRACE:STOP count.

Query Response



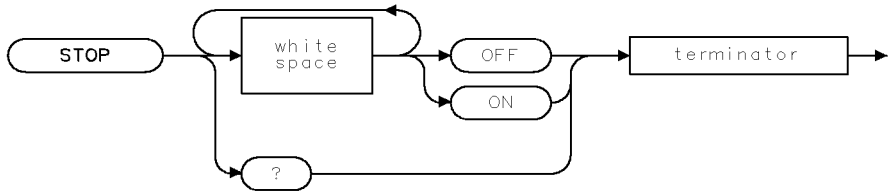
Query Example

```
OUTPUT @Mta;"LIMit:TRAC:COUN?"
ENTER @Mta;Mask_trcnt$
```

TRACe:STOP

Controls acquisition stop.

Syntax



Preset State: OFF

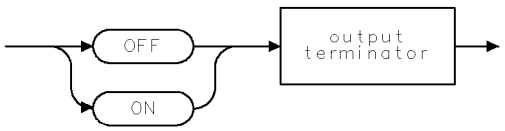
Command Example

```
OUTPUT @Mta;"LIMit:TRAC:STOP ON"
```

Description

The TRACE:STOP command controls acquisition stop on traces count. When TRACE:STOP is turned ON, measurements stop when the number of measurements exceeds the count. The query returns the current state of the TRACE:STOP flag.

Query Response



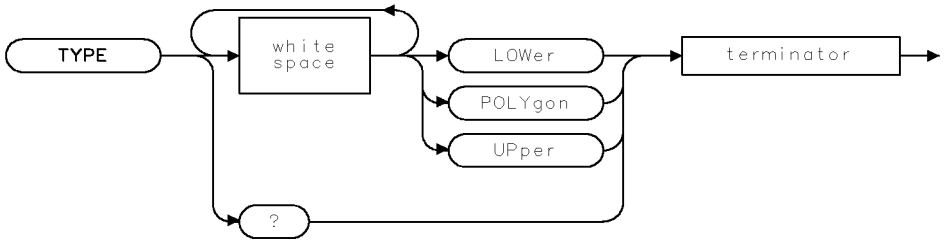
Query Example

```
OUTPUT @Mta;"LIMit:TRAC:STOP?"
ENTER @Mta;Mask_tstop$
```

TYPE

Controls the function of the mask or limit line.

Syntax



Preset State UNDEFINED

Command Example

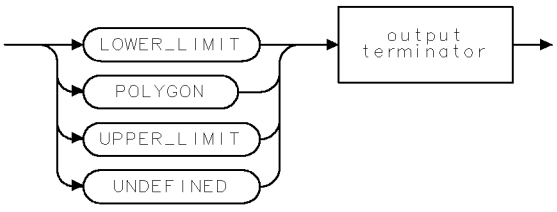
```
OUTPUT @Mta;"LIMit:TYPE:POLY"
```

Description

The TYPE command controls the function of the mask or limit line. When TYPE is UPPER or LOWER, the selected mask or limit line is interpreted as a limit line. TYPE POLYGON is interpreted as a two dimensional mask.

The query return the TYPE of the currently selected mask or limit line.

Query Response



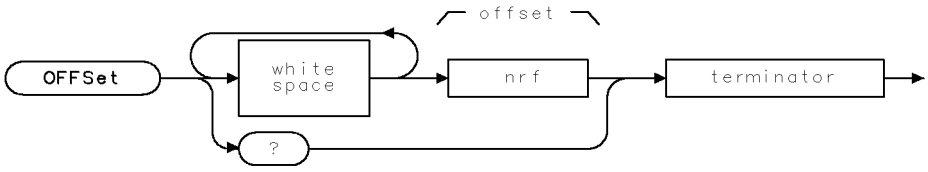
Query Example

```
OUTPUT &Mta;"LIMit:TYPE?"
ENTER @Mta;Limit_type$
```

X:OFFSet

Adjusts the X dimension offset.

Syntax



Preset State: none
Parameter Range: −1E6 through 1E6

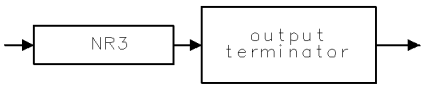
Command Example

```
OUTPUT @Mta;"LIMit:X:OFFS 1.23E-9"
```

Description

The X:OFFSET command adjusts the mask X dimension offset of the currently selected mask. The query returns the value current mask's X:OFFSET.

Query Response



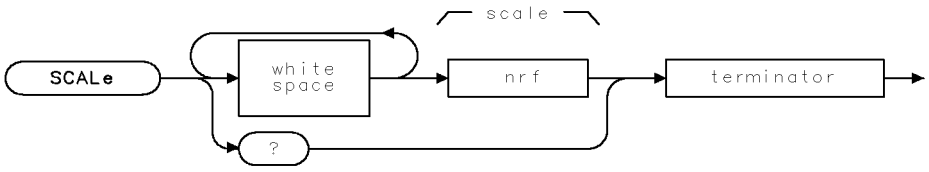
Query Example

```
OUTPUT @Mta;"LIMit:X:OFFS?"
ENTER @Mta;Mask_xoff$
```

X:SCALe

Adjusts the X dimension scaling.

Syntax



xlim1

Preset State: none
Parameter Range: −1E6 through 1E6

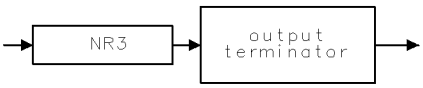
Command Example

```
OUTPUT @Mta;"LIMit:X:SCAL 1.24"
```

Description

The X:SCALE command adjusts the mask X dimension scaling of the currently selected mask. The query returns the value current mask's X:SCALE.

Query Response



QGENNR3

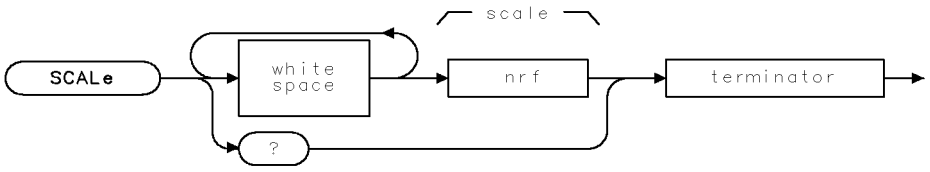
Query Example

```
OUTPUT @Mta;"LIMit:X:SCAL?"
ENTER @Mta;Mask_xscal$
```

XY:SCALe

Adjusts the dimension scaling.

Syntax



Preset State: none
Parameter Range: −1E6 through 1E6

Command Example

```
OUTPUT @Mta;"LIMit:XY:SCAL 1.24"
```

Description

The XY:SCALE command adjusts the mask dimension scaling for both X and Y of the currently selected mask. The query returns the value current mask's X and Y scale factors.

Query Response

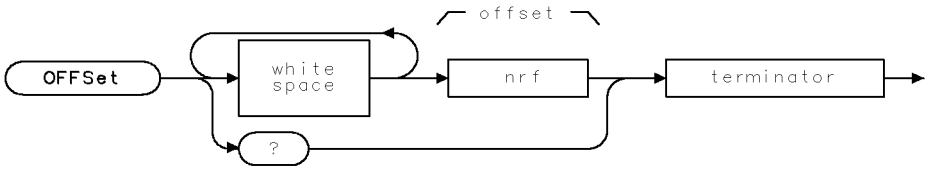
Query Example

```
OUTPUT @Mta;"LIMit:XY:SCAL?"
ENTER @Mta;Mask_scal$
```

Y:OFFSet

Adjusts the Y dimension offset.

Syntax



Preset State: none
Parameter Range: −1E6 through 1E6

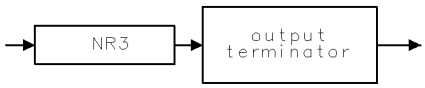
Command Example

```
OUTPUT @Mta;"LIMit:Y:OFFS 1.42E-3"
```

Description

The Y:OFFSET command adjusts the mask Y dimension offset of the currently selected mask. The query returns the value current mask's Y:OFFSET.

Query Response



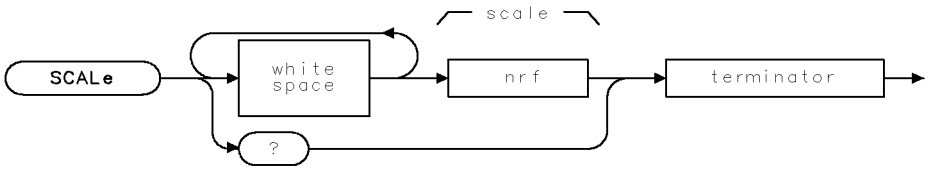
Query Example

```
OUTPUT @Mta;"LIMit:Y:OFFS?"
ENTER @Mta;Mask_yoff$
```

Y:SCALe

Adjusts the Y dimension scaling.

Syntax



Preset State: none
Parameter Range: −1E6 through 1E6

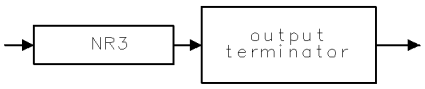
Command Example

```
OUTPUT @Mta;"LIMit:Y:SCAL 0.38"
```

Description

The Y:SCALE command adjusts the mask Y dimension scaling of the currently selected mask. The query returns the value current mask's Y:SCALE.

Query Response



Query Example

```
OUTPUT @Mta;"LIMit:Y:SCAL?"
ENTER @Mta;Mask_yscal$
```

16

Marker Subsystem

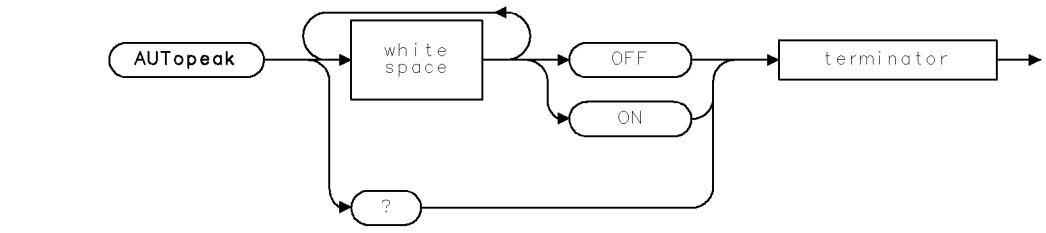
The MARKER subsystem is used to control the two cursor markers in the microwave transition analyzer. See the following list for a command syntax summary of the MARKER subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:MARK<marker_number>		
:AUTopeak	OFF|ON	
:CENTer		[no query]
:HOLD	OFF|ON	
:LOWest?		[query only]
:LPEAk?		[query only]
:NEXT?		[query only]
:PEAK?		[query only]
[:POSition]	<nrf>[[<suffix multiplier>]S| [<suffix multiplier>]HZ]	
:REFerence		[no query]
:SOURce	OFF|CHAN<channel_number>| WMEM<memory_number>| FUNC<function_number>	
:MARK		
:BOTH	<nrf>DB|DBM|DEG|[<suffix multiplier>]V| [<suffix multiplier>]HZ|[<suffix multiplier>]S	[no query]
:DCPeak	OFF|ON	
:DELTa		
[:POSition]	<nrf>	
:STATe	OFF|ON	
:FORmat	SCALar|VECtor	
:INTerpolate	OFF|ON	
:OFF		[no query]
:READout	AUTo|LINear|LOGarithmic,DEGrees|RADian	

AUTopeak

Turns the end-of-sweep marker peak search on or off.

Syntax



Preset State: OFF

Command Example

```
OUTPUT @Mta;"MARK1:AUT ON"
```

Description

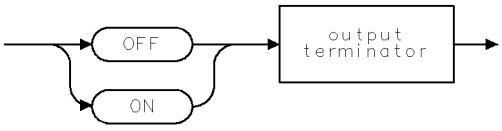
The AUTOPEAK command turns the end-of-sweep-marker peak search on or off. The query returns the current state.

The specified marker is moved to the peak of the trace. If the non-specified marker is placed on a different trace than the specified marker, and the trace formats are identical, then the non-selected marker's horizontal position tracks the selected marker's. Otherwise, the non-selected marker is unaffected.

If the specified marker is off, the marker is placed on the lowest numbered displayed trace (FUNCx). If no trace is currently displayed, trace 1 is displayed and the specified marker is placed on it.

Turning AUTOPEAK on causes a peak search.

Query Response



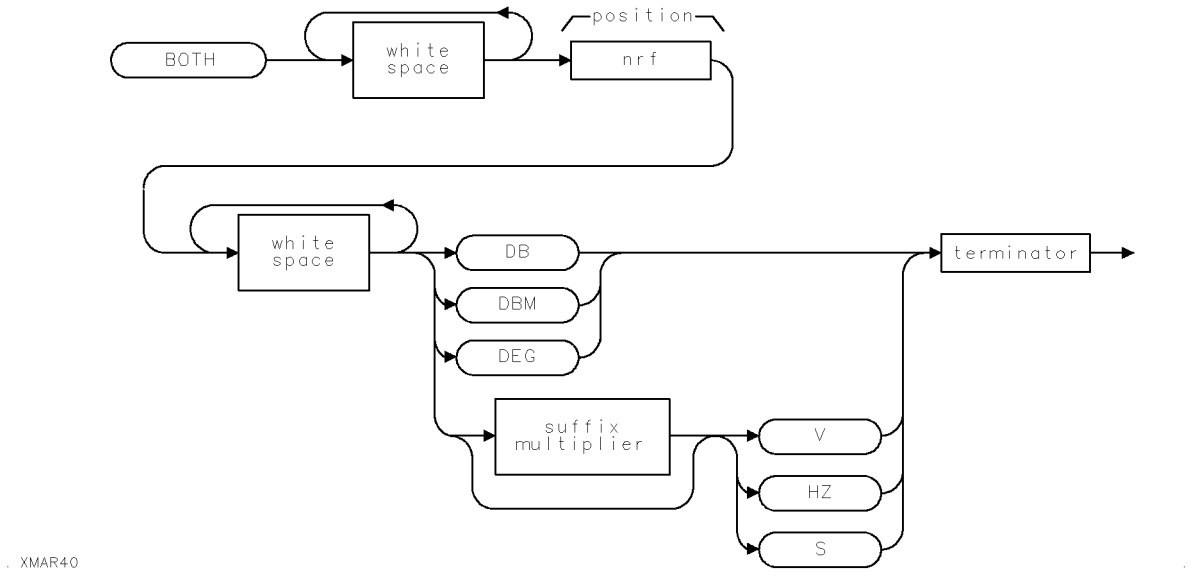
Query Example

```
OUTPUT @Mta;"MARK1:AUT?"
ENTER @Mta;Autopeak$
```

BOTH

Sets the position of both markers.

Syntax



Fundamental Units: seconds, volts, hertz, degrees, dB, or dBm

Command Example

```
OUTPUT @Mta;"MARK:BOTH 150 ps"
```

Description

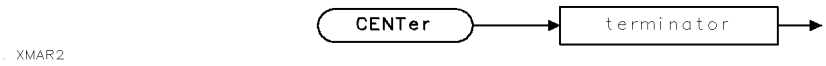
The BOTH command sets the position of both markers. Marker number 1 is placed at the given position and marker number 2 is moved to the same relative position to marker 1.

Note There is no query for this command. Therefore, the normal MARKx:POSITION queries should be used.

CENTer

Moves the selected marker to the center of the display.

Syntax



Command Example

```
OUTPUT @Mta;"MARK1:CENT"
```

Description

The CENTER command causes the selected marker to be moved to the center of the display. The actual function performed depends on the trace domain, marker display domain, and whether zoom transform is on as shown in the table below.

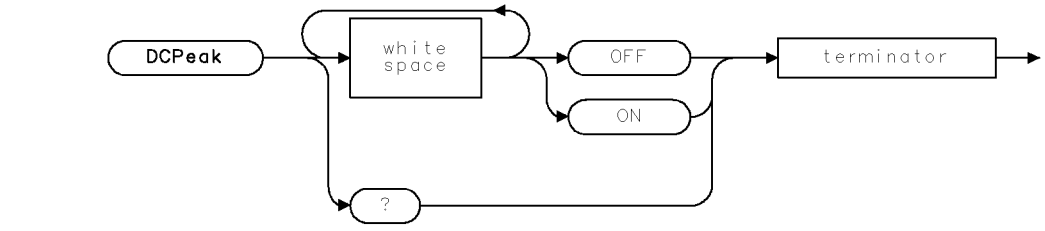
The CENTER command affects settings as shown in this table:

Trace Type (Measure Domain)	Cursor Domain (Display Domain)	Setting Adjusted
Zoom Transform	time domain	Delay
Zoom Transform	frequency domain	Zoom Center Frequency
time_domain	time domain	Delay
time_domain	frequency domain	Seconds/Division
freq_domain	time domain	Not Applicable
freq_domain	frequency domain	Center Frequency

DCPeak

Enables or disables dc for marker peak search.

Syntax



Preset State: ON

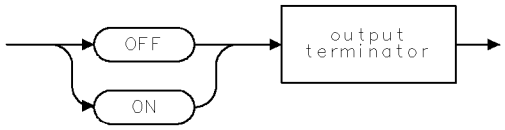
Command Example

```
OUTPUT @Mta;"MARK:DCP OFF"
```

Description

The DCPEAK command enables or disables a mode where marker peak searches of frequency domain data will include the first point of data (normally dc) and the last point of data (normally Nyquist). The query returns the current setting of DCPEAK.

Query Response



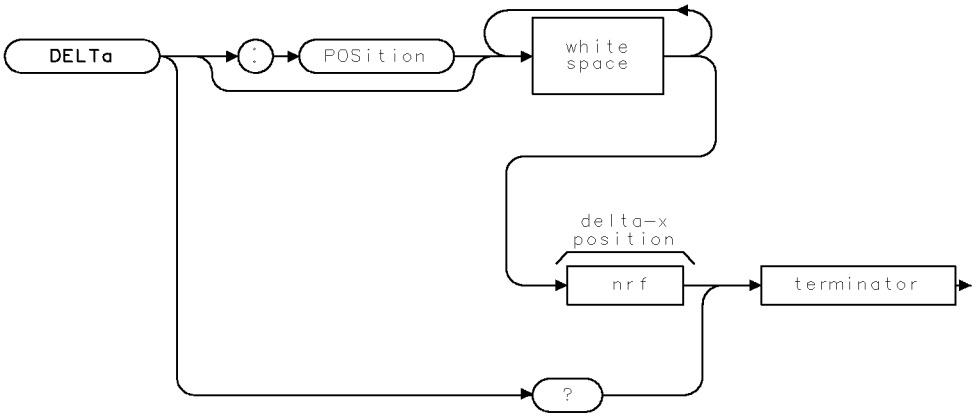
Query Example

```
OUTPUT @Mta;"MARK:DCP?"
ENTER @Mta;Dcpeak$
```

DELTa:POSition

Sets the horizontal position of marker 2.

Syntax



Parameter Range: marker 1 position + delta must be within measurement range.
Related Commands: MARKer:DELTa:STATe

Command Example

```
OUTPUT @Mta;"MARK:DELT 50 us"
```

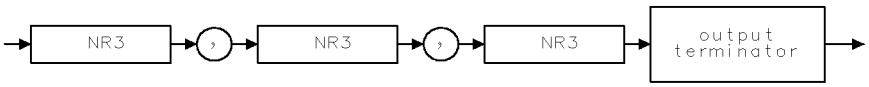
Description

The DELTA:POSITION command sets the horizontal position of marker 2 with respect to marker 1.

The DELTA:POSITION query returns the difference between marker 1 and marker 2, as these values. The first value always corresponds to the horizontal position. The second and third values are real or magnitude and imaginary or phase respectively, depending on the format of the trace.

The difference operation for the query is always a subtraction in the units of the source traces. For example, if marker 1 is at 3 ns, 12 mV and marker 2 is at 10 ns, 20 mV, the result is 7 ns, 8 mV. Note that if the units are logarithmic, the operation is still a subtraction. For example, if marker 1 is at 1 GHz, −20 dBm and marker 2 is at 5 GHz, −35 dBm, the result is 4 GHz, −15 dBm.

Query Response



qgenn6

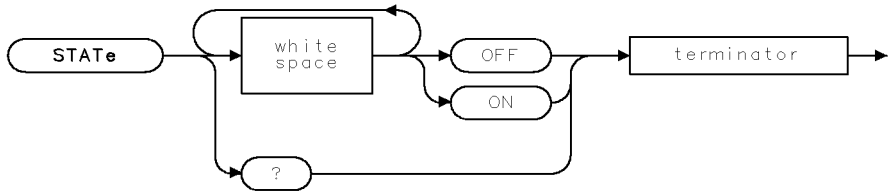
Query Example

```
OUTPUT @Mta;"MARK:DELT?"
ENTER @Mta;X,Y,Z
```

DELTa:STATe

Turns the marker delta mode on or off.

Syntax



Preset State: OFF
Related Commands: MARKer:DELTa:POSition

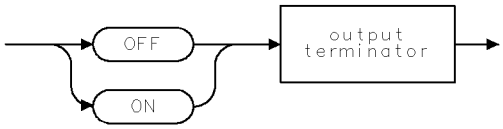
Command Example

```
OUTPUT @Mta;"MARK:DELT:STAT ON"
```

Description

The DELTA:STATE command turns the marker delta mode on or off without changing the relative positions of the marker. The query returns the current value of DELTA:STATE.

Query Response



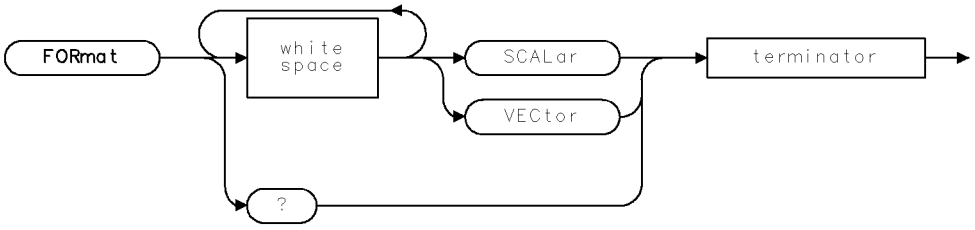
Query Example

```
OUTPUT @Mta;"MARK:DELT:STAT?"
ENTER @Mta;Delta$
```

FORmat

Selects either scalar or vector markers.

Syntax



Preset State: SCALar

Command Example

```
OUTPUT @Mta;"MARK:FOR VEC"
```

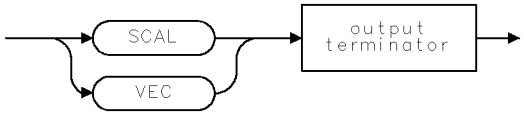
Description

The FORMAT command selects either a scalar or vector implementation for the markers. The query returns the current state.

If FORMAT is set to SCAlar, the markers will simply read out the value of the trace the marker is positioned on.

If FORMAT is set to VECtor, the markers will read out the complex value (that is, either magnitude and phase or real and imaginary) of the trace the marker is positioned on.

Query Response



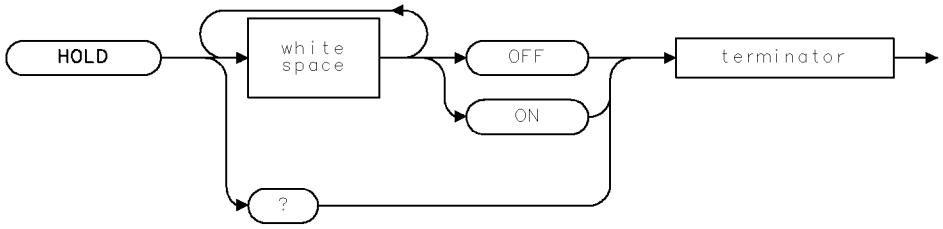
Query Example

```
OUTPUT @Mta;"MARK:FOR?"
ENTER @Mta;Format$
```

HOLD

Enables or disables marker tracking of trace data.

Syntax



Preset State: OFF

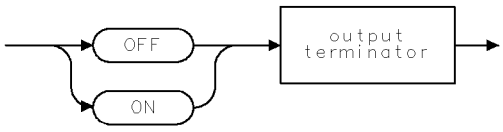
Command Example

```
OUTPUT @Mta;"MARK1:HOLD ON"
```

Description

The HOLD command enables or disables the "tracking" of trace data. Namely, when HOLD is OFF, the specified marker will update its position and value based on trace data whenever new data is available. If HOLD is ON, the specified marker will maintain its position and value regardless of changing trace data. The query returns the current state.

Query Response



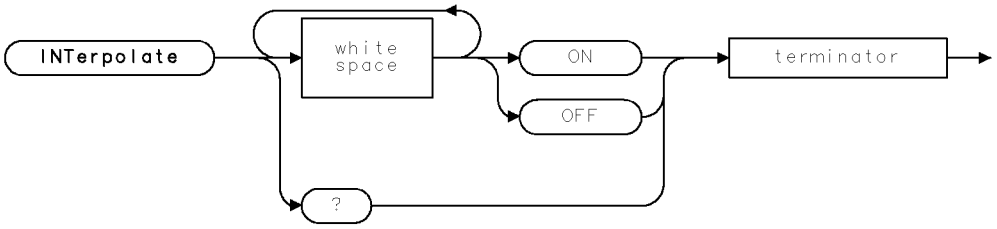
Query Example

```
OUTPUT @Mta;"MARK2:HOLD?"
ENTER @Mta;Hold$
```

INTerpolate

Controls the state of marker interpolation.

Syntax



Preset State: ON
Related Commands: MARKx:LPEAk
MARKx:NEXT
MARKx:PEAK

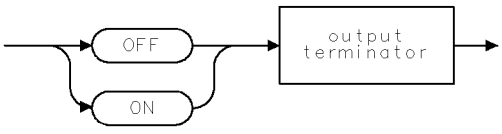
Command Example

```
OUTPUT @Mta;"MARK:INT ON"
```

Description

The INTERPOLATE command controls the state of the marker interpolation. When interpolation is ON, the commands MARKx:LPEAK, MARKx:NEXT, and MARKx:PEAK use interpolation on time domain traces that have been formatted as frequency domain magnitude (linear or log) or frequency domain traces formatted as time domain magnitude (linear or log) to more accurately determine the frequency and amplitude of the peak. When interpolation is OFF, the markers read the value of the actual trace point. The query returns the current state of the Interpolate flag.

Query Response



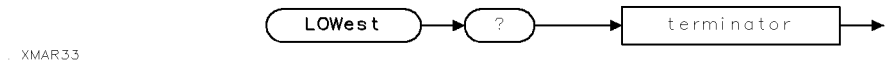
Query Example

```
OUTPUT @Mta;"MARK:INT?"
ENTER @Mta;Mark_interp$
```

LOWest

Causes the specified marker to perform a lowest point search.

Syntax

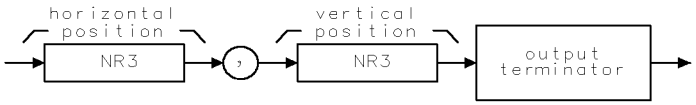


Description

If the specified marker is off, the marker is placed in the lowest numbered displayed trace (FUNCx). If no trace is currently displayed, trace 1 is displayed and the specified marker is placed on it. After the lowest point operation, the horizontal and vertical position of the selected marker is returned. The second marker is unaffected.

Note The MARKx:LOWEST command sets MARK:AUTOPEAK to OFF.

Query Response



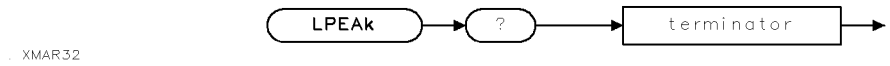
Query Example

```
OUTPUT @Mta;"MARK1:LOW?"
ENTER @Mta;Freq,Magn
```

LPEAk

Performs a frequency domain local peak search.

Syntax



Related Commands: MARKx:INTerpolate

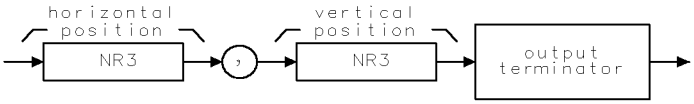
Description

The LPEAK query causes the specified marker to perform a frequency domain local peak search from its current position. If the marker is not positioned on a frequency domain magnitude trace, the marker is placed on the highest point of the trace.

If the specified marker is off, the marker is placed on the lowest numbered displayed trace (FUNCx). If no trace is currently displayed, trace 1 is displayed and the specified marker is placed on it. After the peak operation, the horizontal and vertical position of the selected marker is returned. The second marker is unaffected.

Note The MARKx:LPEAK command sets MARK:AUTOPEAK to OFF.

Query Response



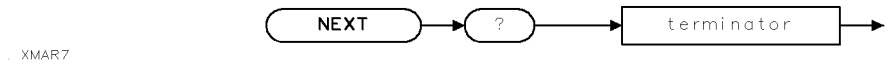
Query Example

```
OUTPUT @Mta;"MARK1:LPEAk?"
ENTER @Mta;Freq,Magn
```

NEXT

Performs a frequency domain next peak search.

Syntax



Related Commands: MARKx:INTerpolate

Description

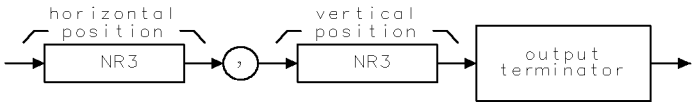
The NEXT query causes the specified marker to perform a frequency domain next peak search from its current position. If the marker is not positioned on a frequency domain magnitude trace, the marker is placed on the lowest point of the trace.

If the specified marker is off, the marker is placed on the lowest numbered displayed trace (FUNCx). If no trace is currently displayed, trace 1 is displayed and the specified marker is placed on it. After the next peak or minimum search operation, the horizontal and vertical position of the selected marker is returned. The second marker is unaffected.

The NEXT query causes the specified marker to perform a next peak search from its current position.

Note The MARKx:NEXT command sets MARK:AUTOPEAK to OFF.

Query Response



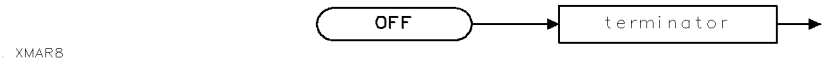
Query Example

```
OUTPUT @Mta;"MARK1:NEXT?"
ENTER @Mta;Freq,Magn
```

OFF

Turns markers off.

Syntax



Command Example

```
OUTPUT @Mta;"MARK:OFF"
```

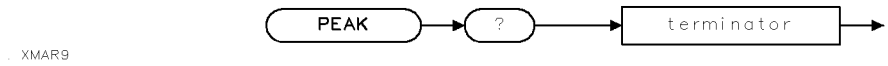
Description

The OFF command turns both markers off.

PEAK

Performs a peak search.

Syntax



Related Commands: MARKx:INTerpolate

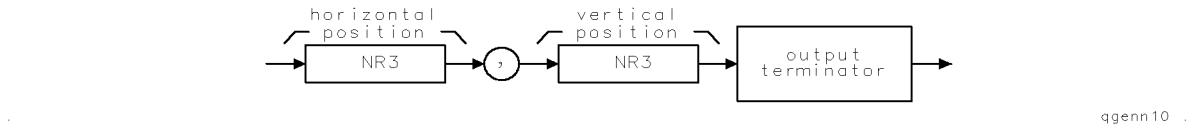
Description

The PEAK query causes the specified marker to perform a peak search.

If the specified marker is off, the marker is placed on the lowest numbered displayed trace (FUNCx). If no trace is currently displayed, trace 1 is displayed and the specified marker is placed on it. After the peak operation, the horizontal and vertical position of the selected marker is returned. The second marker is unaffected.

Note The MARKx:PEAK command sets MARK:AUTOPEAK to OFF.

Query Response



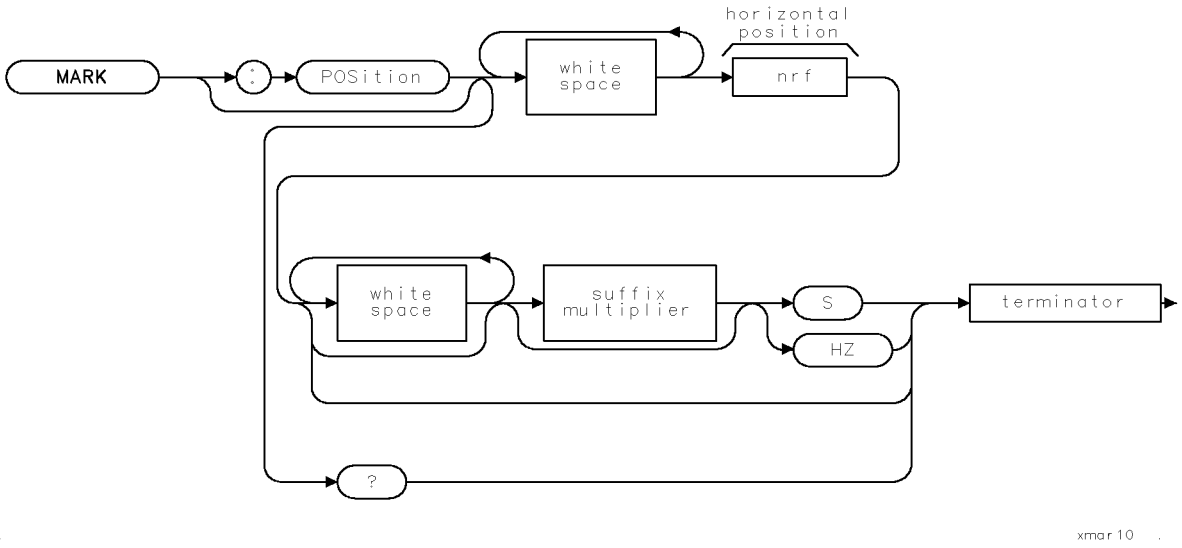
Query Example

```
OUTPUT @Mta;"MARK1:PEAK?"
ENTER @Mta;Freq,Magn
```

POSition

Sets the marker horizontal position.

Syntax



Preset State:	0
Parameter Range:	time domain data: (time_delay − time_range/2) ≤ parameter ≤ (time_delay + time_range/2) − (time_range/length) frequency domain data: start frequency ≤ parameter ≤ stop frequency power sweeps: start power ≤ parameter ≤ stop power
Fundamental Units:	seconds for time domain; hertz for frequency domain

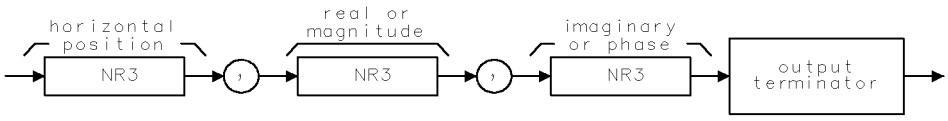
Command Example

```
OUTPUT @Mta;"MARK2:POS 50 ns"
```

Description

The POSITION command sets the horizontal position of the specified marker. The query returns three values. The first value always corresponds to the horizontal position. The second value is real or magnitude and the third value is imaginary or phase, depending on the format of the trace.

Query Response



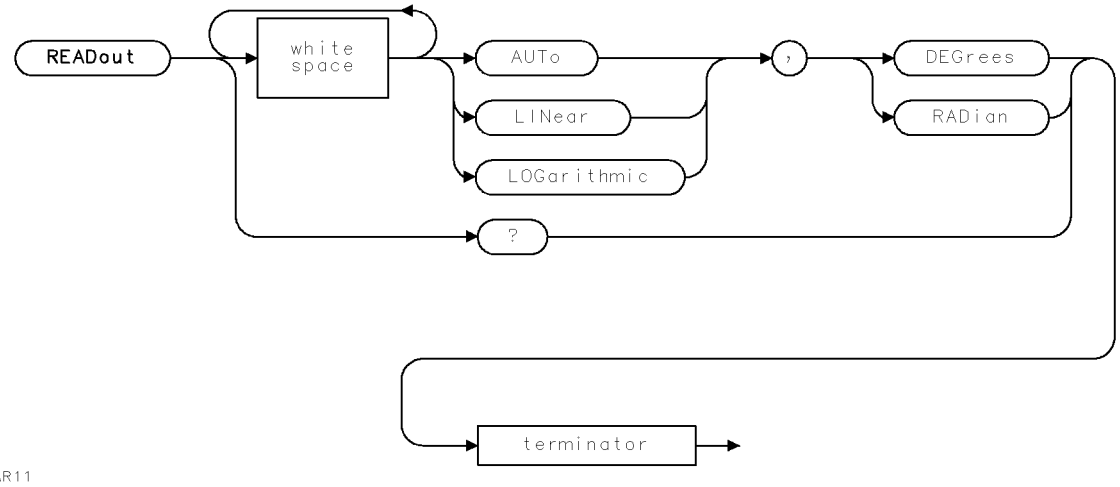
Query Example

```
OUTPUT @Mta;"MARK2:POS?"
ENTER @Mta;X,Y,Z
```

READout

Defines the marker units.

Syntax



Preset State: AUTo, DEGree

Command Example

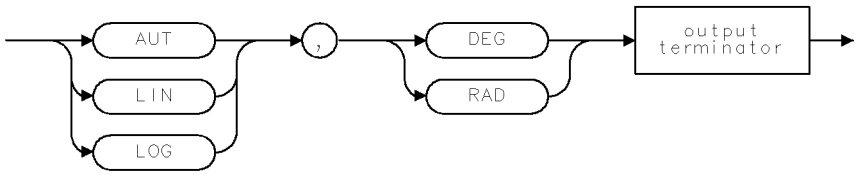
```
OUTPUT @Mta;"MARK:READ LOG,RAD"
```

Description

The READOUT command defines the units that the markers will use. This affects both the graphics display and the values returned via queries. The query returns the current state of READOUT.

- **AUTo** chooses the amplitude units based on the format of the trace defined by the MARKx:SOURCE command.
- **LOGarithmic** uses logarithmic units for the amplitude (that is, dB or dBm).
- **LINear** uses linear units for the amplitude (that is, volts or unitless).
- **DEGree** uses degrees for phase.
- **RADian** uses radians for phase.

Query Response



QMAR11

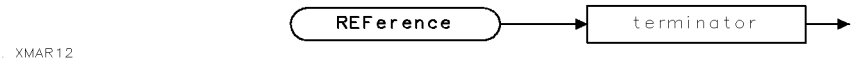
Query Example

```
OUTPUT @Mta;"MARK:READ?"
ENTER @Mta;Readout$
```

REFerence

Sets the reference level to the Y-axis value of the selected marker.

Syntax



Command Example

```
OUTPUT @Mta;"MARK1:REF"
```

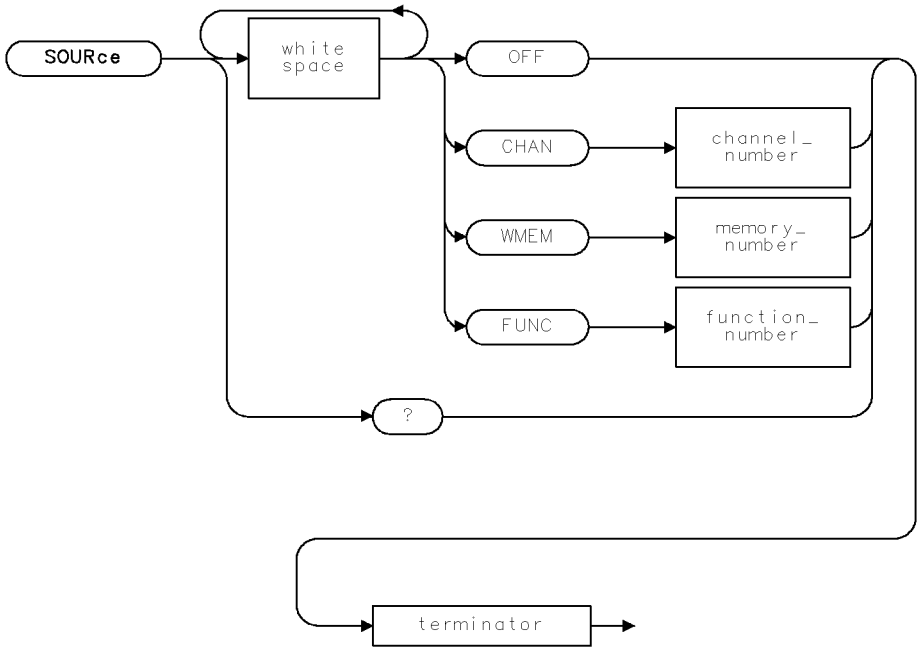
Description

The REFERENCE command causes the reference level, of the trace the marker is positioned on, to be set to the Y-axis value of the selected marker. This command affects the same parameter as DISPLAY:xxx:OFFSET.

SOURce

Defines the trace that a marker is positioned on.

Syntax



Preset State: OFF

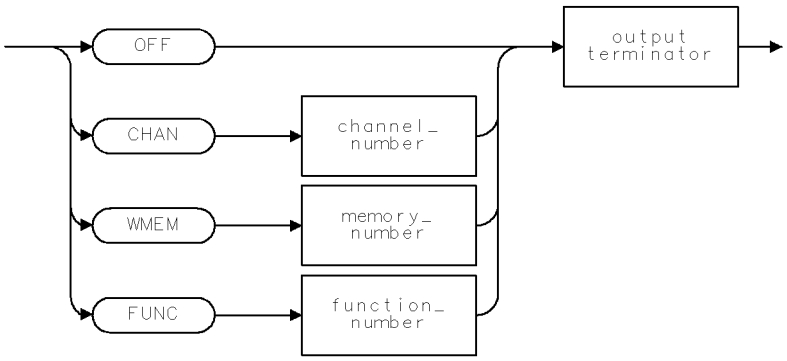
Command Example

```
OUTPUT @Mta;"MARK2:SOUR WMEM1"
```

Description

The SOURCE command defines the trace that a marker is positioned on. The query returns the current value of SOURCE.

Query Response



QMAR13

Query Example

```
OUTPUT @Mta;"MARK1:SOUR?"
ENTER @Mta;Source$
```

17

Mass Memory Subsystem

The Mass Memory subsystem provides mass storage capabilities for the HP 70820A. All mass memory device commands are contained in the MMEMORY subsystem. If the HP Instrument BASIC option is installed, the "msus" (Mass Storage Unit Specifier) may indicate one of 16 volatile RAM disks. The desired mass storage unit is selected with the mass storage unit specifier <*msus*>. <*msus*> is a string with one of the following values:

- "NONE"
- "HPMSIB,row address,column address" (for example, "HPMSIB,0,4")
- "MEMORY, unit" (requires HP Instrument BASIC)

Files are saved in LIF (Logical Interchange Format). The <*filename*> parameter described in the commands in the MMEMORY subsystem is a string. See the following list for a command syntax summary for the MMEMORY subsystem commands.

Note When refering to the following list, <msus> is defined as:

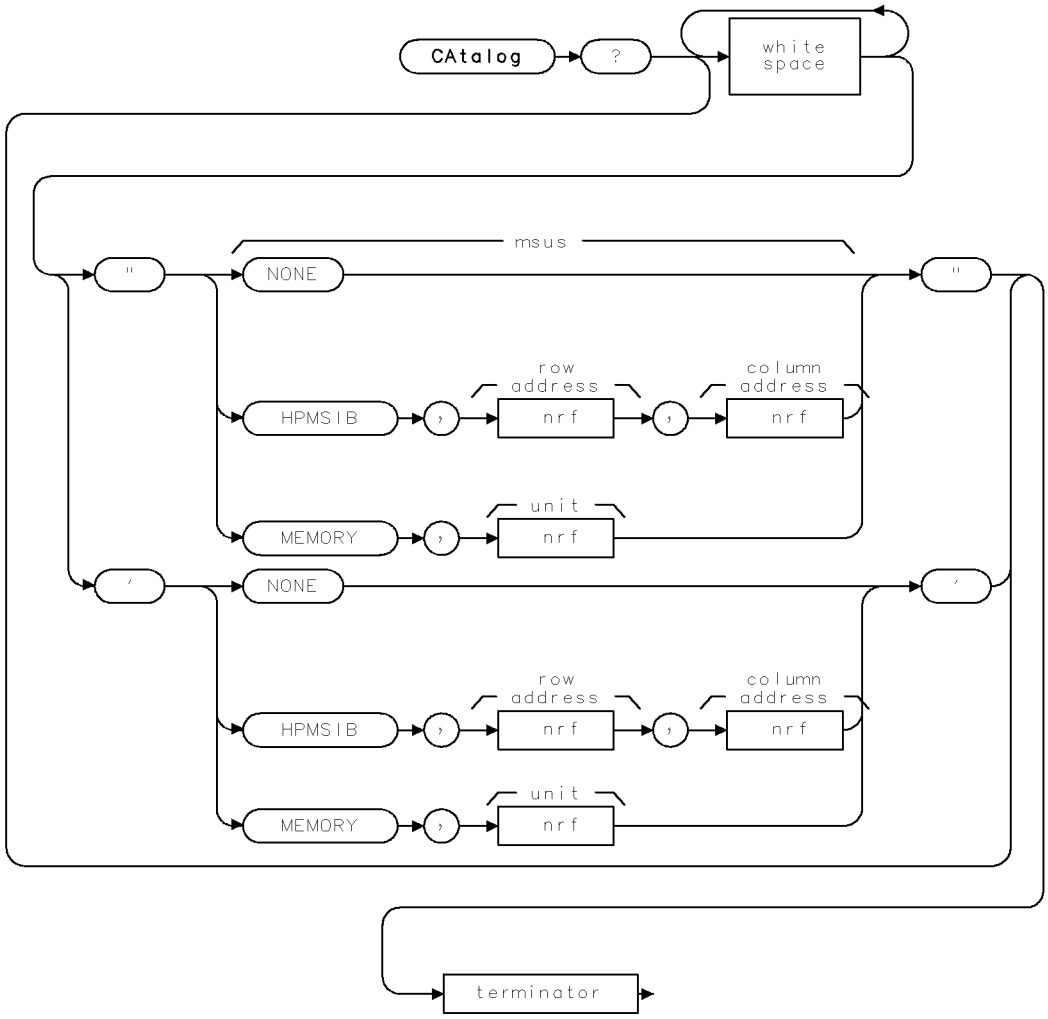
```
(NONE|HPMSIB,<nrf>,<nrf>|MEMORY,<nrf>)
```

KEYWORD	PARAMETER FORM	NOTES
:MMEMory		
:CATalog?	["<msus>"|'<msus>']	[query only]
:DELete	"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:INITialize	["<msus>"|'<msus>']	[no query]
:LOAD		
:CALibration	(CHAN1|CHAN2),"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:CUSTom	"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:HISTogram	"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:PROGram	"{<character>}"[,{"<msus>"|'<msus>']	[no query]
:STATe	"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:TRACe	(CHAN<channel number>|WMEM<memory number>), "{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:USER	"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:MSIS	["<msus>"|'<msus>']	[no query]
:STORe		
:CALibration	(CHAN1|CHAN2),"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:CUSTom	"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:FTRace	(CHAN<channel number>|WMEM<memory number> |FUNC<function number>),"{<character>}"|' {<character>}'[,"<msus>"|'<msus>']	[no query]
:HISTogram	"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:PROGram	"{<character>}"[,"<msus>"|'<msus>']	[no query]
:STATE	<nrf>,"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]
:TRACe	(CHAN<channel number>|WMEM<memory number> |FUNC<function number>),"{<character>}"|' {<character>}'[,"<msus>"|'<msus>']	[no query]
:USER	"{<character>}"|'{<character>}' [,"<msus>"|'<msus>']	[no query]

CATalog

Returns information on the current contents and state of the mass storage media.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Description

The CATALOG query returns information on the current contents and state of the mass storage media. Upon a CATALOG query, the instrument reads the specified mass memory device and returns its directory information in the following format:

<numeric value>,<numeric value> ,<file entry>

The first parameter is the total amount of storage currently used in bytes. The second parameter indicates the total amount of storage available, also in bytes. If the total amount of storage available is not known, 9.91E37 (Not A Number) will be returned. The <*file entry*> is a string.

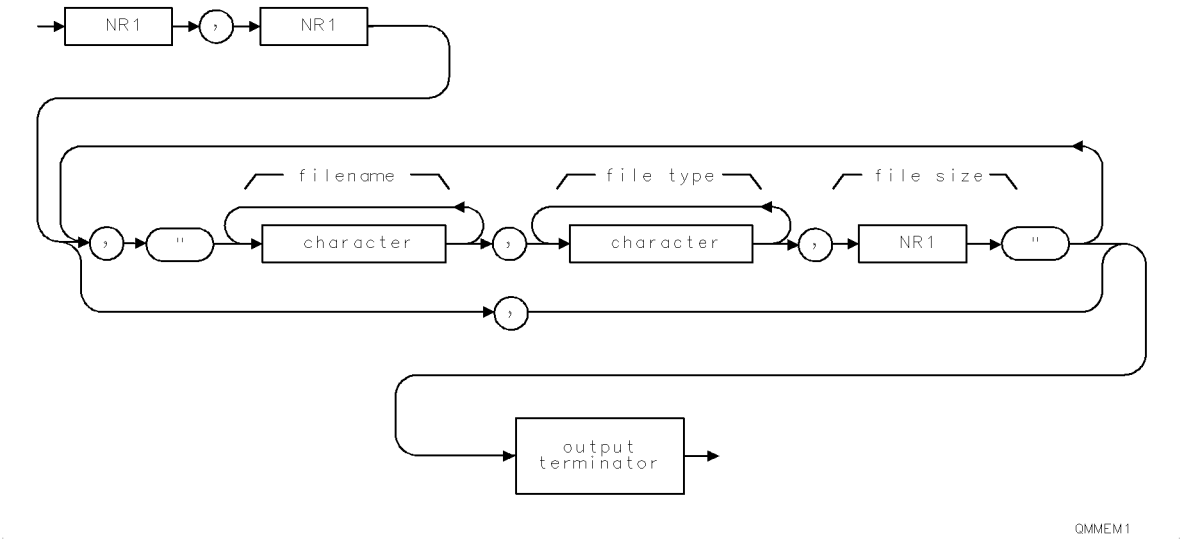
Each <*file entry*> will indicate the name, type, and size of one file in the directory list:

<*file name*>,<*file type*>,<*file size*>.

The <*file name*> is the exact name of the file as it appears in the directory list, including an extension if present. The <*file size*> is the size of the file in bytes. The <*file type*> will be an exact copy of the file type as it appears in the directory. This field will be blank if the information is not available.

The default value of <*msus*> is specified by MMEMORY:MSIS.

Query Response



Query Example

```
DIM Catalog$[10000]
OUTPUT @Mta;"MMEM:CAT? 'HPMSIB,0,4'"
ENTER @Mta;Used,Total,Catalog$
```

For example, the command:

```
PRINT Used,Total,Catalog$
```

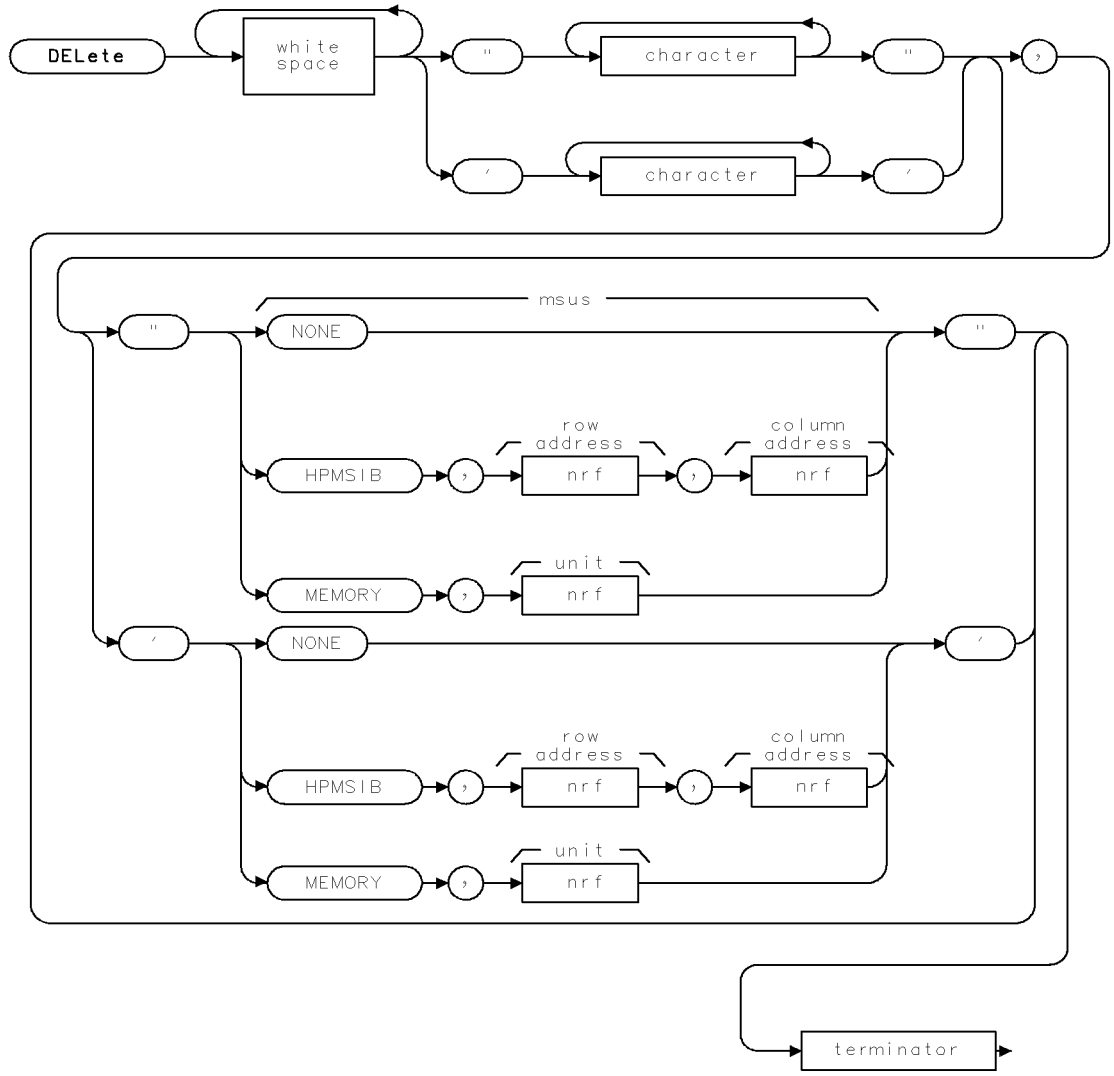
could result in:

```
+768,+2.91840000000E+004,"temp,ASCII,256","s_1,ASCII,512"
```

DELete

Removes a file from the specified mass storage device.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:DEL 'File1'"
```

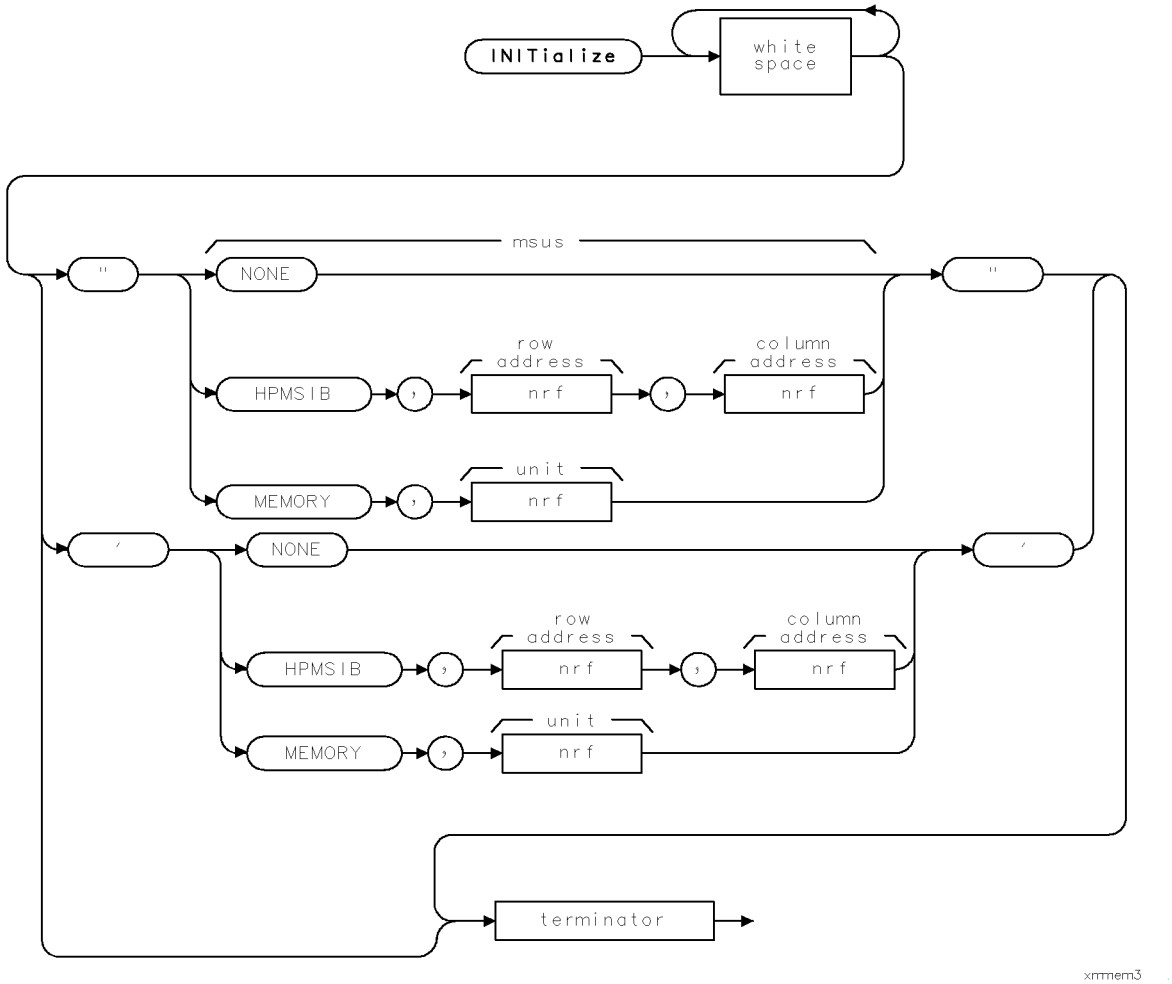
Description

The DELETE command removes a file from the specified mass storage device.

INITialize

Initializes the specified mass storage media.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:INIT 'HPMSIB,0,4'"
```

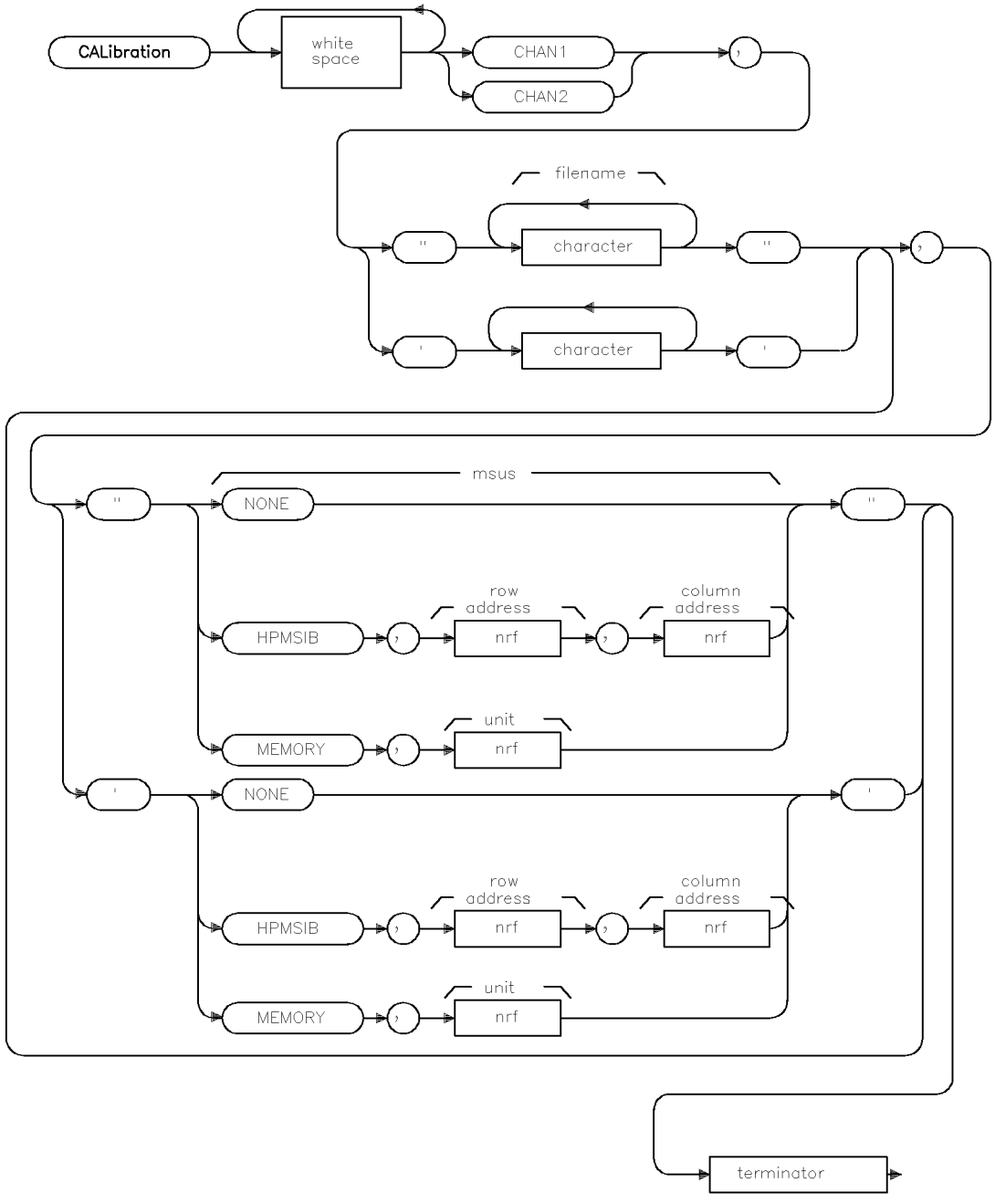
Description

The INITIALIZE command is used to initialize the specified mass storage media. The default value of <*msus*> is NONE (that is, MMEMORY:INITIALIZE; is a no-operation). The second parameter specifies the type of media to be formatted. Currently, only LIF is supported. The third parameter is unused in this implementation but will eventually be used to specify the format, interleave, sector sized, and other media dependent information.

LOAD:CALibration

Loads calibration data from the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:LOAD:CAL CHAN1,'ch1_cal'"
```

Description

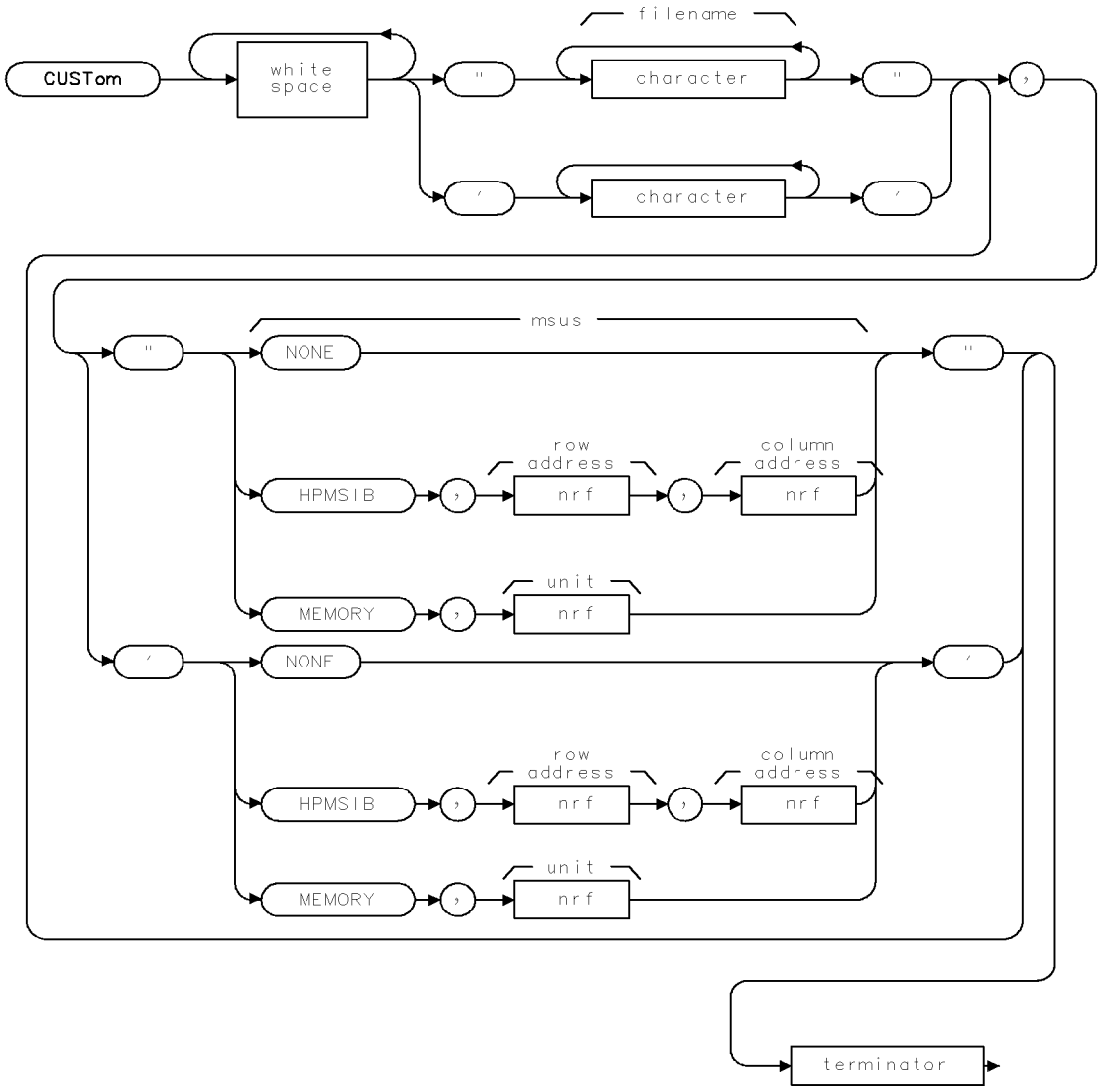
The LOAD:CALIBRATION command loads calibration data from the specified file.

Note CHANNEL1 calibration data cannot be loaded into CHANNEL2 or vice versa.

LOAD:CUSTom

Loads a custom synthesizer definition from the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:LOAD:CUST 'syn83623A'"
```

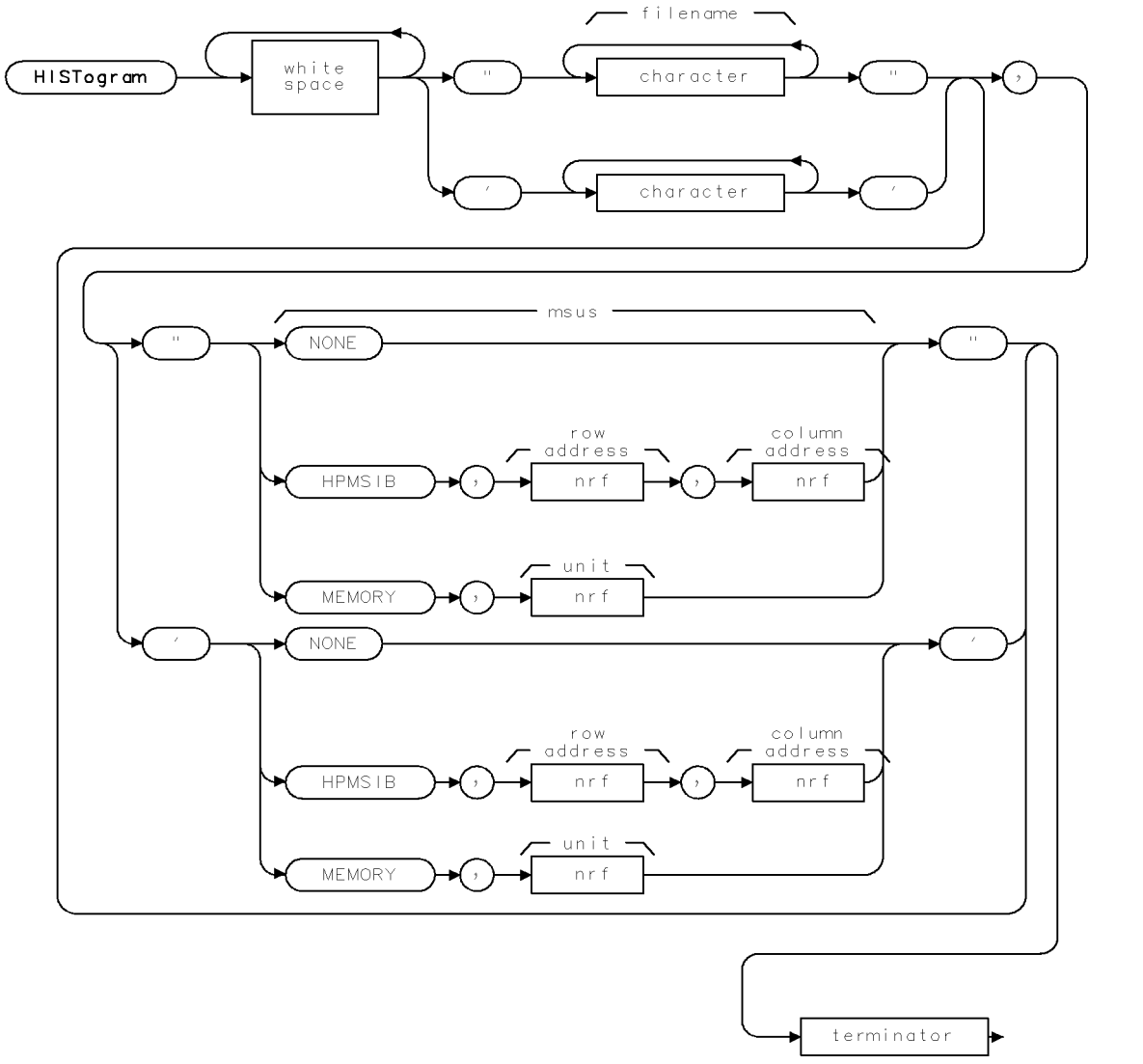
Description

The LOAD:CUSTOM command loads a custom synthesizer definition from the specified file.

LOAD:HISTogram

Loads histogram data from the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:LOAD:HIST 'hist1'"
```

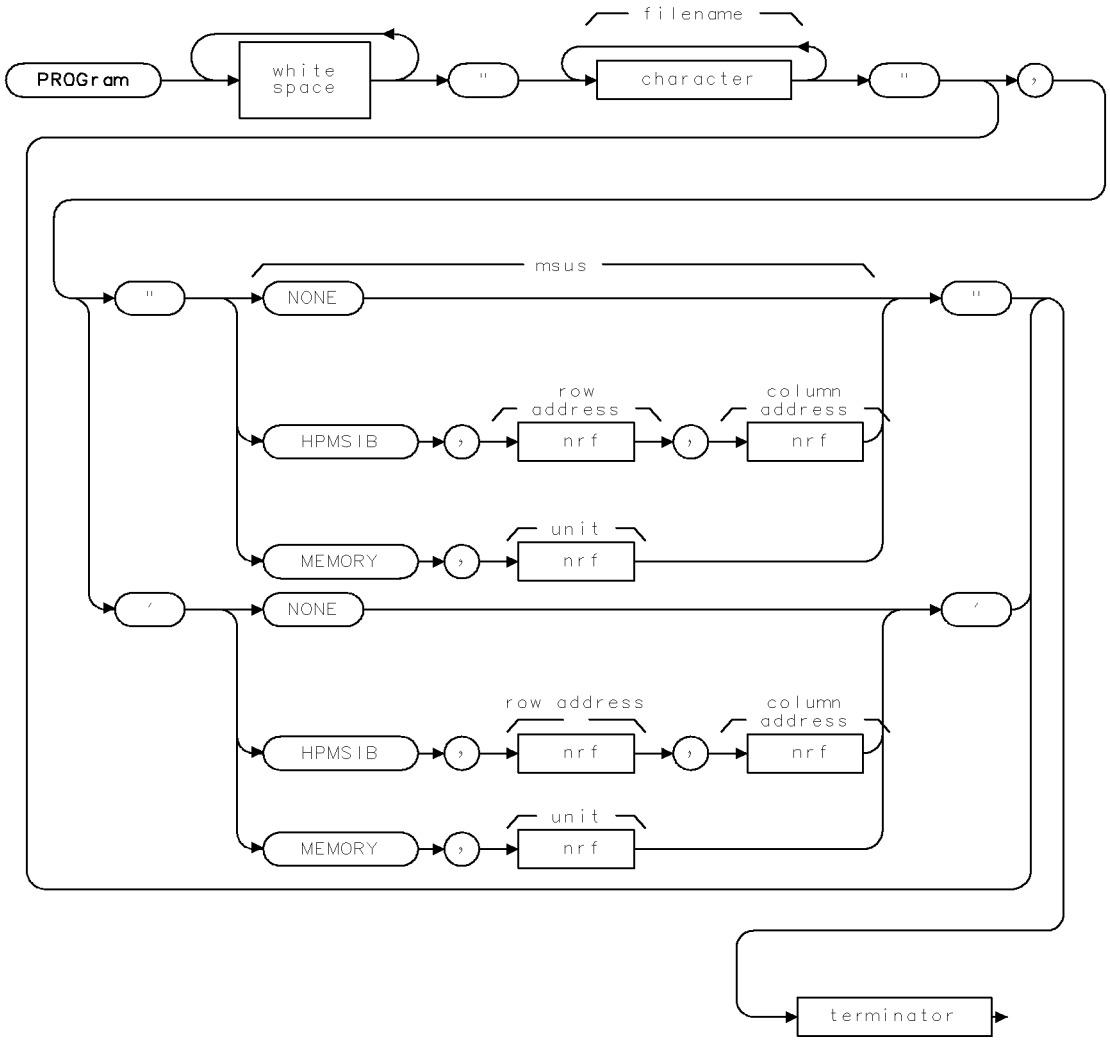
Description

The LOAD:HISTOGRAM command loads histogram data from the specified file.

LOAD:PROGram

Loads the specified file into program.

Syntax



Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:LOAD:PROG 'TEMP'"
```

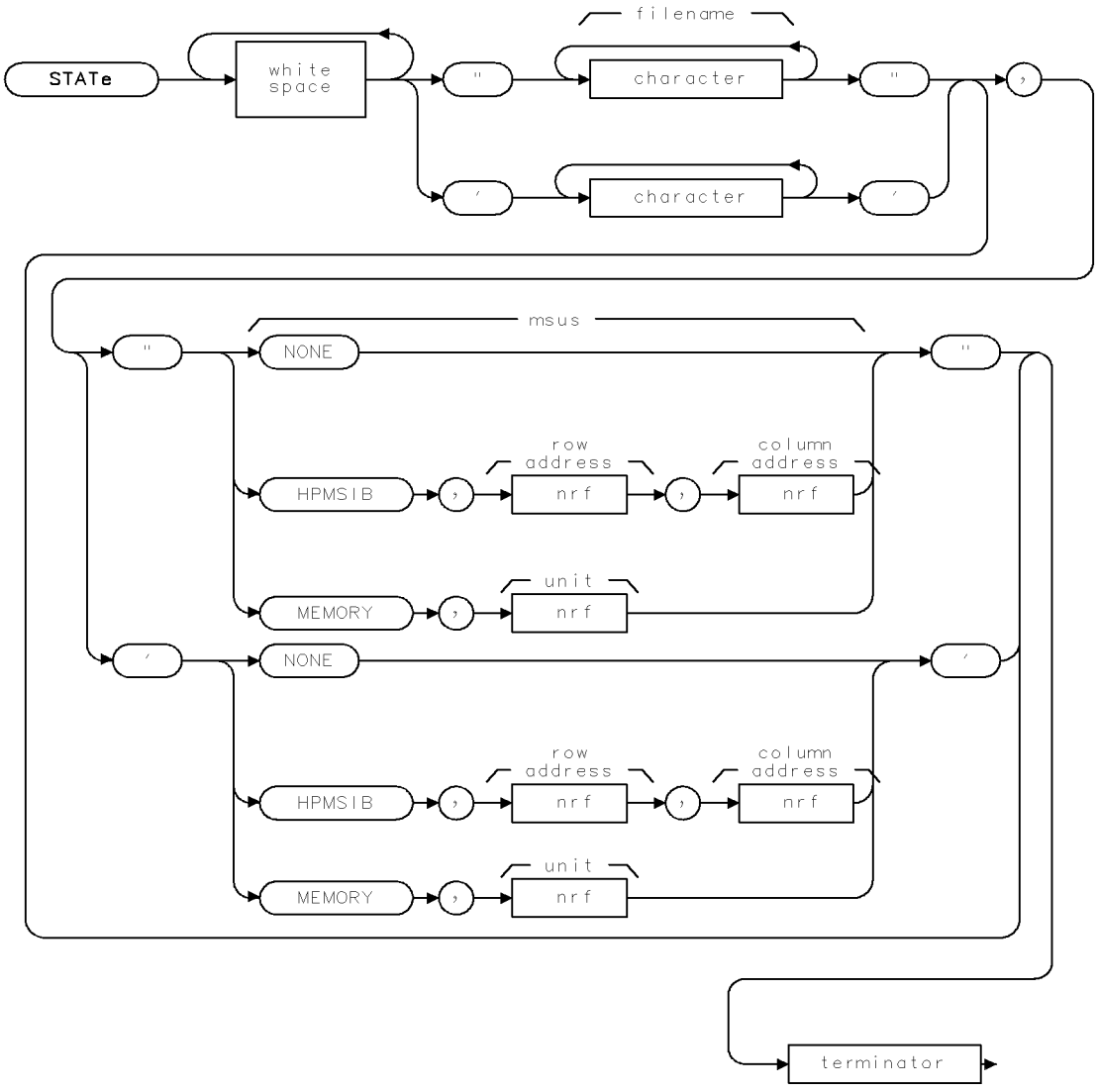
Description

The LOAD:PROGRAM command loads the specified file into the current HP Instrument BASIC program space and runs it. This performs the same function as the HP Instrument BASIC "GET" and "RUN" statements. This command reports error 6100 (option not installed) if the HP Instrument BASIC option is not present.

LOAD:STATe

Loads the specified state from the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31
state = 0 through 6

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:LOAD:STAT 0,'state1'"
```

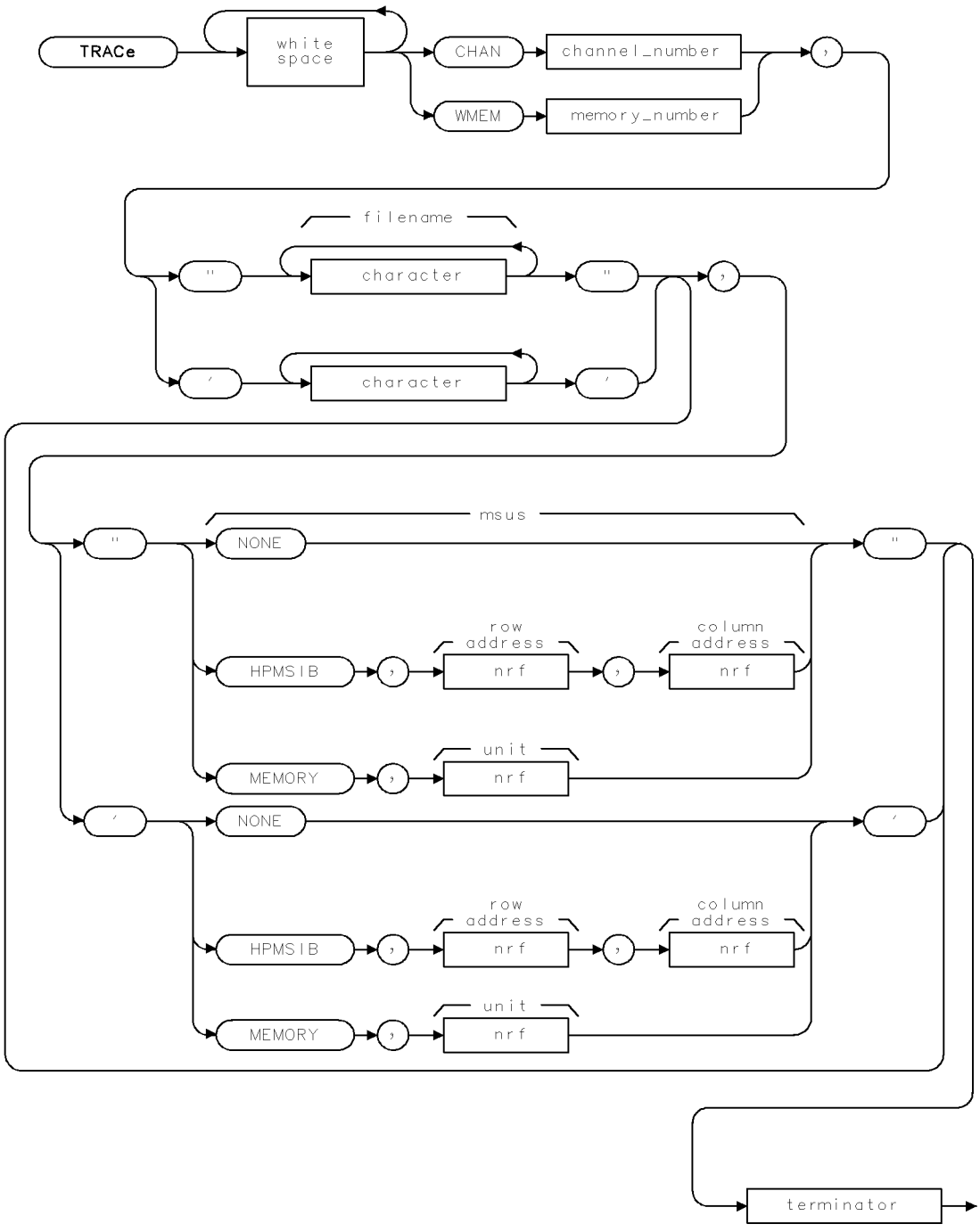
Description

The LOAD:STATE command loads the specified state from the specified file. A state value of 0 corresponds to the operating state (for example, recall the state from the file as opposed to copy it to an internal register).

LOAD:TRACe

Loads the specified trace from the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:LOAD:TRAC CHAN1,'state1'"
```

Description

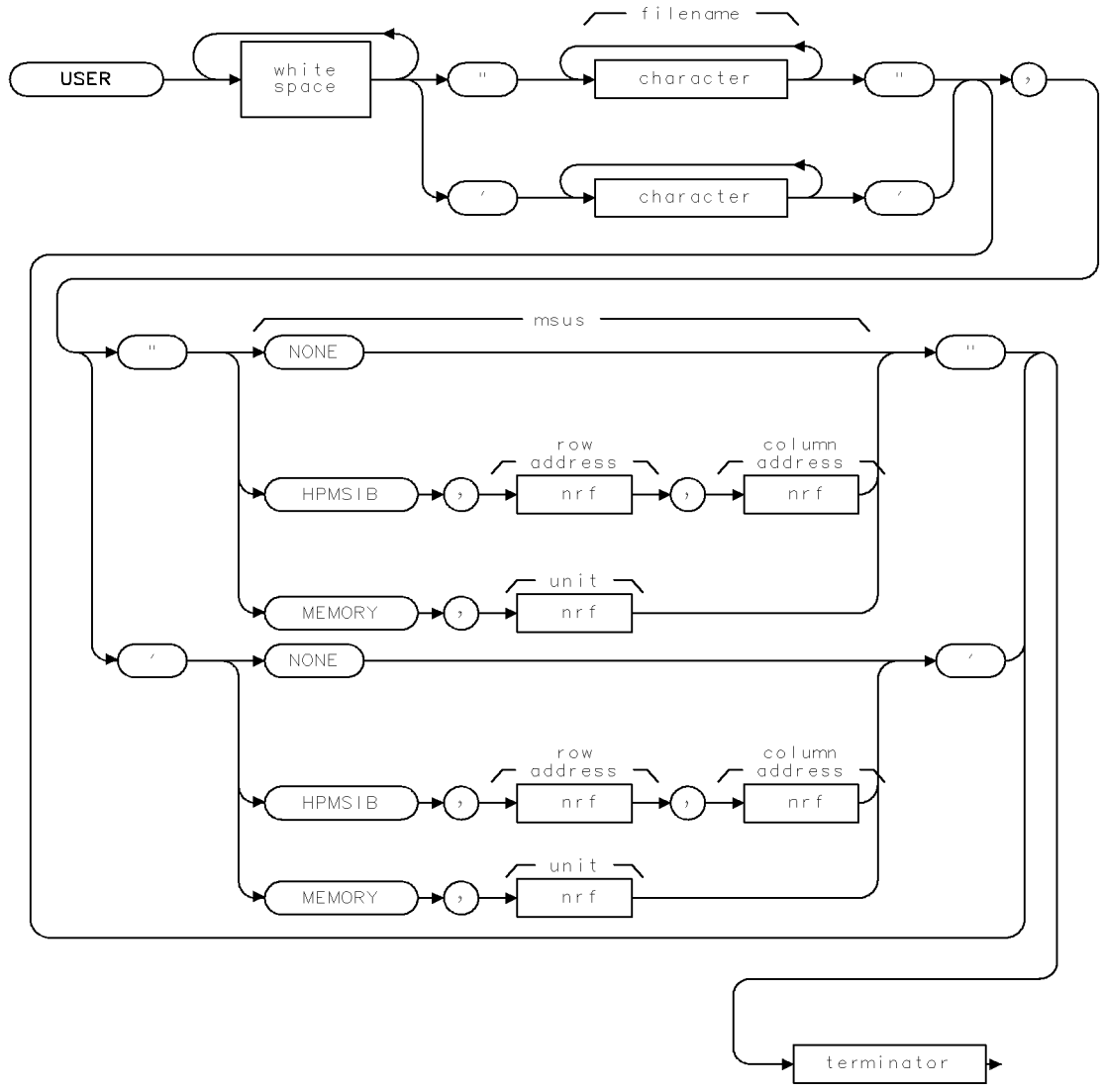
The LOAD:TRACE command loads the specified trace from the specified file. This command is used regardless of how the trace data was stored (for example, with STORE:TRACE or STORE:FTRACE).

Note FUNC1, FUNC2, FUNC3, or FUNC4 cannot be the destination.

LOAD:USER

Loads both channels of user correction data from the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:LOAD:USER 'ch1_user'"
```

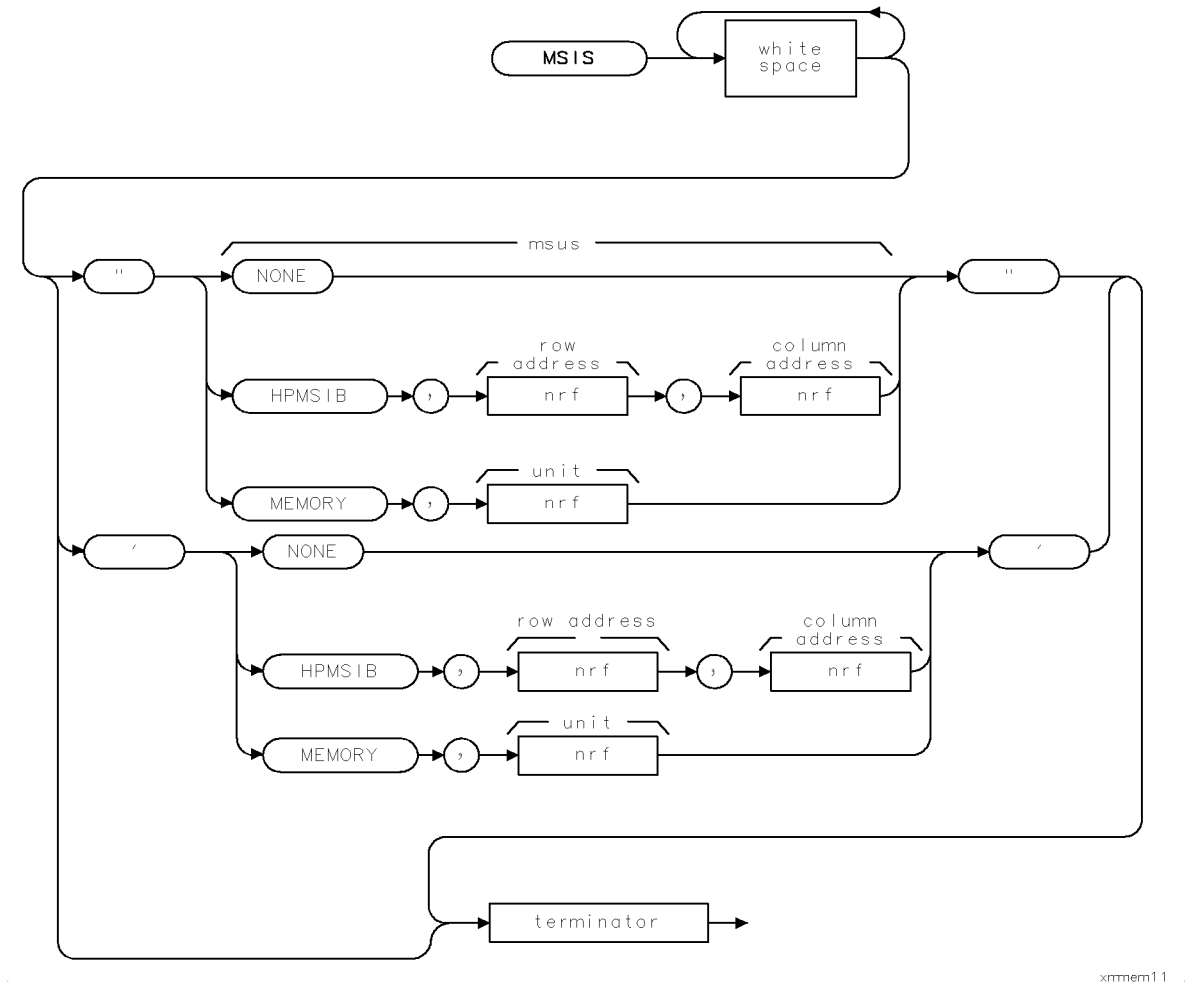
Description

The LOAD:USER command loads both channels of user correction data from the specified file.

MSIS

Selects a default mass storage device.

Syntax



Power On State: NONE
Preset State: Not affected by IP, *RST, or *RCL
Parameter Range: row address = 0 through 7
column address = 0 through 31
unit = 0 through 15

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
```

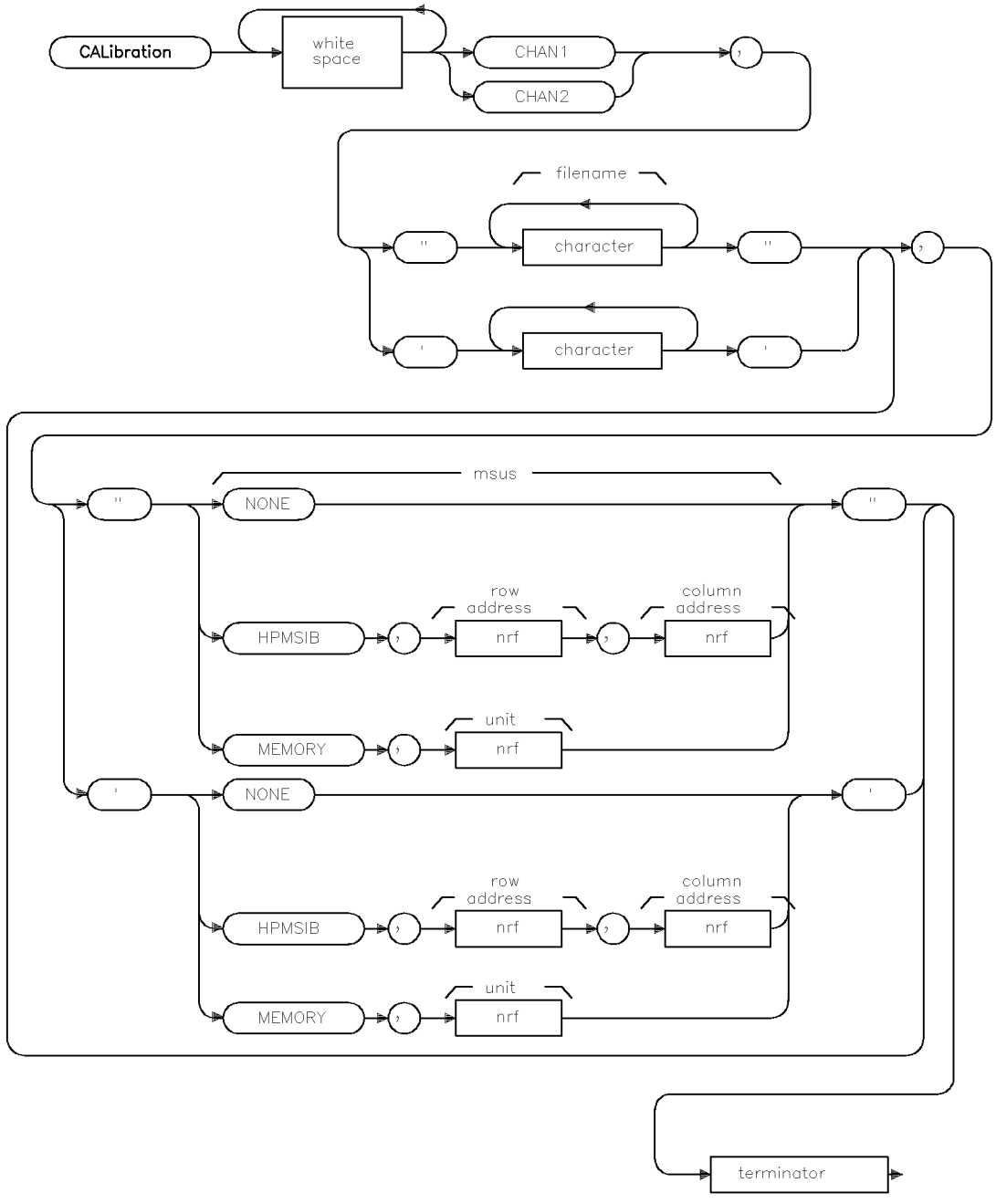
Description

The MSIS command (Mass Storage IS) selects a default mass storage device which is used by all MMEMORY commands and queries with the exception of INITIALIZE. If the parameter is omitted, the value will be set to NONE.

STORe:CALibration

Stores calibration data to the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:STOR:CAL CHAN1,'ch1_cal'"
```

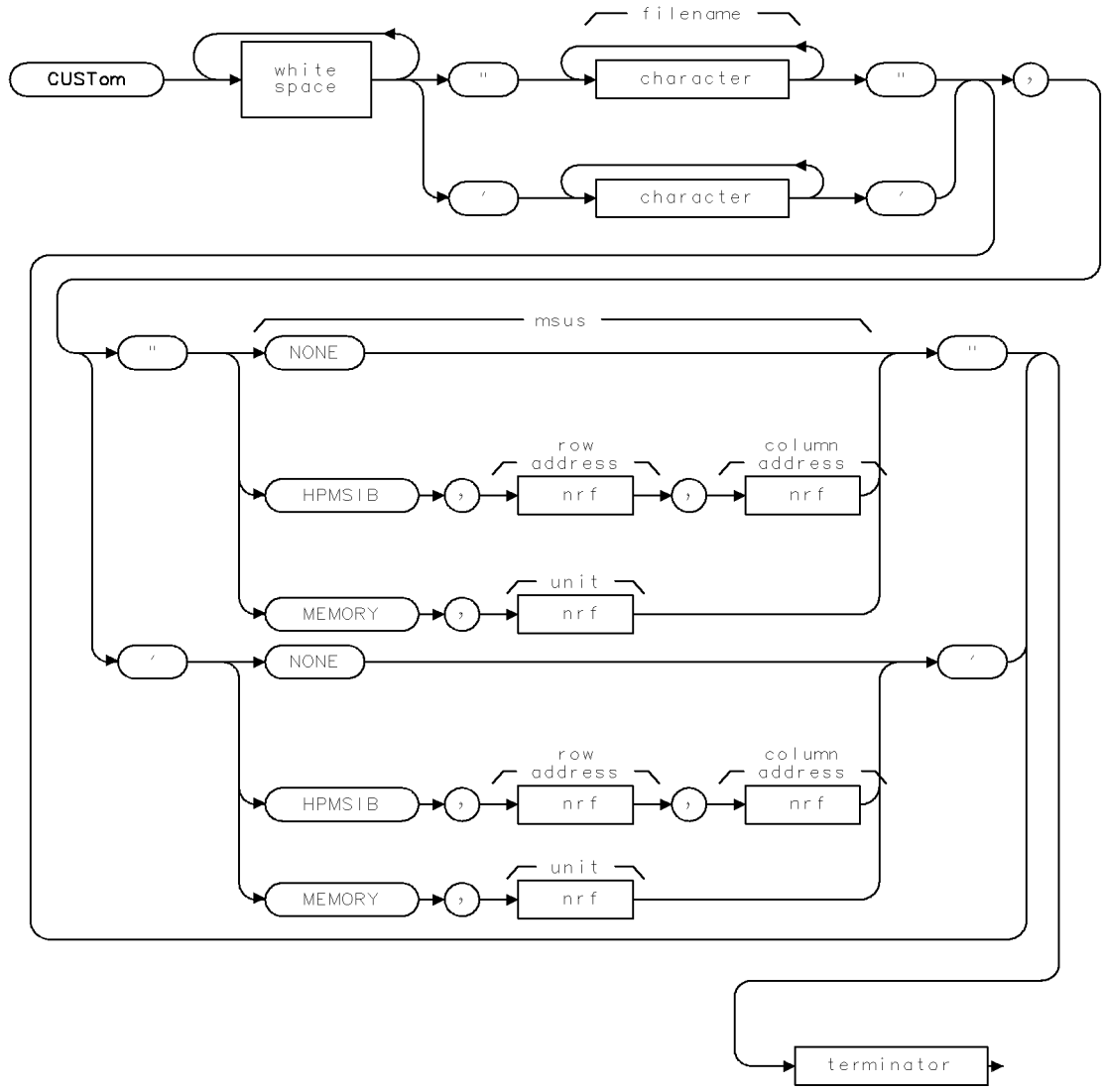
Description

The STORE:CALIBRATION command stores calibration data to the specified file.

STORe:CUSTom

Stores the custom synthesizer definition to the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:STOR:CUST 'syn83623A'"
```

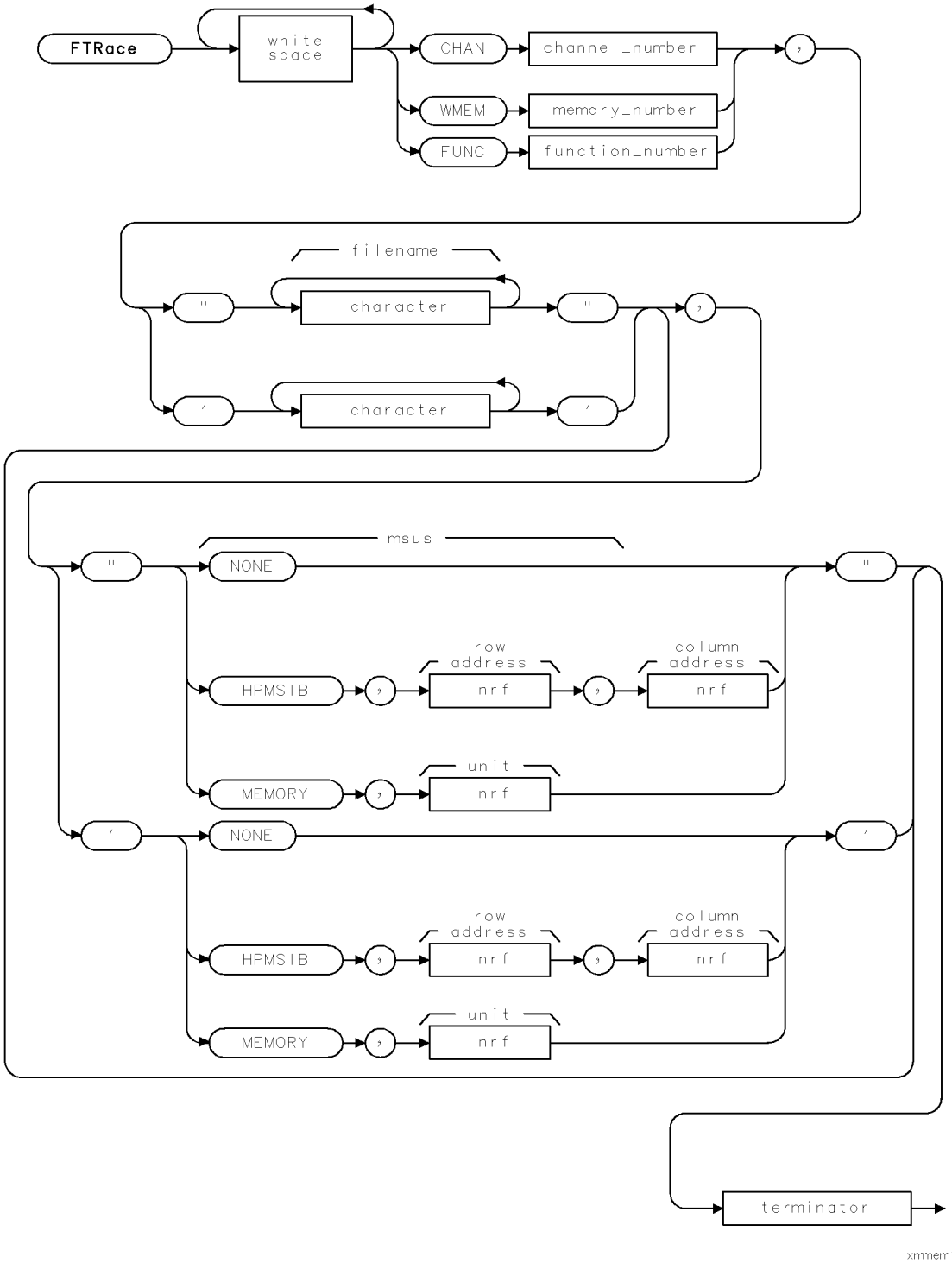
Description

The STORE:CUSTOM command stores the custom synthesizer definition to the specified file.

STORe:FTRace

Stores the specified trace to the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:STOR:FTR FUNC1,'state1'"
```

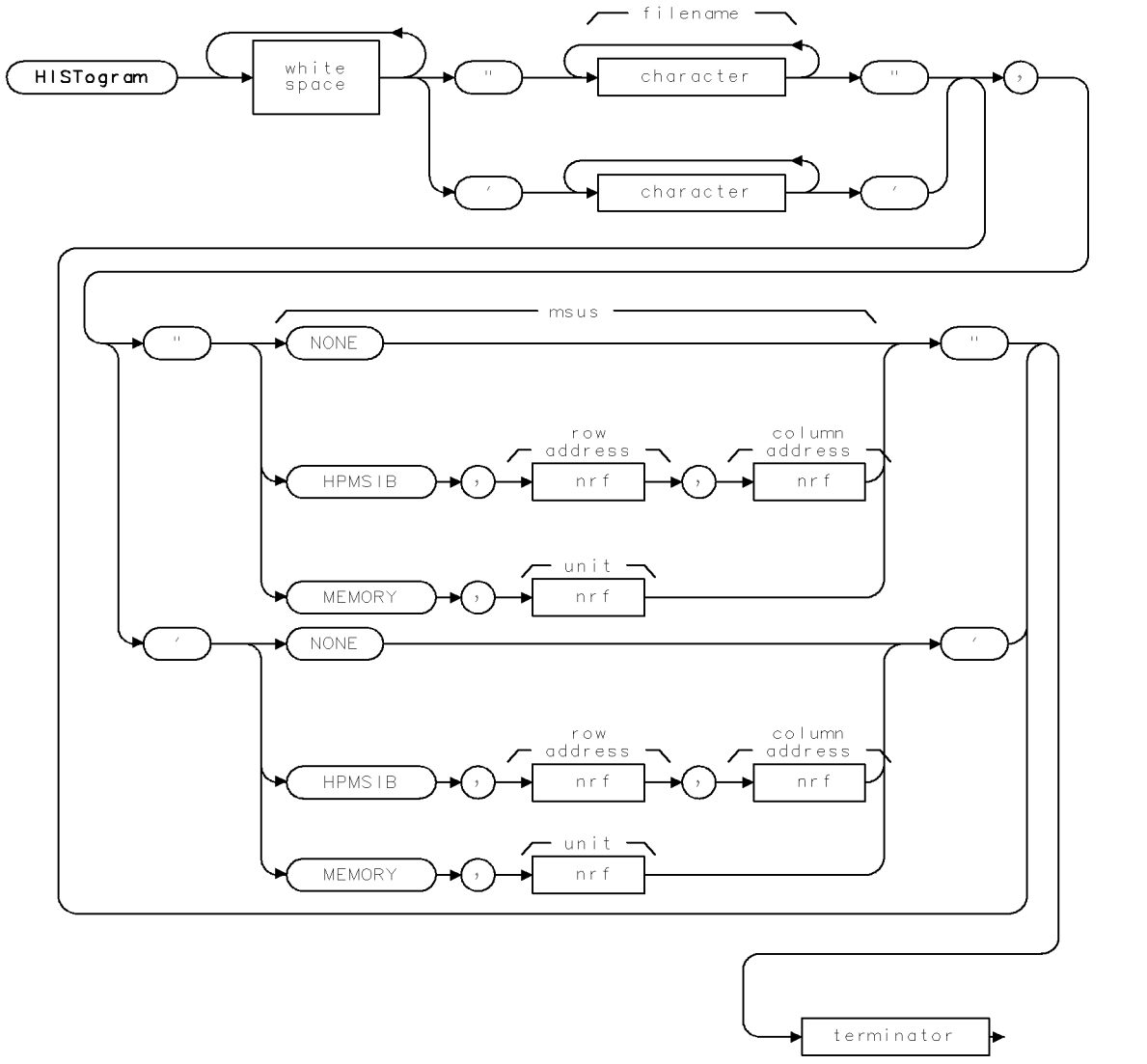
Description

The STORE:FTRACE command stores the specified trace to the specified file. This command stores formatted data (for example, it corresponds to the STORE from, to, FORMAT). Use the STORE:TRACE command to store unformatted data.

STORe:HISTogram

Stores the current histogram data to the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:STOR:HIST 'histogram1'"
```

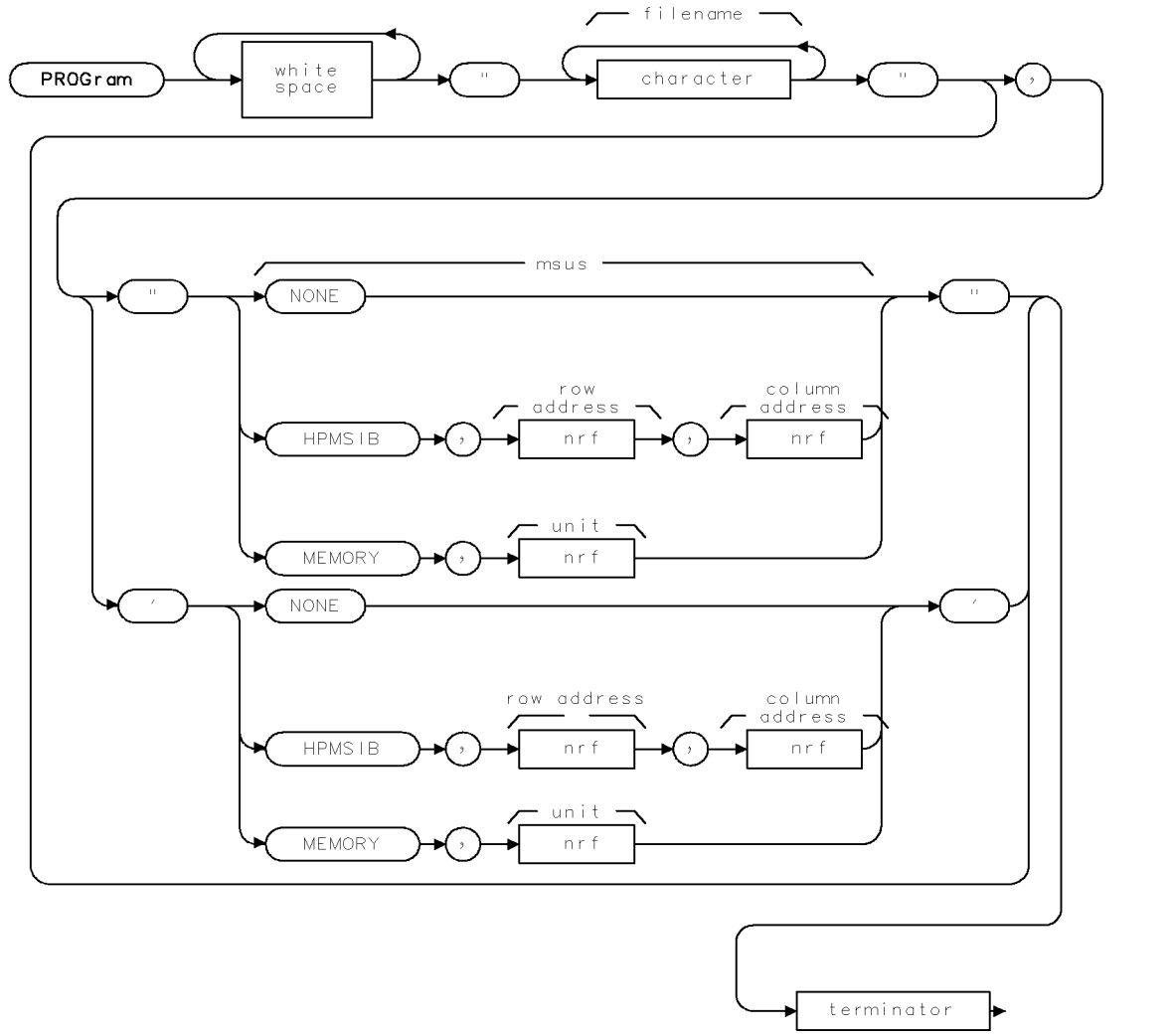
Description

The STORE:HISTOGRAM command stores the current histogram data to the specified file. Valid histogram data must already exist.

STORe:PROGram

Saves the current HP Instrument BASIC program to the specified file.

Syntax



Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:STOR:PROG 'TEMP'"
```

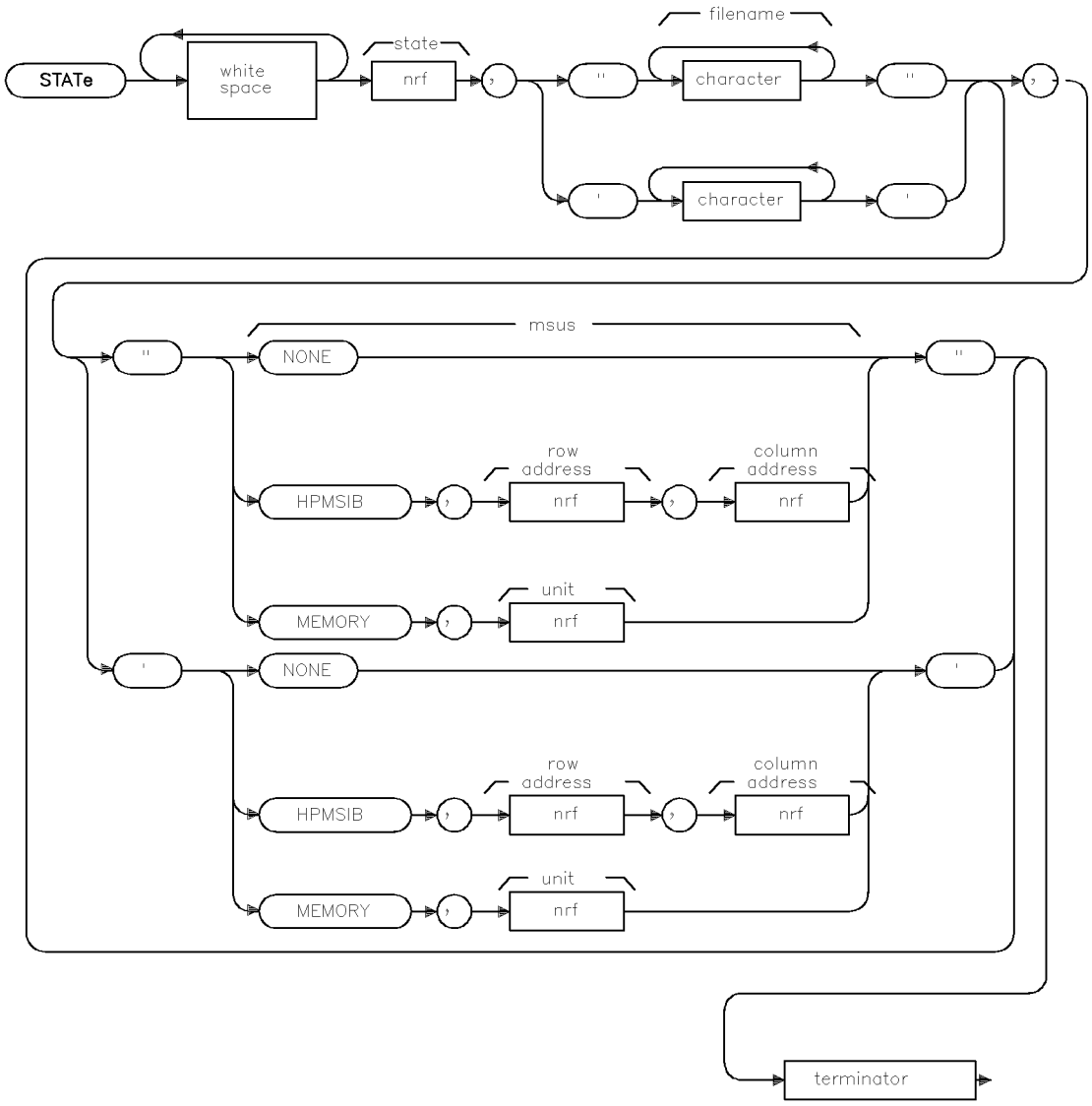
Description

The STORE:PROGRAM command saves the current HP Instrument BASIC program to the specified file. This performs the same function as the HP Instrument BASIC "SAVE" statement. This command reports error 6100 (option not installed) if the HP Instrument BASIC option is not present.

STORe:STATe

Stores the specified state from the specified file.

Syntax



Parameter Range: state = 0 through 6
row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:STOR:STAT 0,'state1'"
```

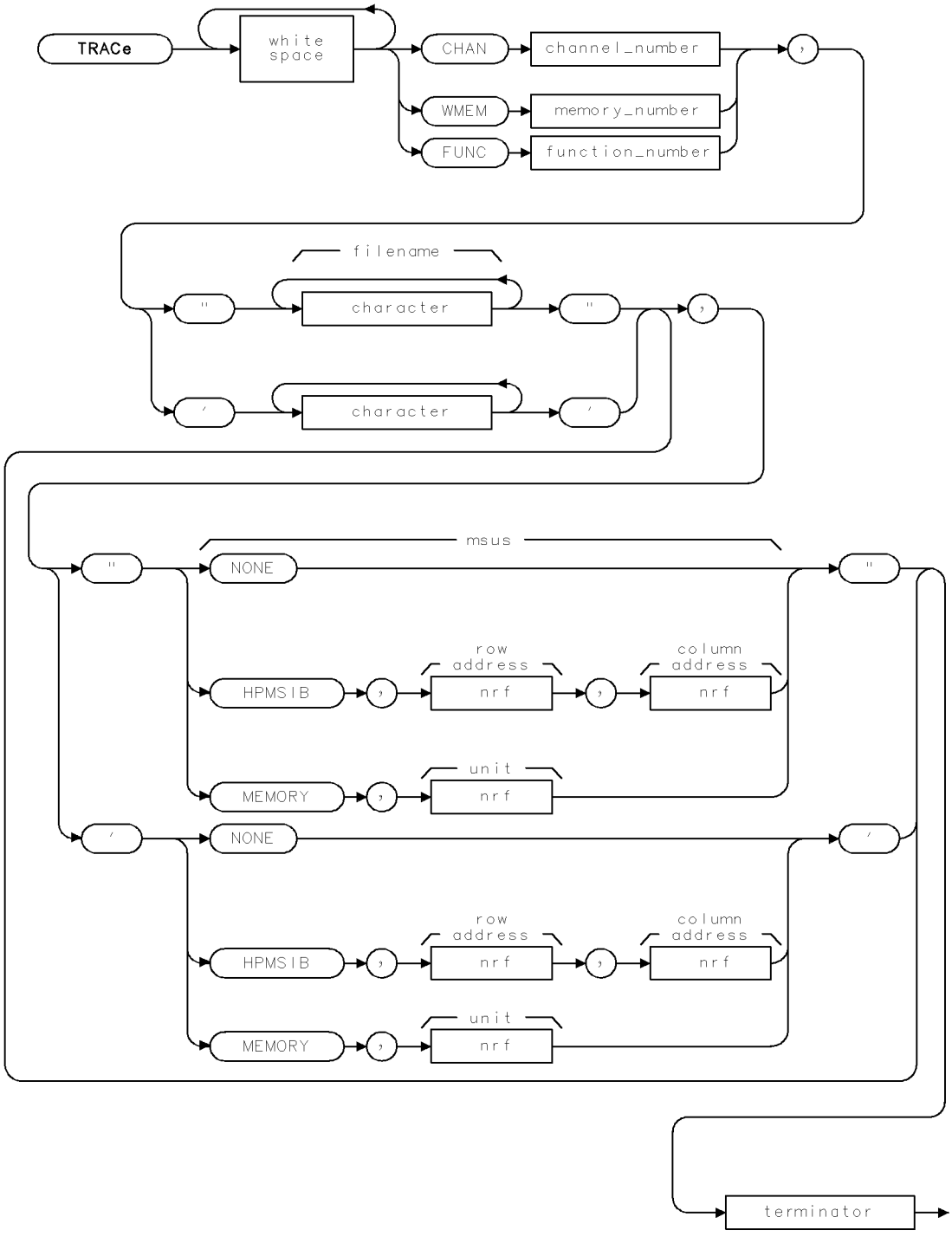
Description

The STORE:STATE command stores the specified state from the specified file. A state value of 0 corresponds to the operating state (for example, save the current state as opposed to copying an internal register).

STORe:TRACe

Stores the specified trace to the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:STOR:TRAC FUNC1,'state1'"
```

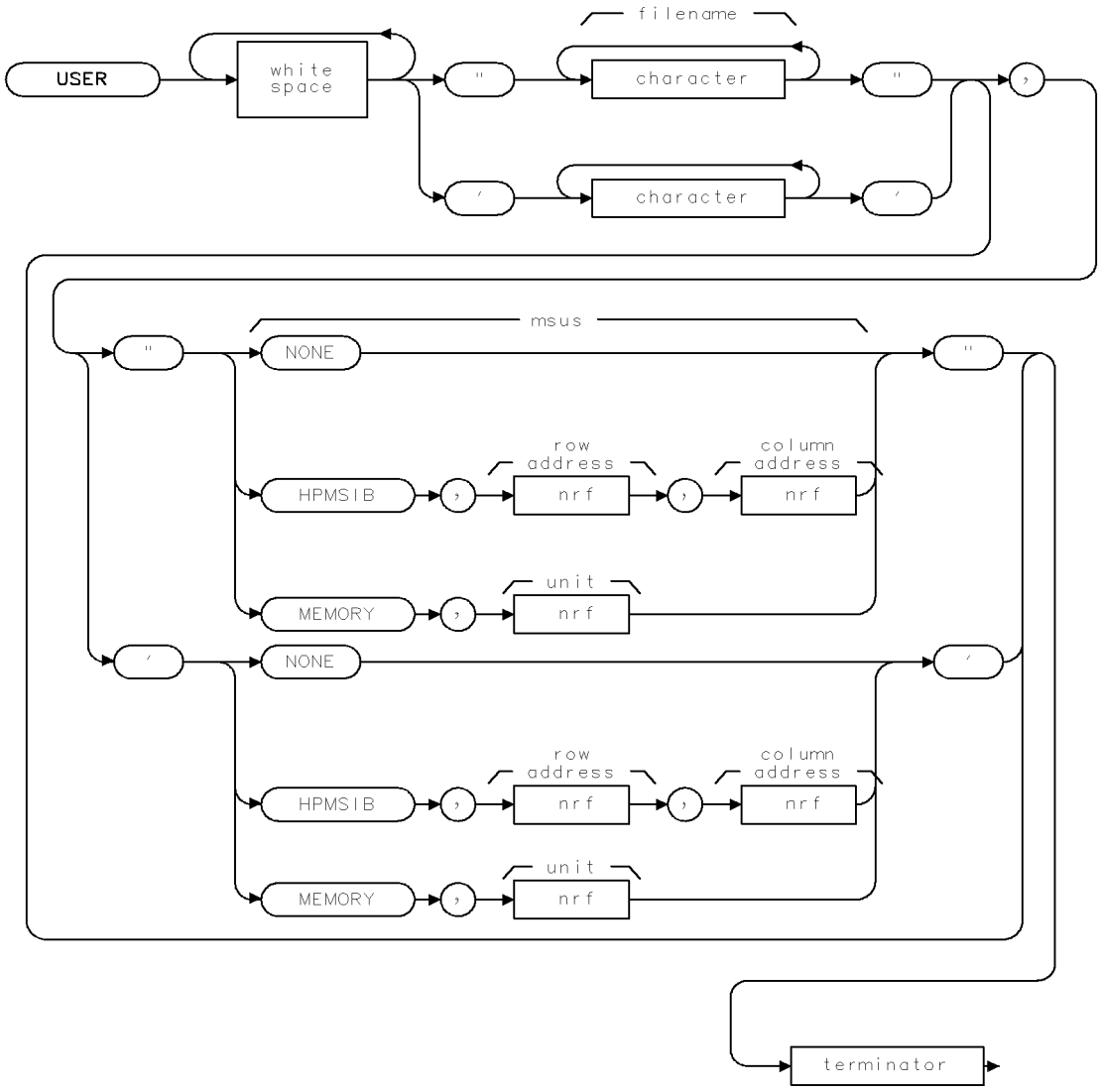
Description

The STORE:TRACE command stores the specified trace to the specified file. This command stores unformatted data (for example, it corresponds to the STORE from, to, VECTOR). Use the STORE:FTRACE command to store formatted data.

STORe:USER

Stores both channels of user correction data to the specified file.

Syntax



Parameter Range: row address = 0 through 7
column address = 0 through 31

Command Example

```
OUTPUT @Mta;"MMEM:MSIS 'HPMSIB,0,4'"
OUTPUT @Mta;"MMEM:STOR:USER 'ch1_user'"
```

Description

The STORE:USER command stores both channels of user correction data to the specified file.

18

Measure Subsystem

The commands in the MEASURE subsystem allow various measurements to be made on a waveform, such as pulse parameter and voltage measurements.

When a measurement cannot be made and a value is requested, FAILVALUE `= 1.0E38` will be returned.

Various terms that are frequently used in the MEASURE subsystem command descriptions are defined below.

EDGE refers to a waveform passing through a voltage level either from lower to higher (positive or rising edge) or from higher to lower (negative or falling edge). There is usually some qualification to prevent noise from looking like a large number of edges. This qualification results in a positive edge being defined as a signal passing successively through the 10 percent, 50 percent, and 90 percent levels (where 0 percent is the BASE and 100 percent is the TOP). A negative edge is defined as a signal passing successively through the 90 percent, 50 percent, and 10 percent levels.

THRESHOLD refers to a level that is used as a reference in making a measurement. For example, if "lower threshold" is used, it refers to the point where the waveform crosses that threshold.

The thresholds may be defined in terms of absolute voltages, or by referring to a percentage on the waveform (such as 10 percent, or 90 percent referenced to TOP and BASE).

HISTOGRAM refers to using a histogram to determine TOP or BASE voltage values. See MEASURE:ALGORITHM for further description. The measure operations require time domain data with linear amplitude. For cases where traces are formatted other than this, the following selection or operations are performed:

- If the input data is frequency domain:
  - and the unformatted data is time domain, use the real portion of the unformatted data.
  - and the unformatted data is frequency domain, declare an error (measurement results are invalid).
- If the input data is time domain:
  - and the input data is linear real or linear magnitude and MINhold or MAXhold is not on, use the input data as is.
  - and the input data is linear real or linear magnitude and MINhold or MAXhold is not on, use every other point of the input data that corresponds to MAXhold.
  - and the input data is phase, use the real portion of the unformatted data.
  - and the input data is logged real, use the real portion of the unformatted data.
  - and the input data is logged magnitude, compute and use the linear magnitude of the input data.

See the following list for a command syntax summary of the MEASURE subsystem commands.

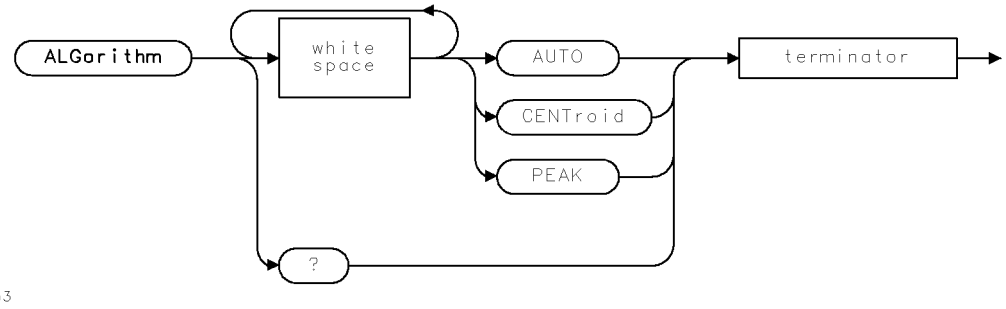
KEYWORD	PARAMETER FORM	NOTES
:MEASure		
:ALGorithm	AUTO|CENTroid|PEAK	
:ALL		
:COUNt	<nrf>	
:CURSor?	DELTa| STARt| STOP	[query only]
:DELay		
:DUTy		
:ESTArt	<nrf>	
:ESTOp	<nrf>	
:FALLtime		
:FREQuency		
:LOWer	<nrf>[PCT]|<nrf>[[<suffix multiplier>]V]	
:MODE	STANdard|USER	
:NWIDth		
:OVERshoot		
:PAVG		
:PERiod		
:PIP		
:PPK		
:PREShoot		
:PRMS		
:PTIMe?	<nrf>[[<suffix multiplier>]S]	[query only]
:PVOLt	<nrf>[[<suffix multiplier>]V],<nrf>	
:PWIDth		
:REFerence	CHAN<channel_number>|WMEM<memory_number> |FUNC<function _number>	
:RISEtime		
:SOURce	CHAN<channel_number>|WMEM<memory_number> |FUNC<function_ number>	
:TDELta?		[query only]
:TMAX?		[query only]
:TMIN?		[query only]
:TPOint?	<nrf>	[query only]
:TSTArt	<nrf>[[<suffix multiplier>]S]	
:TSTOp	<nrf>[[<suffix multiplier>]S]	
:TVOLt	<nrf>[[<suffix multiplier>]V],<nrf>	
:TYPE	AVERage|CONTinuous|OFF|SINGle	
:UNIT	PERCent|VOLTs[,PERCent|VOLTs]	
:UPPer	<nrf>[PCT]|<nrf>[[<suffix multiplier>]V]	

:VAMPlitude?		
:VAVerage?		
:VBASe?		
:VDELta?		[query only]
:VFIFty		[no query]
:VMAX?		
:VMIN?		
:VPOint?	<nrf>	[query only]
:VPP?		
:VREL	<nrf>	
:VRMS?		
:VSTArt	<nrf>[PCT]|<nrf>[[<suffix multiplier>]V]	
:VSTOp	<nrf>[PCT]|<nrf>[[<suffix multiplier>]V]	
:VTIMe?	<nrf>[[<suffix multiplier>]S]	[query only]
:VTOP?		

ALGorithm

Controls the method for determining the TOP and BASE values of a waveform.

Syntax



Preset State: AUTO

Command Example

```
OUTPUT @Mta;"MEAS:ALG PEAK"
```

Description

The ALGORITHM command controls the method for determining the TOP and BASE values of a waveform. The operation of each state is as follows:

PEAK

- Construct a histogram of the waveform's voltage values. Find the "bucket" in the histogram above the center level (that is,*(Vmin + Vmax)/2)* that contains the most data points. Use this value for the TOP.
- Find the "bucket" in the histogram below the center level (that is, *(Vmin + Vmax)/2)* that contains the most data points. Use this value for the BASE.
- If the number of points in the buckets corresponding to the TOP or BASE values is less than 5% of the total number of points in the waveform, use the absolute maximum or minimum values for the TOP and BASE values.

CENTROID

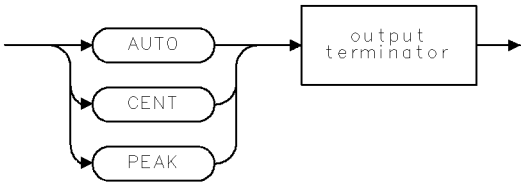
- Construct a histogram of the waveform's voltage values.
- Compute the weighted average of the "buckets" in the histogram from the center level to Vmax. Use this value as the TOP.
- Compute the weighted average of the "buckets" in the histogram from the center level to Vmin. Use this value as the BASE.

AUTO

- The same as PEAK except, if the number of points in the buckets corresponding to the TOP or BASE values is less than 5% of the total number of points in the waveform, use the centroid values for the TOP and BASE values.

The query returns the current state of ALGORITHM.

Query Response



QMEA53

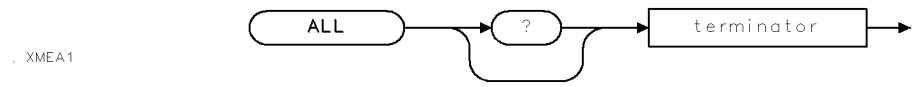
Query Example

```
OUTPUT @Mta;"MEAS:ALG?"
ENTER @Mta;Algorithm$
```

ALL

Makes as many measurements as possible on the defined source waveform.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:ALL"
```

Description

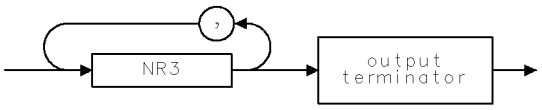
The ALL command makes as many measurements as possible on the defined source waveform. Any response for any measurement that cannot be made (for example, trying to measure a period when there is only one edge in the measurement) will be FAILVALUE. The markers are positioned at the rising or falling edges that define one period.

The query returns the results of the measurements.

The values of the response are as listed in order below:

frequency:	MEASure:FREQency?
period:	MEASure:PERiod?
positive width:	MEASure:PWIDth?
negative width:	MEASure:NWIDth?
rise time:	MEASure:RISE?
fall time:	MEASure:FALL?
amplitude:	MEASure:VAMPlitude?
peak-to-peak:	MEASure:VPP?
preshoot:	MEASure:PREShoot?
overshoot:	MEASure:OVERshoot?
duty cycle:	MEASure:DUTy?
rms voltage:	MEASure:VRMS?
maximum voltage:	MEASure:VMAX?
minimum voltage:	MEASure:VMIN?
top voltage:	MEASure:VTOP?
base voltage:	MEASure:VBASe?

Query Response



. QMEA1

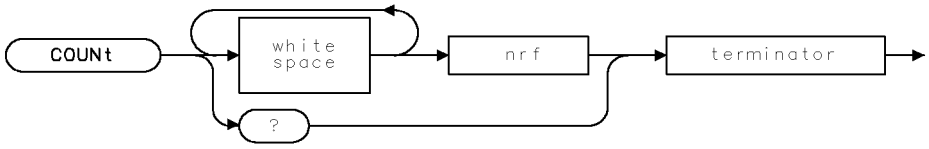
Query Example

```
OUTPUT @Mta;"MEAS:All?"
ENTER @Mta;Freq,Period,Pwid,Nwid,Rise,Fall,Ampl,Pkpk,Pres,
  Over,Duty,Rms,Max,Min,Top,Base
```

COUNt

Sets the average count for averaged measurements.

Syntax



Preset State: 16
Parameter range: width = 1 through 1024
Related commands: MEASure:TYPE

Command Example

```
OUTPUT @Mta;"MEAS:COUN 55"
```

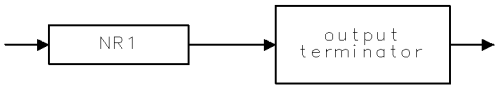
Description

The COUNT command sets the average count for averaged measurements. The query returns the current value of COUNT.

The average operation is implemented as a moving average, that is;

$$value = \frac{(averages - 1) \times old\ value + new\ value}{averages}$$

Query Response



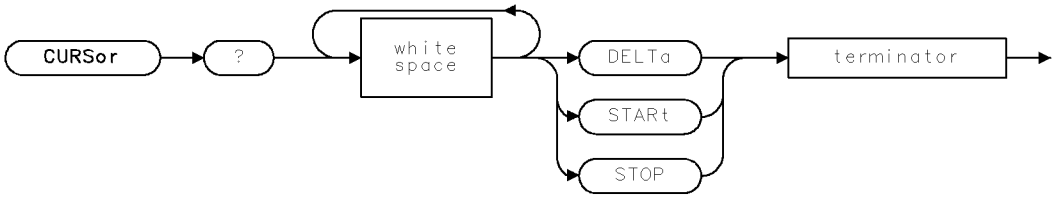
Query Example

```
OUTPUT @Mta;"MEAS:COUN?"
ENTER @Mta;Count
```

CURSor

Returns the specified marker as an ordered pair of time and voltage values.

Syntax



Description

The CURSOR query returns the specified marker as an ordered pair of time and voltage values. DELTA specifies the time and voltage difference between the markers. START refers to marker 1. STOP refers to marker 2.

Query Response

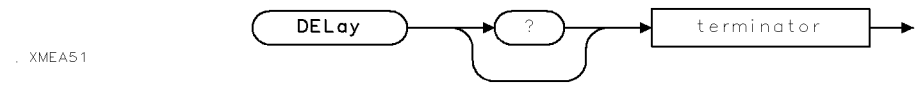
Query Example

```
OUTPUT @Mta;"MEAS:CURS? STAR"
ENTER @Mta;Time,Voltage
```

DELay

Measures the time difference between edges on a trace.

Syntax



Related Commands: MEASure:ESTArt
MEASure:ESTOp
MEASure:REFerence
MEASure:SOURce
MEASure:VSTArt
MEASure:VSTOp

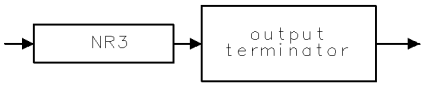
Command Example

```
OUTPUT @Mta;"MEAS:SOUR FUNC1"
OUTPUT @Mta;"MEAS:REF FUNC2"
OUTPUT @Mta;"MEAS:VSTA 75 PCT"
OUTPUT @Mta;"MEAS:ESTA 1"
OUTPUT @Mta;"MEAS:VSTO 50 PCT"
OUTPUT @Mta;"MEAS:ESTO -1"
OUTPUT @Mta;"MEAS:DEL"
```

Description

The DELAY command measures the time difference between an edge on the trace specified by MEASURE:SOURCE and an edge on the trace specified by MEASURE:REFERENCE. The edge on MEASURE:SOURCE is defined with the MEASURE:ESTART (edge and slope) and MEASURE:VSTART (threshold) commands. The edge on MEASURE:REFERENCE is defined with the MEASURE:ESTOP (edge and slope) and MEASURE:VSTOP (threshold) commands.

Query Response



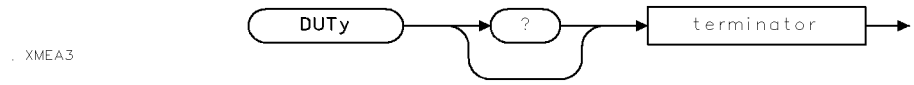
Query Example

```
OUTPUT @Mta;"MEAS:DELAY?"
ENTER @Mta;Delay
```

DUTy

Measures the duty cycle of the signal.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:DUT"
```

Description

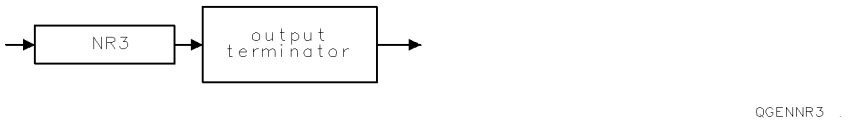
The DUTY cycle command measures the duty cycle of the signal. The duty cycle is defined as:

duty cycle (%) = (positive pulse width/period) × 100%

The positive pulse width is defined as the delta time between a rising edge of the waveform and the following falling edge. The period is defined as the delta time between two adjacent rising or falling edges. If the measurement cannot be made, FAILVALUE will be returned. The markers are placed at the rising or falling edges that define one period of the waveform.

The query returns the measured duty cycle.

Query Response



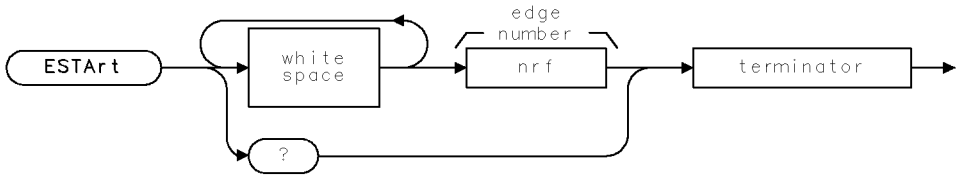
Query Example

```
OUTPUT @Mta;"MEAS:DUT?"
ENTER @Mta;Duty_cycle
```

ESTArt
(Start Marker Edge Number)

Positions marker 1 at a specified intersection of the threshold and the waveform.

Syntax



Preset State: 1
Parameter Range: edge number = 1 to number of edges on display screen

Command Example

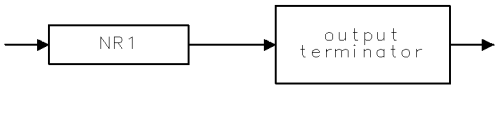
```
OUTPUT @Mta;"MEAS:ESTA 1"
```

Description

The ESTART command positions marker 1 at a specified intersection of the threshold defined by MEASURE:VSTART and the waveform specified by MEASURE:SOURCE. The sign of the parameter indicates the slope and the value indicates the intersection number. For example, −3 specifies the third negative slope intersection of the waveform and the threshold.

The query returns the currently specified edge number of marker 1.

Query Response



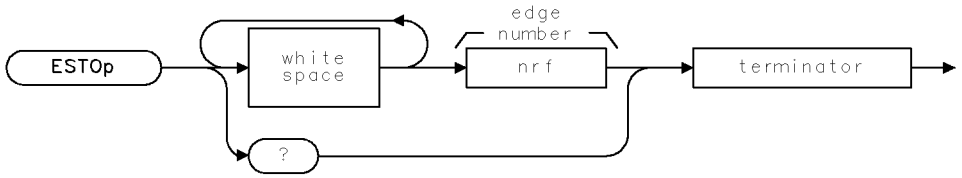
Query Example

```
OUTPUT @Mta;"MEAS:ESTA?"
ENTER @Mta;Edge
```

ESTOp
(Stop Marker Edge Number)

Positions marker 2 at a specified intersection of the threshold and the waveform.

Syntax



Preset State: 1
Parameter Range: edge number = 1 to number of edges on display screen

Command Example

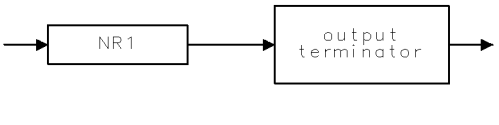
```
OUTPUT @Mta;"MEAS:ESTO 1"
```

Description

The ESTOP command positions marker 2 at a specified intersection of the threshold defined by MEASURE:VSTOP and the waveform specified by MEASURE:REFERENCE. The sign of the parameter indicates the slope and the value indicates the intersection number. For example, −3 specifies the third negative slope intersection of the waveform and threshold.

The query returns the currently specified edge number of marker 2.

Query Response



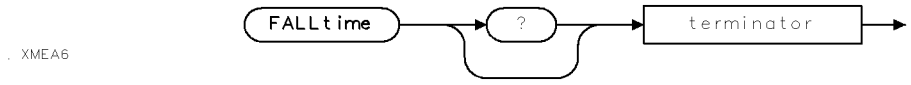
Query Example

```
OUTPUT @Mta;"MEAS:ESTO?"
ENTER @Mta;Edge
```

FALLtime

Measures fall time.

Syntax



Command Example

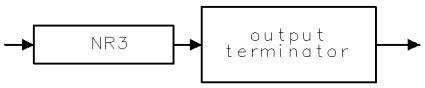
```
OUTPUT @Mta;"MEAS:FALL"
```

Description

The FALLTIME command measures the fall time of the first falling edge that intersects both the upper and lower voltage thresholds. If the measurement cannot be made, FAILVALUE is returned. The markers are positioned at the intersections of the upper or lower thresholds and the waveform.

The query returns the measured fall time.

Query Response



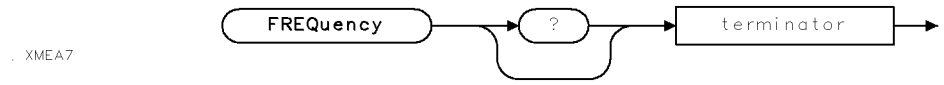
Query Example

```
OUTPUT @Mta;"MEAS:FALL?"
ENTER @Mta;Fall_time
```

FREQuency

Measures frequency.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:FREQ"
```

Description

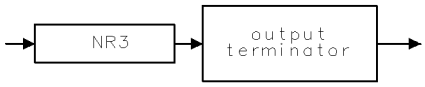
The FREQUENCY command measures the frequency of the first complete period of the waveform. The frequency is computed from the measured period:

frequency = 1/period.

The period is defined as the delta time between two adjacent rising or falling edges. If the measurement cannot be made, FAILVALUE is returned. The markers are positioned at the points used to determine the period.

The query returns the measured frequency of the waveform.

Query Response



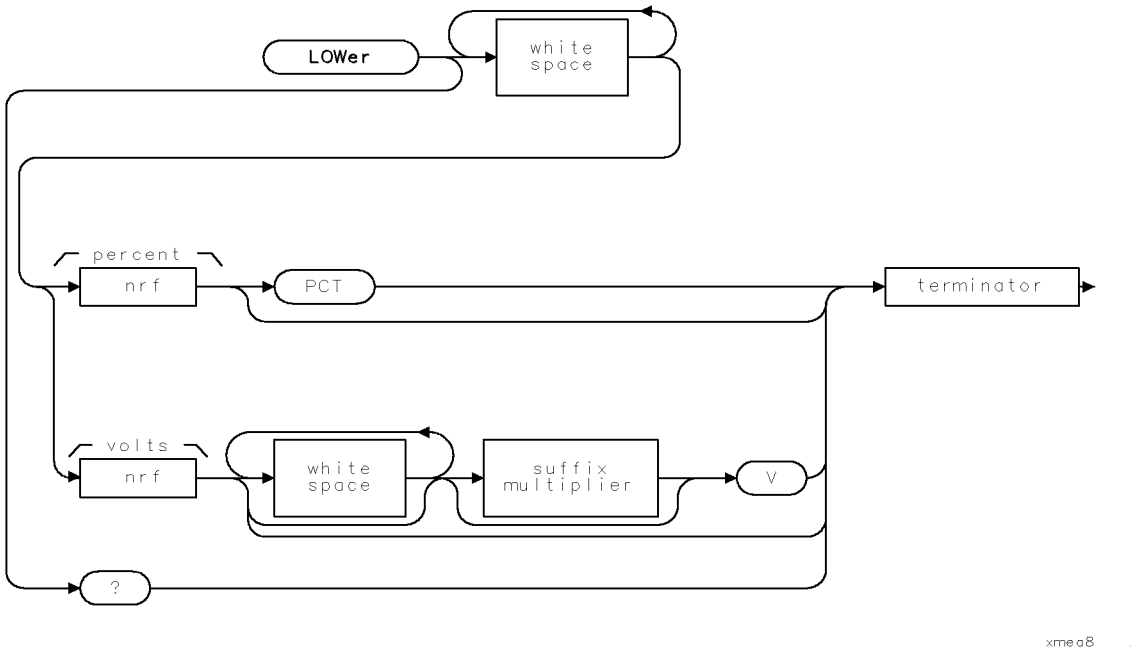
Query Example

```
OUTPUT @Mta;"MEAS:FREQ?"
ENTER @Mta;Freq
```

LOWer

Sets the lower measurement threshold.

Syntax



Preset State: 10%
Parameter Range: 0 to 50 for percent; VMIN to MEASure:UPPer for volts
Fundamental Unit: volts

Command Example

See the query example below.

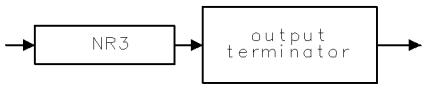
Description

The LOWER command sets the lower measurement threshold in either percent or volts. This value is used for the lower threshold for rise and fall time measurements when the MODE command of the MEASURE subsystem is specified as USER. When the lower threshold value is entered in volts, the range checking is done when a measurement is performed.

The LOWER threshold command works in conjunction with the MODE, UNIT, and UPPER threshold commands of the MEASURE subsystem. Refer to the respective command description in this section for more detailed information.

The query returns the current value of the lower measurement threshold.

Query Response



QGENNR3

Query Example

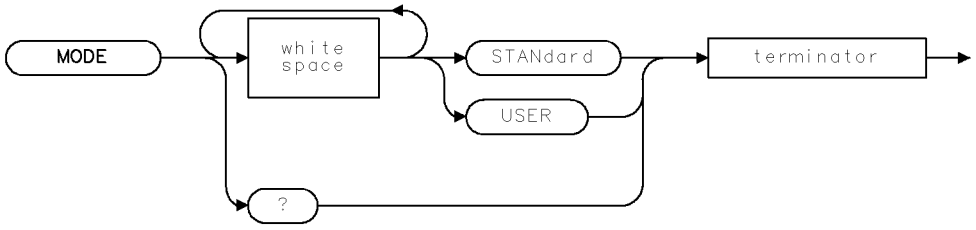
For a 20 percent to 80 percent rise time measurement

```
OUTPUT @Mta;"MEAS:MODE USER";
OUTPUT @Mta;"MEAS:UNIT PERC";        ! Units in percent.
OUTPUT @Mta;"MEAS:LOW 20";           ! Lower threshold at 20
percent.
OUTPUT @Mta;"MEAS:UPP 80";           ! Upper threshold at 80
percent.
OUTPUT @Mta;"MEAS:RISE?"             ! Measure rise time.
ENTER @Mta;Rise
```

MODE

Select measurement thresholds.

Syntax



Preset State: STANdard

Command Example

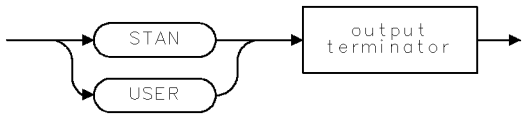
```
OUTPUT @Mta;"MEAS:MODE USER"
```

Description

The MODE command determines how rise and fall time measurements are to be made. STANDARD specifies the upper and lower thresholds as 10 percent and 90 percent. USER specifies the upper and lower threshold values defined by the LOWER threshold, UPPER threshold, and UNIT commands.

The query returns the current measurement mode.

Query Response



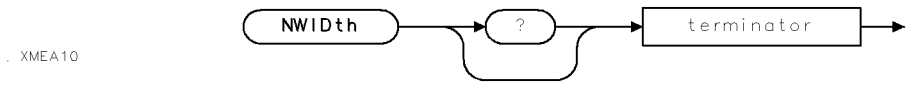
Query Example

```
OUTPUT @Mta;"MEAS:MODE?"
ENTER @Mta;Mode$
```

NWIDth
(Negative Pulse Width)

Measures the negative pulse width.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:NWID"
```

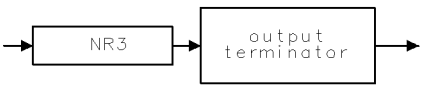
Description

The NWIDTH command measures the negative pulse width. The negative pulse width is defined as the delta time between the first falling edge and the following rising edge of the waveform. That is to say:

negative pulse width = −(time of first falling edge) + (time of next rising edge)

The query returns the measured negative pulse width of the waveform. If the measurement cannot be made, FAILVALUE is returned. The markers are placed at the points used to determine the negative pulse width.

Query Response



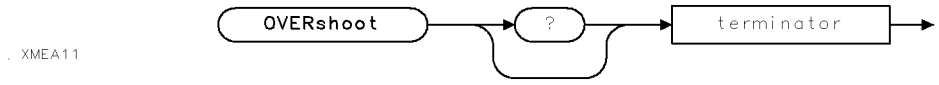
Query Example

```
OUTPUT @Mta;"MEAS:NWID?"
ENTER @Mta;Neg_width
```

OVERshoot

Measures the overshoot.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:OVER"
```

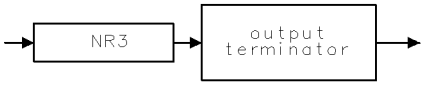
Description

The OVERSHOOT command measures the overshoot of a waveform. The markers are positioned at the rising or falling edges of one period. The overshoot measurement uses the following algorithm:

- If the first edge is positive, then overshoot = Vmax − TOP
- If the first edge is negative, then overshoot = BASE − Vmin

The query returns the measured overshoot value.

Query Response



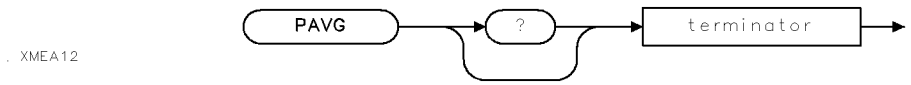
Query Example

```
OUTPUT @Mta;"MEAS:OVER?"
ENTER @Mta;Overshoot
```

PAVG (Average Power)

Measures the average power.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:PAVG"
```

Description

The PAVG command measures the average power of one complete period of the waveform. The markers are positioned on the edges that define one period.

The computation of average power is as follows:

$$average\ power = SUM(V(j)^2/50)/n$$

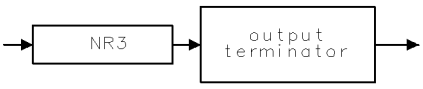
where j indexes over one period of the waveform.

The result is in dBm.

Note The average power operation assumes that the input is the envelope of the measured data so the conversion to dBm includes an adjustment that converts to rms power for a sinusoid. The amount of this adjustment is 3.01 dB ($10 \times LOG_{10}(2)$).

The query returns the measured average power.

Query Response



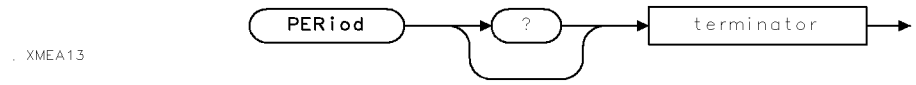
Query Example

```
OUTPUT @Mta;"MEAS:PAVG?"
ENTER @Mta;Pavg
```

PERiod

Measures the period.

Syntax



Command Example

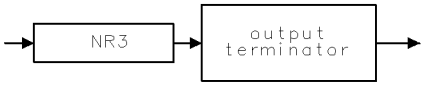
```
OUTPUT @Mta;"MEAS:PER"
```

Description

The PERIOD command measures the period of the waveform by determining the delta time between two adjacent rising or falling edges. If the measurement cannot be made, FAILVALUE is returned. The markers are positioned at the points used to determine the period.

The query returns the measured period.

Query Response



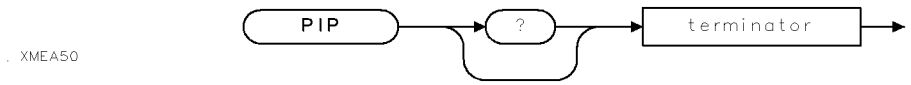
Query Example

```
OUTPUT @Mta;"MEAS:PER?"
ENTER @Mta;Period
```

PIP
(Peak Instantaneous Power)

Measures the peak instantaneous power.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:PIP"
```

Description

The PIP command measures the peak instantaneous power of the waveform.

Peak instantaneous power is measured as:

$$v = max(abs(vmax), abs(vmin))$$

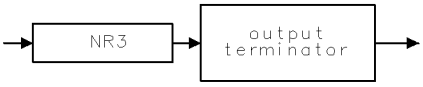
$$pip = v^2/50$$

The result is in dBm.

Note The conversion to dBm includes an adjustment that converts rms power for a sinusoid. The amount of this adjustment is 3.01 dB ($10 \times LOG_{10}(2)$).

The query returns the measured peak instantaneous power.

Query Response



QGENNR3

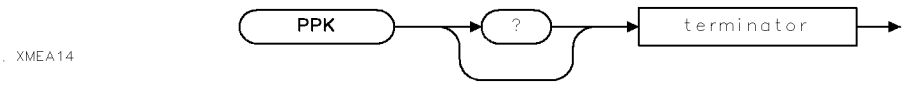
Query Example

```
OUTPUT @Mta;"MEAS:PIP?"
ENTER @Mta;Pip
```

PPK
(Peak Power)

Measures the peak power.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:PPK"
```

Description

The PPK command measures the peak power of the waveform.

PPK is intended for use with RF pulse signals. Peak power is defined as the total transmitted energy for one cycle divided by the pulse width. The actual computation of peak power is as follows:

1. Determine the 50 percent positive pulse width by using the following equation:

$$0.5 \times (TOP + BASE)$$

2. Compute the total transmitted energy:

$$energy = SUM\ dt \times Pi$$

where dt is the time width of each data point and Pi is the instantaneous power.

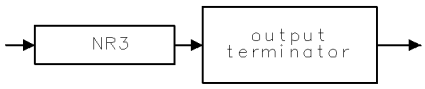
Note If there are one or more cycles of the signal, the summation occurs over just one cycle. Otherwise, the summation occurs over the entire data set.

3. Divide the total transmitted energy by the pulse width.

The result is in dBm. The query returns the measured peak power.

Note This operation assumes that the input is the envelope of the measured data. Therefore, the conversion to dBm includes an adjustment to convert to rms power for a sinusoid. The amount of this adjustment is 3.01 dB (10(2)).

Query Response



QGENNR3

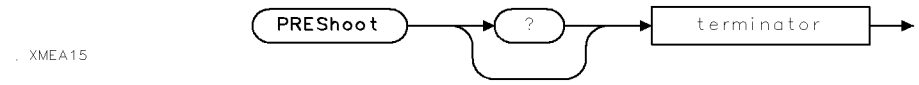
Query Example

```
OUTPUT @Mta;"MEAS:PPK?"
ENTER @Mta;Ppk
```

PREShoot

Measures the preshoot of the defined waveform.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:PRES"
```

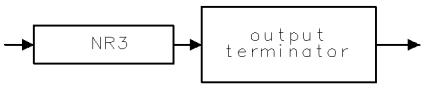
Description

The PRESHOOT command measures the preshoot of the defined waveform. The markers are positioned at the rising or falling edges of one period. The preshoot measurement uses the following algorithm:

- If the first edge is positive, then preshoot = BASE – Vmin
- If the first edge is negative, then preshoot = Vmax – TOP

The query returns the measured overshoot value.

Query Response



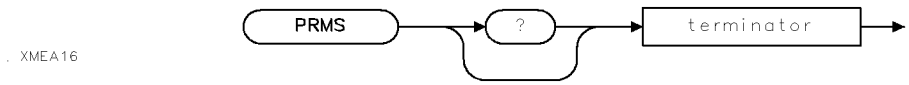
Query Example

```
OUTPUT @Mta;"MEAS:PRES?"
ENTER @Mta;Preshoot
```

PRMS
(RMS Power)

Measures the rms power.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:PRMS"
```

Description

The PRMS command measures the rms power of one complete period of the waveform. The markers are positioned on the edges that define one period.

The computation of rms power is as follows:

$$rms\ power = SQRT(SUM((V(j)^2/50)^2)/n)$$

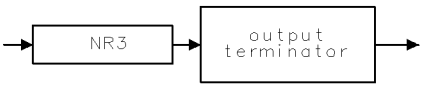
where j indexes over one period of the waveform.

The result is in dBm.

Note The rms power operation assumes that the input is the envelope of the measured data so the conversion to dBm includes an adjustment that converts to rms power for a sinusoid. The amount of this adjustment is 3.01 dB ($10_{10}(2)$).

The query returns the measured rms power.

Query Response



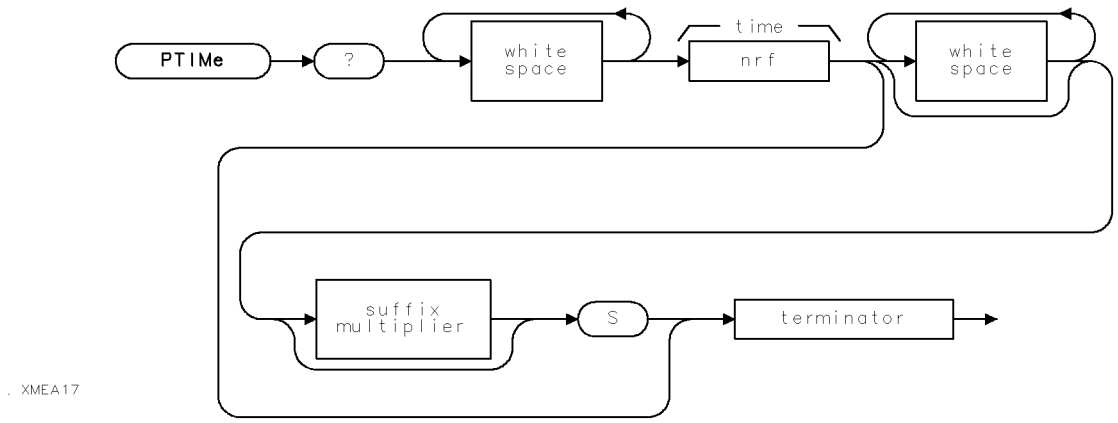
Query Example

```
OUTPUT @Mta;"MEAS:PRMS?"
ENTER @Mta;Prms
```

PTIMe
(Preceding Point of Requested Time)

Returns the nearest point at or preceding the requested time.

Syntax



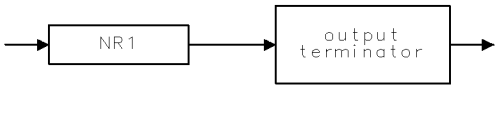
Parameter Range: time delay − time range/2 ≤ time ≤
= (time delay+time range/2) − (time range/trace length)

Fundamental Unit: seconds

Description

The PTIME query returns the nearest point at or preceding the requested time. (The first point is the zero point.)

Query Response



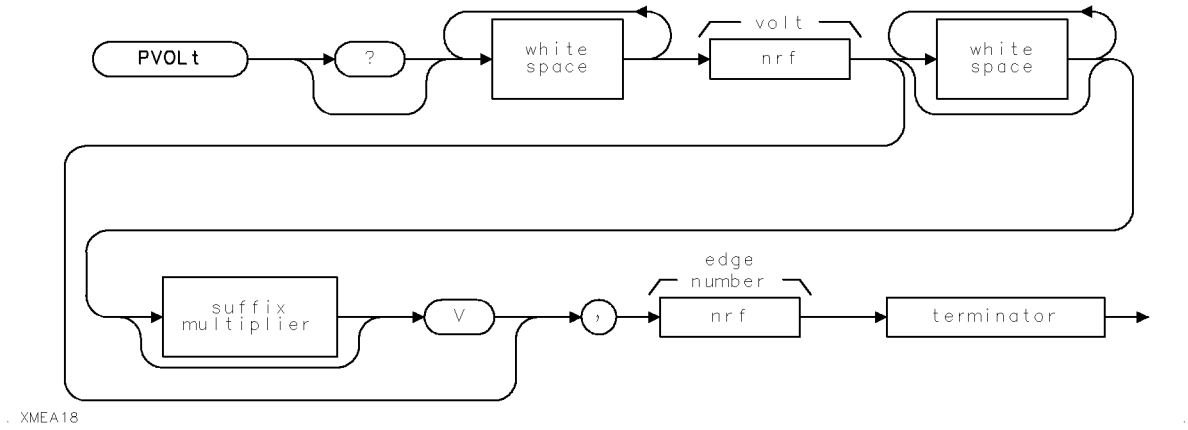
Query Example

```
OUTPUT @Mta;"MEAS:PTIM? 5 ns"
ENTER @Mta;Point
```

PVOLt
(Point of Specified Voltage/Intersection)

Positions marker 1 at the intersection of the waveform and the specified voltage level.

Syntax



Parameter Range: time delay − time range/2 ≤ volt ≤
(time delay+time range/2) − (time range/trace length)
(Note: accepts −32768 through 32767; however, for successful operation, enough edges in data are required.)

Fundamental Unit: volts

Command Example

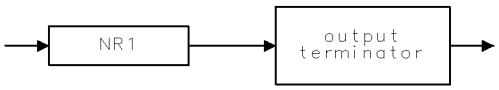
```
OUTPUT @Mta;"MEAS:PVOL 10 mV, -3"
```

Description

The PVOLT command positions marker 1 at the intersection of the waveform and the specified voltage level. If the edge parameter is positive, the intersection must be positive-going. If the edge parameter is negative, the intersection must be negative-going.

The query returns the point number of the intersection.

Query Response



. QGENN1

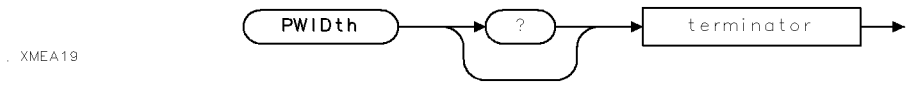
Query Example

```
OUTPUT @Mta;"MEAS:PVOL? 10 mV, -3"
ENTER @Mta;Point
```

PWIDth
(Positive Pulse Width)

Measures the positive pulse width.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:PWID"
```

Description

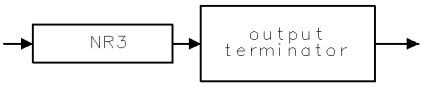
The PWIDTH command measures the positive pulse width. The positive pulse width is defined as the delta time between the first rising edge and the following falling edge of the waveform. That is,

positive pulse width = −(time of first rising edge) + (time of next falling edge)

If the measurement cannot be made, FAILVALUE is returned. The markers are placed at the points used to determine the positive pulse width.

The query returns the measured positive pulse width of the waveform.

Query Response



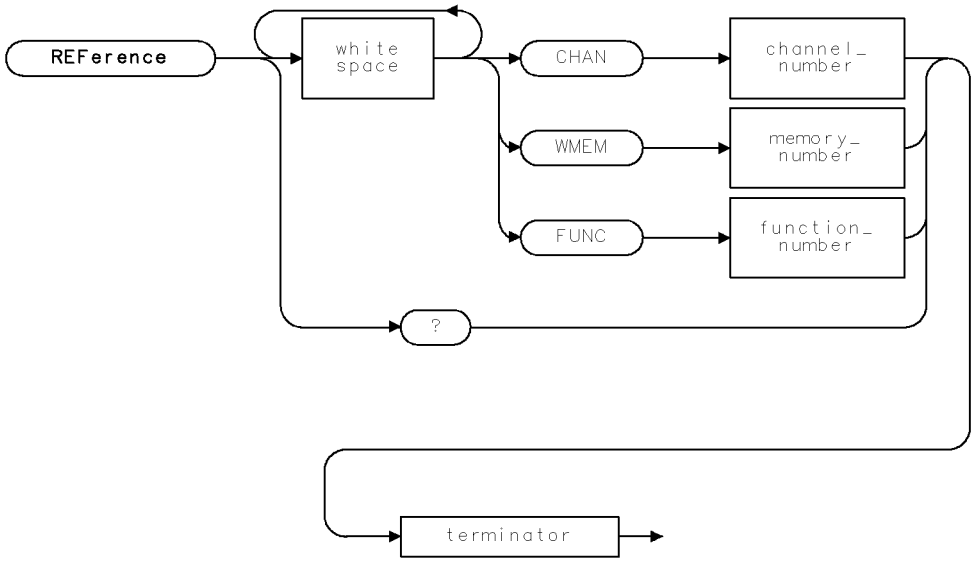
Query Example

```
OUTPUT @Mta;"MEAS:PWID?"
ENTER @Mta;Pos_width
```

REFerence

Selects the reference source of data for subsequent operations.

Syntax



Preset State: FUNC1

Command Example

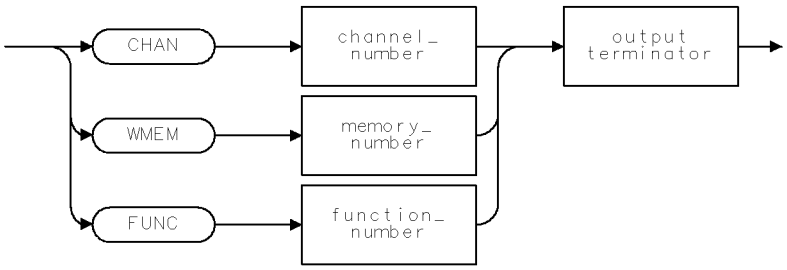
```
OUTPUT @Mta;"MEAS:REF WMEM2"
```

Description

The REFERENCE command selects the reference source of data for subsequent operations in the MEASURE subsystem. The source may be selected from any available channel, waveform memory, or function.

The query returns the current source.

Query Response



. QMEA21 .

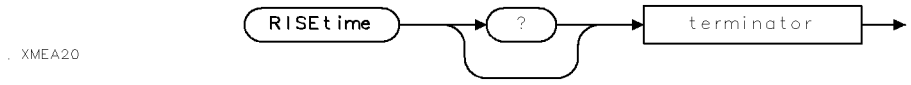
Query Example

```
OUTPUT @Mta;"MEAS:REF?"
ENTER @Mta;Reference$
```

RISEtime

Measures the rise time.

Syntax



Command Example

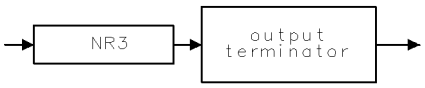
```
OUTPUT @Mta;"MEAS:RISE"
```

Description

The RISETIME command measures the rise time of the first rising edge that intersects both the upper and lower voltage thresholds. If the measurement cannot be made, FAILVALUE is returned. The markers are positioned at the intersections of the upper or lower thresholds and the waveform.

The query returns the measured rise time.

Query Response



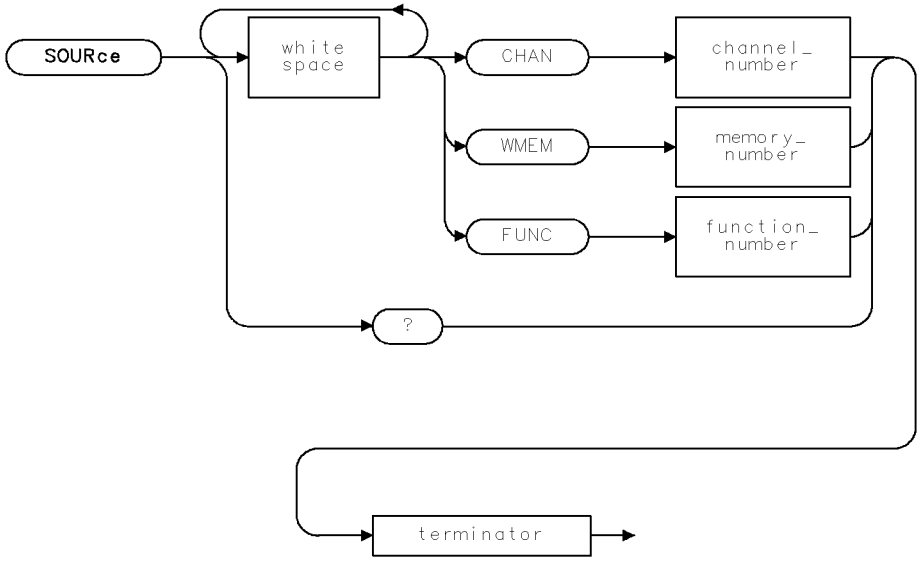
Query Example

```
OUTPUT @Mta;"MEAS:RISE?"
ENTER @Mta;Rise_time
```

SOURce

Sets the measurement data source.

Syntax



Preset State: FUNC1

Command Example

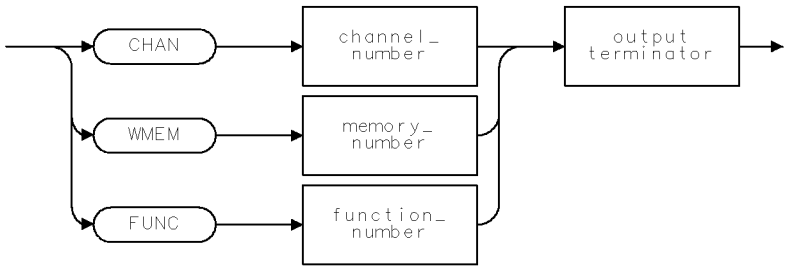
```
OUTPUT @Mta;"MEAS:SOUR WMEM2"
```

Description

The SOURCE command selects the source of data for subsequent operations in the MEASURE subsystem. The source may be selected from any available channel, waveform memory, or function.

The query returns the current source.

Query Response



. QMEA21

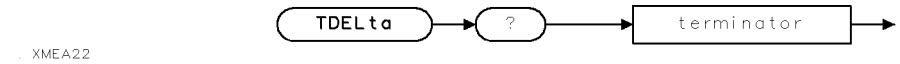
Query Example

```
OUTPUT @Mta;"MEAS:SOUR?"
ENTER @Mta;Source$
```

TDELta
(Time Delta)

Returns the time difference between marker 1 and marker 2.

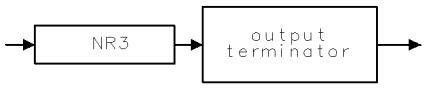
Syntax



Description

The TDELTA query returns the time difference between marker 1 and marker 2.

Query Response



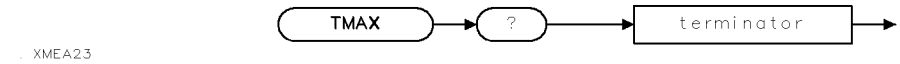
Query Example

```
OUTPUT @Mta;"MEAS:TDEL?"
ENTER @Mta;Time_delta
```

TMAX
(Time of First Occurrence of Maximum Voltage)

Returns the time of VMAX.

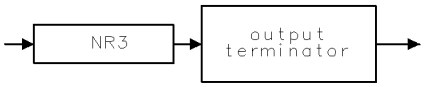
Syntax



Description

The TMAX query returns the time at which the first occurrence of the maximum voltage (VMAX) occurred.

Query Response



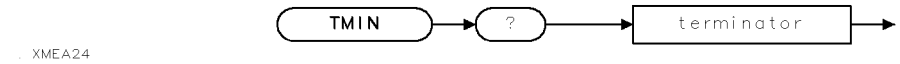
Query Example

```
OUTPUT @Mta;"MEAS:TMAX?"
ENTER @Mta;Time_of_vmax
```

TMIN
(Time of First Occurrence of Minimum Voltage)

Returns the time of VMIN.

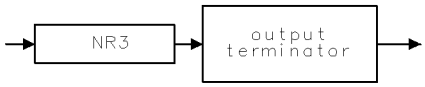
Syntax



Description

The TMIN query returns the time at which the first occurrence of the minimum voltage (VMIN) occurred.

Query Response



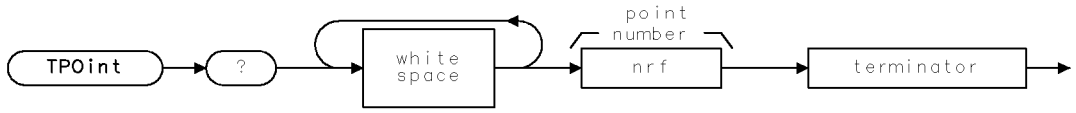
Query Example

```
OUTPUT @Mta;"MEAS:TMIN?"
ENTER @Mta;Time_of_vmin
```

TPOint
(Time of Specified Point)

Returns the time of the specified point.

Syntax



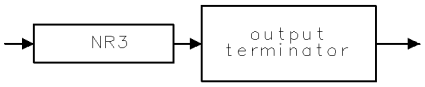
XMEA25

Parameter Range: point number = 0 to trace length − 1

Description

The TPOINT query returns the time of the specified point (zero is the first point.)

Query Response



QGENNR3

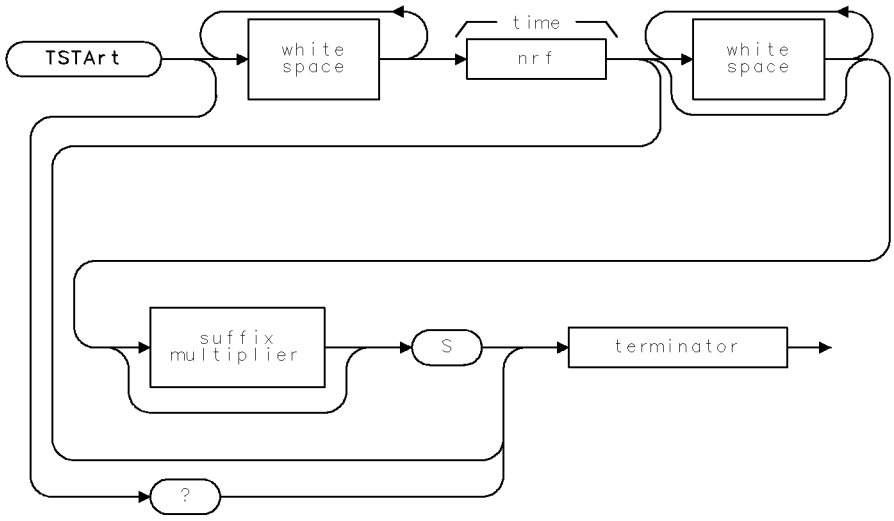
Query Example

```
OUTPUT @Mta;"MEAS:TPO? 50"
ENTER @Mta;Time_of_point50
```

TSTArt
(Time of Start Marker)

Sets the time position of marker 1.

Syntax



XMEA26

Preset State: 0
Parameter Range: minimum time ≤ time ≤ maximum time;
minimum time = time_delay − time_range/2,
maximum time = time_delay + time_range/2 −
: (time_range/trace_length)
Fundamental Unit: seconds

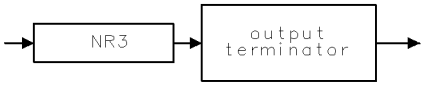
Command Example

```
OUTPUT @Mta;"MEAS:TSTA 10 us"
```

Description

The TSTART command sets the time position of marker 1. The query returns the position of marker 1.

Query Response



QGENNR3

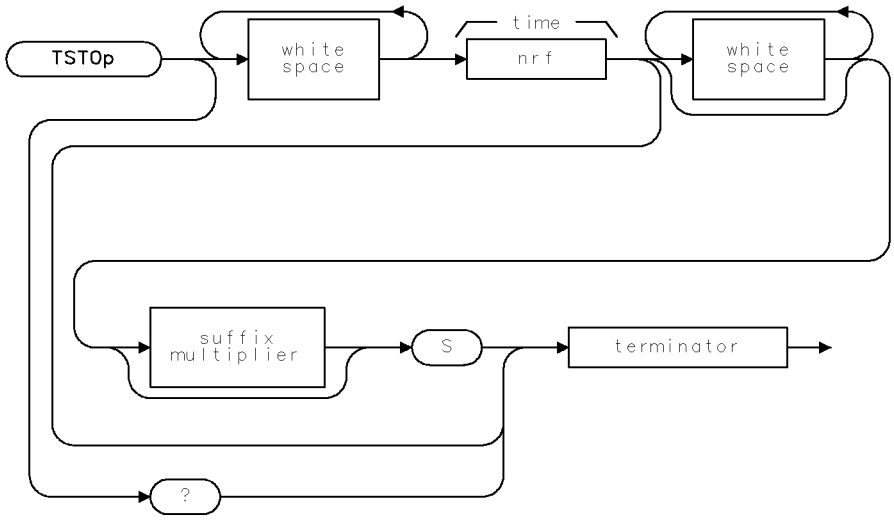
Query Example

```
OUTPUT @Mta;"MEAS:TSTA?"
ENTER @Mta;Start_marker
```

TSTOp
(Time of Stop Marker)

Sets the time position of marker 2.

Syntax



Preset State: 0
Parameter Range: minimum time ≤ time ≤ maximum time;
minimum time = time_delay − time_range/2,
maximum time = time_delay + time_range/2 −
(time_range/trace_length)
Fundamental Unit: seconds

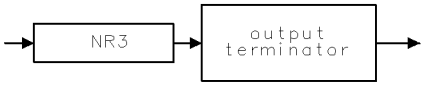
Command Example

```
OUTPUT @Mta;"MEAS:TSTO 10 us"
```

Description

The TSTOP command sets the time position of marker 2. The query returns the position of marker 2.

Query Response



QGENNR3

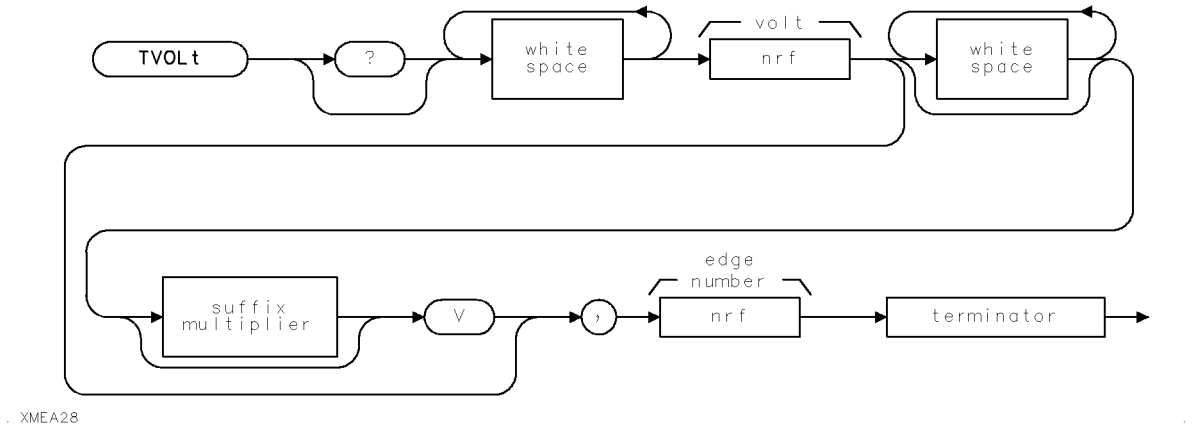
Query Example

```
OUTPUT @Mta;"MEAS:TSTO?"
ENTER @Mta;Stop_marker
```

TVOLt
(Time of Specified Voltage/Intersection)

Positions marker 1 at the intersection of the waveform and the specified voltage level.

Syntax



Parameter Range: volt offset − volt range/2 ≤ voltage level ≤ = volt offset + volt range/2

Fundamental Unit: volts

Command Example

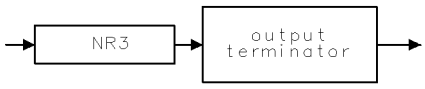
```
OUTPUT @Mta;"MEAS:TVOL 10 mV, -3"
```

Description

The TVOLT command positions marker 1 at the intersection of the waveform and the specified voltage level. If the edge parameter is positive, the intersection must be positive going. If the edge parameter is negative, the intersection must be negative going.

The query returns the time position of the intersection.

Query Response



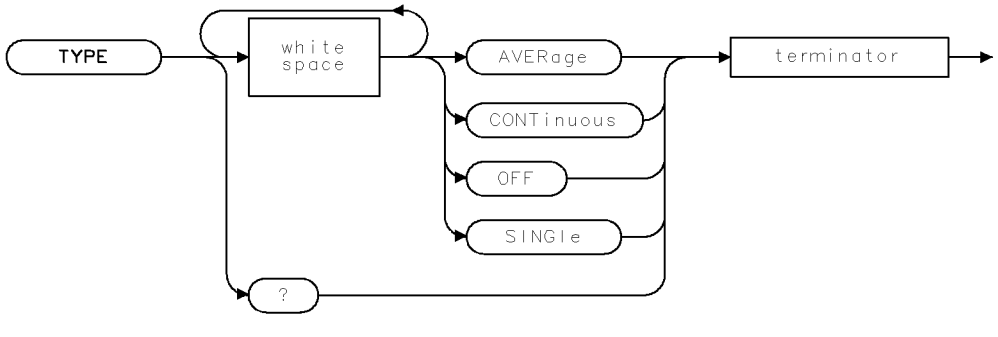
Query Example

```
OUTPUT @Mta;"MEAS:TVOL? 10 mV, -3"
ENTER @Mta;Time
```

TYPE

Controls the operation of the measurement subsystem.

Syntax



Preset State: OFF
Related Commands: MEASure:COUNt

Command Example

```
OUTPUT @Mta;"MEAS:TYPE CONT"
```

Description

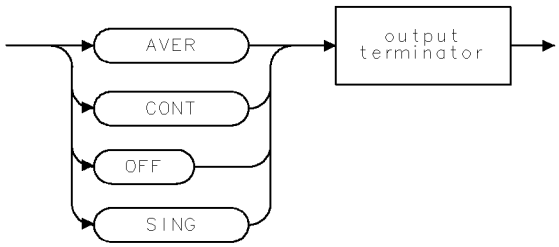
The TYPE command controls the operation of the measurement subsystem.

The query returns the current state of TYPE.

The possible states are:

OFF: Measure update is off, measurement results are removed from the screen.
SINGle: Perform a single measurement.
CONTinuous: Perform a measurement after each data acquisition.
AVERage: Perform a measurement after each data acquisition and average the results.

Query Response



QMEA48

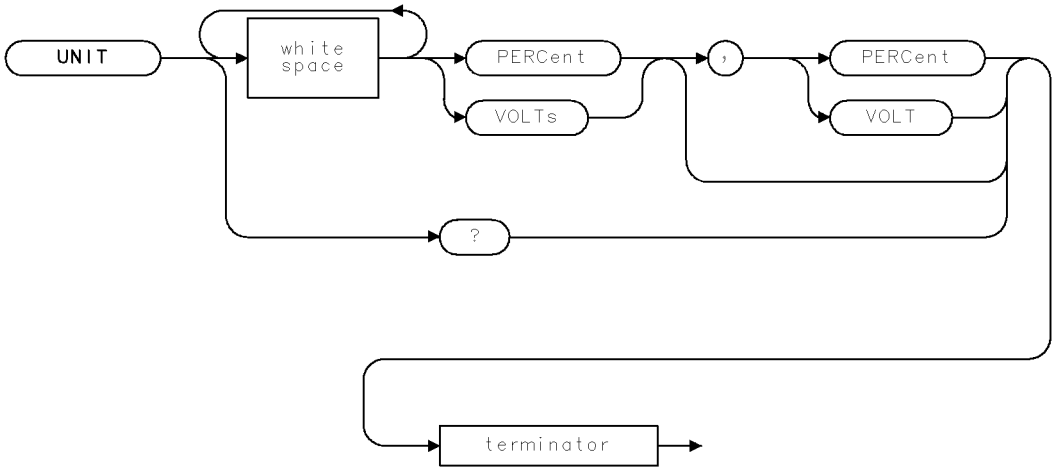
Query Example

```
OUTPUT @Mta;"MEAS:TYPE?"
ENTER @Mta;Type$
```

UNIT

Determines how the parameters for the commands MEASURE:LOWER and MEASURE:UPPER will be interpreted.

Syntax



XMEA29

Preset State: PERCent, PERCent

Command Example

See the query example below.

Description

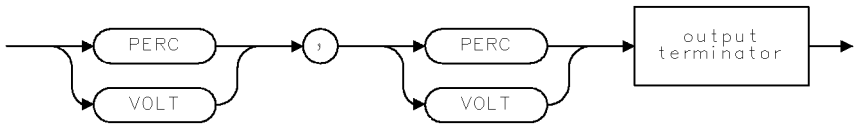
The UNIT command determines how the parameters for the commands MEASURE:LOWER and MEASURE:UPPER will be interpreted. Units may be specified as either PERCENT or VOLTS.

The first parameter describes the units for the UPPER threshold. The second, optional, parameter describes the units for the LOWER threshold. If the second parameter is omitted, the units for the LOWER threshold are set to the same value as the units for the UPPER threshold.

The UNIT command functions in conjunction with the MODE, UPPER threshold, and LOWER threshold commands of the MEASURE subsystem. Refer to the respective command descriptions in this section for more detailed information.

The query returns the currently specified threshold units.

Query Response



QMEA29

Query Example

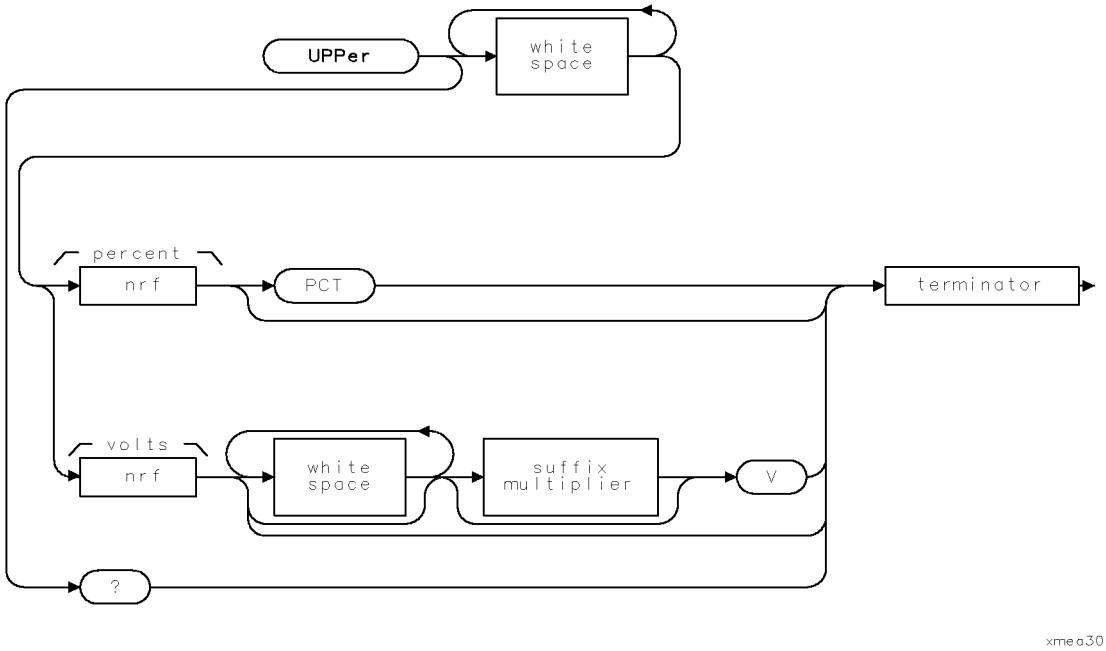
For a 20 percent, 80 percent rise time measurement

```
OUTPUT @Mta;"MEAS:MODE USER";
OUTPUT @Mta;"MEAS:UNIT PERC";          ! Units = percent.
OUTPUT @Mta;"MEAS:LOW 20";             ! Lower threshold at 20
percent.
OUTPUT @Mta;"MEAS:UPP 80";             ! Upper threshold at 80
percent.
OUTPUT @Mta;"MEAS:RISE?";              ! Rise time.
ENTER @Mta;Rise_time
```

UPPer

Sets the upper measurement threshold.

Syntax



Preset State: 90%
Parameter Range: percent = 50 to 100; MEASure:LOWer to VMAX for volts
Fundamental Unit: volts

Command Example

See the query example below.

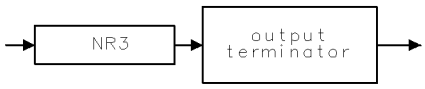
Description

The UPPER command sets the upper measurement threshold in either percent or volts. This value is used for the upper threshold for rise and fall time measurements when the MODE command of the MEASURE subsystem is specified as USER. When the upper threshold value is entered in volts, the range checking is done when a measurement is performed.

UPPER threshold works in conjunction with the LOWER threshold, MODE, and UNIT commands of the MEASURE subsystem. Refer to the respective command description in this section for more detailed information.

The query returns the current value of the upper measurement threshold.

Query Response



QGENNR3

Query Example

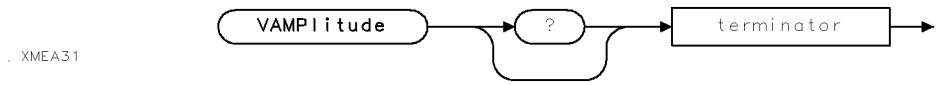
For a 20 percent to 80 percent rise time measurement

```
OUTPUT @Mta;"MEAS:MODE USER";
OUTPUT @Mta;"MEAS:UNIT PERC";          ! Units in percent.
OUTPUT @Mta;"MEAS:LOW 20";             ! Lower threshold at 20
percent.
OUTPUT @Mta;"MEAS:UPP 80";             ! Upper threshold at 80
percent.
OUTPUT @Mta;"MEAS:RISE?"               ! Measure rise time.
ENTER @Mta;Rise
```

VAMPlitude (Signal Amplitude)

Measures the signal amplitude.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:VAMP"
```

Description

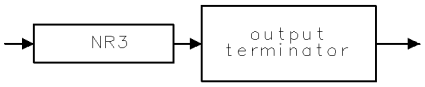
The VAMPLITUDE command measures the signal amplitude in volts and is based on:

amplitude = Vtop − Vbase.

The markers are positioned at their current time position and Vtop and Vbase.

The query returns the current signal amplitude value.

Query Response



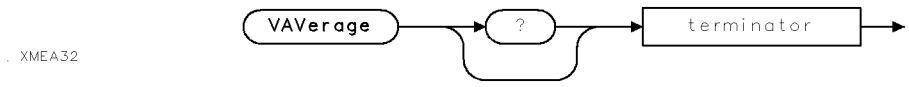
Query Example

```
OUTPUT @Mta;"MEAS:VAMP?"
ENTER @Mta;Amplitude
```

VAVerage (Average Voltage)

Measures the average voltage.

Syntax



Command Example

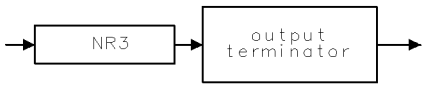
```
OUTPUT @Mta;"MEAS:VAV"
```

Description

The VAVERAGE command measures the average voltage of the waveform.

The query returns the measured average voltage value.

Query Response



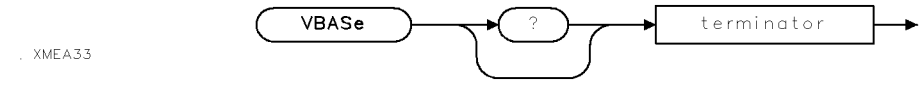
Query Example

```
OUTPUT @Mta;"MEAS:VAV?"
ENTER @Mta;Vaverage
```

VBASe
(Base Voltage Level)

Measures the base level.

Syntax



Command Example

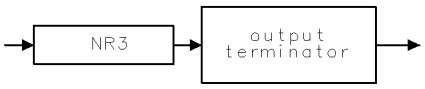
```
OUTPUT @Mta;"MEAS:VBAS"
```

Description

The VBASE command measures the voltage level of the base of the waveform. The markers are positioned at their current time position and Vtop and Vbase.

The query returns the measured base voltage level.

Query Response



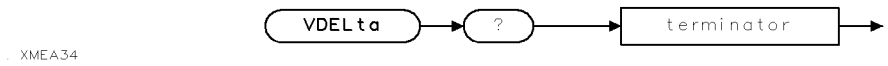
Query Example

```
OUTPUT @Mta;"MEAS:VBAS?"
ENTER @Mta;Vbase
```

VDELta
(Voltage Marker Delta)

Returns the voltage difference between marker 1 and marker 2.

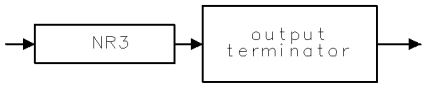
Syntax



Description

The VDELTA query returns the voltage difference between marker 1 and marker 2.

Query Response



Query Example

```
OUTPUT @Mta;"MEAS:VDEL?"
ENTER @Mta;Vdelta
```

VFIFty
(Voltage Markers to 50 Percent)

Sets markers to 50% of TOP and BASE values.

Syntax

Command Example

```
OUTPUT @Mta;"MEAS:VFIF"
```

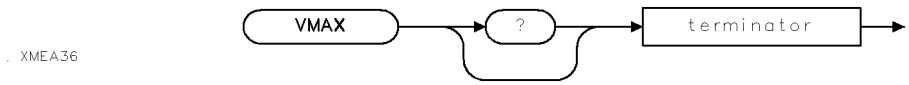
Description

The VFIFTY command measures the TOP and BASE values of the waveform and sets the markers at the 50 percent voltage level (0.5 × (TOP + BASE)).

VMAX
(Maximum Voltage of Signal)

Measures the maximum voltage.

Syntax



Command Example

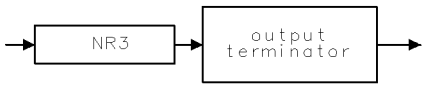
```
OUTPUT @Mta;"MEAS:VMAX"
```

Description

The VMAX command measures the maximum voltage of the waveform. The markers are positioned at VMAX and VMIN.

The query returns the measured maximum voltage of the waveform.

Query Response



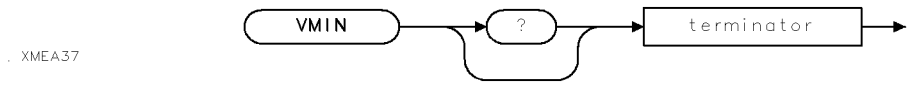
Query Example

```
OUTPUT @Mta;"MEAS:VMAX?"
ENTER @Mta;Vmax
```

VMIN
(Minimum Voltage of Signal)

Measures the minimum voltage.

Syntax



Command Example

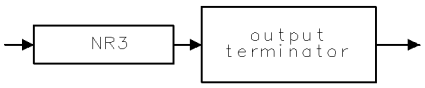
```
OUTPUT @Mta;"MEAS:VMIN"
```

Description

The VMIN command measures the minimum voltage of the waveform. The markers are positioned at VMAX and VMIN.

The query returns the measured minimum voltage of the waveform.

Query Response



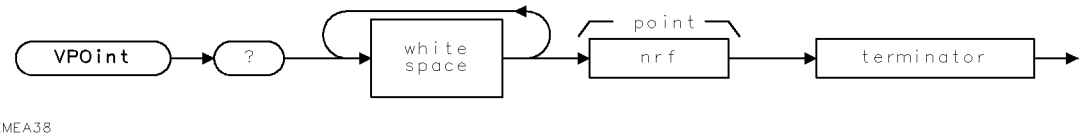
Query Example

```
OUTPUT @Mta;"MEAS:VMIN?"
ENTER @Mta;Vmin
```

VPOint
(Voltage of Specified Point)

Returns the value (in volts) of the specified point.

Syntax

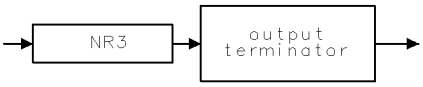


Parameter Range: point = 0 to trace length − 1

Description

The VPOINT query returns the value (in volts) of the specified point. (The first point is the zero point.)

Query Response



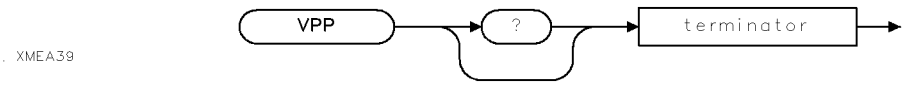
Query Example

```
OUTPUT @Mta;"MEAS:VPO? 25"
ENTER @Mta;V_at_point
```

VPP
(Signal Voltage Peak-to-Peak)

Measures the peak-to-peak voltage.

Syntax



Command Example

```
OUTPUT @Mta;"MEAS:VPP"
```

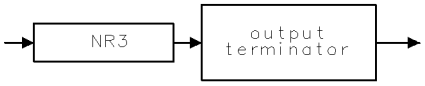
Description

The VPP command measures the peak-to-peak voltage of the waveform and is based on the following:

$$Vpp = Vmax - Vmin$$

The query returns the measured peak-to-peak voltage value. The markers are positioned at their current time and VMAX and VMIN.

Query Response



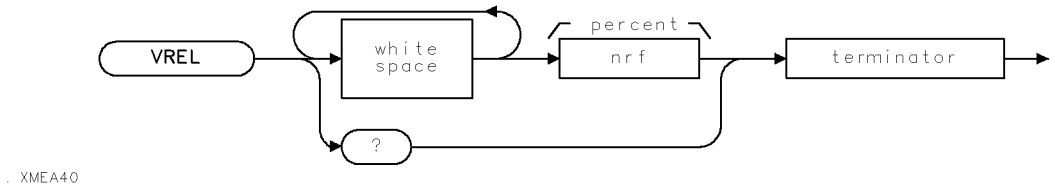
Query Example

```
OUTPUT @Mta;"MEAS:VPP?"
ENTER @Mta;Vpp
```

VREL
(Relative Voltage Marker Positioning)

Sets the marker voltage positions as a function of the last established values.

Syntax



Preset State: 100
Parameter Range: percent = 10, 20, 30, 40, 50, 100

Command Example

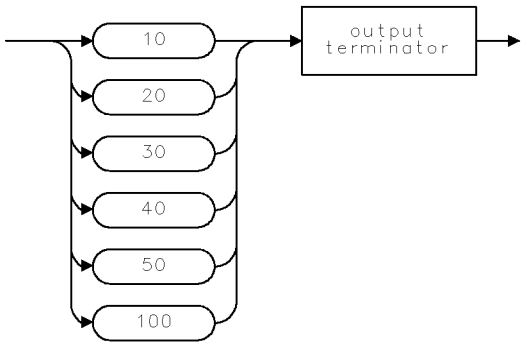
```
OUTPUT @Mta;"MEAS:VREL 20"
```

Description

The VREL command sets the marker voltage positions as a function of the last established values. For example, execute the VAMPLITUDE command toposition the voltage markers at the TOP and BASE of the waveform, then execute `VREL 10` to position marker 1 at 10 percent and marker 2 at 90 percent of the original positions.

The query returns the current relative voltage marker positioning.

Query Response



QMEA40

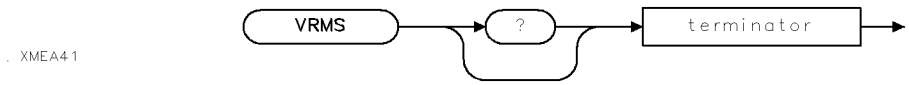
Query Example

```
OUTPUT @Mta;"MEAS:VREL?"
ENTER @Mta;Relative
```

VRMS
(RMS Voltage)

Measures the rms voltage.

Syntax



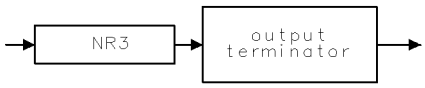
Command Example

```
OUTPUT @Mta;"MEAS:VRMS"
```

Description

The VRMS command measures the rms voltage of one complete period of the waveform. The query returns the current rms voltage. The markers are positioned on the edges that define one period.

Query Response



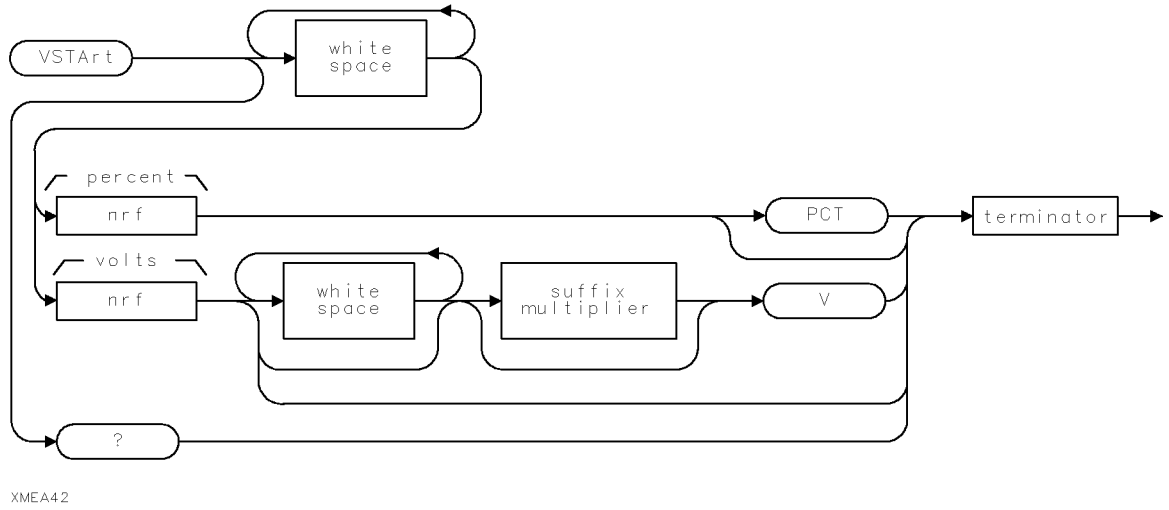
Query Example

```
OUTPUT @Mta;"MEAS:VRMS?"
ENTER @Mta;Vrms
```

VSTArt

Sets the threshold used by the ESTART and DELAY commands.

Syntax



Parameter Range: The least restrictive of:
volt offset−(0.625×volt range)≤ value≤volt offset+(0.625×volt range)
offset−(0.625×range)≤ value≤offset+(0.625×range)
where:
volt offset = WAVEFORM:VORIGIN
volt range = WAVEFORM:VRANGE
offset = DISPLAY:xxx:OFFSET
range = DISPLAY:xxx:RANGE
(that is, the marker can be positioned up to 12.5% outside the full scale measurement)
Percent = −100% through +200%

Fundamental Unit: volts

Command Example

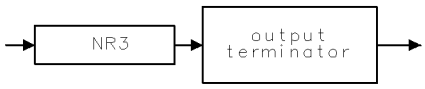
```
OUTPUT @Mta;"MEAS:VSTA 5.0 uV"
```

Description

The VSTART command sets the threshold used by the ESTART and DELAY commands on the trace defined by MEASURE:SOURCE. The threshold may be specified as an absolute level in volts or as a relative level in percent. If specified in percent, the level will be computed as BASE + percent * (TOP – BASE) / 100.

The query returns the current threshold.

Query Response



QGENNR3

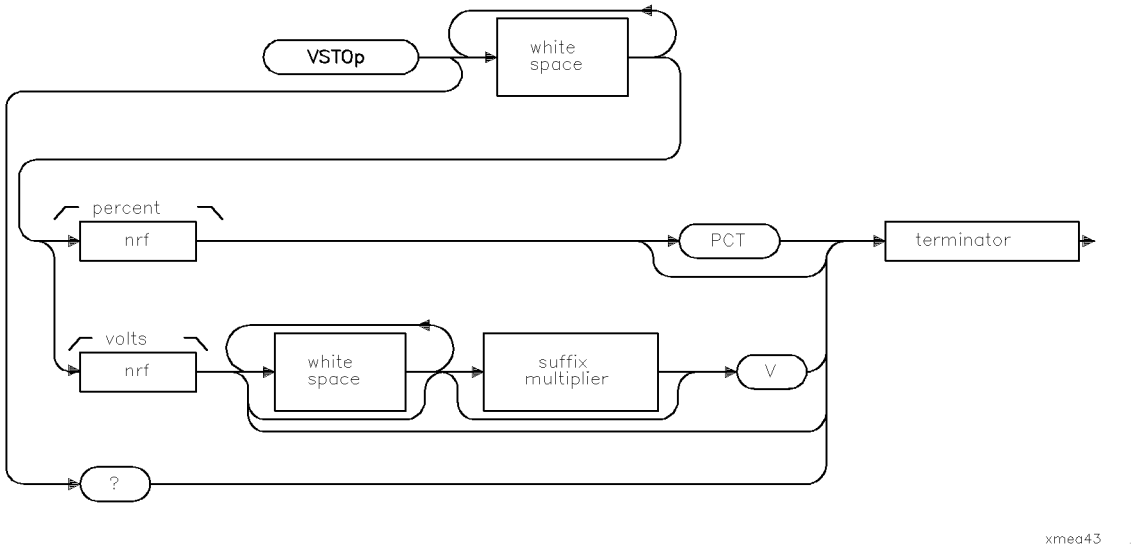
Query Example

```
OUTPUT @Mta;"MEAS:VSTA?"
ENTER @Mta;Vstart
```

VSTOp

Sets the threshold used by the ESTOP and DELAY commands.

Syntax



Parameter Range: The least restrictive of:
volt offset$-(0.625\times$volt range$)\leq$ value$\leq$volt offset$+(0.625\times$volt range)
offset$-(0.625\times$range$)\leq$ value$\leq$offset$+(0.625\times$range)
where:
volt offset = WAVEFORM:VORIGIN
volt range = WAVEFORM:VRANGE
offset = DISPLAY:xxx:OFFSET
range = DISPLAY:xxx:RANGE
(that is, the marker can be positioned up to 12.5% outside the full scale measurement)
Percent = -100% through $+200\%$

Fundamental Unit: volts

Command Example

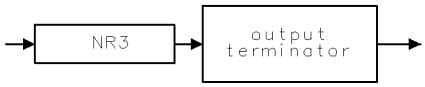
```
OUTPUT @Mta;"MEAS:VSTO 5.0 uV"
```

Description

The VSTOP command sets the threshold used by the ESTOP and DELAY commands on the trace defined by MEASURE:REFERENCE. The threshold may be specified as an absolute level in volts or as a relative level in percent. If specified in percent, the level will be computed as BASE + percent * (TOP - BASE) / 100.

The query returns the current threshold.

Query Response



QGENNR3

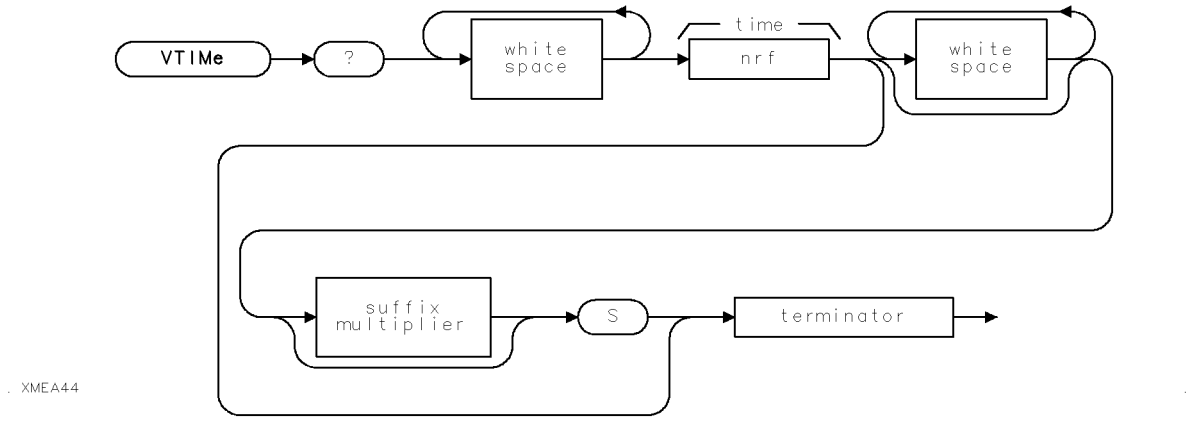
Query Example

```
OUTPUT @Mta;"MEAS:VSTO?"
ENTER @Mta;Vstop
```

VTIMe (Voltage at Specified Time)

Returns the value (in volts) of the waveform at the specified time.

Syntax



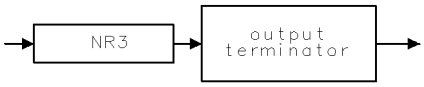
Parameter Range: time delay − time range/2 ≤ time ≤
= (time delay+time range/2) − (time range/trace length)

Fundamental Unit: seconds

Description

The VTIME query returns the value (in volts) of the waveform at the specified time. If the specified time does not lie on top of a measured point, the result is interpolated from the two closest points.

Query Response



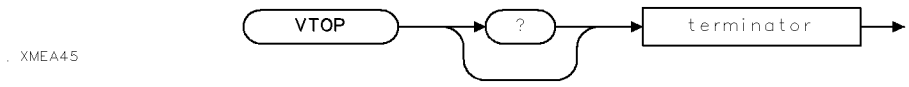
Query Example

```
OUTPUT @Mta;"MEAS:VTIM? 0"
ENTER @Mta;V_at_t
```

VTOP
(Top Voltage Level)

Measures the top voltage level.

Syntax



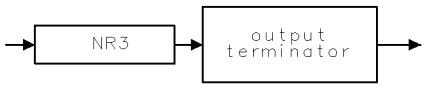
Command Example

```
OUTPUT @Mta;"MEAS:VTOP"
```

Description

The VTOP command measures the voltage level of the top of the waveform. The query returns the measured top voltage level. The markers are positioned at their current time position and Vtop and Vbase.

Query Response



Query Example

```
OUTPUT @Mta;"MEAS:VTOP?"
ENTER @Mta;Vtop
```

19

Modulator Source Subsystem

The MODULATOR SOURCE subsystem controls the pulse generator in the microwave transition analyzer. See the following list for a command syntax summary of the MODULATOR SOURCE subsystem commands.

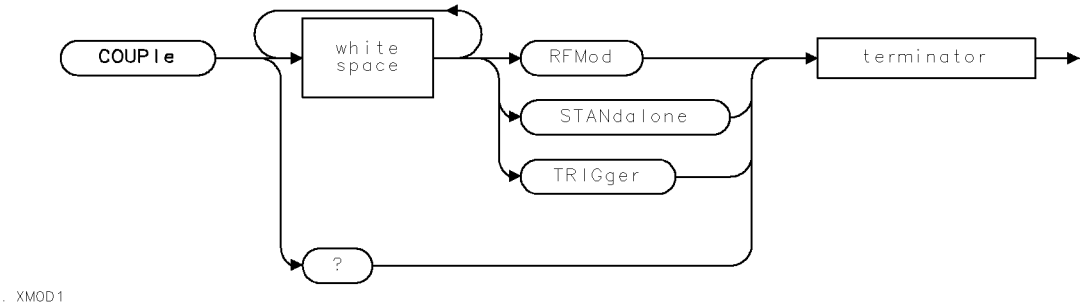
Note This means that if the 10 MHz reference is removed or fails, the internal modulator source stops working. When the module switches from internal to external reference, there will be a brief period when the modulator source is turned off. If the module is set to external reference and the reference is disconnected and then reconnected, the modulator source may go into a random state.

KEYWORD	PARAMETER FORM	NOTES
:MODulator		
:COUPle	RFMod|STANdalone| TRIGger	
:DUTY	<nrf>[PCT]	
:FREQuency	<nrf>[[<suffix multiplier>]HZ]	
:PERiod	<nrf>[[<suffix multiplier>]S]	
:POLarity	NEGative|POSitive	
:STATe	OFF|INTernal|MANual	
:WIDth	<nrf>[[<suffix multiplier>]S]	

COUPle

Couples the modulator to the signal frequency.

Syntax



Preset State: RFMod
Related Commands: MODulator:STATe
SWEep:TIME:TRANslate

Command Example

```
OUTPUT @Mta;"MOD:COUP OFF"
```

Description

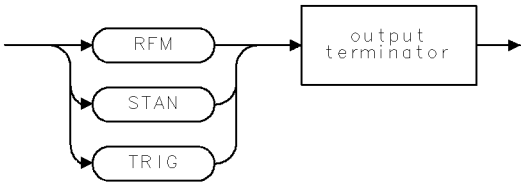
The COUPLE command enables or disables the coupling between the modulator source frequency and the frequency set by SWEEP:SIGNAL. When COUPLE is either RFMOD or TRIGGER and the modulator is turned on and the frequency is changed, then SWEEP:SIGNAL will be automatically set to the modulator frequency. For example, use COUPLE RFMOD if the modulator source is modulating the RF signal fed into channel 1. If COUPLE is STANDALONE and the modulator source frequency is changed, SWEEP:SIGNAL must also be changed to the same value.

The query returns the current state of COUPLE.

The MODULATOR:STATE and MODULATOR:COUPLE commands affect the state of the instrument as follows:

State	Couple	Mode
OFF	not applicable	Non-pulsed mode operation: the modulator frequency does not couple to anything.
INTERNAL	STANDALONE	Non-pulsed mode operation: the modulator frequency does not couple to anything.
INTERNAL	TRIGGER	Non-pulsed mode operation: the modulator frequency and the signal frequency couple bi-directionally.
INTERNAL	RFMOD	Pulsed mode operation: The modulator frequency and the signal frequency couple bi-directionally. The synthesizer frequency and the carrier frequency couple bi-directionally. On transition into this state, the carrier frequency is set to the current synthesizer frequency.
MANUAL	STANDALONE	Non-pulsed mode operation: the modulator frequency does not couple to anything.
MANUAL	TRIGGER	Non-pulsed mode operation: the modulator frequency couples to the signal frequency (but not vice-versa).
MANUAL	RFMOD	Pulsed mode operation: the modulator frequency couples to the signal frequency (but not visa-versa). The synthesizer frequency and the carrier frequency couple bi-directionally. On transition into this state, the carrier frequency is set to the current synthesizer frequency.

Query Response



QMOD1

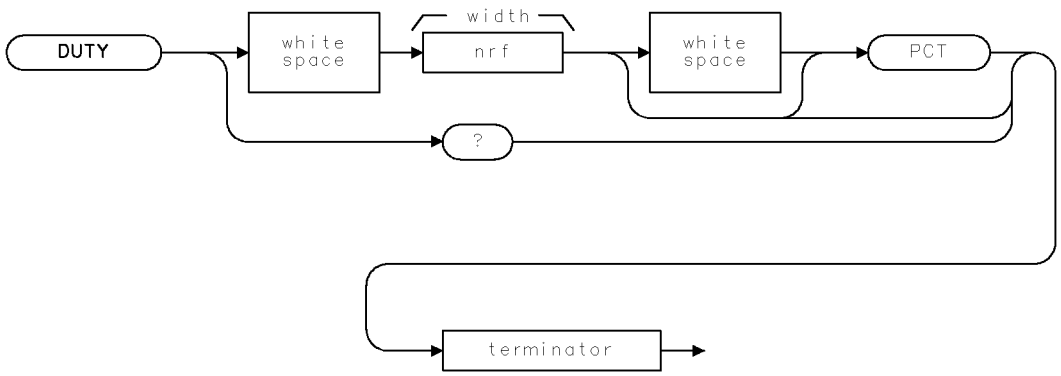
Query Example

```
OUTPUT @Mta;"MOD:COUP?"
ENTER @Mta;Couple$
```

DUTY

Sets the duty cycle of the modulator.

Syntax



Preset State: 10%
Parameter Range: 100 ns ≤ width ≤ period − 100 ns;
width = period × duty cycle/100

Command Example

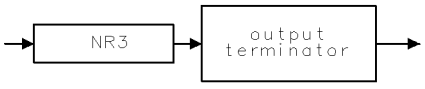
```
OUTPUT @Mta;"MOD:DUTY 25"
```

Description

The DUTY command sets the duty cycle of the modulator source pulse in percent. If POLARITY is set to positive, the percent refers to the width of the positive portion of the pulse, otherwise the negative portion. This command sets the same parameter as the MODULATOR:WIDTH command.

The query returns the current state of DUTY.

Query Response



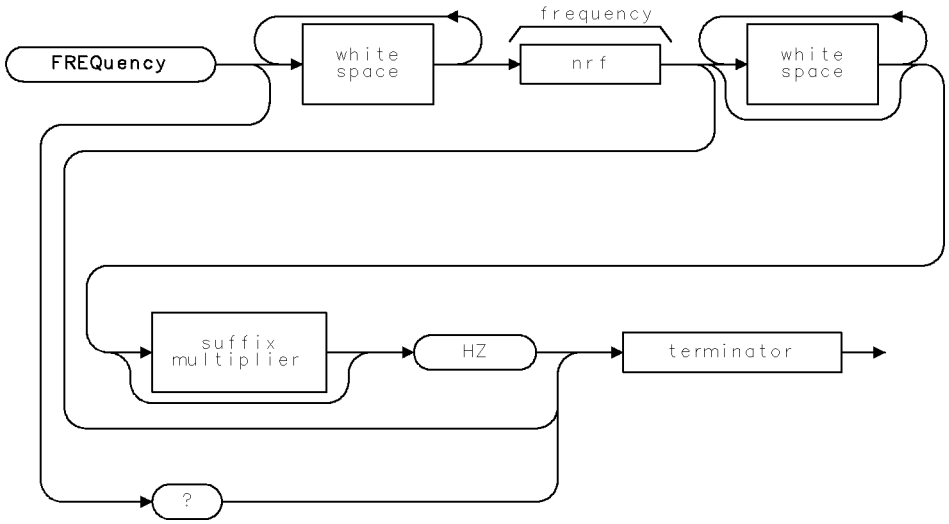
Query Example

```
OUTPUT @Mta;"MOD:DUTY?"
ENTER @Mta;Duty_cycle
```

FREQuency

Sets the frequency of the modulator.

Syntax



XMOD3

Preset State:	10 kHz
Parameter Range:	frequency = 10 MHz/65535 through 10 MHz/2;
	frequency = 152.59 Hz through 5 MHz
Fundamental Unit:	hertz

Command Example

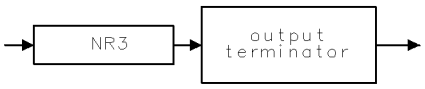
```
OUTPUT @Mta;"MOD:FREQ 25 kHz"
```

Description

The FREQUENCY command sets the frequency of the modulator source. If the COUPLE parameter is on, this also sets the value of SWEEP:SIGNAL. This command sets the same parameter as the MODULATOR:PERIOD command.

The query returns the current state of FREQUENCY.

Query Response



QGENNR3

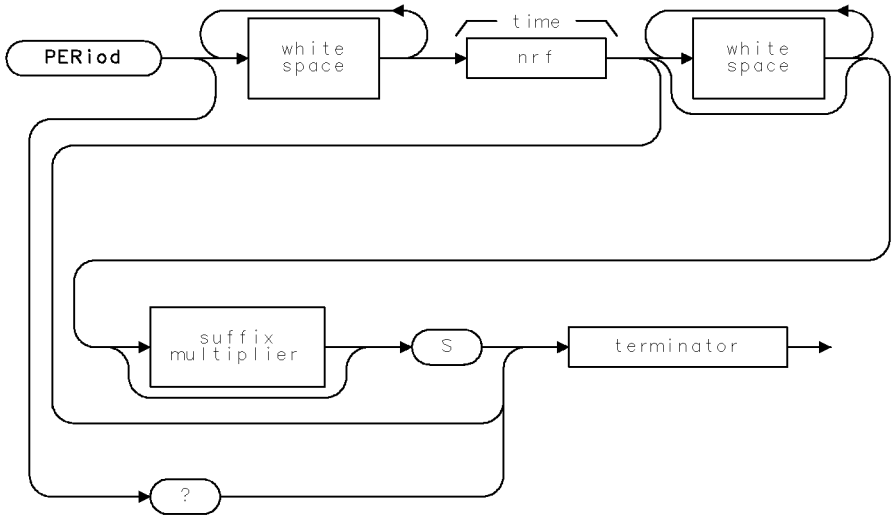
Query Example

```
OUTPUT @Mta;"MOD:FREQ?"
ENTER @Mta;Freq
```

PERiod

Sets the period of the modulator.

Syntax



Preset State: 100 μs
Parameter Range: 200 ns (2×100 ns) ≤ time ≤ 6.5535 ms (65535×100 ns)
Fundamental Unit: seconds

Command Example

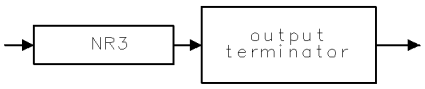
```
OUTPUT @Mta;"MOD:PER 250 us"
```

Description

The PERIOD command sets the period of the modulator source. If the COUPLE parameter is on, the value of SWEEP:SIGNAL is set to 1/period. This command sets the same parameter as the MODULATOR:FREQUENCY command.

The query returns the current state of PERIOD.

Query Response



QGENNR3

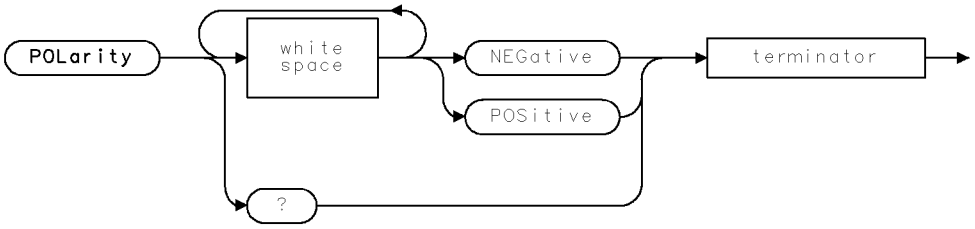
Query Example

```
OUTPUT @Mta;"MOD:PER?"
ENTER @Mta;Period
```

POLarity

Sets the modulator pulse polarity.

Syntax



Preset State: POSitive

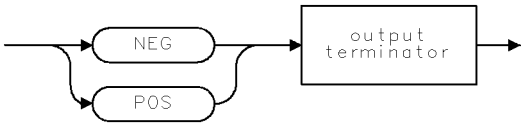
Command Example

```
OUTPUT @Mta;"MOD:POL NEG"
```

Description

The POLARITY command determines whether the duty cycle and width commands apply to the positive portion of the pulse (POSITIVE) or the negative portion of the pulse (NEGATIVE). The query returns the current state of POLARITY.

Query Response



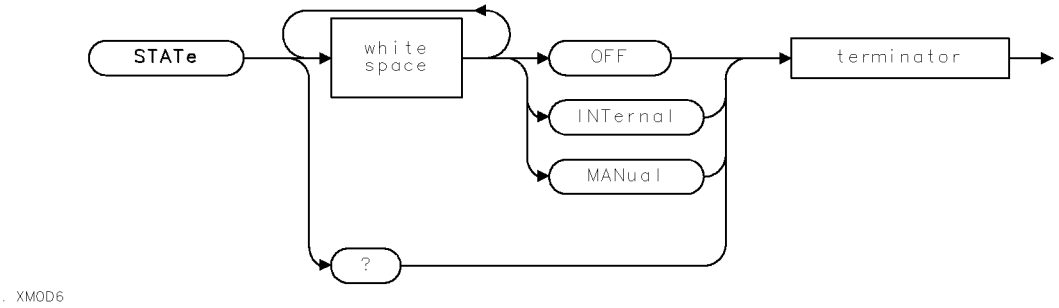
Query Example

```
OUTPUT @Mta;"MOD:POL?"
ENTER @Mta;Polarity$
```

STATe

Controls the state of the modulator source.

Syntax



Preset State: OFF
Related Commands: MODulator:STATe
SWEep:TIME:TRANslate

Command Example

```
OUTPUT @Mta;"MOD:STAT INT"
```

Description

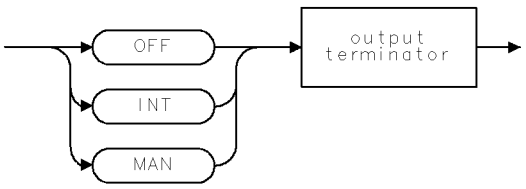
The STATE command controls the state of the modulator source. The query returns the current state of STATE.

When STATE transitions to OFF (that is, the current state is not OFF and the new state is OFF), this command will also turn off the pulse modulator of the synthesizer, if any (see SOURCE:PULSE:STATE).

The MODULATOR:STATE and MODULATOR:COUPLE commands affect the state of the instrument as follows:

STATE	COUPLE	MODE
OFF	not applicable	Non-pulsed mode operation: the modulator frequency does not couple to anything
INTERNAL	STANDALONE	Non-pulsed mode operation: the modulator frequency does not couple to anything
INTERNAL	TRIGGER	Non-pulsed mode operation: the modulator frequency and the signal frequency couple bi-directionally.
INTERNAL	RFMOD	Pulsed mode operation: The modulator frequency and the signal frequency couple bi-directionally. The synthesizer frequency and the carrier frequency couple bi-directionally. On transition into this state, the carrier frequency is set to the current synthesizer frequency.
MANUAL	STANDALONE	Non-pulsed mode operation: the modulator frequency does not couple to anything.
MANUAL	TRIGGER	Non-pulsed mode operation: the modulator frequency couples to the signal frequency (but not vice-versa).
MANUAL	RFMOD	Pulsed mode operation: the modulator frequency couples to the signal frequency (but not vice-versa). The synthesizer frequency and the carrier frequency couple bi-directionally. On transition into this state, the carrier frequency is set to the current synthesizer frequency.

Query Response



QMOD6

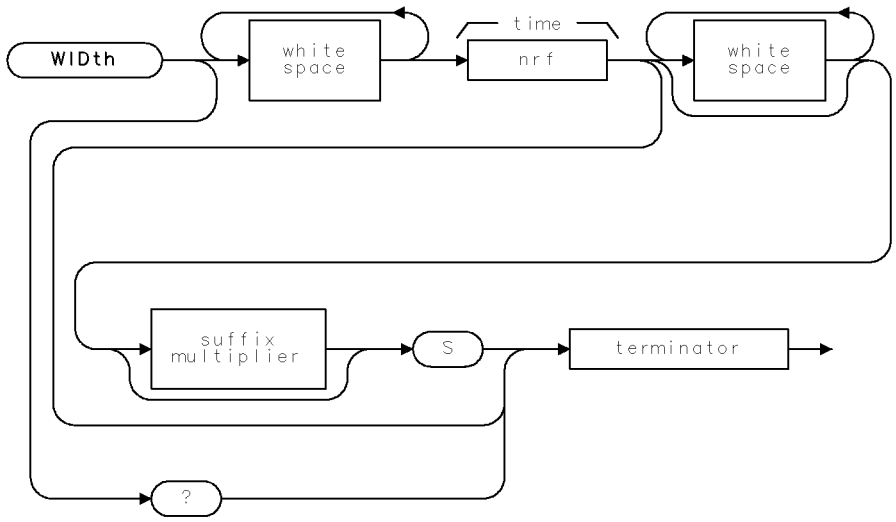
Query Example

```
OUTPUT @Mta;"MOD:STAT?"
ENTER @Mta;State$
```

WIDth

Sets the pulse width of the modulator.

Syntax



Preset State: 10 μs
Parameter Range: time = 100 ns through 6.5534 ms
Fundamental Unit: seconds

Command Example

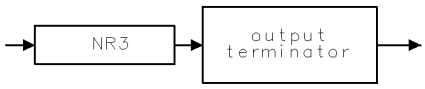
```
OUTPUT @Mta;"MOD:WID 1 us"
```

Description

The WIDTH command sets the width of the modulator source pulse. If the POLARITY command is set to POSITIVE, the WIDTH command sets the width of the positive portion of the pulse, otherwise it sets the negative portion. This command sets the same parameter as MODULATOR:DUTY. The query returns the current state of WIDTH.

If the parameter sent to the WIDTH command is greater than the period − 100 ns, then the period will be set to WIDTH + 100 ns.

Query Response



QGENNR3

Query Example

```
OUTPUT @Mta;"MOD:WID?"
ENTER @Mta;Width
```

20

Peak Power Meter Subsystem

Note The PEAK POWER subsystem is only available with Option H10.

The PEAK POWER subsystem controls the operation of the microwave transition analyzer as a statistical peak power meter. The basic operation of the peak power meter mode is to sample the input across an interval (see PPOWER:WIDTH) a large number of times (determined by PPOWER:COUNT or PPOWER:SRES) and use the maximum value for the peak power determination. The normal triggering and vertical scaling modes are used (see CHANx and TRIGGER subsystems).

The basic operation of the statistical power meter does not require the microwave transition analyzer to know either the modulation frequency or the carrier frequency. However, to improve the measurement results, the RF corrections will be applied assuming the carrier frequency is specified by PULSE:CARRIER. If this assumption is invalid, RF corrections may be disabled with the CALIBRATE:CORRECT:RF command.

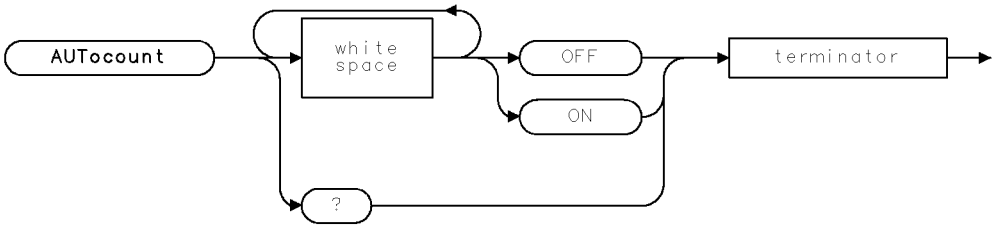
See the following list for a command syntax summary of the PEAK POWER subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:PPOWer		
:AUTocount	OFF|ON	
:COUNt	<nrf>	
:MEASure		[no query]
:RESults?	CHAN1|CHAN2	[query only]
:SRATe	<nrf>[[<suffix multiplier>]HZ]	
:SRESolution	<nrf>	
:TYPE	AUTOmatic|DITher|MANual	
:WIDth	<nrf>[[<suffix multiplier>]S]	

AUTocount

Enables or disables the computation of PPOWER:COUNT.

Syntax



Preset State: OFF

Command Example

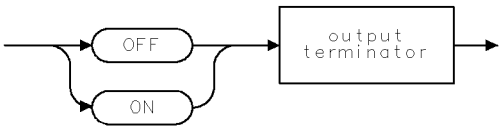
```
OUTPUT @Mta;"PPOW:AUT ON"
```

Description

The AUTOCOUNT command enables or disables the computation of PPOWER:COUNT for a specified PPOWER:SRESOLUTION. The query returns the current setting of AUTOCOUNT.

The state of AUTOCOUNT is affected by both PPOWER:COUNT and PPOWER:SRESOLUTION.

Query Response



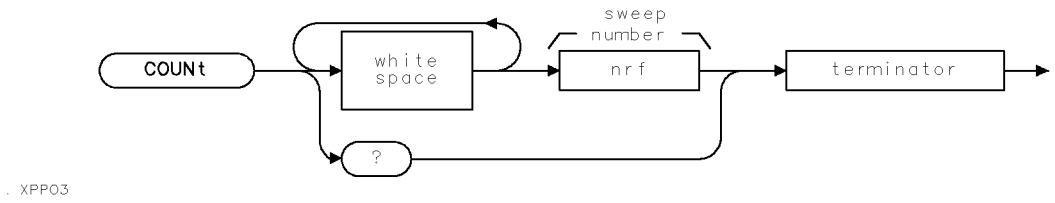
Query Example

```
OUTPUT @Mta;"PPOW:AUT?"
ENTER @Mta;Autocount$
```

COUNt

Sets the number of sweeps that will comprise one peak power measurement.

Syntax



Parameter Range: sweep number = 1 through 16384
Preset State: 128

Command Example

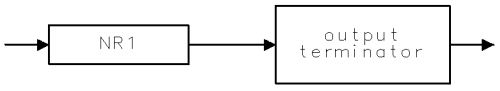
```
OUTPUT @Mta;"PPOW:COUN 1000"
OUTPUT @Mta;"PPOW:SRES?"
ENTER @Mta;Resolution
PRINT "A count of 1000 yields ";Resolution
```

Description

The COUNT command sets the number of sweeps that will comprise one power measurement. This command affects the PPOWER:SRESOLUTION state and sets the PPOWER:AUTO state to OFF.

The query returns the current setting for COUNT.

Query Response



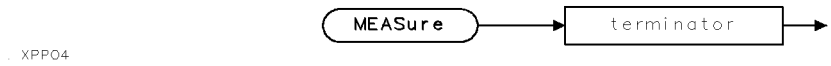
Query Example

```
OUTPUT @Mta;"PPOW:COUN?"
ENTER @Mta;Count
```

MEASure

Invokes a peak power measurement.

Syntax



Command Example

```
OUTPUT @Mta;"PPOW:MEAS"
```

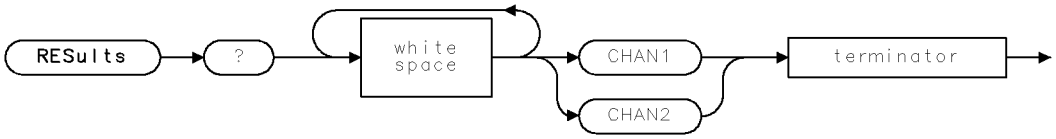
Description

The MEASURE command invokes a peak power measurement.

RESults

Returns the results of a peak power measurement.

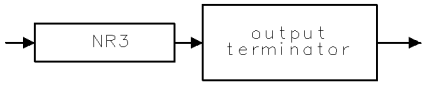
Syntax



Description

The RESULTS query returns the results for the specified channel of the peak power measurement invoked by PPOWER:MEASURE.

Query Response



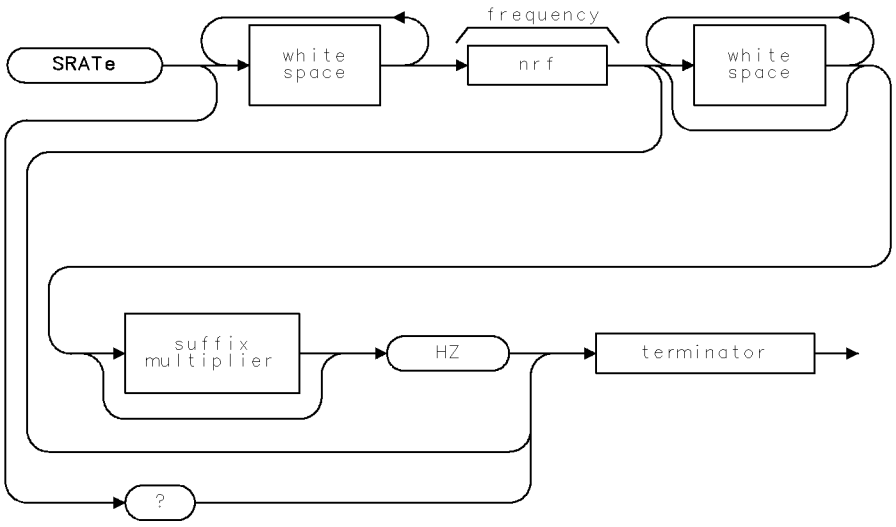
Query Example

```
OUTPUT @Mta;"PPOW:MEAS"
OUTPUT @Mta;"PPOW:RES? CHAN1"
ENTER @Mta;Chan1_power
```

SRATe (Sample Rate)

Sets the sample rate.

Syntax



XPP07

Parameter Range: frequency = 10 MHz through 20 MHz
Preset State: 20 MHz
Fundamental Unit: hertz

Command Example

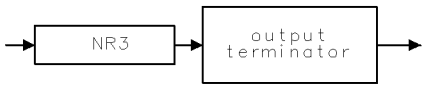
```
OUTPUT @Mta;"PPOW:SRAT 19.78 MHz"
```

Description

The SRATE command sets the sample rate used in the peak power meter mode of operation. This command forces PPOWER:TYPE to MANUAL.

The query returns the current sample rate.

Query Response



QGENNR3

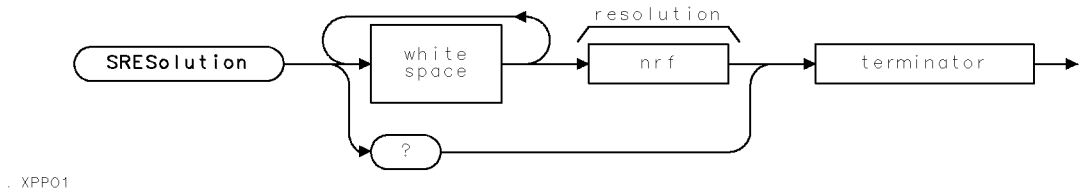
Query Example

```
OUTPUT @Mta;"PPOW:SRAT?"
ENTER @Mta;Sample_rate
```

SRESolution

Specifies the statistical resolution of a peak power meter measurement in dBc.

Syntax



Parameter Range: resolution = 0.01 through 100
Preset State: 0.1

Command Example

```
OUTPUT @Mta;"PPOW:SRES .01"
OUTPUT @Mta;"PPOW:COUN?"
ENTER @Mta;Count
PRINT "For .01 dBm, count is ";Count
```

Description

The SRESOLUTION command specifies the statistical resolution of a peak power meter measurement in dBc. A measurement made with a given statistical resolution setting has a 99.5 percent confidence level of sampling the peak value of the signal to within the specified resolution. This command affects the PPOWER:COUNT state and sets the AUTOCOUNT mode to ON. The query returns the current SRESOLUTION setting.

The assumptions made on the signal for the statistical resolution computations are:

- There is no system noise or noise on the input.
- The carrier is assumed to be sinusoidal.
- The amplitude of the pulse envelope does not change significantly over an interval of 100 ns or 5 percent of the pulse width, whichever is less.
- The sampling is effectively random with respect to the carrier of the input (there is some phase or frequency dithering done to help ensure this).

The probability of any given sample being within the desired range of the positive (or negative) peak of the carrier is given by:

$$probability\ of\ point = 4 \times ARCCOS(10^{(-resolution/20)})/(2 \times PI)$$

The number of points acquired in one measurement on the interval of interest is:

$$points = min(0.05 \times width, 100.0^{-9}) \times sample\ rate$$

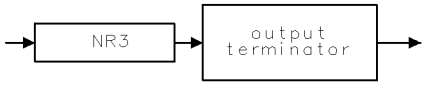
The probability of one sweep acquiring at least one point on the peak is:

$$probability\ of\ sweep = 1.0 - (1.0 - probability\ of\ point)^{points}$$

Since the microwave transition analyzer attempts to "randomize" the sampling process by adjusting the sampler phase or sampler frequency between sweeps, a minimum of eight sweeps will always be computed. The number of sweeps required for 99.5 percent confidence of acquiring the peak is:

$$sweeps = max(8, (log(1 - 0.995)/log(1 - probability\ of\ sweep)))$$

Query Response



QGENNR3

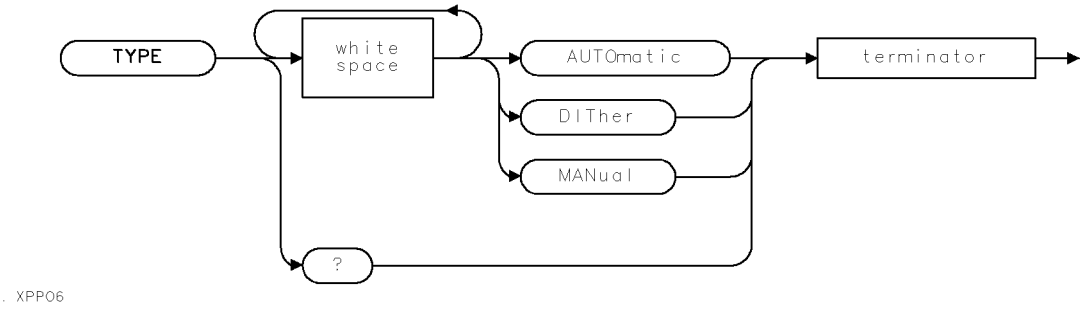
Query Example

```
OUTPUT @Mta;"PPOW:SRES?"
ENTER @Mta;Resolution
```

TYPE

Sets data acquisition mode for peak power measurements.

Syntax



Preset State: AUTOmatic

Command Example

```
OUTPUT @Mta;"PPOW:TYPE MAN"
```

Description

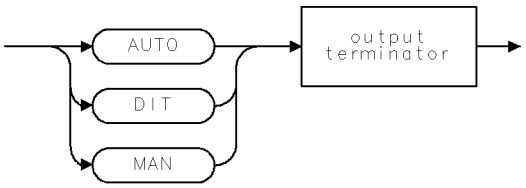
The TYPE command specifies the method by which the module will acquire data for the power meter mode of operation.

The options for TYPE are:

- **AUTO** In the AUTOMATIC mode, the module will sample at a fixed sampling rate and "dither" the phase in a pseudo-random sequence between each trace. The fixed sample rate will be computed based on the carrier frequency (as defined by PULSE:CARRIER). This mode is useful when the carrier frequency is known and stable.
- **MAN** In the MANUAL mode, the module will sample at a fixed sampling rate specified by the user (see PPOWER:SRATE) and "dither" the phase in a pseudo random sequence between each trace.
- **DIT** In the DITHER mode, the module will vary the sample rate between each measurement. The sample rate will range from 16 MHz to 20 MHz. This mode is useful when the carrier frequency is unknown or unstable. Because of the changing sample rates, this mode is currently slower than either the AUTO or MAN modes.

The query returns the current setting of TYPE.

Query Response



. QPP06

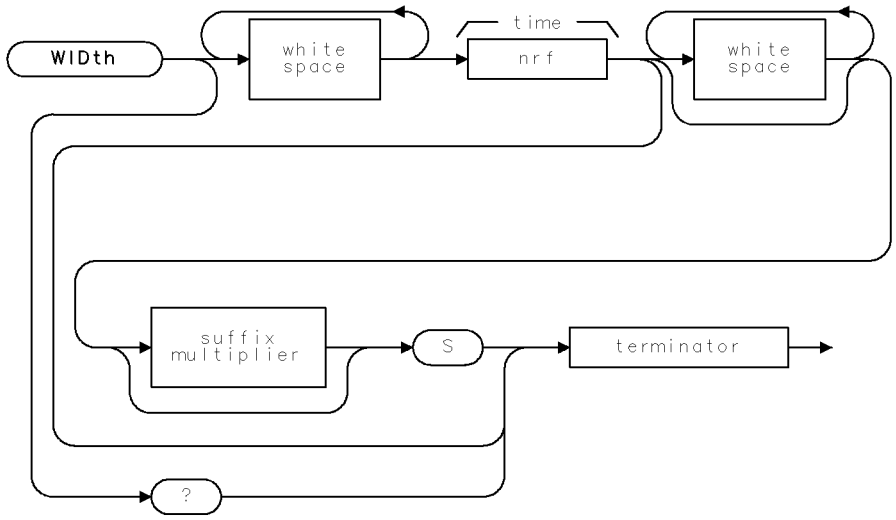
Query Example

```
OUTPUT @Mta;"PPOW:TYPE?"
ENTER @Mta;Meter_type$
```

WIDth

Sets the measurement interval.

Syntax



XPP08

Parameter Range: time = 1 ns through 10 ms
Preset State: 1 μs
Fundamental Unit: seconds

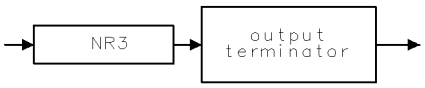
Command Example

```
OUTPUT @Mta;"PPOW:WID 5 us"
```

Description

The WIDTH command specifies the width of the time interval to be considered for the peak power measurement. The query returns the current setting for WIDTH.

Query Response



QGENNR3

Query Example

```
OUTPUT @Mta;"PPOW:WID?"
ENTER @Mta;Width
```

21

Program Subsystem

The PROGRAM subsystem provides the administrative features needed to generate and control one or more user programmed tasks resident in an instrument. Two distinct methods of accessing a particular program are provided in the PROGRAM subsystem. One method employs EXPLICIT reference for each command, allowing access to all programs, but requires the program name to be always specified. The other method allows a single program to be SELECTED at a time, and the related commands accessed only the currently selected program.

All the commands under SELECTED access only the program that has been selected by the NAME command. All the commands under the EXPLICIT mode directly reference the desired program by name, thus allowing access to a program without having to change the selected program.

An *RST received from a remote controller, via the device interface, shall cause all programs to be stopped.

An *RST generated by a program running in the device shall cause the same effect as a *RST received from a remote controller, with the exception that the program that generated the *RST shall not be forced to stop. It is necessary for a *RST to behave in this manner so that a program, when run on either a remote controller or the device, shall have the same effect. See the following list for a command syntax summary of the PROGRAM subsystem commands.

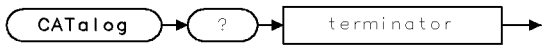
Note The current implementation of the PROGRAM subsystem in the HP 70820A only supports a single program.

KEYWORD	PARAMETER FORM	NOTES
:PROGram		
:CATalog?		[query only]
:EXPLicit		
:DEFine	“{<character>}”|’{<character>}’ ,#<digit><count><block data>	
:DELete	“{<character>}”|’{<character>}’	[no query]
:EXECute	“{<character>}”|’{<character>}’ “{<character>}”|’{<character>}’	[no query]
:MALLocate	“{<character>}”|’{<character>}’ ,<nrf>|DAFault	
:NUMBer	“{<character>}”|’{<character>}’ ,“{<character>}”|’{<character>}’ {,<nrf>}	
:STATe	“{<character>}”|’{<character>}’ ,CONTinue|PAUSe|RUN|STOP	
:STRing	“{<character>}”|’{<character>}’ ,“{<character>}”|’{<character>}’ {,“<character>”|’<character>’}	
:WAIT	“{<character>}”|’{<character>}’	
[:SELected]		
:DEFine	#<digit><count><block data>	
:DELete		
[:SELected]		[no query]
:ALL		[no query]
:EXECute	“{<character>}”|’{<character>}’	[no query]
:MALLocate	<nrf>|DEFault	
:NAME	“{<character>}”|’{<character>}’	
:NUMBer	“{<character>}”|’{<character>}’ [,<nrf>]	
:STATe	CONTinue|PAUSe|RUN|STOP	
:STRing	“{<character>}”|’{<character>}’ {,“<character>”|’<character>’}	
:WAIT		

CATalog

Lists all the defined programs.

Syntax

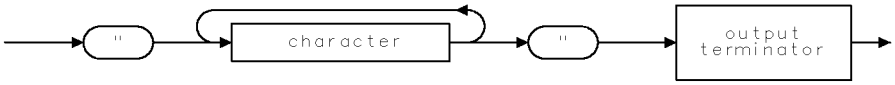


xprg1

Description

The CATALOG query commands lists all the defined programs. The response is a list of comma separated strings. Each string contains the name of a program. If no programs are currently defined, then response is a null string (""). The HP 70820A currently only supports a single HP Instrument BASIC program running at a time. Because of this, the CATALOG query will always return a single name (or null string if no program is loaded) that can be used to identify the current program.

Query Response



qgenn9

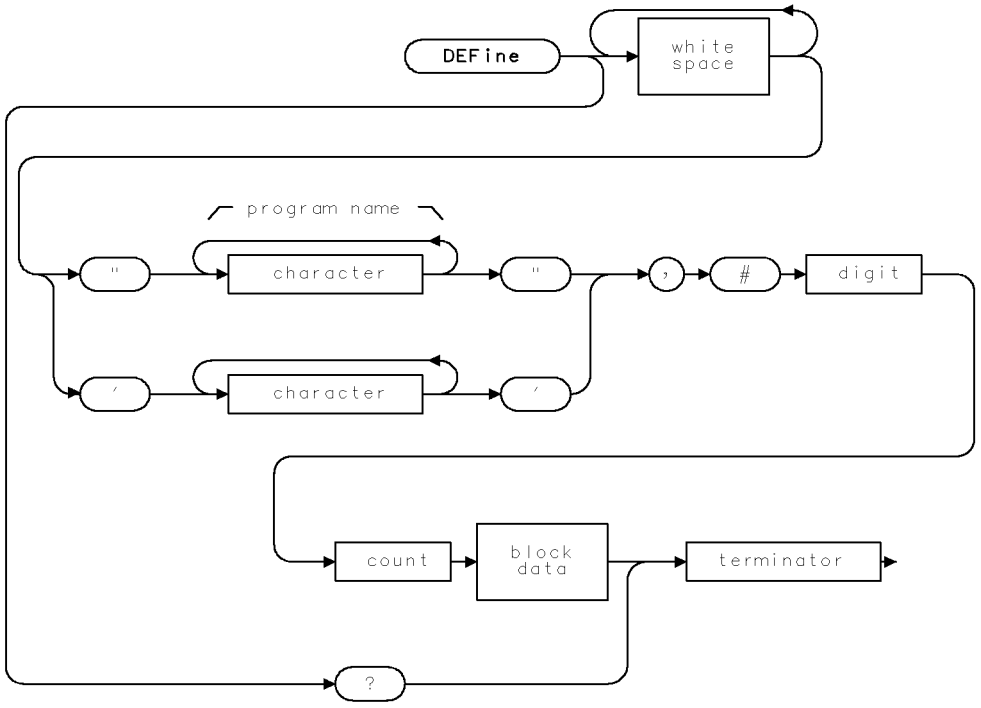
Query Example

```
OUTPUT @Mta;"PROG:CAT?"
ENTER @Mta;Catalog$
IF Catalog$="""""""" THEN  ! To BASIC this represents ""
  PRINT "No program is loaded"
ELSE
  OUTPUT @Mta;"PROG:NAME ";Catalog$
END IF
```

EXPLicit:DEFine

Used to create and download programs.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:DEL:ALL"
                    ! Note that the count (20) includes
                    ! the CR and LF that are sent.
OUTPUT @Mta;"PROG:EXPL:DEF 'REMOTE',#22010 PRINT 5"
OUTPUT @Mta;"20 END"
OUTPUT @Mta;"" END   ! Send a LF with EOI asserted
```

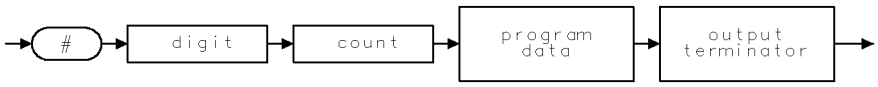
Description

The DEFINE command is used to create and download programs. In the DEFINE command, the specified program name represents a unique name. That is, to download and overwrite an existing program with the same program name, the existing program must be DELETED first. If such an attempt is made, without deleting first, an "Illegal program name" error, −282, is generated. The program is block data and the contents are device dependent. When overflow occurs a program error, −28X, is generated.

In the DEFINE query, the specified program name is the name of an existing program; otherwise an "Illegal program name" error, −282, is generated. The program is uploaded as definite length arbitary block response data.

The DEFINE query will generate an "Illegal program name" error (−282) if there is not a program currently loaded.

Query Response



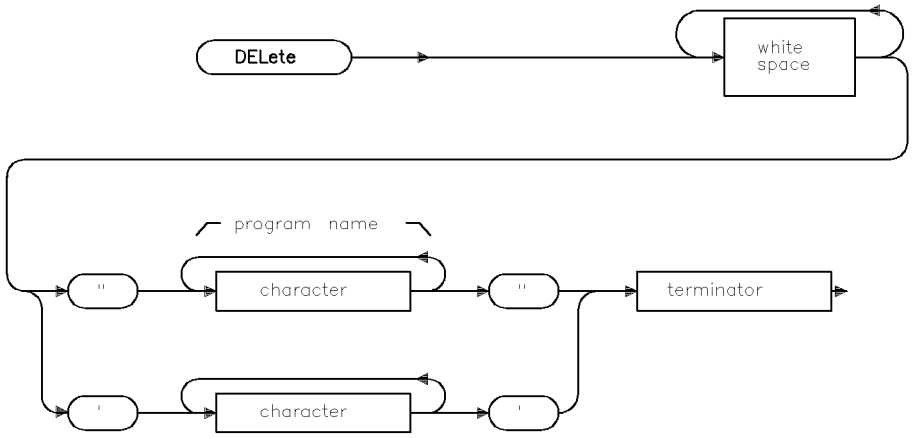
Query Example

```
DIM Line$[128]
OUTPUT @Mta;"PROG:EXPL:DEF? 'REMOTE'"
ENTER @Mta USING "#,A,D";Header$,Digit$
ENTER @Mta USING "#,"&Digit$&"D";Byte_count
LOOP
  ENTER @Mta;Line$
  EXIT IF LEN(Line$)=0
  PRINT Line$
END LOOP
```

EXPLicit:DELete

Deletes the specified downloaded program.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:EXPL:DEL 'REMOTE'"
```

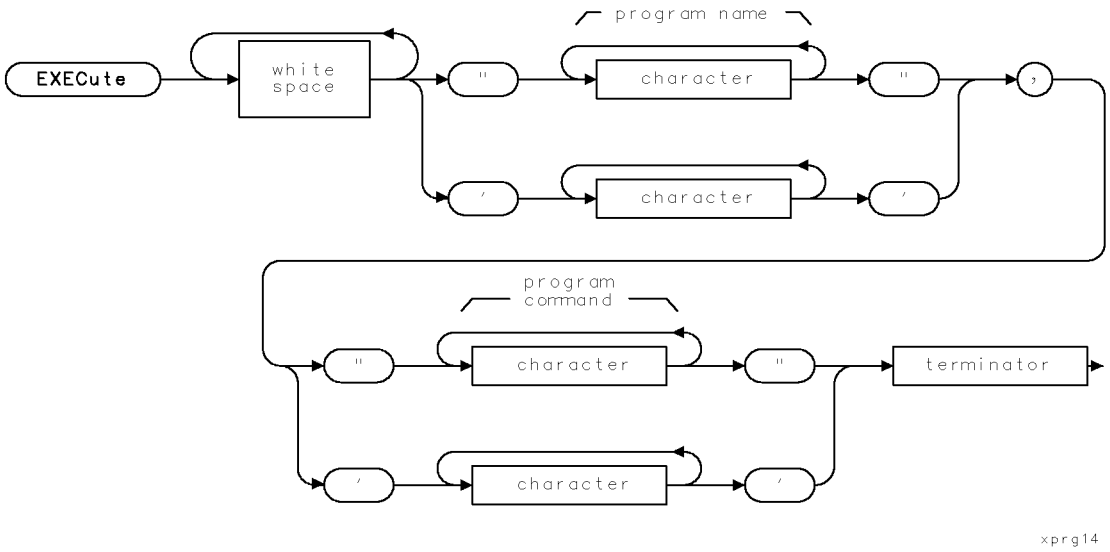
Description

The DELETE command deletes the specified downloaded program.

EXPLicit:EXECute

Executes the specified command in the specified program environment.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:EXPL:EXEC 'REMOTE','PRINT 5/2'"
```

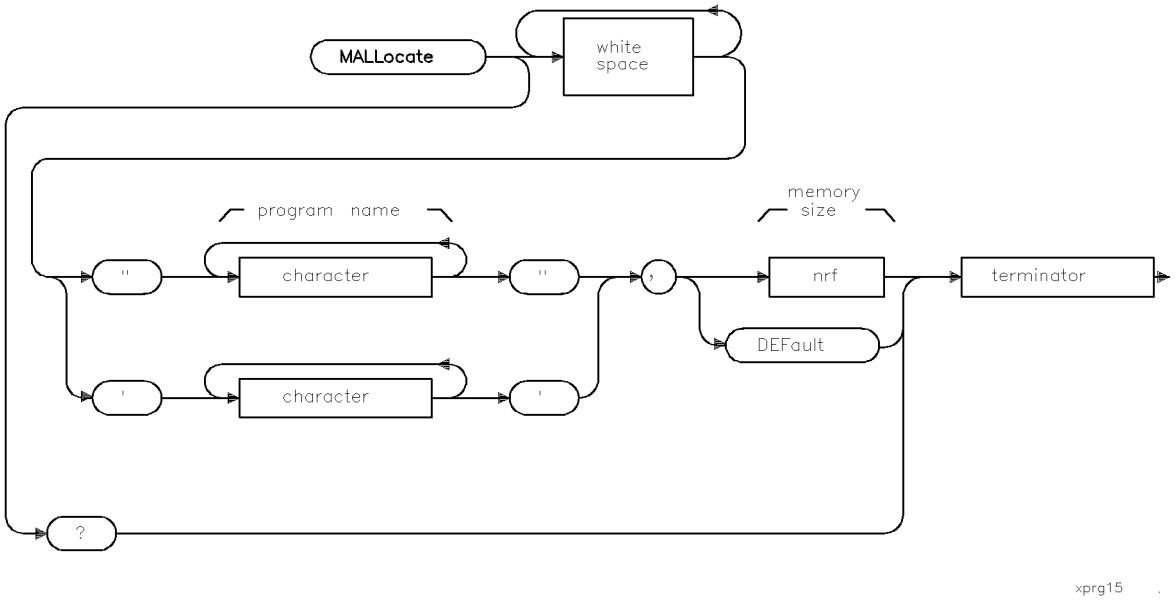
Description

The EXECUTE command executes the specified command in the specified program environment. The command string data represents any legal program command. If the string data is not legal, then a "Program syntax error," 285, is generated. The specified program is in either the PAUSED or STOPPED state before the EXECUTE command is allowed. If the program is in the RUN state, a "Program currently running" error, −284, is generated.

EXPLicit:MALLocate

Reserves memory space in the device for use by the specified program.

Syntax



Parameter Range: 1122 through maximum
maximum is dependent on the program size and how many keys are defined with the DISPlay:MENU system.

Command Example

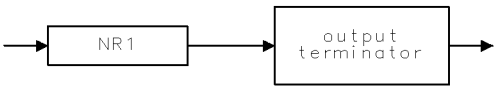
```
OUTPUT @Mta;"PROG:EXPL:MALL 'REMOTE',50000"
```

Description

The MEMORY ALLOCATE command reserves memory space in the device for use by the specified program. The specified program is the name of an existing program, otherwise an "Illegal program name" error, −282, is generated. If DEFAULT is specified, then the device calculates the amount of memory needed. You can also specify the required memory in bytes. If the required memory is greater than the space available, then a parameter error "Data out of range", −222, is generated.

A program must be loaded before the MALLOCATE command or query may be used. *RST has no effect on the value of MALLOCATE.

Query Response



. QGENN1

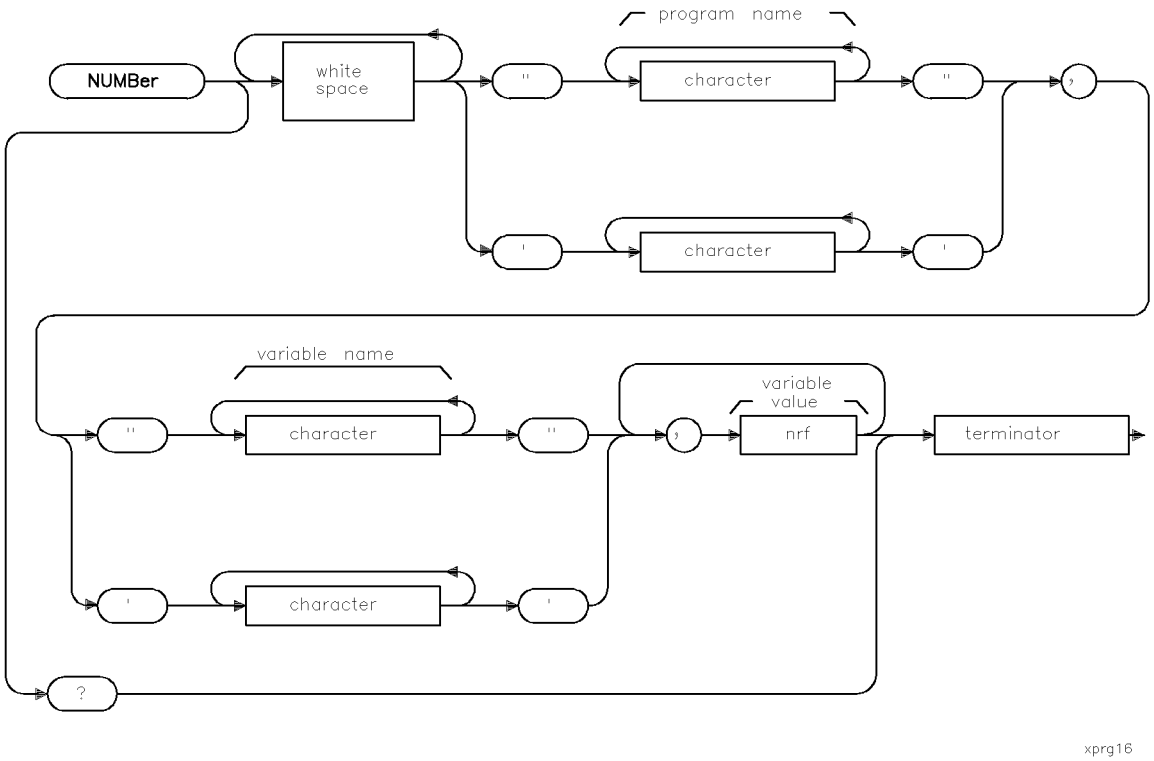
Query Example

```
OUTPUT @Mta;"PROG:EXPL:MALL? 'REMOTE'"
ENTER @Mta;Stack_size
```

EXPLicit:NUMBer

Used to set and query the contents of numeric program variables and arrays.

Syntax



Command Example

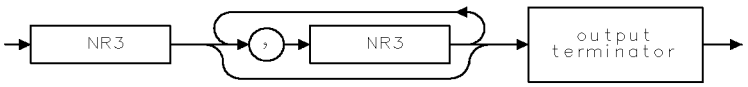
```
OUTPUT @Mta;"PROG:EXPL:NUMB 'REMOTE','a',2.5"
```

Description

The NUMBER command is used to set and query the contents of numeric program variables and arrays in the specified program. The specified program must be a DEFINED program, otherwise an "Illegal program name" error, −282, is generated. The variable is the name of an existing variable in the specified program, otherwise an "Illegal variable name" error, −283, is generated. The variable can be either character data or string data. The variable values is a list of comma separated numeric values which are used to set the variable. If the specified variable cannot hold all of the specified numeric values, then a "Parameter not allowed" error, −108 , is generated.

The query returns the contents of the variable as a comma separated list.

Query Response



qgenn7

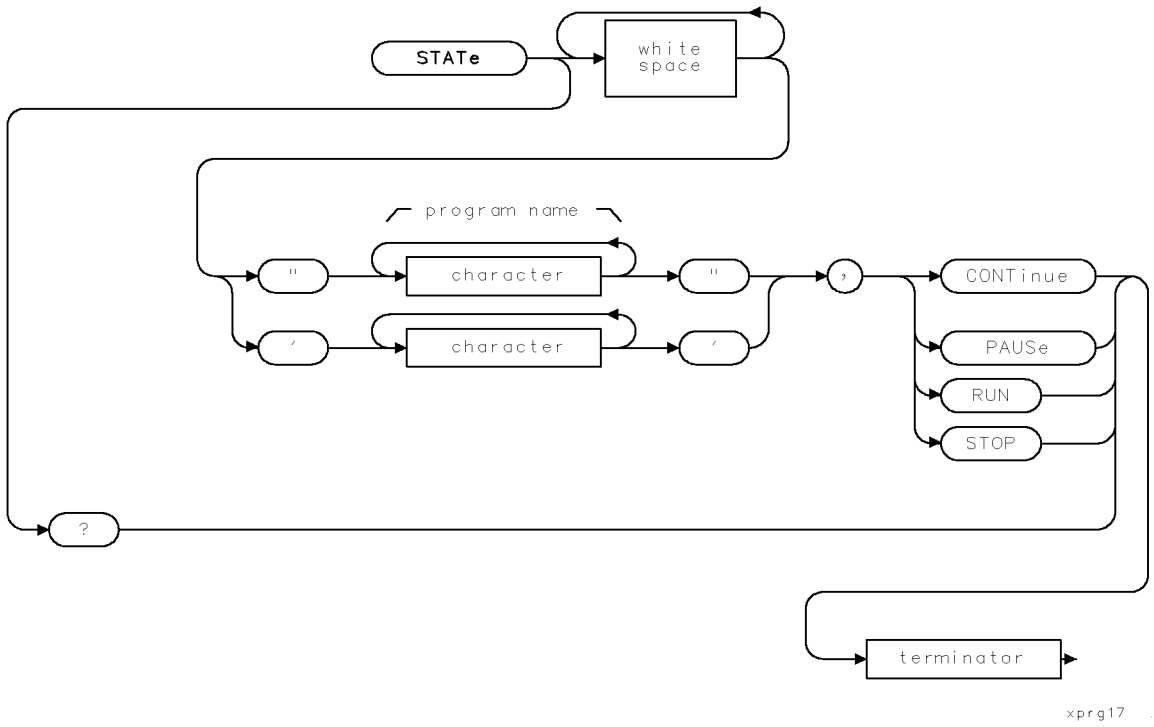
Query Example

```
OUTPUT @Mta;"PROG:EXPL:NUMB? 'REMOTE','a'"
ENTER @Mta;A
```

EXPLicit:STATe

Used to either set the state or query the state of a specified program.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:EXPL:STAT 'REMOTE',RUN"
```

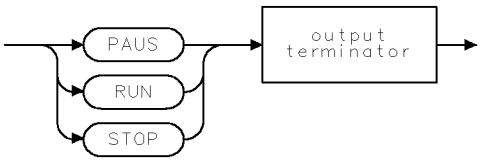
Description

The STATE command is used to either set the state or query the state of a specified program. The following table defines the effect of setting the STATE to the desired value from each of the possible current states. In certain cases a parameter error "Settings conflict", −221, is generated.

The states are described as follows:

Running:	The program is currently executing.
Paused:	The program has reached a break in execution but can be continued.
Stopped:	Execution has been terminated.

Query Response



qprg17

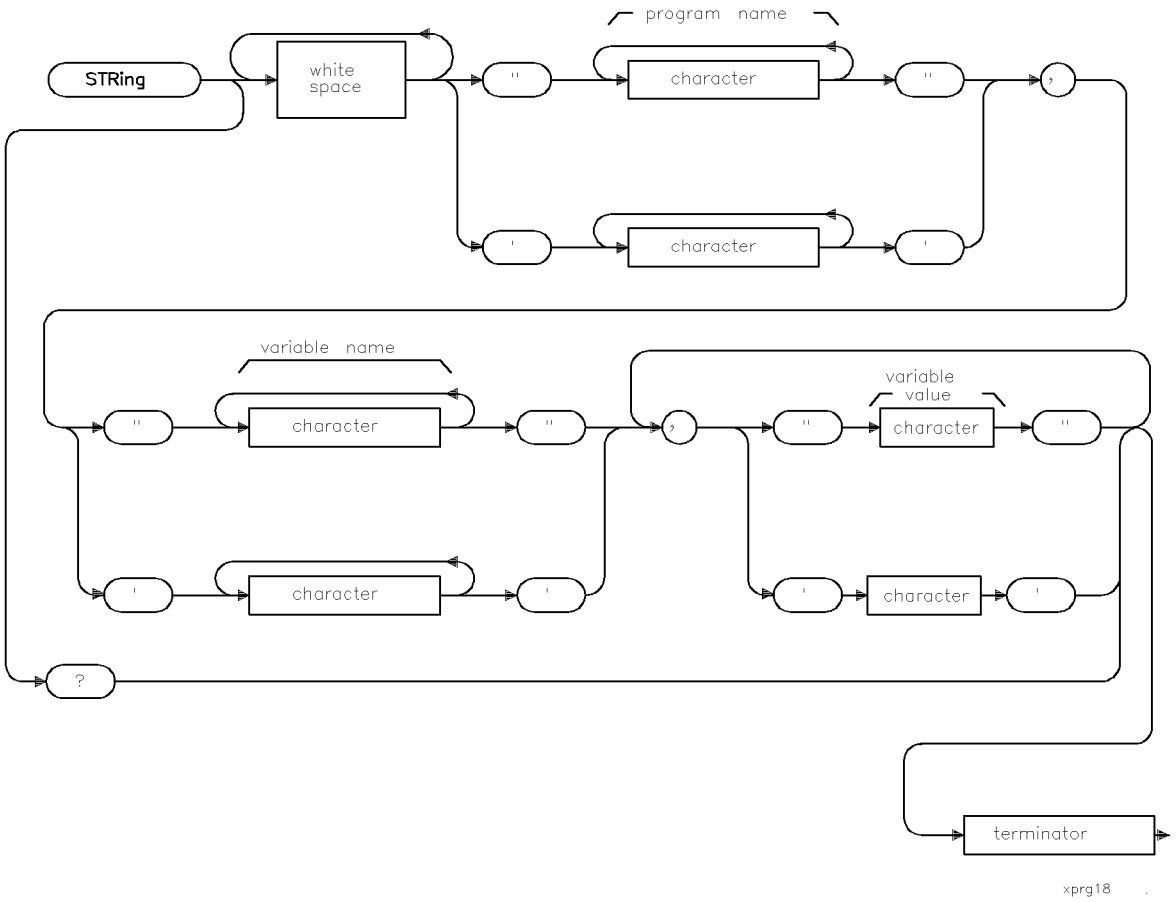
Query Example

```
OUTPUT @Mta;"PROG:EXPL:STAT? 'REMOTE'"
ENTER @Mta;State$
```

EXPLicit:STRing

Used to set and query the contents of string program variables and arrays.

Syntax



Command Example

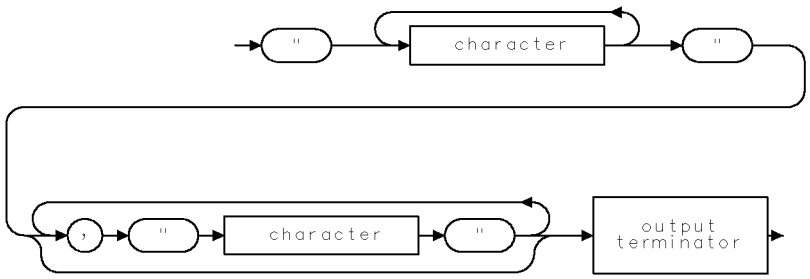
```
OUTPUT @Mta;"PROG:EXPL:STR 'REMOTE','b','Hello world'"
```

Description

The STRING command is used to set and query the contents of string program variables and arrays in the specified program. The specified program must be a DEFINED program, otherwise an "Illegal program name" error, −282, is generated. The variable specified is the name of an existing variable in the specified program, otherwise an "Illegal variable name" error, −283, is generated. The variable name can be either character data or string data. Variable values is a list of comma separated strings which are used to set the variable name. If the specified variable cannot hold all of the specified string values, then a "Parameter not allowed" error, −108, is generated. If a string value is too long then it will be truncated when stored in the programs variable.

The query returns the contents of the variable as a comma separated list.

Query Response



qgenn12

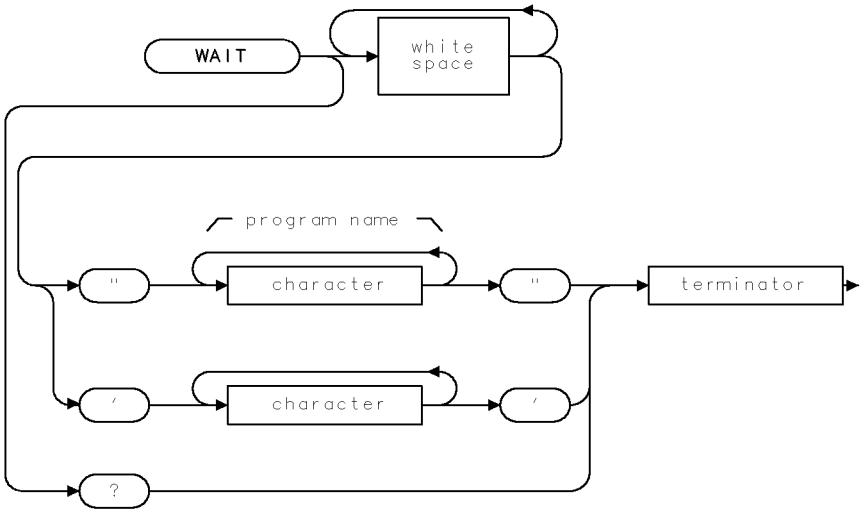
Query Example

```
OUTPUT @Mta;"PROG:EXPL:STR? 'REMOTE','b'"
ENTER @Mta;B$
```

EXPLicit:WAIT

Prevents commands or queries from being executed.

Syntax



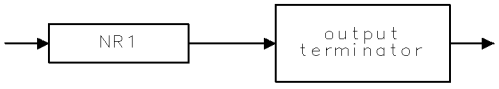
Command Example

```
OUTPUT @Mta;"PROG:EXPL:WAIT 'REMOTE'"
```

Description

The WAIT command and its query prevent commands or queries from being executed, until the defined program exits from the RUN state, that is either STOPPED or PAUSED. For the query, a 1 is returned in NR1 format at this time, when the program is either STOPPED or PAUSED.

Query Response



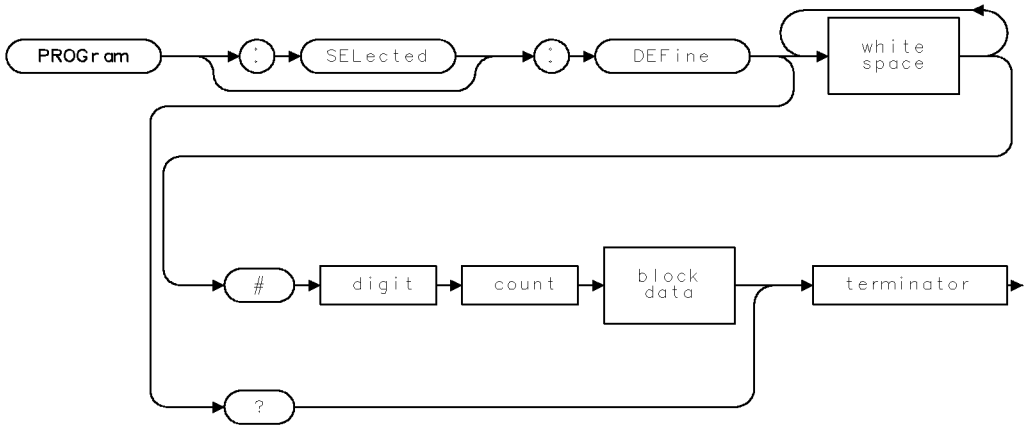
Query Example

```
OUTPUT @Mta;"PROG:EXPL:WAIT? 'REMOTE'"
ENTER @Mta;A$
```

SELected:DEFine

Used to create and download programs.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:DEL:ALL"
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
                    ! Note that the count (20) includes
                    ! the CRs and LFs that are sent.
OUTPUT @Mta;"PROG:DEF #22010 PRINT 5"
OUTPUT @Mta;"20 END"
OUTPUT @Mta;"" END   ! Send a LF with EOI asserted
```

Description

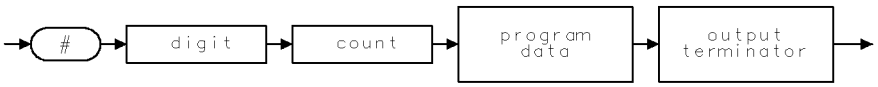
The DEFINE command is used to create and download programs. The DEFINE query is used to upload programs. The program name used for the definition is the currently selected program name. In the DEFINE command, the selected program name shall represent a unique name. That is, to download and overwrite an existing program with the same program name, the existing program must be DELETED first. If such an attempt is made without deleting first, an "Illegal program name" error, −282, shall be generated.

The program shall be block data. The contents are device dependent. When overflow occurs a program error, −28X, shall be generated.

In the DEFINE query, the selected program shall be returned. The program shall be uploaded as definite length arbitary block response data.

The DEFINE query will generate an "Illegal program name" error (−282) if there is not a program currently loaded.

Query Response



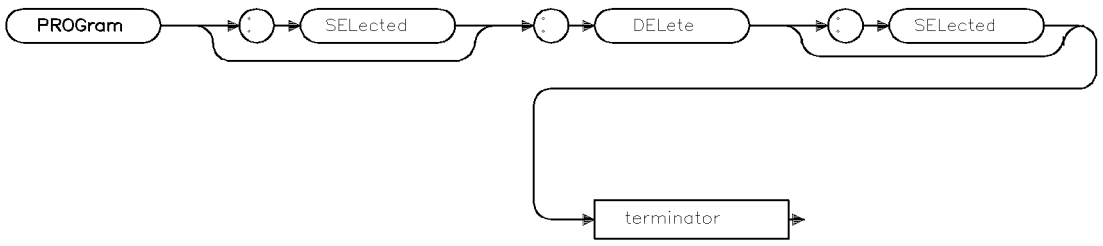
Query Example

```
DIM Line$[128]
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @MTA;"PROG:DEF?"
ENTER @Mta USING "#,A,D,";Header$,Digit$
ENTER @Mta USING "#,"&Digit$&"D";Byte_count
LOOP
  ENTER @Mta;Line$
  EXIT IF LEN(Line$)=0
  PRINT Line$
END LOOP
```

SELected:DELete:SELected

Deletes the selected program.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:DEL"
```

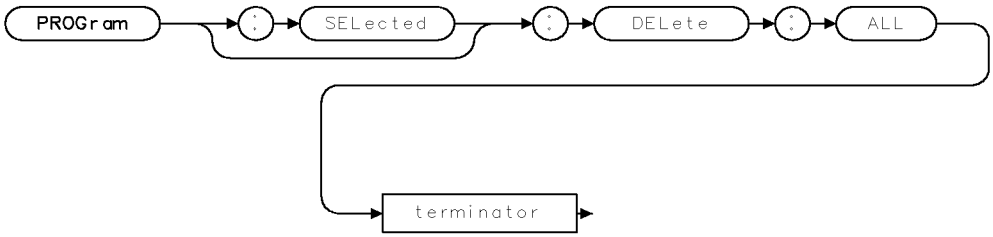
Description

Deletes the selected program. If the program is in the RUN state, a "Program currently running " error, −284, occurs and no programs are deleted.

SELected:DELete:ALL

Deletes all programs in device.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:DEL:ALL"
```

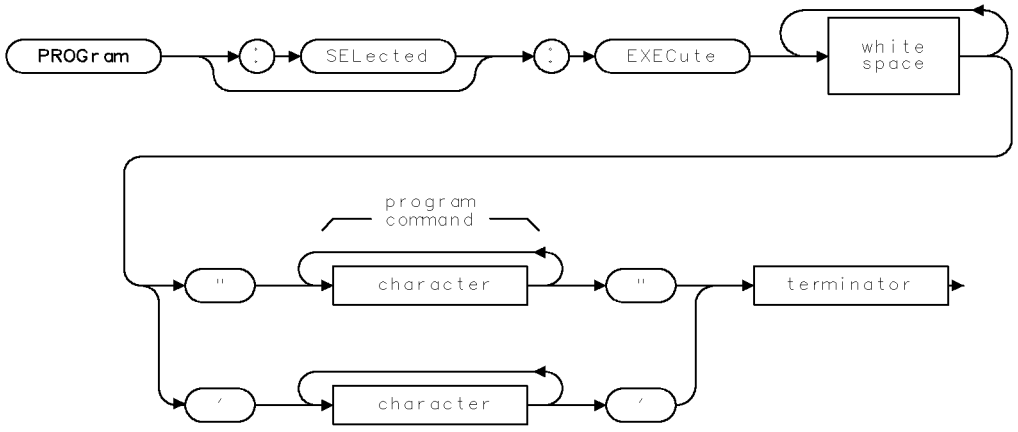
Description

Deletes all programs in device. If any of the programs are in the RUN state, a "Program currently running" error, −284, occurs and no programs are deleted.

SELected:EXECute

Executes a program command in the selected program environment.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:EXEC 'PRINT 5/2'"
```

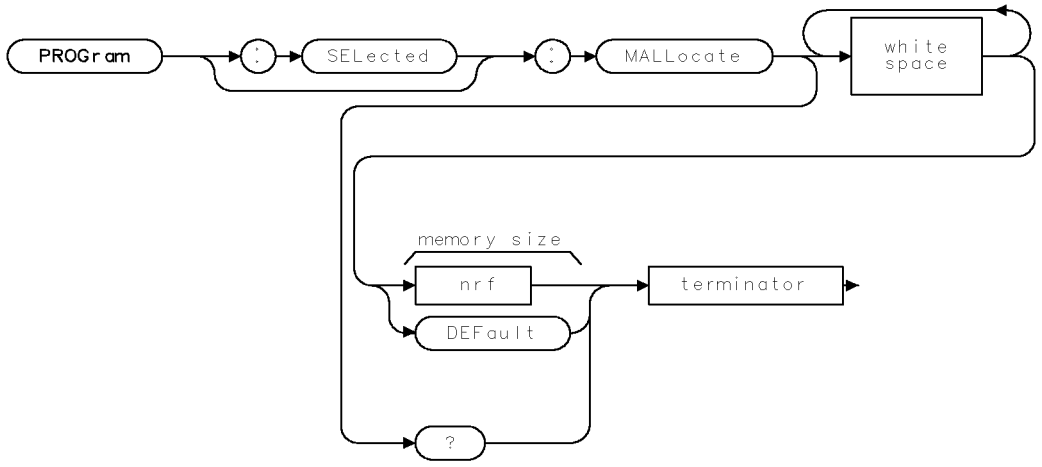
Description

The EXECUTE command executes a program command in the selected program environment. The program command is string data representing any legal program command. If the string data is not legal, then a "Program syntax error", −285, is generated. The selected program is in either the PAUSED or STOPPED state before the EXECUTE command is allowed. If the program is in the RUN state, a "Program currently running" error, −284, is generated.

SELected:MALLocate

Reserves memory space in the device for use by the selected program.

Syntax



Parameter range: 1122 through maximum
Maximum is dependent on program size
and how many keys are defined with
the DISPlay:MENU system.

Command Example

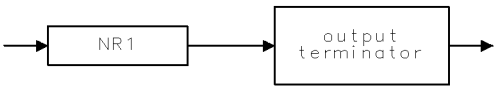
```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:MALL 50000"
```

Description

The MEMORY ALLOCATE command reserves memory space in the device for use by the selected program. If DEFAULT is specified, the device calculates the amount of memory needed. You can also specify the required memory in bytes. If the required memory is greater than the space available, then a parameter error "Data out of range", −222, is generated. *RST has no effect on the value of MALLOCATE.

A program name must already be defined and a program loaded before the MALLOCATE command or query may be used.

Query Response



. QGENN1

Query Example

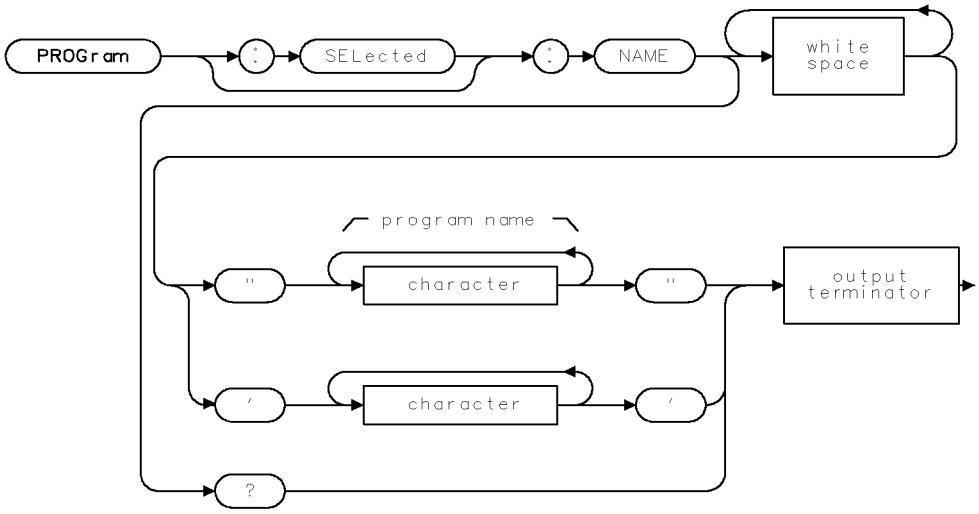
Query example:

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:MALL?"
ENTER @Mta;Stack_size
```

SELected:NAME

Defines the name of the program to be selected.

Syntax



xprg7

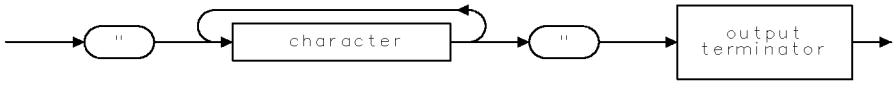
Command Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
```

Description

The NAME command defines the name of the program to be selected. If the program name already exists, then that existing program is selected. If the program name does not exist, then the new name is selected, but no program is defined by this selection. The program name is character data.

Query Response



qgenn9

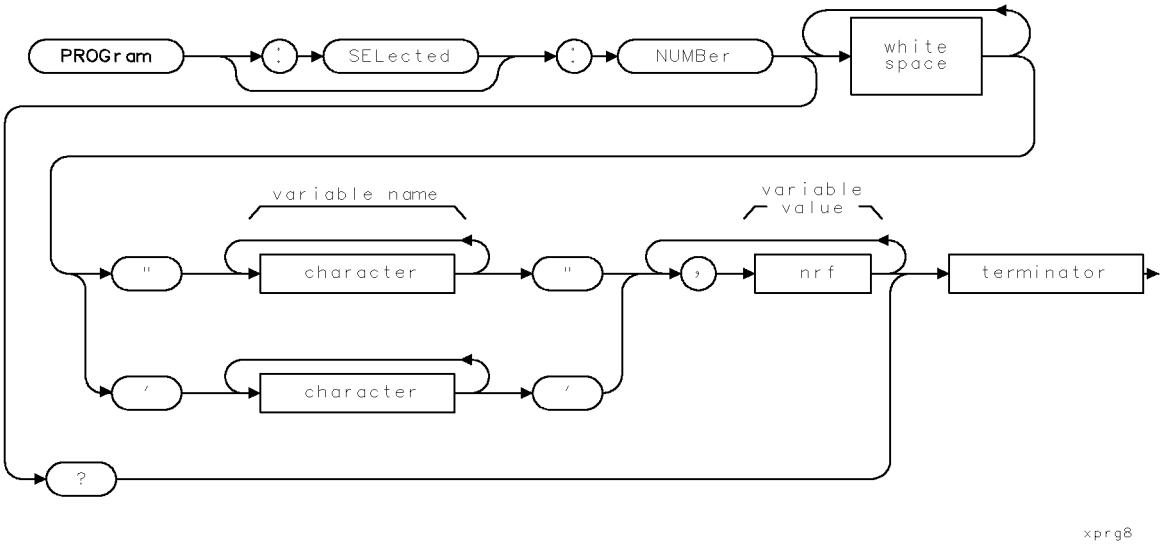
Query Example

```
OUTPUT @Mta;"PROG:NAME?"
ENTER @Mta;Name$
```

SELected:NUMBer

Used to set and query the contents of numeric program variables and arrays.

Syntax



Command Example

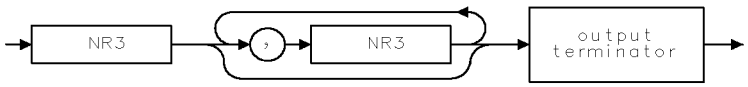
```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:NUMB 'a',2.5"
```

Description

The NUMBER command is used to set and query the contents of numeric program variables and arrays in the currently selected program. The currently selected program must be a DEFINED program, otherwise an "Illegal program name" error, −282, is generated. The variable specified is the name of an existing variable in the selected program, otherwise an "Illegal variable name" error, −283, is generated. The name can be either character data or string data. The associated values is a list of comma separated numeric values which are used to set the variable name. If the specified variable cannot hold all of the specified numeric values, a "Parameter not allowed" error, −108 , is generated.

The query returns the contents of the variable as a comma separated list.

Query Response



qgenn7

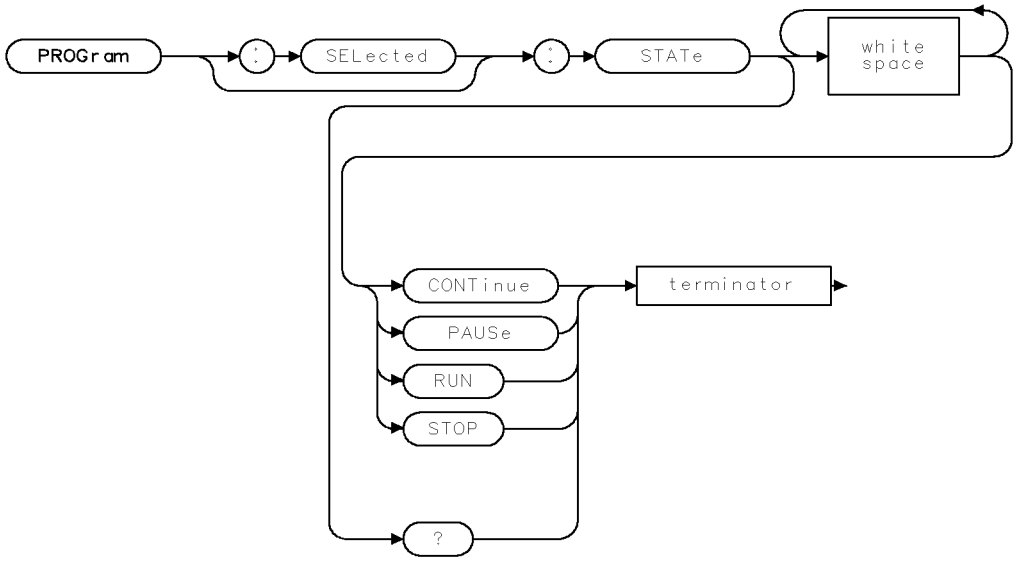
Query Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:NUMB? 'a'"
ENTER @Mta;A
```

SELected:STATe

Used to either set the state or query the state of a selected program.

Syntax



Command Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:STAT RUN"
```

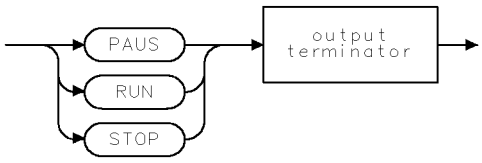
Description

The STATE command is used to either set the state or query the state of a selected program. The following table defines the effect of setting the STATE to the desired value from each of the possible current states. In certain cases a parameter error "Settings conflict", −221, is generated.

The states are described as follows:

Running:	The program is currently executing.
Paused:	The program has reached a break in execution but can be continued.
Stopped:	Execution has been terminated.

Query Response



qprg9

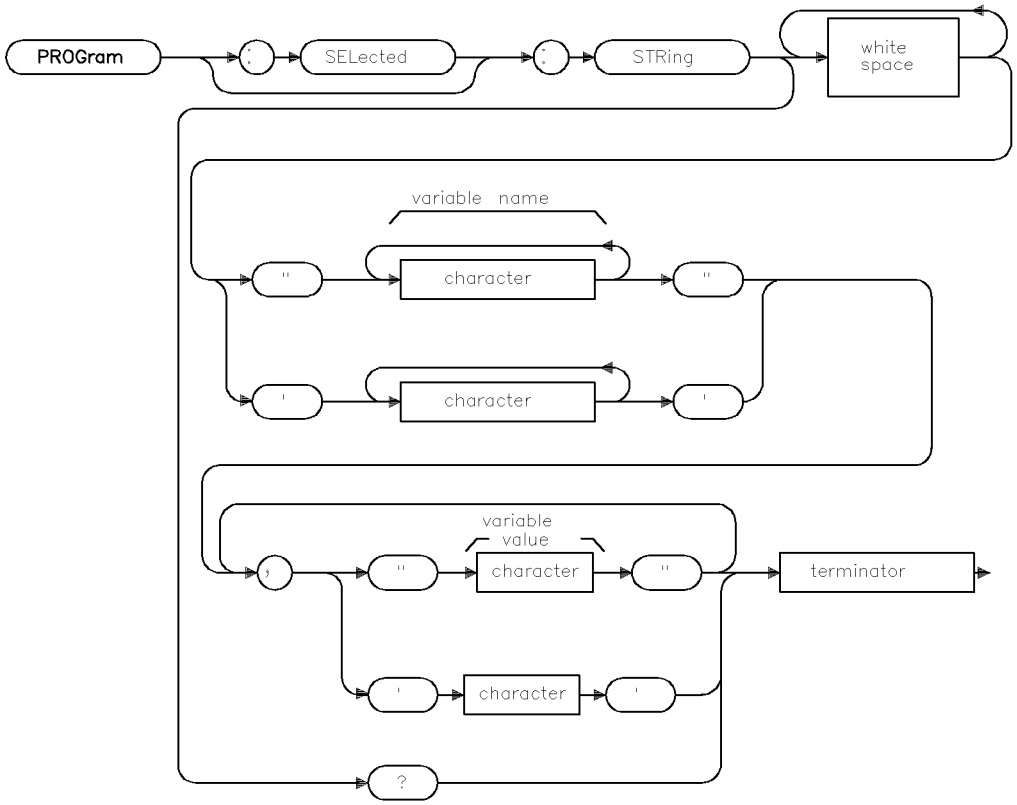
Query Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:STAT?"
ENTER @Mta;State$
```

SELected:STRing

Sets and queries the contents of string program variables and arrays.

Syntax



Command Example

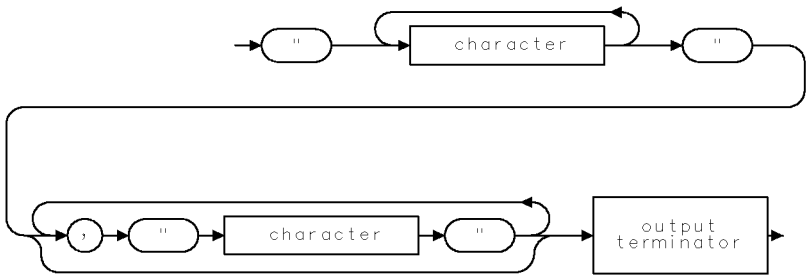
```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:STR 'b','Hello world'"
```

Description

The STRING command sets and queries the contents of string program variables and arrays in the currently selected program. The currently selected program must be a DEFINED program, otherwise an "Illegal program name" error, −282, is generated. The variable specified is the name of an existing variable in the selected program, otherwise an "Illegal variable name" error, −283, is generated. The name can be either character data or string data. String values are a list of comma separated strings which are used to set the variable. If the specified variable cannot hold all of the specified string values then a "Parameter not allowed" error, −108, is generated. If a string value is too long then it will be truncated when stored in the programs variable.

The query returns the contents of the variable as a comma separated list.

Query Response



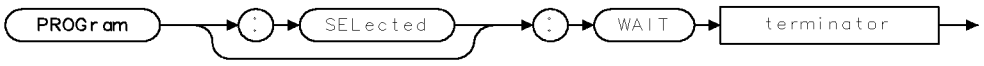
Query Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:STR? 'b'"
ENTER @Mta;B$
```

SELected:WAIT

Prevents further commands or queries to be executed.

Syntax



xprg11

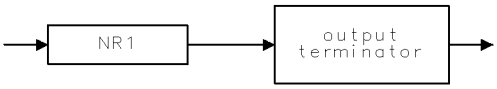
Command Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:WAIT"
```

Description

The WAIT command and its query prevents further commands or queries to be executed, until the selected program exits from the RUN state, that is either STOPPED or PAUSED. For the query, a 1 is returned in NR1 format when the program is either STOPPED or PAUSED.

Query Response



QGENN1

Query Example

```
OUTPUT @Mta;"PROG:NAME 'REMOTE'"
OUTPUT @Mta;"PROG:WAIT?"
ENTER @Mta;A$
```

22

RF Pulse Subsystem

The RF PULSE subsystem controls various parameters relating to RF pulses. In general, for correct operation with RF pulses, the following conditions must be met:

- **INPUTS:** the RF pulse must be present at one input and the CW carrier at the other input.
- **CARRIER FREQUENCY:** the RF carrier must be properly positioned in both the IF and the decimated data. See the commands PULSE:ADJUST and PULSE:CARRIER
- **ASYNCHRONOUS:** if the RF pulse profile mode is on (that is, SWEEP:FILTER is ON) the carrier must be asynchronous with respect to the modulation.

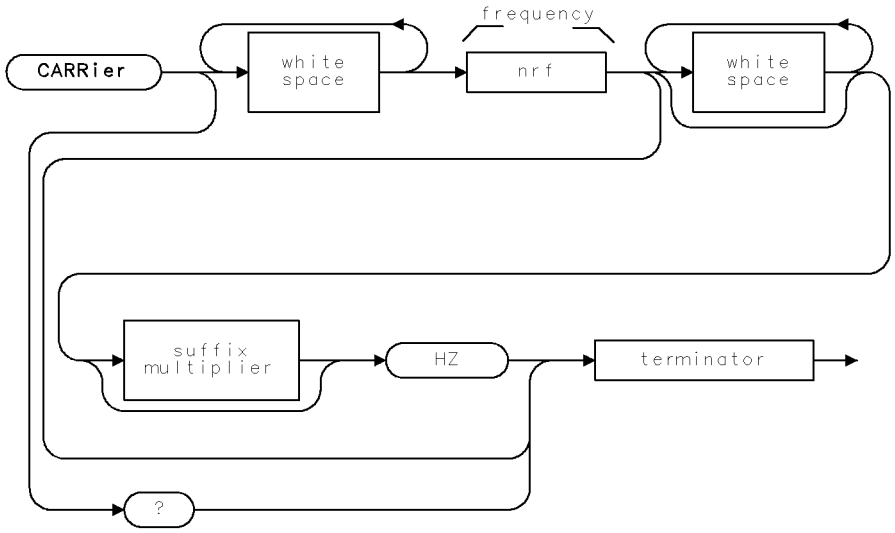
See the following list for a command syntax summary of the RF PULSE subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:PULSe		
:CARRier	<nrf>[[<suffix multiplier>]HZ]	
:MCARrier		[no query]

CARRier

Specifies the frequency of the carrier for RF pulses.

Syntax



Preset State: 3.123456789 GHz
Parameter Range: frequency = 500 kHz through 100 GHz
Fundamental Unit: hertz
Related Commands: MODulator:STATe

Command Example

```
OUTPUT @Mta;"PULS:CARR 5 GHz"
```

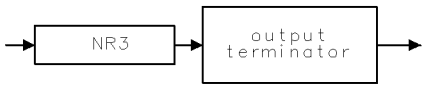
Description

The CARRIER command specifies the frequency of the carrier for RF pulses. The value for CARRIER is only used when MODULATOR:STATE is INTERNAL.

Note The value for carrier can be affected by other commands.

The query returns the current state of CARRIER.

Query Response



QGENNR3

Query Example

```
OUTPUT @Mta;"PULS:CARR?"
ENTER @Mta;Carrier_freq
```

MCARrier
(Measure Carrier)

Directs the HP 70820A to measure the frequency of the carrier.

Syntax

Command Example

```
OUTPUT @Mta;"PULS:MCAR"
```

Description

The MCARRIER command directs the microwave transition analyzer to measure the frequency of the carrier for the RF pulse profile mode of operation. The CW carrier is assumed to be present on the non-trigger source input (that is, if the trigger source is channel 1, this command assumes the carrier is present on channel 2). This overwrites the parameter set by the PULSE:CARRIER command. The results can be queried via the PULSE:CARRIER query.

23

Secure Subsystem

The SECURE subsystem commands provide a convenient means of destroying or blanking various information in the module. See the following list for a command syntax summary of the SECURE subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:SECure		
:MATH		[no query]
:MEMory	<nrf>|ALL	[no query]
:SCReen		[no query]
:STATe	<nrf>|ALL	[no query]
:UCAL	<nrf>|ALL	[no query]

MATH

Clears unused equations from the math equation buffer.

Syntax

xsec5

Command Example

```
OUTPUT @Mta;"SEC:MATH"
```

Description

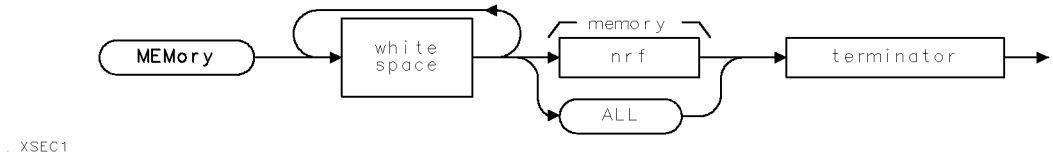
The MATH command clears unused equations from the math equation buffer. The math equation buffer acts as a recall buffer for manual entry of equations.

Note Any equation that is in use by a function is not cleared. To perform a complete clear-of-user-entered equations, precede the SECURE:MATH command with *RST.

MEMory

Erases the contents of waveform memory.

Syntax



Parameter Range: memory = 1 through 4

Command Example

```
OUTPUT @Mta;"SEC:MEM ALL"
```

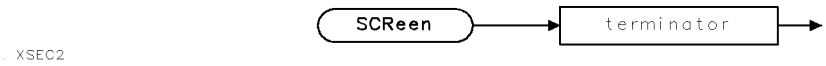
Description

The MEMORY command erases the contents of either a single waveform memory or all waveform memories.

SCReen

Blanks various portions of the graphics display.

Syntax



Command Example

```
OUTPUT @Mta;"SEC:SCR"
```

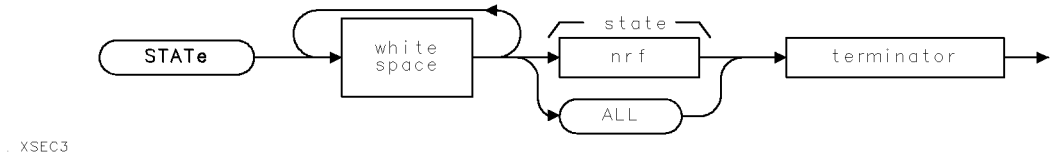
Description

The SCREEN command causes various portions of the graphics display to be blanked. The only way to reenable the graphics display is to preset the instrument (that is, *RST). The state of screen secured is stored as part of the instrument state (that is, `... SECURE:SCREEN; :*SAV 1;*RST;*RCL 1; ...` will still have the appropriate portions of the graphics display blanked).

STATe

Erases the contents of recall registers.

Syntax



Parameter Range: state = 1 through 6

Command Example

```
OUTPUT @Mta;"SEC:STAT 1"
```

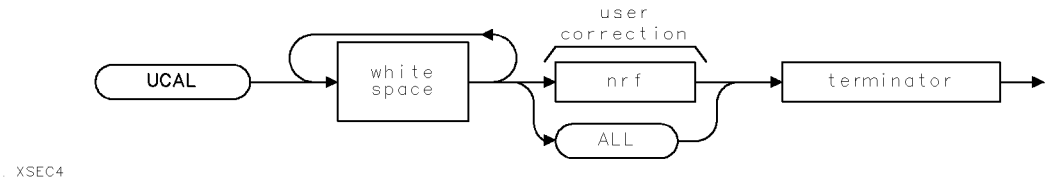
Description

The STATE command erases the contents of either a single recall state register or all recall state registers.

UCAL
(User Correction Array)

Erases the contents of user correction.

Syntax



Parameter Range: user correction = 1 or 2

Command Example

```
OUTPUT @Mta;"SEC:UCAL ALL"
```

Description

The UCAL command erases the contents of either a single user correction array or all user correction arrays.

24

Source Subsystem

The SOURCE subsystem commands control the configuration and the settings for an external source. See the following list for a command syntax summary of the SOURCE subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:SOURce		
:BERT		
:CLOCk	<nrf>[[<suffix multiplier>]HZ]	
:DIVisor	<nrf>	
:READ		
:CLOCk	OFF|ON	
:DIVisor	OFF|ON	
[:STATe]	OFF|ON	
:SIGNal	CLOCk|CDK|PATTern|PDK|PTRigger)	
:CONFigure		
:ADDRess	<nrf>[,<nrf>]	
:BUS	HPIB|HPMSIB	
:COLumn	<nrf>	
:CUSTom	"{<character>}"|'{<character>}'	[no query]
:GENeric	OFF|ON	
:HANDshake	OFF|ON	
:HP8510		
:ADDRess	<nrf>	
:DEVice	<nrf>	
[:STATe]	OFF|ON	
:RESet	OFF|ON	
:ROW	<nrf>	
:TYPE	CUSTom|HP3325|HP3335|HP7032X|HP834X| HP836XX|HP8644|HP8645|HP8657X|HP8665| HP8672X|HP8673X|NONE	
:COUPle	OFF|ON	
:FREQuency		
:CW	<nrf>[[<suffix multiplier>]HZ]	
:POWer		
:LEVel	<nrf>[DBM]	
:STATe	OFF|ON	
:PULSe		
:STATe	OFF|ON	
:USER	"{<character>}"|'{<character>}'	

Bit Error Rate Test

The BERT (Bit Error Rate Test) subsystem is intended to simplify using the HP 70820A with a BERT system, specifically, the HP 70841A, HP 70842A, HP 70845A, and HP 70846A modules.

Under most circumstances the HP 70820A is required to know the frequency of the input. Without this subsystem the user would be required to furnish the signal frequency (note that since the data is normally a psuedo-random sequence, the HP 70820A signal acquistion routines will not work). For example, with a 1 GHz clock frequency and a pattern length of $2^{15}-1$, the signal frequency is :

$$signal = \frac{1\ GHz}{2^{15}-1} \sim\ 30.518509476\ kHz$$

Using this subsystem, the HP 70820A can be instructed to ask the BERT what the pertinent parameters are, and then the HP 70820A will compute the signal frequency.

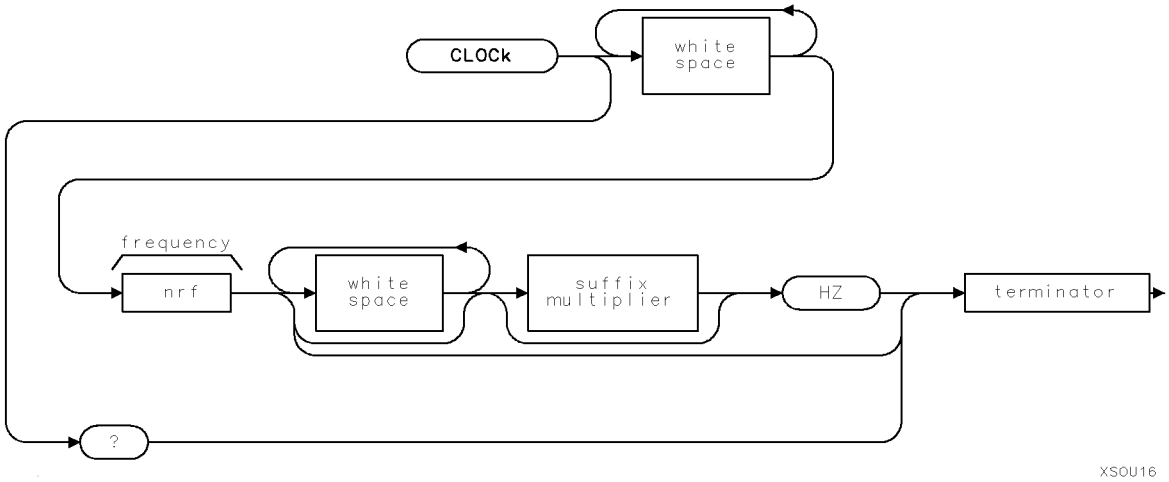
The BERT subsystem, assumes that a custom synthesizer has been defined and selected. Note that having this custom synthesizer selected will enable menus that will allow front panel access to the functions in this subsystem. The following is a BASIC program to define a custom synthesizer for the HP 7084xA modules.

```
10    ASSIGN @Ssa TO 711
20    DIM A$[5000]
30    Syn_addr=0
40    !
50    A$="SOUR:CONF:CUST "
60    A$=A$&"'7084?A*',"           ! name
70    A$=A$&"'*IDN?',"             ! ID query
80    A$=A$&"'H*P*,*7084?A,*',"    ! ID response
90    A$=A$&"'',"                  ! Reset instruction
100   A$=A$&"'',"                  ! Preset instruction(s)
110   A$=A$&"0,"                   ! freq stable mask
120   A$=A$&"0,"                   ! ampl stable mask
130   A$=A$&"'',"                  ! stable clear
140   A$=A$&".745Hz,"              ! minimum freq (min clock/max div for BERT)
150   A$=A$&"3GHz,"                ! maximum freq (max clock/min div for BERT)
160   A$=A$&"0,"                   ! freq resolution 1
170   A$=A$&"100MHz,"              ! f1 (BERT minimum clock rate)
180   A$=A$&"0,"                   ! freq resolution 2
190   A$=A$&"3GHz,"                ! f2 (BERT maximum clock rate)
200   A$=A$&"0,"                   ! freq resolution 3
210   A$=A$&"32,"                  ! f3 (BERT clock divide ratio)
220   A$=A$&"0,"                   ! freq resolution 4
230   A$=A$&"1GHz,"                ! preset freq (default BERT clock freq)
240   A$=A$&"'',"                  ! freq text
250   A$=A$&"'',"                  ! freq term
260   A$=A$&"1,"                   ! minimum ampl (BERT minimum divisor)
270   A$=A$&"134.217728E6," ! 2^27 ! maximum ampl (BERT maximum divisor)
280   A$=A$&"1,"                   ! amplitude resolution
290   A$=A$&"1,"                   ! preset amplitude (default BERT divisor)
300   A$=A$&"'',"                  ! amplitude text
310   A$=A$&"'',"                  ! amplitude term
320   A$=A$&"16384,"               ! ability  (this is a BERT)
330   A$=A$&"'ROM VALUES',"        ! AM on (clock freq query for BERT)
340   A$=A$&"'',"                  ! AM off (meas clock freq query for BERT)
350   A$=A$&"'',"                  ! FM on (divide ratio query for BERT)
360   A$=A$&"'',"                  ! FM off (pattern query for BERT)
370   A$=A$&"'',"                  ! PM on (user pattern length query for BERT)
380   A$=A$&"'',"                  ! PM off (trigger mode query for BERT)
390   A$=A$&"'',"                  ! RF on
400   A$=A$&"'',"                  ! RF off
410   A$=A$&"'',"                  ! before sweep
420   A$=A$&"'',"                  ! after sweep
430   A$=A$&"'*OPC?',"             ! handshake
440   A$=A$&"'',"                  ! Generic on (trigger mode = clock for BERT)
450   A$=A$&"'',"                  ! Generic off (trigger mode = pattern for
                                     BERT)
460   A$=A$&"''"                   ! Generic label
470   OUTPUT @Ssa;A$
480 ! OUTPUT @Ssa;"SOUR:CONF:TYPE CUST;BUS HPIB;ADDR ";Syn_addr
490   END
```

BERT:CLOCk

Sets the value the HP 70820A will use as the clock frequency of the BERT source.

Syntax



Parameter Range: Dependent on the BERT
Default State: 1 GHz

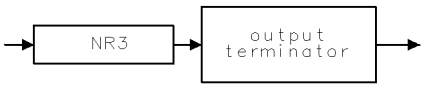
Command Example

```
OUTPUT @Mta;"SOUR:BERT:CLOC 2 GHz"
```

Description

The BERT:CLOCK command sets the value the HP 70820A will use as the clock frequency of the BERT source. This does NOT set the clock frequency of the actual BERT system. This is essentially a “manual override” of the HP 70820A “polling” of the BERT clock. The query returns the current setting of BERT:CLOCK.

Query Response



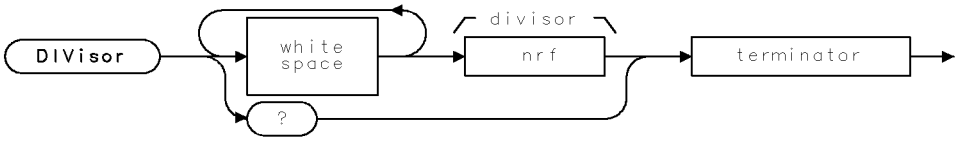
Query Example

```
OUTPUT @Mta;"SOUR:BERT:CLOC?"
ENTER @Mta;Clock_rate
```

BERT:DIVisor

Sets the value the HP 70820A will use as the divide ratio.

Syntax



Parameter Range: $1 \leq$ divisor $\leq$ dependent on the BERT
Default State: 1

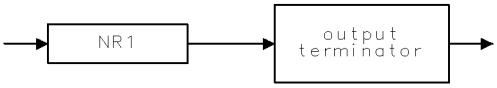
Command Example

```
OUTPUT @Mta;"SOUR:BERT:DIV 1024"
```

Description

The BERT:DIVISOR command sets the value the HP 70820A will use as the divide ratio between the clock frequency and the trigger or pattern frequency of the BERT. This does NOT set the divide ratio of the actual BERT system. This is essentially a "manual override" of the HP 70820A "polling" of the BERT divide ratio. The query returns the current setting of BERT:DIVISOR.

Query Response



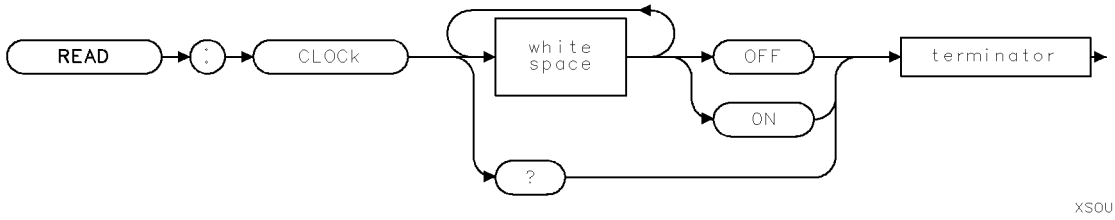
Query Example

```
OUTPUT @Mta;"SOUR:BERT:DIV?"
ENTER @Mta;Divisor
```

BERT:READ:CLOCk

Determines if value of BERT:CLOCK will be affected when executing BERT:READ.

Syntax



Default State: ON

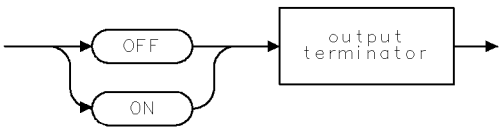
Description

The BERT:READ:CLOCK command determines if the value of BERT:CLOCK will be affected when executing the BERT:READ command. The query returns the current setting of BERT:READ:CLOCK.

Command Example

```
OUTPUT @Mta;"SOUR:BERT:READ:CLOC OFF"
```

Query Response



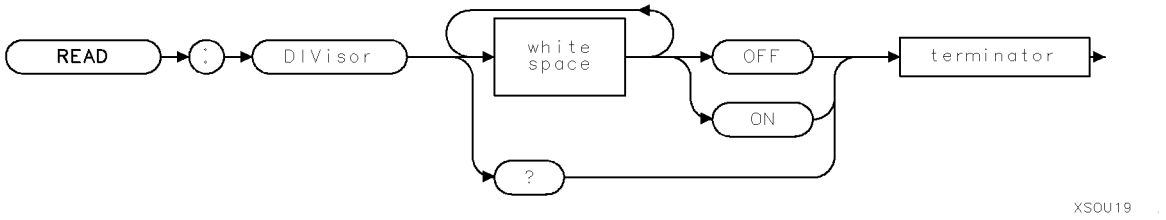
Query Example

```
OUTPUT @Mta;"SOUR:BERT:READ:CLOC?"
ENTER @Mta;Clock$
```

BERT:READ:DIVisor

Determines if the value of BERT:DIVISOR will be affected when executing BERT:READ.

Syntax



Default State: ON

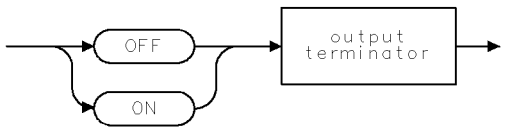
Command Example

```
OUTPUT @Mta;"SOUR:BERT:READ:DIV OFF"
```

Description

The BERT:READ:DIVISOR command determines if the value of BERT:DIVISOR will be affected when executing the BERT:READ command. The query returns the current setting of BERT:READ:DIVISOR.

Query Response



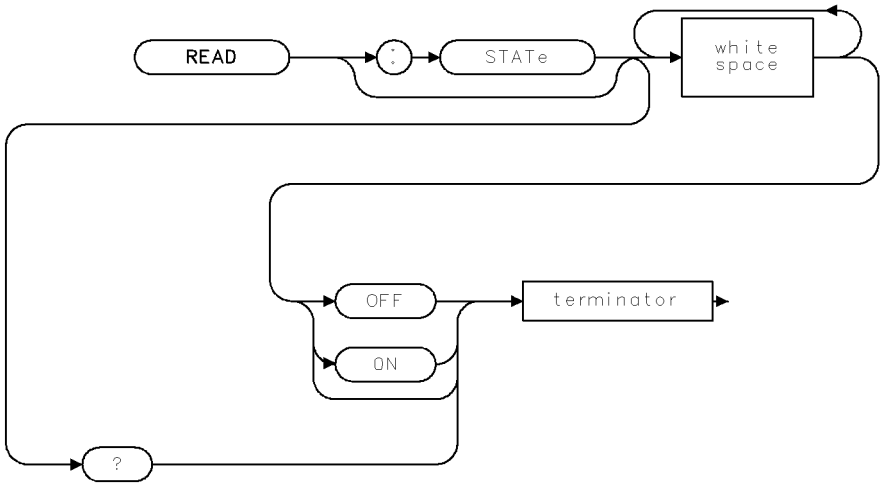
Query Example

```
OUTPUT @Mta;"SOUR:BERT:READ:DIV?"
ENTER @Mta;Divisor$
```

BERT:READ:STATe

Determines when the clock frequency and divisor are read from the BERT.

Syntax



Default state: OFF

Command Example

```
OUTPUT @Mta;"SOUR:BERT:READ"          ! Poll BERT immediately
OUTPUT @Mta;"SOUR:BERT:READ:STAT ON" ! Poll BERT continuously
```

Description

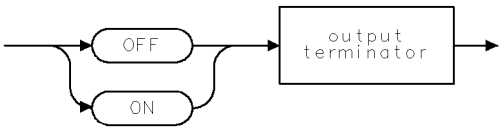
The BERT:READ:STATE command determines when the BERT is polled for the values of clock frequency and divisor. The results of the poll are available via BERT:CLOCK and BERT:DIVISOR. The query returns the current setting of BERT:READ:STATE.

OFF: Disables the polling of the BERT before sweeps.

ON: Enables the polling of the BERT for CLOCK frequency and DIVISOR before sweeps are started.

Default: Perform a poll of the BERT immediately. Does not change the current polling STATE.

Query Response



QGENON

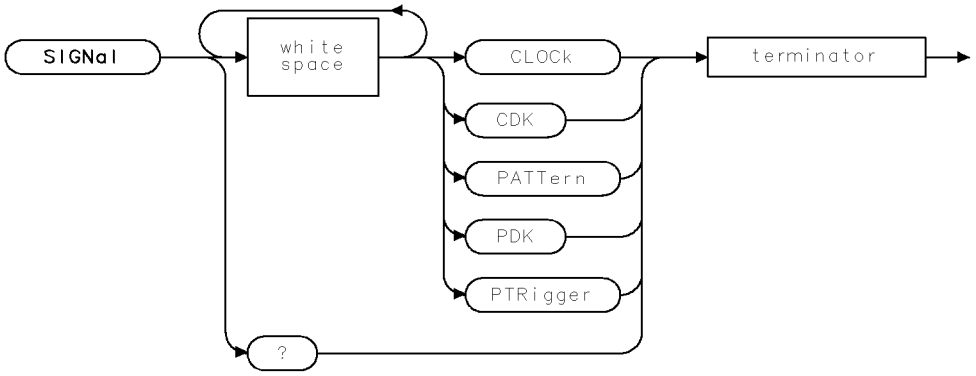
Query Example

```
OUTPUT @Mta;"SOUR:BERT:READ?"
ENTER @Mta;State$
```

BERT:SIGNal

Determines how BERT:CLOCK and BERT:DIVISOR will couple into SWEEP:SIGNAL or PULSE:CARRIER.

Syntax



Default State: CLOCk
Related Commands: PULSe:CARRier
SWEep:SIGnal

Command Example

```
OUTPUT @Mta;"SOUR:BERT:SIGN PATT"
```

Description

The BERT:SIGNAL command determines how BERT:CLOCK and BERT:DIVISOR will couple into SWEEP:SIGNAL or PULSE:CARRIER. This command affects the state of SOURCE:GENERIC (which is used to control the trigger output of the HP 70841A). The query returns the current setting of BERT:SIGNAL.

CLOCK: The value coupled into SWEEP:SIGNAL or PULSE:CARRIER is BERT:CLOCK. BERT:DIVISOR has no effect. SOURCE:GENERIC is unchanged.

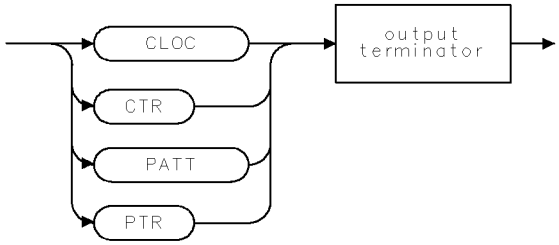
CDK (CLOCK Divided by K): The value coupled into SWEEP:SIGNAL or PULSE:CARRIER is BERT:CLOCK / k. The value of "k" is dependent on the pattern generator. For the HP 70841A, the value is 32. SOURCE:GENERIC is set to ON (trigger out = CLOCK).

PATTERN: The value coupled into SWEEP:SIGNAL or PULSE:CARRIER is BERT:CLOCK / BERT:DIVISOR. SOURCE:GENERIC is unchanged.

PDK (Pattern Divided by K): The value coupled into SWEEP:SIGNAL or PULSE:CARRIER is BERT:CLOCK / BERT:DIVISOR / k. The value of "k" is dependent on the pattern generator. For the HP 70841A the value is 32. SOURCE:GENERIC is set to ON (trigger out = CLOCK).

PTRIGGER: The value coupled into SWEEP:SIGNAL or PULSE:CARRIER is BERT:CLOCK / BERT:DIVISOR / n. The value of "n" is normally 1 but for certain pattern generators (notably the HP 70841A) may have a value of 1 through 32. "n" represents the ratio of the pattern frequency to the trigger frequency. SOURCE:GENERIC is set to OFF (trigger out = PATTERN).

Query Response



QSOU21

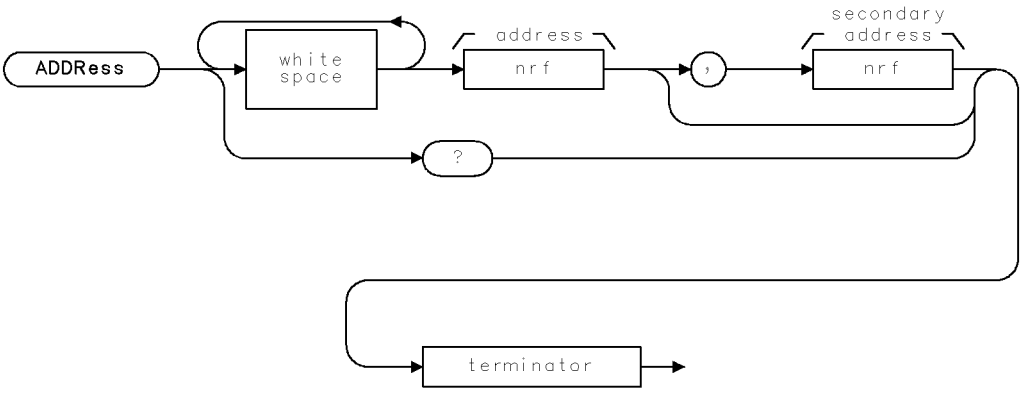
Query Example

```
OUTPUT @Mta;"SOUR:BERT:SIGN?"
ENTER @Mta;Signal$
```

CONFigure:ADDRess

Defines the address of the external synthesizer.

Syntax



Parameter Range:	address = 0 through 255; secondary address = 0 through 30
Preset State:	Not affected by *RST or IP
Power-On State:	see CONFigure:SEARch (CONFIGURE subsystem)

Command Example

Example 1

```
OUTPUT @Mta;"SOUR:CONF:ADDR 17;BUS HPMSIB;TYPE HP836XX;"
```

Example 2

```
OUTPUT @Mta;"SOUR:CONF:ADDR 17,5;BUS HPIB;TYPE HP834X;"
```

Description

The CONFIGURE:ADDRESS command defines the address of the external synthesizer. The query returns the current setting of the address.

This command works in conjunction with the SOURCE:CONFIGURE:TYPE and SOURCE:CONFIGURE:BUS commands to define the synthesizer. After receipt of a program message terminator (generally a line feed), the synthesizer is set to a preset state.

The optional second parameter determines whether secondary addressing on HP-IB should be used. If the parameter is omitted, then primary addressing will be used on HP-IB. If the parameter is sent, then secondary addressing will be used on HP-IB.

Note This parameter has no effect if BUS is set to HPMSIB.

The query returns −1 for the secondary address if there is no defined secondary address (that is, the secondary address was omitted in the previous ADDRESS command).

Note Sending the commands `SOURCE:CONFIGURE:ROW` n and `SOURCE:CONFIGURE:COLUMN` m is equivalent to `SOURCE:CONFIGURE:ADDRESS` $32 \times n + m$.

Query Response

qgenn8

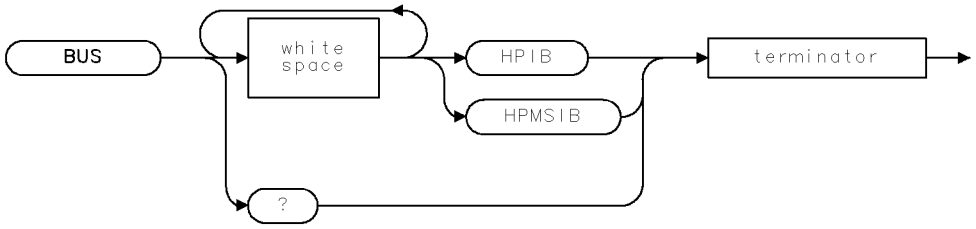
Query Example

```
OUTPUT @Mta;"SOUR:CONF:ADDR?"
ENTER @Mta;Address, Secondary
```

CONFigure:BUS

Specifies either HP-IB or HP-MSIB for the external synthesizer.

Syntax



Preset State: Not affected by *RST or IP
Power-On State: see CONFigure:SEARch in the CONFIGURE subsystem

Command Example

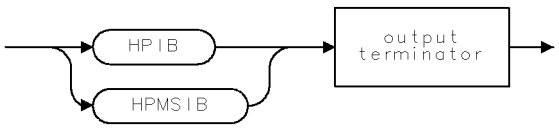
```
OUTPUT @Mta;"SOUR:CONF:BUS HPIB;TYPE HP8340"
```

Description

The CONFIGURE:BUS command defines the bus (either HP-IB or HP- MSIB) of the external synthesizer. The query returns the current setting of the bus.

This command works in conjunction with the SOURCE:CONFIGURE:TYPE and SOURCE:CONFIGURE:ADDRESS commands to define the synthesizer. After receipt of a program message terminator (generally a line feed), the synthesizer is set to a preset state.

Query Response



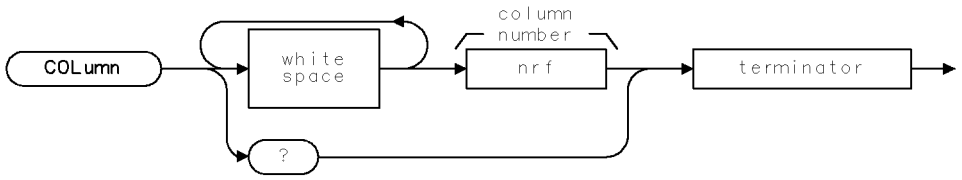
Query Example

```
OUTPUT @Mta;"SOUR:CONF:BUS?"
ENTER @Mta;Bus$
```

CONFigure:COLumn

Defines the column portion of the external synthesizer address.

Syntax



Parameter Range: column number = 0 through 31
Preset State: Not affected by *RST or IP
Power-On State: see CONFigure:SEARch in the CONFIGURE subsystem

Command Example

```
OUTPUT @Mta;"SOUR:CONF:ROW 0;COL 17;TYPE HP8340"
```

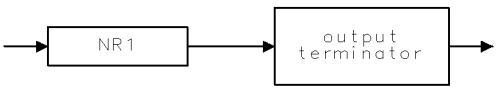
Description

The CONFIGURE:COLUMN command defines the column portion of the address of the external synthesizer (that is, bits 0 through 4). The query returns the current setting of the column.

This command works in conjunction with the SOURCE:CONFIGURE:TYPE, SOURCE:CONFIGURE:BUS, and SOURCE:CONFIGURE:ROW commands to define the synthesizer. After receipt of a program message terminator (generally a line feed), the synthesizer is set to a preset state.

Note ROW n; COLumn m; is equivalent to ADDRess $32 \times n + m$

Query Response



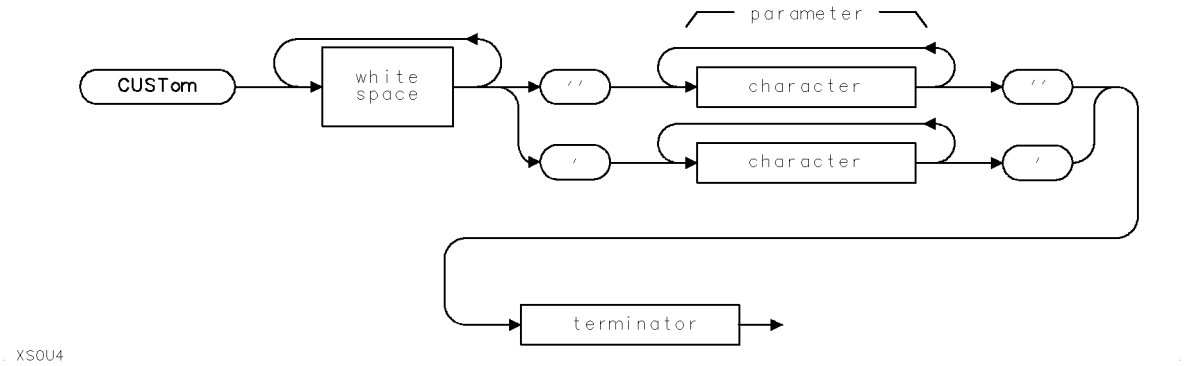
Query Example

```
OUTPUT @Mta;"SOUR:CONF:COL?"
ENTER @Mta;Column
```

CONFigure:CUSTom

Defines the control parameters for the CUSTOM synthesizer source.

Syntax



Preset State: not affected by *RST, IP, or power cycle
Parameter Range: the total amount of text stored is limited to $\leq$ 256 bytes

Command Example

See the example at the end of this command description.

Description

The CONFIGURE:CUSTOM command defines the control parameters for the CUSTOM synthesizer source (selected with SOURCE:CONFIGURE:TYPE ...).

The parameters, in order, for this command are:

1. **synth name:** The text expected from the synthesizer on an OUTPUT IDENTITY command (that is, the text that appears in the address map of the modular displays). This text is placed on the menu key for selecting CUSTOM as well as being used as an identity check on HP-MSIB. **Syntax:** string
2. **idn query:** The remote programming command to send to the synthesizer to determine its name. This is used as an identity check for both HP-MSIB and HP-IB). **Syntax:** string
3. **idn response:** The expected response to the idn query. **Syntax:** string
4. **reset text:** The reset instruction for the synthesizer. This instruction also recognizes 'DEVICE CLEAR' which will cause a selected device clear to be sent to the source. **Syntax:** string
5. **preset text:** The text sent to initialize the synthesizer. **Syntax:** string
6. **freq stable mask:** This masks the serial poll byte for the bit which indicates the frequency is stable after a change. Send 0 if not supported. **Syntax:** <NR1> (ASCII integer)
7. **ampl stable mask:** This masks the serial poll bye for the bit which indicates the amplitude is stable after a change. Send 0 if not supported. **Syntax:** <NR1> (ASCII integer)

8. **stable clear:** The text sent to the synthesizer to clear the frequency and amplitude stable bits. If a response is expected, precede the text with a question mark (which is not sent). Send a null string if not supported. **Syntax:** string
9. **min freq:** The minimum frequency the synthesizer can be set to. **Syntax:** <NR3> [xHz]
10. **max freq:** The maximum frequency the synthesizer can be set to. **Syntax:** <NR3> [xHz]
11. **freq resol 1:** The frequency resolution of the synthesizer for $0 < f < f1$. **Syntax:** <NR3> [xHz]
12. **f1** The first frequency breakpoint for determining the resolution of the synthesizer. **Syntax:** <NR3> [xHz]
13. **freq resol 2:** The frequency resolution of the synthesizer for $f1 \leq f < f2$. **Syntax:** <NR3> [xHz]
14. **f2** The second frequency breakpoint for determining the resolution of the synthesizer. **Syntax:** <NR3> [xHz]
15. **freq resol 3:** The frequency resolution of the synthesizer for $f2 \leq f < f3$. **Syntax:** <NR3> [xHz]
16. **f3** The third frequency breakpoint for determining the resolution of the synthesizer. **Syntax:** <NR3> [xHz]
17. **freq resol 4:** The frequency resolution of the synthesizer for $f \geq f3$. **Syntax:** <NR3> [xHz]
18. **preset freq:** The frequency to set the synthesizer to at instrument preset. **Syntax:** <NR3> [xHz]
19. **freq text:** The remote programming command for frequency. A numeric argument plus suffix will be appended to form the complete command. **Syntax:** string
20. **freq suffix:** The text to use as the suffix for the frequency remote programming command. See below for a more complete discussion of suffixes. **Syntax:** string
21. **min ampl:** The minimum amplitude the synthesizer can be set to. **Syntax:** <NR3> [dBm]
22. **max ampl:** The maximum amplitude the synthesizer can be set to. **Syntax:** <NR3> [dBm]
23. **ampl resol:** The amplitude resolution of the synthesizer. **Syntax:** <NR3> [dBm]
24. **preset ampl:** The amplitude to set the synthesizer to at instrument preset. **Syntax:** <NR3> [dBm]
25. **ampl text:** The remote programming command for amplitude. A numeric argument plus suffix will be appended to form the complete command. **Syntax:** string
26. **ampl suffix:** The text to use as the suffix for the amplitude remote programming command. See below for a more complete discussion of suffixes. **Syntax:** string

27. **ability:** This integer describes the capability of the synthesizer as follows:
    - bit 0: AM supported, AM on and AM off fields should be non-null
    - bit 1: FM supported, FM on and FM off fields should be non-null
    - bit 2: PM supported, PM on and PM off fields should be non-null
    - bit 3: RF on/off supported, RF on and RF off fields should be non-null
    - bit 4: reserved (set to 0)
    - bit 5: Serial poll, set to 1 if the module does NOT support serial poll. The frequency and amlitude stable masks should be 0.
    - bit 6: reserved (set to 0)
    - bit 7: reserved (set to 0)
    - bit 8: Query for minimum frequency
    - bit 9: Query for maximum frequency
    - bit 10: Query for minimum amplitude
    - bit 11: Query for maximum amplitude
    - bit 12: Wait aproximately 750 ms after sending the reset instruction
    - bit 13: reserved (set to 0)
    - bit 14: reserved (set to 0)
    - bit 15: reserved (set to 0)

    Syntax: NR1 (ASCII integer)
28. **AM on:** Text to sent to the synthesizer to turn on AM. **Syntax:** string, send a null string if not supported
29. **AM off:** Text to sent to the synthesizer to turn off AM. **Syntax:** string, send a null string if not supported
30. **FM on:** Text to send to the synthesizer to turn on FM. **Syntax:** string, send a null string if not supported
31. **FM off:** Text to send to the synthesizer to turn off FM. **Syntax:** string, send a null string if not supported
32. **PM on:** Text to send to the synthesizer to turn on PM. **Syntax:** string, send a null string if not supported
33. **PM off:** Text to send to the synthesizer to turn off PM. **Syntax:** string, send a null string if not supported
34. **RF on:** Text to send to the synthesizer to turn on the RF. **Syntax:** string, send a null string if not supported
35. **RF off:** Text to send to the synthesizer to turn off the RF. **Syntax:** string, send a null string if not supported
36. **display off:** Text to send to the synthesizer to turn off the synthesizer display. **Syntax:** string

37. **display on:** Text to send to the synthesizer to turn on the synthesizer display. **Syntax:** string

38. **handshake:** Text to send to the synthesizer as a query to determine when the previous commands have been parsed. This query should generate a response terminated by LF or EOI. For synthesizers supporting IEEE488.2, use the *OPC? query. **Syntax:** string

The string comparisons on 'synth name' and 'idn response' support the following special characters:

*	matches any string. For example: "H*P*83620*" matches "HEWLETT-PACKARD,83620A,.... "
?	matches any character. For example: "8362?" matches "83623"
|	multiple string. String1|string2 does a string compare of the input with string1 and, if there is no match, with string2. For example, "83620|83623" matches "83623"
\	cancels the special meaning of the following character, use \\ to match \. For example: "TEST*" matches "TEST*"

The suffixes for the frequency and amplitude remote commands follow these rules:

- if the first character of the suffix is a space, append the suffix with no changes.
- if the first character of the suffix is alphanumeric, precede the suffix with the appropriate multiplier. For example: 1000 Hz becomes 1 kHz.
- if the first character of the suffix is ?, determine the normal suffix multiplier and use it to "index" through a comma-delimited suffix string to find the appropriate suffix. For example, `suffix = "? HZ,kKZ,MMZ,x "` would produce the following: 10 -> 10 HZ, 1000 -> 1 KZ, 1000000 -> 1 MZ. For any other multiplier, the suffix is space.

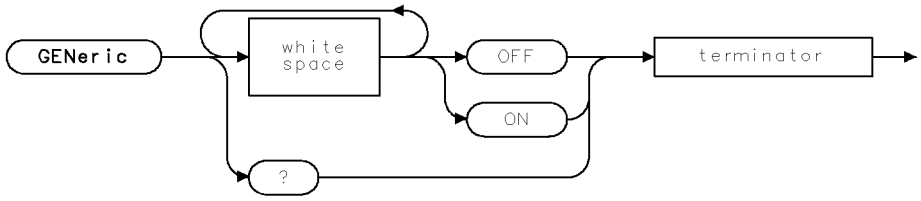
```
10    ASSIGN @Mta TO 710
20    DIM A$[5000]
30    Syn_addr=0
40    !
50    A$="SOUR:CONF:CUST "
60    A$=A$&"'83623',"              ! name
70    A$=A$&"'*IDN?',"              ! ID query
80    A$=A$&"'H*P*,*83623A*',"      ! ID response
81    A$=A$&"'*RST;',"              ! Reset instruction
90    A$=A$&"':STAT:PRES;:STAT:OPER:PTR 0;NTR 2;ENAB 2;"
91    A$=A$&":*SRE 50;:POW:STAT 0;:PULS:STAT0;"&CHR$(10)&"',"
100   A$=A$&"128,"                  ! freq stable mask
110   A$=A$&"0,"                    ! ampl stable mask
120   A$=A$&"'?STAT:OPER:EVEN?',"   ! stable clear
130   A$=A$&"10MHz,"                ! minimum freq
140   A$=A$&"20GHz,"                ! maximum freq
150   A$=A$&"1,"                    ! freq resolution 1
160   A$=A$&"100GHz,"               ! f1 (resolution breakpoint)'
170   A$=A$&"1,"                    ! freq resolution 2
180   A$=A$&"100GHz,"               ! f2 (resolution breakpoint)'
190   A$=A$&"1,"                    ! freq resolution 3
200   A$=A$&"100GHz,"               ! f3 (resolution breakpoint)'
```

```
210   A$=A$&"1,"                    ! freq resolution 4
220   A$=A$&"1GHz,"                 ! preset freq
230   A$=A$&"'FREQ:CW ',"           ! freq text
240   A$=A$&"'HZ',"                 ! freq term
250   A$=A$&"-110,"                 ! minimum ampl
260   A$=A$&"20,"                   ! maximum ampl
270   A$=A$&"0.01,"                 ! amplitude resolution
280   A$=A$&"-10,"                  ! preset amplitude
290   A$=A$&"'POW:LEV ',"           ! amplitude text
300   A$=A$&"' DBM',"               ! amplitude term
310   A$=A$&"12,"                   ! ability
320   A$=A$&"'',"                   ! AM on
330   A$=A$&"'',"                   ! AM off
340   A$=A$&"'',"                   ! FM on
350   A$=A$&"'',"                   ! FM off
360   A$=A$&"'PULS:SOUR EXT;STAT 1'," ! PM on
370   A$=A$&"'PULS:STAT 0',"        ! PM off
380   A$=A$&"'POW:STAT 1',"         ! RF on
390   A$=A$&"'POW:STAT 0',"         ! RF off
400   A$=A$&"'DISP:STAT 0',"        ! display off
410   A$=A$&"'DISP:STAT 1'"         ! display on
415   A$=A$&"'*OPC?'"               ! handshake
420   OUTPUT @Mta;A$
430   OUTPUT @Mta;"SOUR:CONF:TYPE CUST;BUS HPIB;ADDR ";Syn_addr
440   END
```

CONFigure:GENeric

Turns a user defined function of the external synthesizer ON or OFF.

Syntax



Preset State: OFF (if supported by synthesizer)

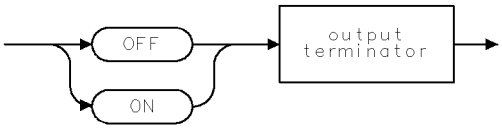
Command Example

```
OUTPUT @Mta;"SOUR:CONF:GEN ON"
```

Description

The GENERIC command turns a user defined function of the external synthesizer ON or OFF. The operation is defined in the "GENERIC ON" and "GENERIC OFF" fields of the custom synthesizer definition. The query returns the current setting.

Query Response



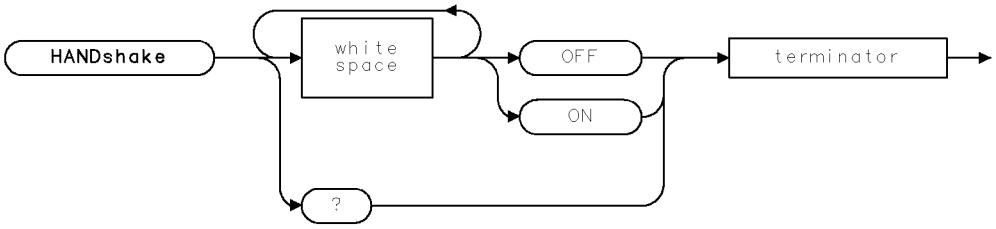
Query Example

```
OUTPUT @Mta;"SOUR:CONF:GEN?"
ENTER @Mta;Generic$
```

CONFigure:HANDshake

Enables or disables the use of SRQ for the external synthesizer.

Syntax



Preset State: ON
Related Commands: SWEep:FREQuency:DWELl
SWEep:POWer:DWELl

Command Example

```
OUTPUT @Mta;"SOUR:CONF:HAND OFF"
```

Description

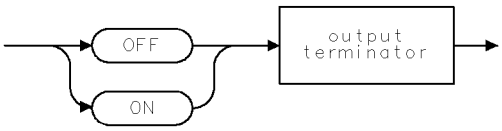
The CONFIGURE:HANDSHAKE command enables or disables the use of SRQ on HP-IB for determining when the external synthesizer is stable. The query returns the current state of HANDSHAKE.

The state of HANDSHAKE has no effect if SOURCE:CONFIGURE:BUS is HPMSIB.

The state of HANDSHAKE should normally be left ON unless the synthesizer is being used in a situation where SRQ may not operate normally (for example, if the synthesizer is controlled via a “pass through” mode of another instrument that does not support “pass through SRQ”.

When HANDSHAKE is OFF, it is up to the user to set the dwell time of the measurement larger than the longest programming and settling time of the synthesizer (which can include relay switching on power sweeps).

Query Response



QGENON

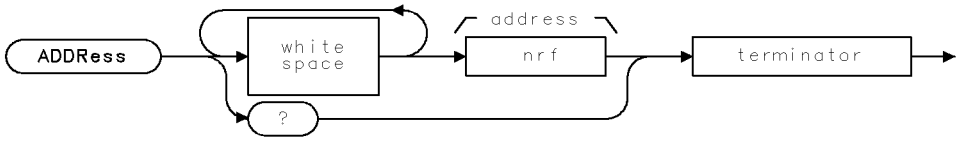
Query Example

```
OUTPUT @Mta;"SOUR:CONF:HAND?"
ENTER @Mta;Handshake$
```

CONFigure:HP8510:ADDRess

Defines the HP-IB address of the HP 8510 for pass-through mode.

Syntax



Power-On State:	19
Preset State:	Not affected by IP, *RST, or *RCL
Parameter Range:	address = 0 through 31
Related Commands:	SOURce:CONFigure:HP8510:DEVice
	SOURce:CONFigure:HP8510:STATe

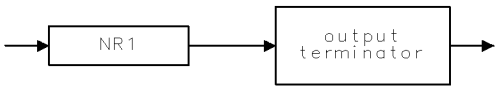
Command Example

```
OUTPUT @Mta;"SOUR:CONF:HP8510:ADDR 15"
```

Description

The ADDRESS command defines the HP-IB address of the HP 8510 to use for pass-through mode. The query returns the current setting of ADDRESS.

Query Response



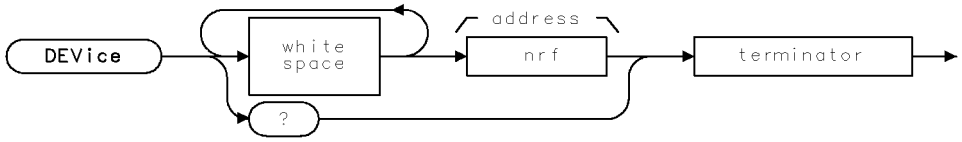
Query Example

```
OUTPUT @Mta;"SOUR:CONF:HP8510:ADDR?"
ENTER @Mta;Address
```

CONFigure:HP8510:DEVice

Defines the HP-IB address of the device on the HP 8510 system bus.

Syntax



Power on State: 19
Preset State: Not affected by IP, *RST, or *RCL
Parameter Range: address = 0 through 31
Related Commands: SOURce:CONFigure:HP8510:ADDRess
SOURce:CONFigure:HP8510:STATe

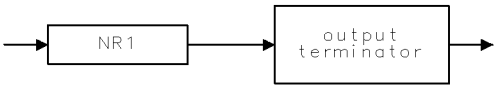
Command Example

```
OUTPUT @Mta;"SOUR:CONF:HP8510:DEV 17"
```

Description

The DEVICE command defines the HP-IB address of the device (usually a synthesizer) on the HP 8510 system bus to use in pass-through mode. The query returns the current setting of DEVICE.

Query Response



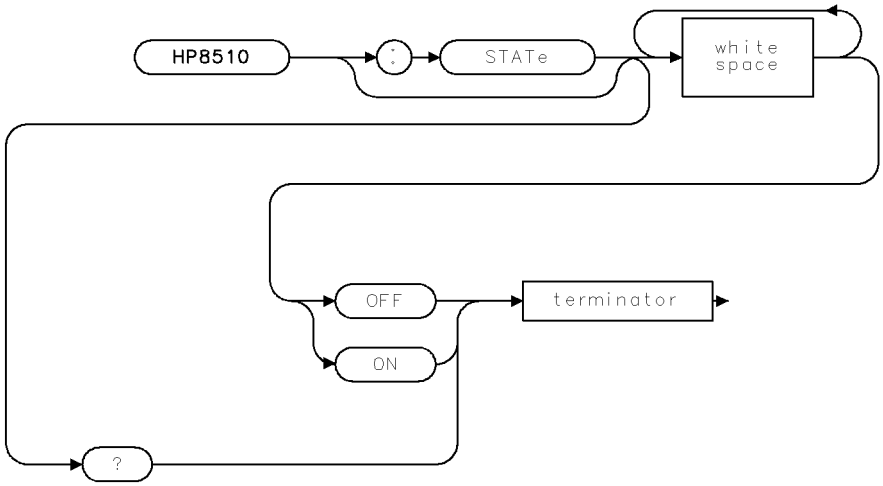
Query Example

```
OUTPUT @Mta;"SOUR:CONF:HP8510:DEV?"
ENTER @Mta;Device
```

CONFigure:HP8510:STATe

Enables or disables the pass-through mode.

Syntax



Power-On State: OFF
Preset State: Not affected by IP, *RST, or *RCL

Command Example

```
OUTPUT @Mta;"SOUR:CONF:HP8510 ON"
```

Description

The STATE command enables or disables the pass-through mode of an HP 8510. The query returns the current value of STATE.

A typical sequence for using pass-through would be:

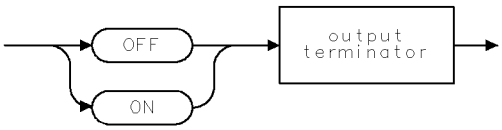
- define the HP 8510 address with SOURCE:CONFIGURE:HP8510:ADDRESS
- define the device address with SOURCE:CONFIGURE:HP8510:DEVICE
- turn pass-through mode ON
- the HP 70820A sends “ADDRPASS nn” to address xx
  - xx is the address of the HP 8510
  - nn is the address of the device on the HP 8510 system bus (usually a synthesizer)
- configure the synthesizer on the HP 70820A. Use the address of the HP 8510 system bus (usually 17) as the address of the synthesizer.

A typical sequence for using the pass-through mode for measurements would be:

- unconfigure the synthesizer
- turn pass-through mode OFF
- use the HP 70820A to address the HP 8510 to indicate pass-through mode is finished

- use the HP 70820A unaddress the HP-IB and unassert REN

Query Response



QGENON

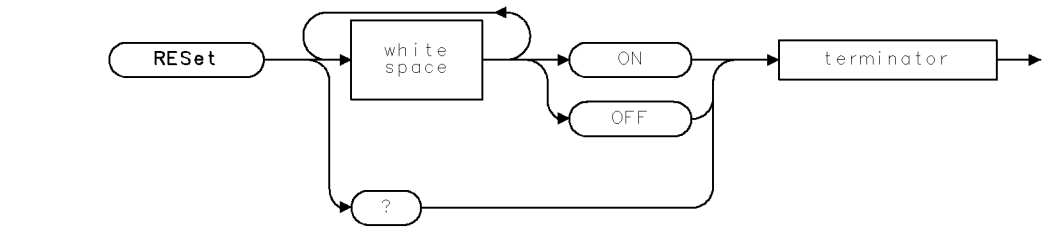
Query Example

```
OUTPUT @Mta;"SOUR:CONF:HP8510?"
ENTER @Mta;Passthru$
```

CONFigure:RESet

Controls the synthesizer reset instruction.

Syntax



Power-On State: ON
Preset State: not affected by IP, *RST, or *RCL

Command Example

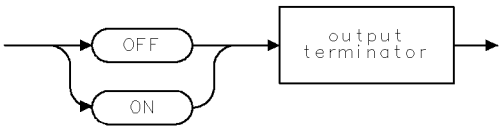
```
OUTPUT @Mta;"SOUR:CONF:RES OFF"
```

Description

The CONFIGURE:RESET command enables or disables the reset instruction that may be sent to the synthesizer when the synthesizer is being initialized (for example, at IP, *RST, or at configuration). This allows the user to set up states in the synthesizer not supported by the microwave transition analyzer and prevents the microwave transition analyzer from clearing these states (for example, FM).

The query returns the current state of CONFIGURE:RESET.

Query Response



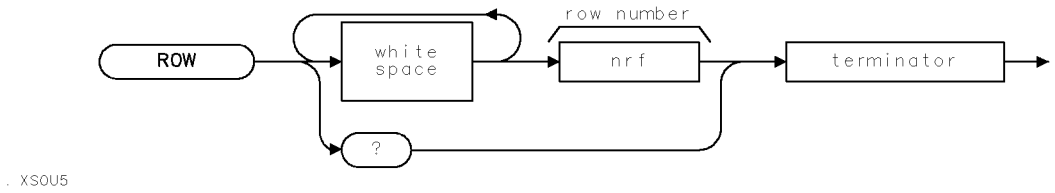
Query Example

```
OUTPUT @Mta;"SOUR:CONF:RES?"
ENTER @Mta;Reset$
```

CONFigure:ROW

Defines the row portion of the external synthesizer address.

Syntax



Preset State: not affected by *RST, IP, or power cycle
Parameter Range: row number = 0 through 7

Command Example

```
OUTPUT @Mta;"SOUR:CONF:ROW 0;COL 17;TYPE HP8340"
```

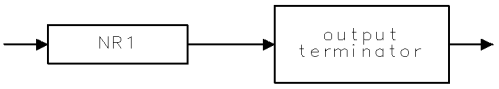
Description

The CONFIGURE:ROW command defines the row portion of the address of the external synthesizer (that is, bits 5 through 7). The query returns the current setting of the row.

This command works in conjunction with the SOURCE:CONFIGURE:TYPE, SOURCE:CONFIGURE:BUS, and SOURCE:CONFIGURE:COLUMN commands to define the synthesizer. After receipt of a program message terminator (generally a line feed), the synthesizer is set to a preset state.

Note ROW n; COLumn m; is equivalent to ADDRess $8 \times n + m$

Query Response



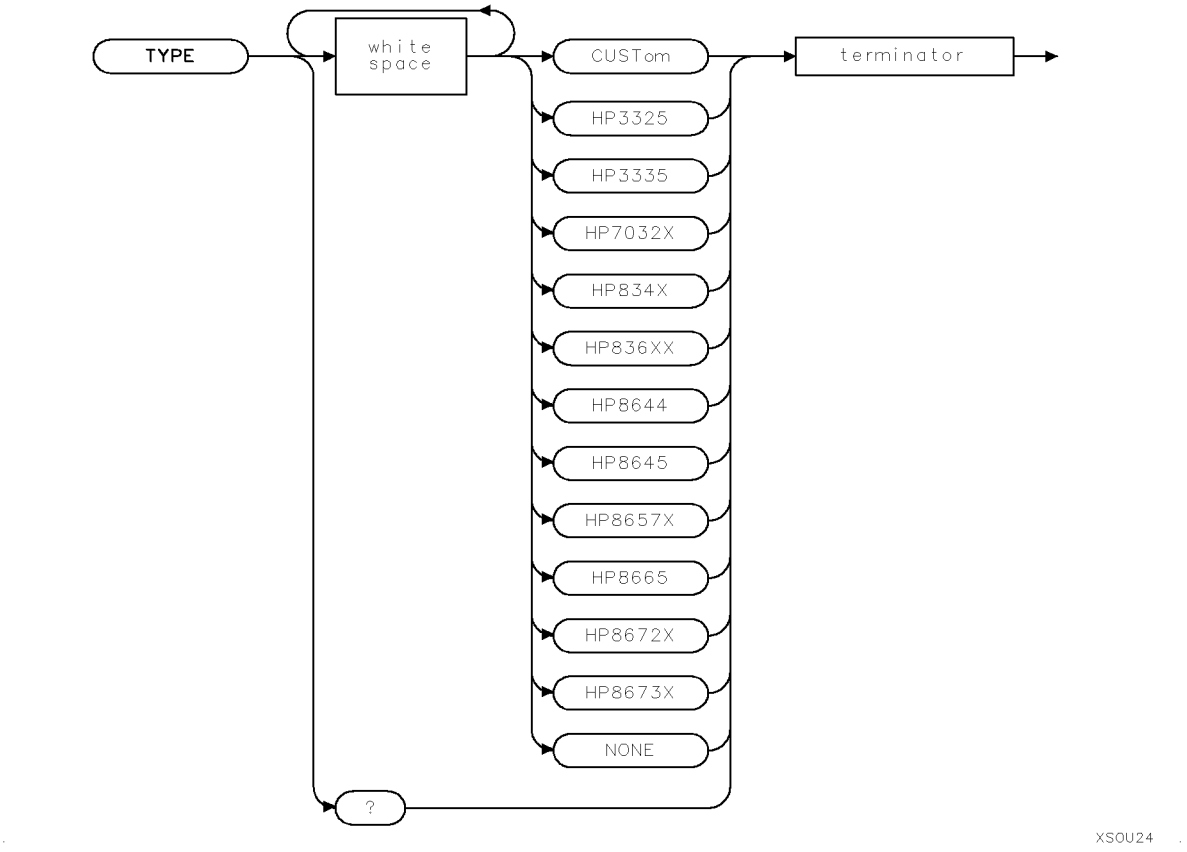
Query Example

```
OUTPUT @Mta;"SOUR:CONF:ROW?"
ENTER @Mta;Row
```

CONFigure:TYPE

Sets the model number of the external synthesizer.

Syntax



Preset State: Not affected by *RST or IP
Power-On State: see CONFigure:SEARch in the CONFIGURE subsystem

Command Example

```
OUTPUT @Mta;"SOUR:CONF:TYPE HP8340;BUS HPIB;ADDR 30;"
```

Description

The CONFIGURE:TYPE command sets the model number of the external synthesizer. The query returns the current setting of TYPE.

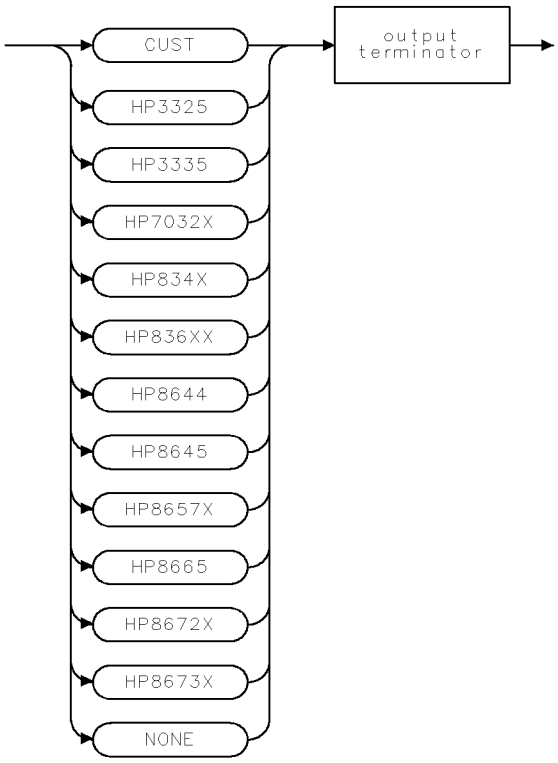
This command works in conjunction with the SOURCE:CONFIGURE:ADDRESS and SOURCE:CONFIGURE:BUS commands to define the synthesizer. After receipt of a program message terminator (generally a line feed), the synthesizer is set to a preset state.

Note

Different synthesizers have various capabilities which can affect the performance and operation of the microwave transition analyzer. Things to watch for:

- Minimum and maximum frequency settability. This can affect the preset value for signal frequency.
- Minimum and maximum amplitude settability.
- Modulation capability (AM, FM, PM)
- Source power on and off capability. If the synthesizer has this capability, the power level will be preset to a relatively high value (−10 dBm). If the synthesizer does not have this capability, the power level will be preset to the minimum value.
- Signal stable capability. If the synthesizer does not have the capability to tell the HP 70820A when the synthesizer output is stable (for both frequency and amplitude), various measurements run "open loop". It is up to the user to enter a dwell time to ensure that the synthesizer output is stable before measurements are made (HP 3325, HP 3335, and HP 7032X fall into this category).

Query Response



QSOU24

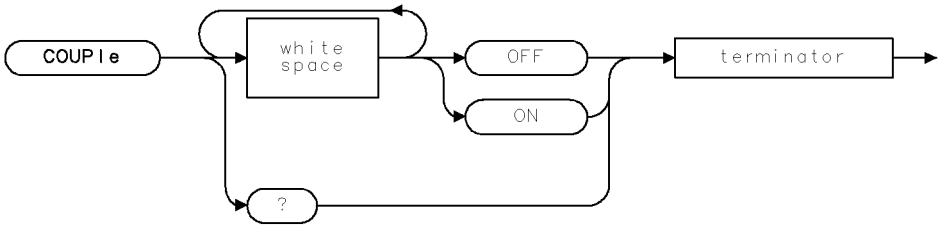
Query Example

```
OUTPUT @Mta;"SOUR:CONF:TYPE?"
ENTER @Mta;Model$
```

COUPle

Couples the external synthesizer frequency and the signal frequency.

Syntax



Preset State: If a valid synthesizer is present: ON;
If a valid synthesizer is not present: OFF

Related Commands: PULse:CARRier
SOURce:CONFigure:TYPE
SWEep:SIGnal

Command Example

```
OUTPUT @Mta;"SOUR:COUP OFF"
```

Description

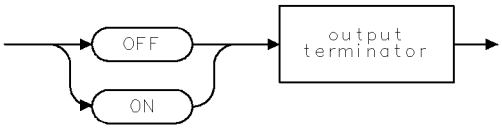
The COUPLE command links the setting of the external synthesizer frequency to either:

- the signal frequency defined by SWEEP:SIGNAL if not an RF pulse measurement, or
- the carrier frequency defined by PULSE:CARRIER if an RF pulse measurement.

If an external synthesizer is not present or if the SOURCE:CONFIGURE:TYPE is set to NONE, trying to set COUPLE to ON will generate an error.

The query returns the current state of COUPLE.

Query Response



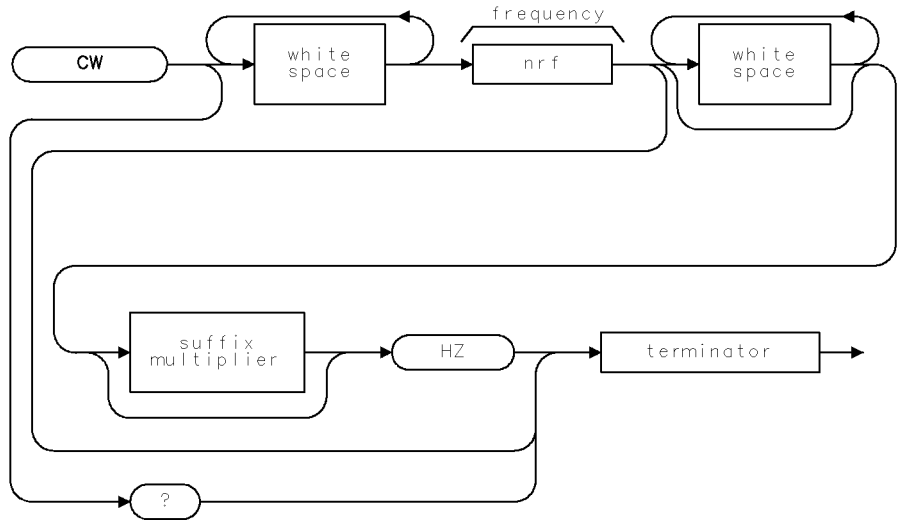
Query Example

```
OUTPUT @Mta;"SOUR:COUP?"
ENTER @Mta;Couple$
```

FREQuency:CW

Sets the frequency of the external synthesizer.

Syntax



XSOU8

Preset State: dependent on external synthesizer
Parameter Range: dependent on external synthesizer
Fundamental Unit: hertz

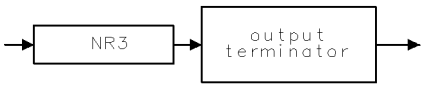
Command Example

```
OUTPUT @Mta;"SOUR:FREQ:CW 2.5 GHz"
```

Description

The FREQUENCY:CW command sets the frequency of the external synthesizer. The query returns the current setting.

Query Response



QGENNR3

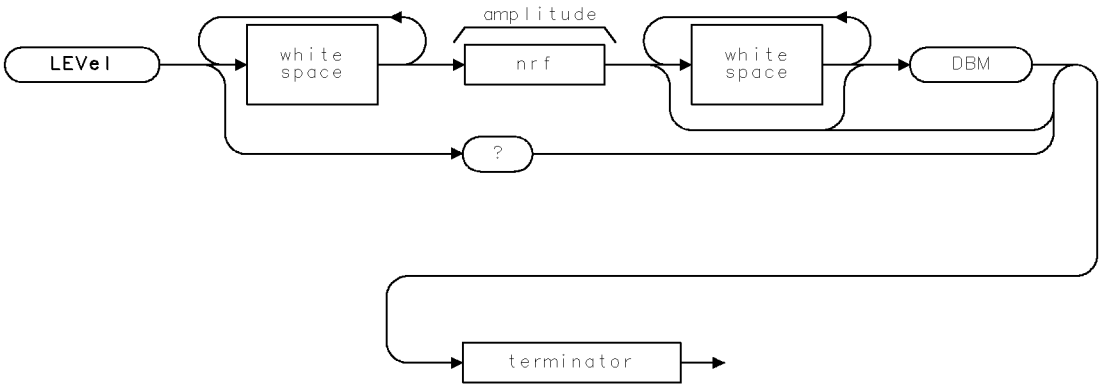
Query Example

```
OUTPUT @Mta;"SOUR:FREQ:CW?"
ENTER @Mta;Frequency
```

POWer:LEVel

Sets the power level of the external synthesizer.

Syntax



XSOU9

Preset State: dependent on external synthesizer
Parameter Range: dependent on external synthesizer
Fundamental Unit: dBm

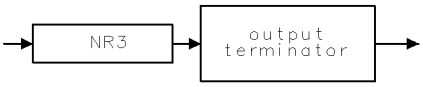
Command Example

```
OUTPUT @Mta;"SOUR:POW:LEV -20 dBm"
```

Description

The POWER:LEVEL command sets the power level of the external synthesizer. The query returns the current setting.

Query Response



QGENNR3

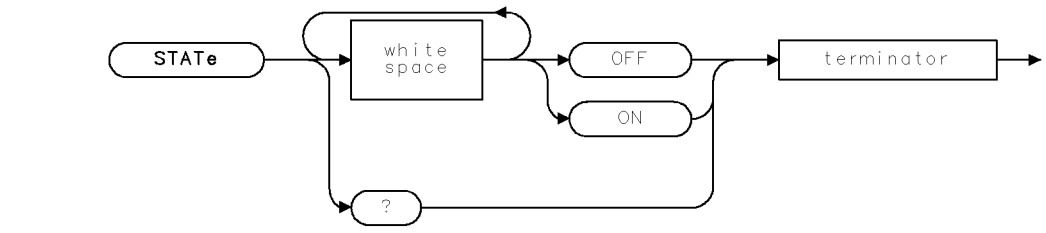
Query Example

```
OUTPUT @Mta;"SOUR:POW:LEV?"
ENTER @Mta;Power_level
```

POWer:STATe

Turns the RF of the external synthesizer on or off.

Syntax



Preset State: OFF (if supported by synthesizer)

Command Example

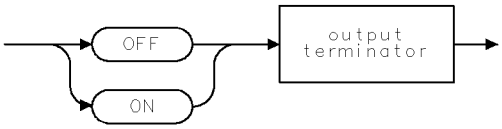
```
OUTPUT @Mta;"SOUR:POW:STAT ON"
```

Description

The POWER:STATE command turns the RF of the external synthesizer on or off. The query returns the current setting.

Note Not all of the synthesizers supported by the HP 70820A have the capability to turn the RF output on and off.

Query Response



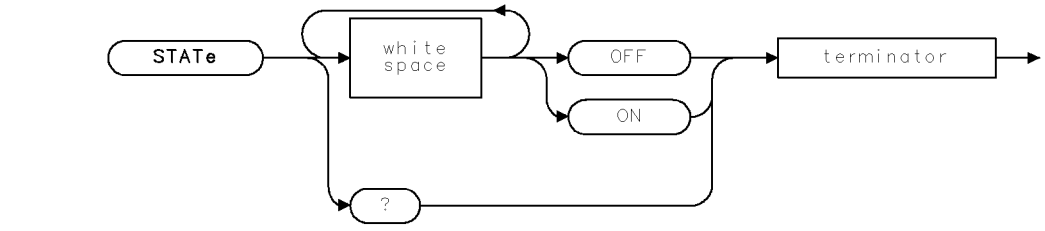
Query Example

```
OUTPUT @Mta;"SOUR:POW:STAT?"
ENTER @Mta;State$
```

PULSe:STATe

Turns the external pulse modulation state of the external synthesizer on or off.

Syntax



Preset State: OFF

Command Example

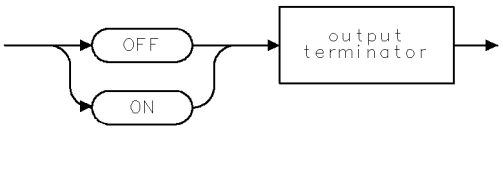
```
OUTPUT @Mta;"SOUR:PULS:STAT ON"
```

Description

The PULSE:STATE command turns the external pulse modulation state of the external synthesizer on or off. The query returns the current setting.

Note Not all of the synthesizers supported by the HP 70820A have pulse modulation capability.

Query Response



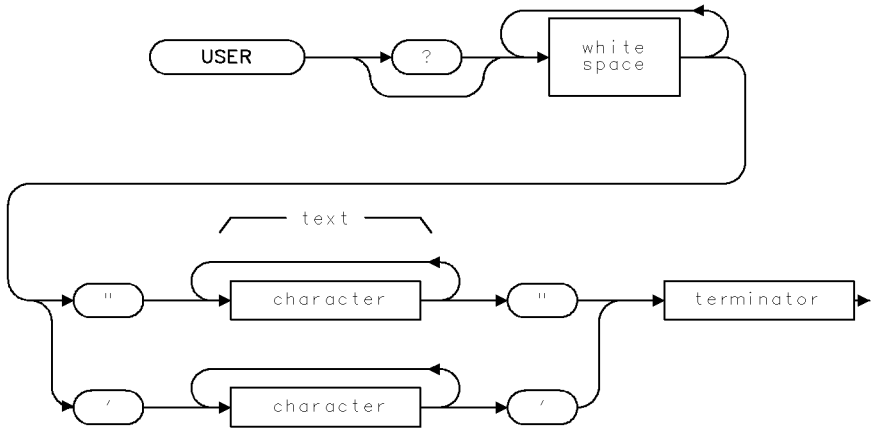
Query Example

```
OUTPUT @Mta;"SOUR:PULS:STAT?"
ENTER @Mta;State$
```

USER

Sends the string parameter to the external synthesizer.

Syntax



Command Example

```
OUTPUT @Mta;"SOUR:USER ':POW:SEAR 0'"
```

Description

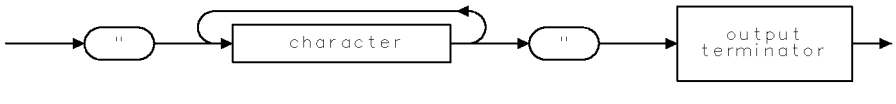
The USER command sends the string parameter to the external synthesizer. This provides the user with access to synthesizer functions not directly supported by the HP 70820A.

The USER query sends the string parameter to the external synthesizer and returns the synthesizer's response. The response of the synthesizer is assumed to be an alphanumeric string terminated by a line feed.

A line feed will be appended to the text sent to the synthesizer for both the command and query forms.

Note The SOURCE:USER command should not be used to alter any parameters that the HP 70820A controls since many of the measurement setup parameters depend on the settings of the synthesizer.

Query Response



Query Example

```
DIM A$[100]
OUTPUT @Mta;"SOUR:USER? '*IDN?'"
ENTER @Mta;Synthesizer_idn$
```

25

Status Subsystem

The STATUS subsystem commands access the non-IEEE488.2 status structures for this instrument. The following additional status structures are supported:

STANDARD OPERATION STATUS REGISTER

Note Many of the "unused" bits are defined and may be implemented in future versions.

- bit 0: calibrating—set at the start of calibration, cleared at the end of calibration
- bit 1: unused (defined as settling)
- bit 2: unused (defined as ranging)
- bit 3: unused (defined as sweeping)
- bit 4: unused (defined as measuring)
- bit 5: waiting for trigger—set when the measurement is waiting for a trigger, cleared when a trigger occurs
- bit 6: unused (defined as waiting for arm)
- bit 7: unused (defined as correcting)
- bit 8: unused (defined as instrument-specific)
- bit 9: unused (defined as instrument-specific)
- bit 10: self test in progress (instrument-specific)—set at the start of self-test, cleared at the end of self-test
- bit 11: ready (instrument-specific)—set by the :READy command, cleared by a program message terminator
- bit 12: unused (defined as instrument-specific)
- bit 13: unused (reserved)
- bit 14: unused (defined as user-defined program running)
- bit 15: zero

These registers consist of a CONDITION register which reflects the current state of the module and of an EVENT register which reflects changes in the CONDITION register dependent on a transition filter.

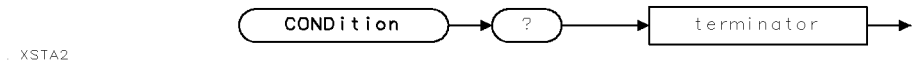
See the following list for a command syntax summary of the STATUS subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:STATus		
:OPERation?		
:CONDition?		[query only]
:ENABle	<nrf>	
[:EVENt]?		[query only]
:NTRansition	<nrf>	
:PTRansition	<nrf>	
:PRESet		[no query]

OPERation:CONDition

Returns the current status of the STANDARD OPERATION STATUS CONDITION register.

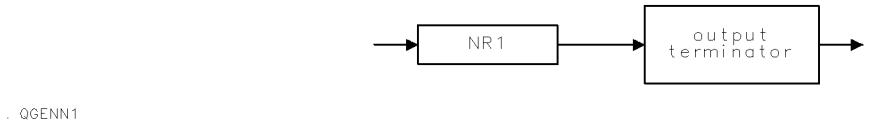
Syntax



Description

The CONDITION query returns the current status of the STANDARD OPERATION STATUS CONDITION register.

Query Response



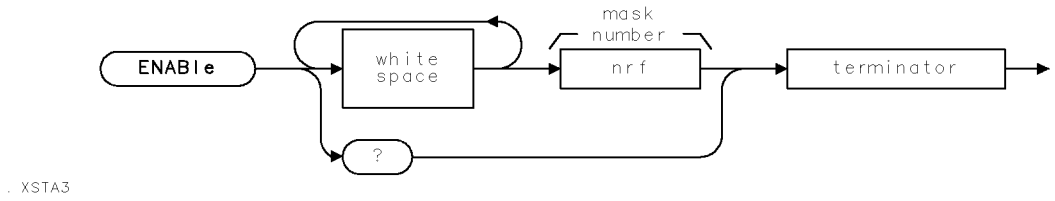
Query Example

```
OUTPUT @Mta;"STAT:OPER:COND?"
ENTER @Mta;Condition
```

OPERation:ENABle

Sets the enable mask for the STANDARD OPERATION STATUS EVENT register.

Syntax



Preset State: 0
Parameter Range: mask number = 0 through 32767
Related Commands: *SRE
*STB?

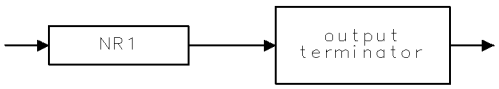
Command Example

```
OUTPUT @Mta;"STAT:OPER:ENAB 1024"
```

Description

The ENABLE command sets which bits of the STANDARD OPERATION STATUS EVENT register will cause bit 7 to be set in the Status Byte Register (see *SRE and *STB in the Common Command set). The query returns the current state of ENABLE.

Query Response



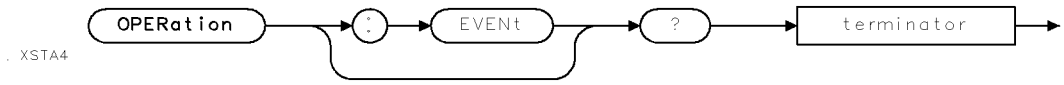
Query Example

```
OUTPUT @Mta;"STAT:OPER:ENAB?"
ENTER @Mta;Enable_mask
```

OPERation:EVENt

Returns the current status of the STANDARD OPERATION STATUS EVENT register.

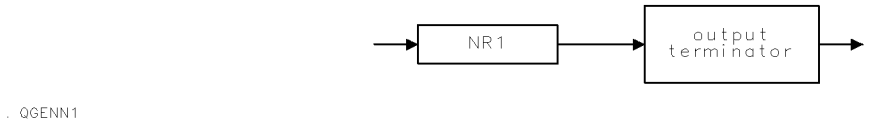
Syntax



Description

The EVENT query returns the current status of the STANDARD OPERATION STATUS EVENT register. This is a destructive read command query.

Query Response



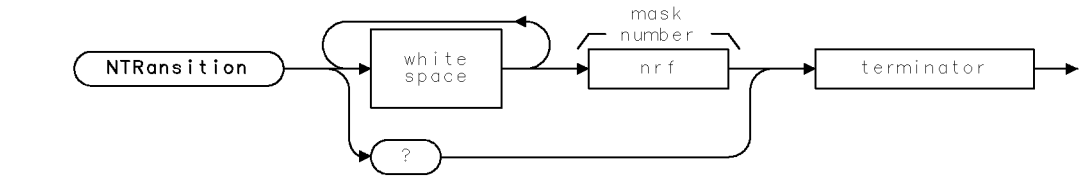
Query Example

```
OUTPUT @Mta;"STAT:OPER:EVEN?"
ENTER @Mta;Condition
```

OPERation:NTRansition (Negative Transition)

Sets the negative transition filter for the STANDARD OPERATION STATUS CONDITION register.

Syntax



Preset State: 0
Parameter Range: mask number = 0 through 32767

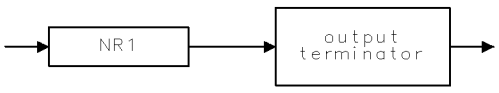
Command Example

```
OUTPUT @Mta;"STAT:OPER:NTR 1024"          ! Self-test bit
OUTPUT @Mta;"STAT:OPER:PTR 0"
OUTPUT @Mta;"STAT:OPER:ENAB 1024"         ! Self-test bit
OUTPUT @Mta;"*SRE 128"
OUTPUT @Mta;"*TST?"
!
! The module will assert SRQ when the self-test is
! finished.
ENTER @Mta;Pass$
```

Description

The NTRANSITION command defines which bits of the STANDARD OPERATION STATUS CONDITION register will set a corresponding bit of the STANDARD OPERATION STATUS EVENT register with a negative transition. The query returns the current state of NTRANSITION.

Query Response



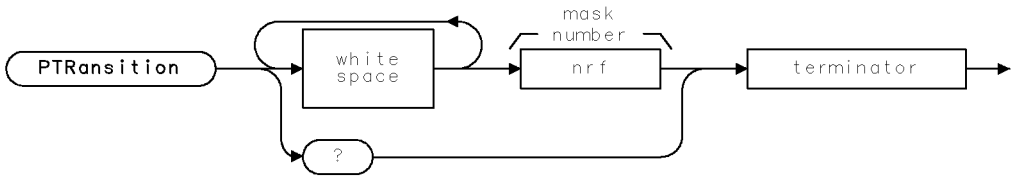
Query Example

```
OUTPUT @Mta;"STAT:OPER:NTR?"
ENTER @Mta;Negative_mask
```

OPERation:PTRansition (Positive Transition)

Sets the positive transition filter for the STANDARD OPERATION STATUS CONDITION register.

Syntax



Preset State: 32767
Parameter Range: mask number = 0 through 32767

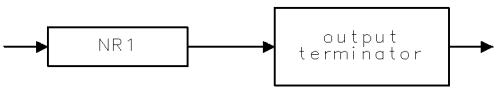
Command Example

```
OUTPUT @Mta;"STAT:OPER:NTR 0"
OUTPUT @Mta;"STAT:OPER:PTR 32"        ! Wait for trigger bit
OUTPUT @Mta;"STAT:OPER:ENAB 32"       ! Wait for trigger bit
OUTPUT @Mta;"*SRE 128"
OUTPUT @Mta;"RUN"
!
! The module will assert SRQ when the measurement setup
! is complete and the module is waiting for a trigger
!
```

Description

The PTRANSITION command defines which bits of the STANDARD OPERATION STATUS CONDITION register will set a corresponding bit of the STANDARD OPERATION STATUS EVENT register with a positive transition. The query returns the current state of PTRANSITION.

Query Response



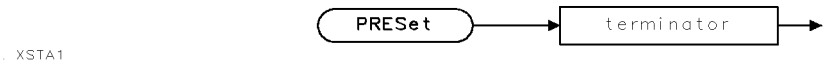
Query Example

```
OUTPUT @Mta;"STAT:OPER:PTR?"
ENTER @Mta;Positive_mask
```

PRESet

Presets the STATUS subsystem.

Syntax



Command Example

```
OUTPUT @Mta;"STAT:PRES"
```

Description

The PRESET command presets the STATUS subsystem. The effect is to set all PTRANSITION filter bits to 1, all NTRANSITION filter bits to 0, and all ENABLE bits to 0.

26

Sweep Subsystem

The SWEEP subsystem commands control the microwave transition analyzer horizontal axis functions. See the following list for a command syntax summary of the SWEEP subsystem commands.

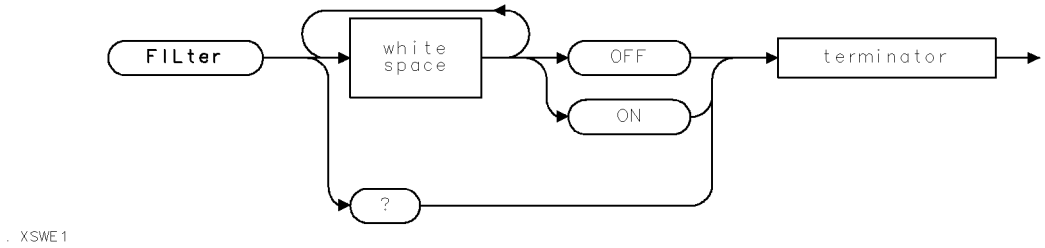
KEYWORD	PARAMETER FORM	NOTES
:SWEep		
:FILter	OFF|ON	
:FREQuency		
:DIVisor	<nrf>	
:DWELl	<nrf>[[<suffix multiplier>]S]	
:IFBandwidth	<nrf>[[<suffix multiplier>]HZ]	
:MULTiplier	<nrf>	
:OFFSet	<nrf>[[<suffix multiplier>]HZ]	
:STARt	<nrf>[[<suffix multiplier>]HZ]	
:STOP	<nrf>[[<suffix multiplier>]HZ]	
:FUZZy	FREQuency|LENGth|OFF|TIME	
:MODE	ASIN|AUTo|SING|TRIG	
:OPTimize	OFF|ON	
:POWer		
:DWELl	<nrf>[[<suffix multiplier>]S]	
:IFBandwidth	<nrf>[[<suffix multiplier>]HZ]	
:STARt	<nrf>[DBM]	
:STOP	<nrf>[DBM]	
:REFerence	INTernal|EXTernal	
:SIGnal	<nrf>[[<suffix multiplier>]HZ]	
:SINgle	OFF|ON	
:SLISt	<nrf>[[<suffix multiplier>]HZ]	
	[,<nrf>[[<suffix multiplier>]HZ]	
	[,<nrf>[[<suffix multiplier>]HZ]	
	[,<nrf>[[<suffix multiplier>]HZ]	
	[,<nrf>[[<suffix multiplier>]HZ]]]]]	
:STRack		
:COUNt	<nrf>	
[:STATe]	OFF|ON	
:VERify	OFF|ON	
:TIME		
:ANALytic	OFF|ON	
:CYCLes	OFF|ON	
:DELay	<nrf>[CYCLes|[<suffix multiplier>]S]	
:FILter		
:BWIDth	<nrf>[[<suffix multiplier>]HZ]	
[:STATe]	OFF|ON	
:VIDeo	(CHAN1|CHAN2,ON|OFF)|?(CHAN1|CHAN2)	

:OVERsweep		
:DECimate	<nrf>[[<suffix multiplier>]S]	
:DETect	AM|FILTer|FM|MAXimum|MINimum|PM|SAMPle	
:FILTer		
:APERture	<nrf>	
:CHANx		
:CENTer	<nrf>[[<suffix multiplier>]HZ]	
:FORMat	FREQuency|MAGNitude|PHASe|SAMPle	
:TRACk	AUTo|CHAN1|CHAN2|MANual	
[:STATe]	OFF|ON	
:OFFSet	<nrf>[[<suffix multiplier>]S]	
:POINts	<nrf>[[<suffix multiplier>]S]	
[:STATe]	OFF|ON	
:RANGe	<nrf>[CYCLes|[<suffix multiplier>]S]	
:SRATe	<nrf>[[<suffix multiplier>]HZ]	
:TARGet		
:SRATe	<nrf>[[<suffix multiplier>]HZ]	
[:STATe]	AUTO|MANual	
:TRANslate	AUTO|OFF|ON	
:ZOOM		
:CENTer	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:REZoom		[no query]
:SPAN	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:STARt	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
[:STATe]	OFF|ON	
:STOP	<nrf>[CYCLes|[<suffix multiplier>]HZ]	
:TYPE	FREQuency|POWer|TIME	

FILter

Sets the state of an IF filtering process.

Syntax



Command Example

```
OUTPUT @Mta;"SWE:FIL ON"
```

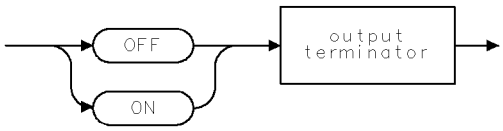
Description

The FILTER command sets the state of an IF filtering process. The query returns the current state of FILTER.

The FILTER command does not correspond directly to the state of a hardware filter but is interpreted based on the state of the instrument as follows:

- **TIME SWEEP:**
  - SINGLE SHOT: corresponds to hardware filter
  - REPETITIVE:
    - Modulation source off: corresponds to hardware filter
    - Modulation source on: corresponds to a software filter
- **FREQUENCY SWEEP**
  - SINGLE xxx: normal stepped frequency sweep
- **POWER SWEEP:** no effect; the hardware filter is automatically used if the signal position in the IF and the setting of resolution bandwidth warrants it.

Query Response



QGENON

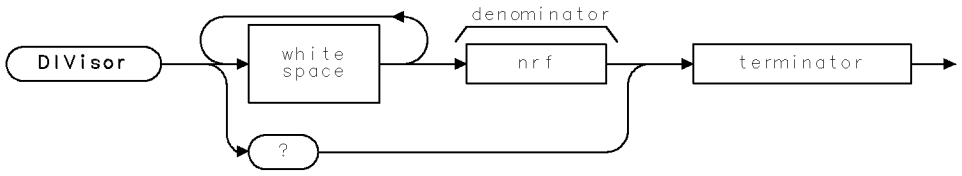
Query Example

```
OUTPUT @Mta;"SWE:FIL?"
ENTER @Mta;Filter$
```

FREQuency:DIVisor

Specifies the denominator of the frequency multiplier.

Syntax



Preset State: 1
Parameter Range: denominator = 1 through 1024

Command Example

```
OUTPUT @Mta;"SWE:FREQ:DIV 3"
```

Description

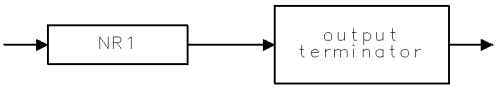
The FREQUENCY:DIVISOR command specifies the denominator of the multiplier in the sweep equation:

$$RF = (numerator/denominator) \times FREQ + offset.$$

The sweep equation is used for stepped frequency and power sweeps (for example, SWEEP:TYPE is FREQUENCY or POWER). RF is the frequency expected at the input, *FREQ* is the synthesizer frequency, numerator is specified by SWEEP:FREQUENCY:MULTIPLIER, denominator is specified by SWEEP:FREQUENCY:DIVISOR, and offset is specified by SWEEP:FREQUENCY:OFFSET.

The query returns the current value of denominator.

Query Response



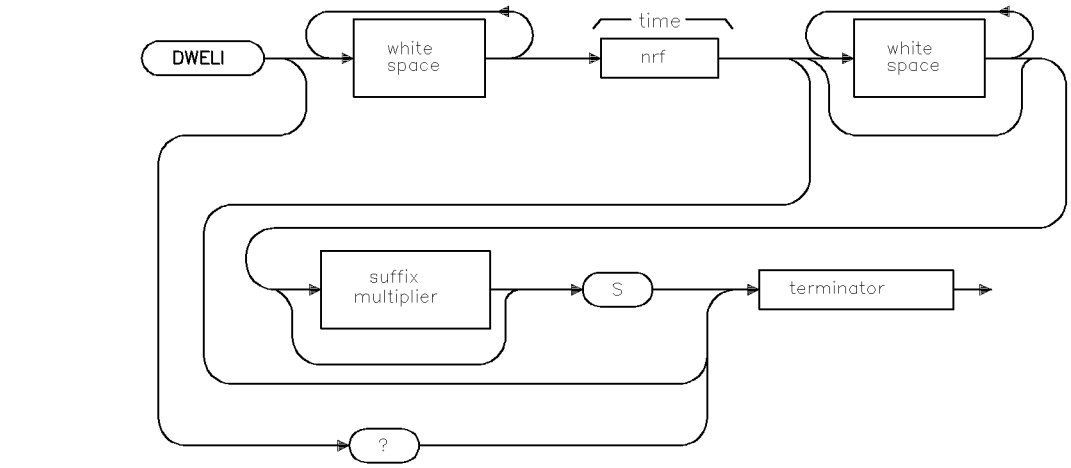
Query Example

```
OUTPUT @Mta;"SWE:FREQ:DIV?"
ENTER @Mta;Denominator
```

FREQuency:DWELl

Sets the dwell time.

Syntax



Preset State: 100 ms
Parameter Range: time = 0 through 1 second (with 10 ms resolution)
Fundamental Unit: seconds

Command Example

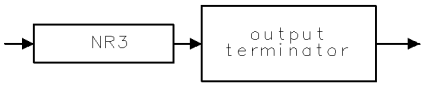
```
OUTPUT @Mta;"SWE:FREQ:DWEL 100 ms"
```

Description

The FREQUENCY:DWELL command specifies the minimum amount of time between programming the synthesizer and the start of a measurement. The query returns the current setting.

The command and query can also be accessed via SWEEP:POWER:DWELL.

Query Response



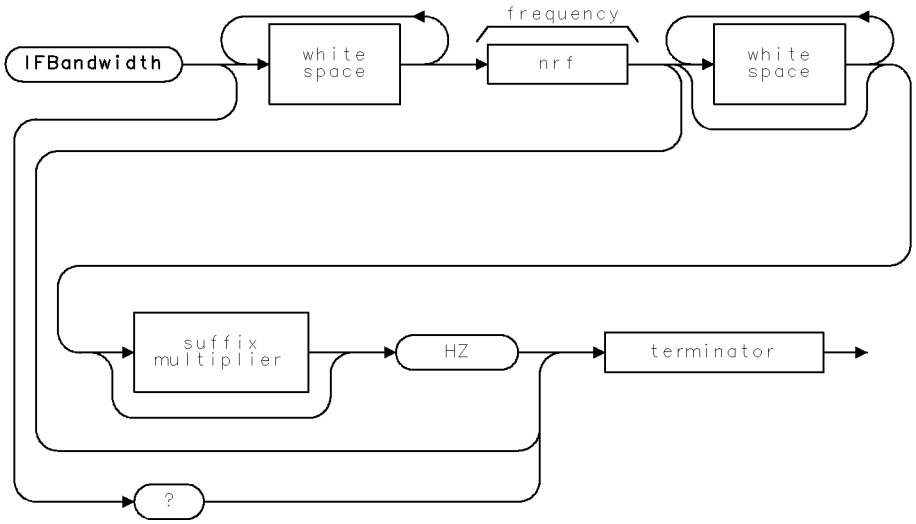
Query Example

```
OUTPUT @Mta;"SWE:FREQ:DWEL?"
ENTER @Mta;Dwell_time
```

FREQuency:IFBandwidth

Specifies the approximate resolution bandwidth used for frequency and power sweeps.

Syntax



Preset State: 1 kHz
Parameter Range: frequency = 10 Hz through 50 kHz
Fundamental Unit: hertz

Command Example

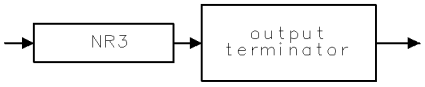
```
OUTPUT @Mta;"SWE:FREQ:IFB 1 kHz"
```

Description

The FREQUENCY:IFBANDWIDTH command specifies the approximate resolution bandwidth used for frequency and power sweeps. The query returns the current setting.

The command and query can also be accessed via SWEEP:POWER:IFBANDWIDTH.

Query Response



QGENNR3

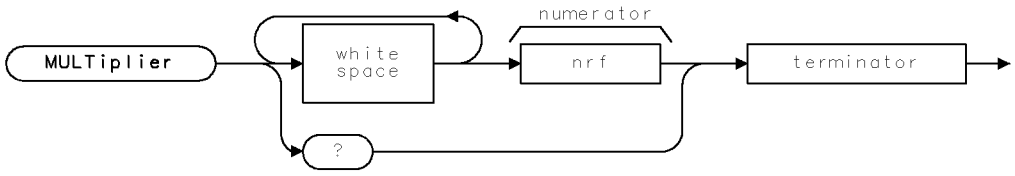
Query Example

```
OUTPUT @Mta;"SWE:FREQ:IFB?"
ENTER @Mta;If_bandwidth
```

FREQuency:MULTiplier

Specifies the numerator of the frequency multiplier.

Syntax



Preset State: 1
Parameter Range: numerator = −1024 through +1024

Command Example

```
OUTPUT @Mta;"SWE:FREQ:MULT 3"
```

Description

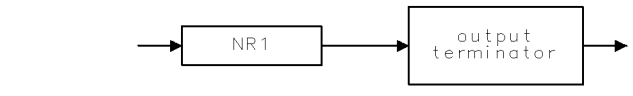
The FREQUENCY:MULTIPLIER command specifies the numerator of the multiplier in the sweep equation:

$$RF = (numerator/denominator) \times FREQ + offset.$$

The sweep equation is used for stepped frequency and power sweeps (for example, SWEEP:TYPE is FREQUENCY or POWER). RF is the frequency expected at the input, *FREQ* is the synthesizer frequency, numerator is specified by SWEEP:FREQUENCY:MULTIPLIER, denominator is specified by SWEEP:FREQUENCY:MULTIPLIER, and offset is specified by SWEEP:FREQUENCY:OFFSET.

The query returns the current value of numerator.

Query Response



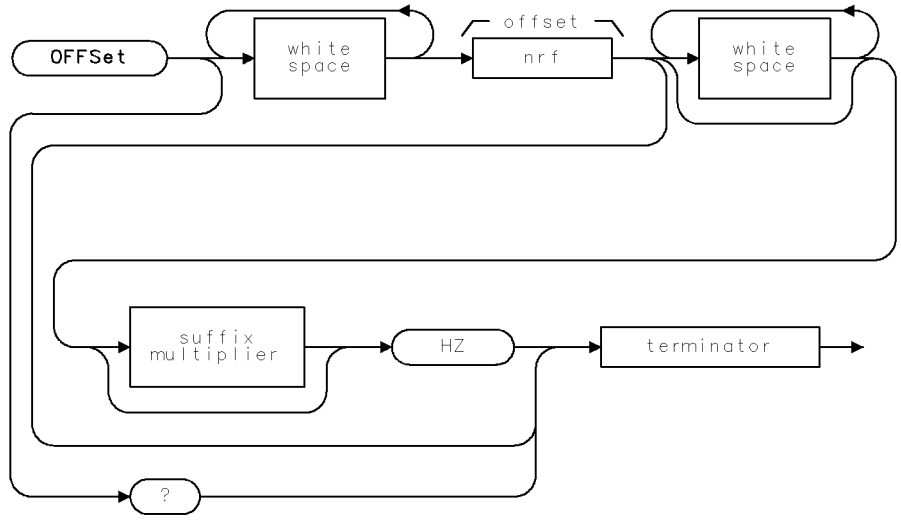
Query Example

```
OUTPUT @Mta;"SWE:FREQ:MULT?"
ENTER @Mta;Numerator
```

FREQuency:OFFSet

Specifies the frequency offset.

Syntax



XSWE14

Preset State: 0
Parameter Range: frequency = −100 GHz through +100 GHz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"SWE:FREQ:OFFS 1 MHz"
```

Description

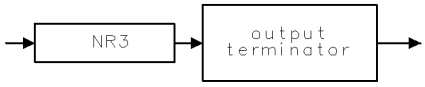
The FREQUENCY:OFFSET command specifies the frequency offset in the sweep equation:

$$RF = (numerator/denominator) \times FREQ + offset.$$

The sweep equation is used for stepped frequency and power sweeps (for example, SWEEP:TYPE is FREQUENCY or POWER). RF is the frequency expected at the input, *FREQ* is the synthesizer frequency, numerator is specified by SWEEP:FREQUENCY:MULTIPLIER, denominator is specified by SWEEP:FREQUENCY:MULTIPLIER, and offset is specified by SWEEP:FREQUENCY:OFFSET.

The query returns the current value of offset.

Query Response



QGENNR3

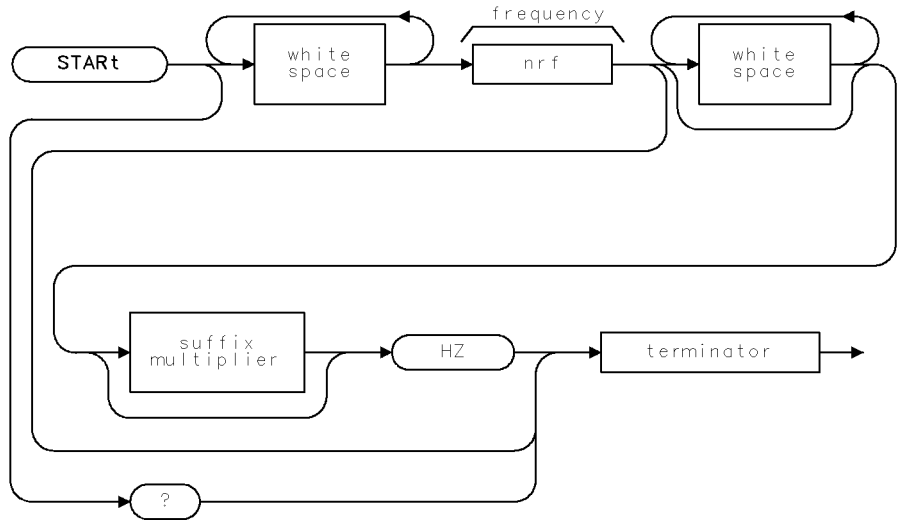
Query Example

```
OUTPUT @Mta;"SWE:FREQ:OFFS?"
ENTER @Mta;Freq_offset
```

FREQuency:STARt

Specifies the start frequency used for frequency sweeps.

Syntax



XSWE15

Preset State: dependent on synthesizer configured (if any)
Parameter Range: dependent on synthesizer configured (if any)
Fundamental Unit: hertz

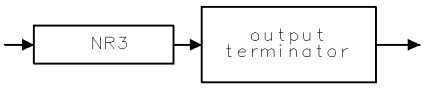
Command Example

```
OUTPUT @Mta;"SWE:FREQ:STAR 100 MHz"
```

Description

The FREQUENCY:START command specifies the start frequency used for frequency sweeps. The query returns the current setting.

Query Response



QGENNR3

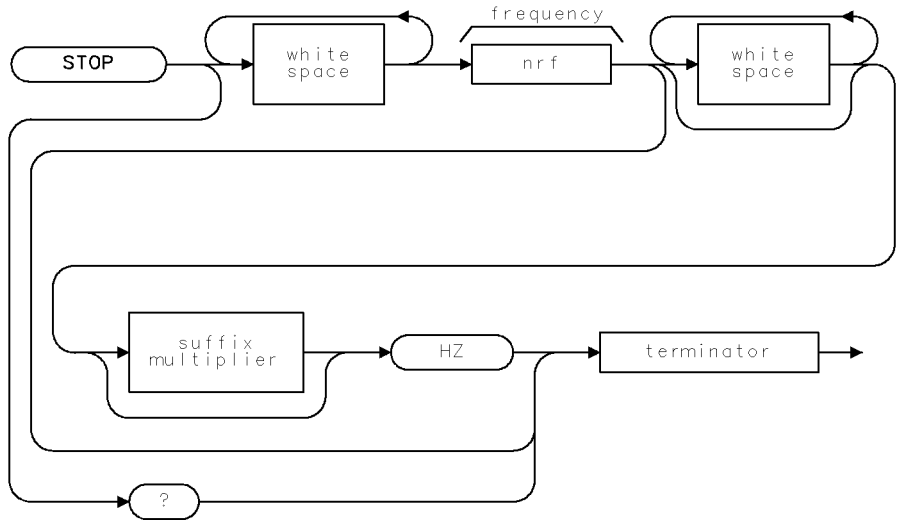
Query Example

```
OUTPUT @Mta;"SWE:FREQ:STAR?"
ENTER @Mta;Start_freq
```

FREQuency:STOP

Specifies the stop frequency used for frequency sweeps.

Syntax



XSWE16

Preset State: dependent on synthesizer configured (if any)
Parameter Range: dependent on synthesizer configured (if any)
Fundamental Unit: hertz

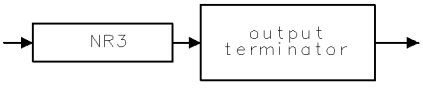
Command Example

```
OUTPUT @Mta;"SWE:FREQ:STOP 100 MHz"
```

Description

The FREQUENCY:STOP command specifies the stop frequency used for frequency sweeps. The query returns the current setting.

Query Response



QGENNR3

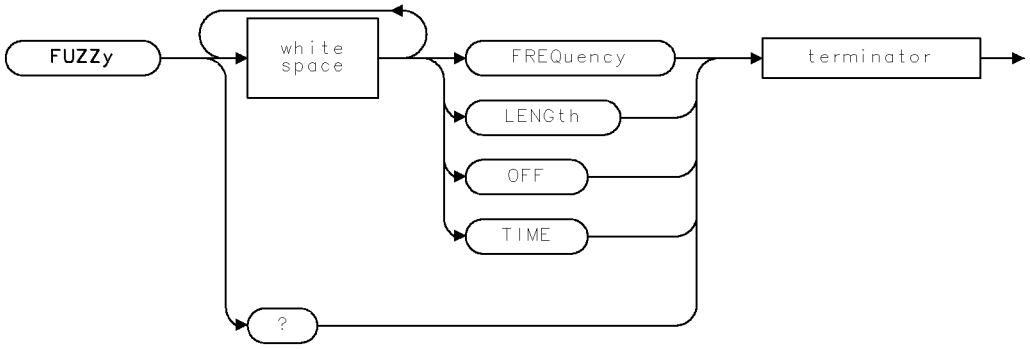
Query Example

```
OUTPUT @Mta;"SWE:FREQ:STOP?"
ENTER @Mta;Stop_freq
```

FUZZy

Sets optimization mode for pulsed RF measurements.

Syntax



Preset State: LENGth

Command Example

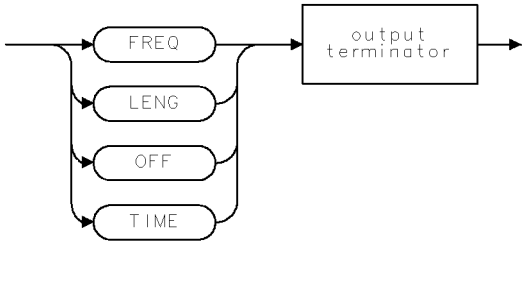
```
OUTPUT @Mta;"SWE:FUZZ TIME"
```

Description

The FUZZY command determines what, if any, setup parameters may be adjusted to optimize pulsed RF measurements. The query returns the current state of FUZZY.

- **OFF:** no adjustments are allowed
- **FREQuency:** the carrier frequency may be adjusted (this option is available only if the module controls a synthesizer)
- **LENGth:** the trace length (ACQUIRE:POINTS) may be adjusted
- **TIME:** the time range may be adjusted

Query Response



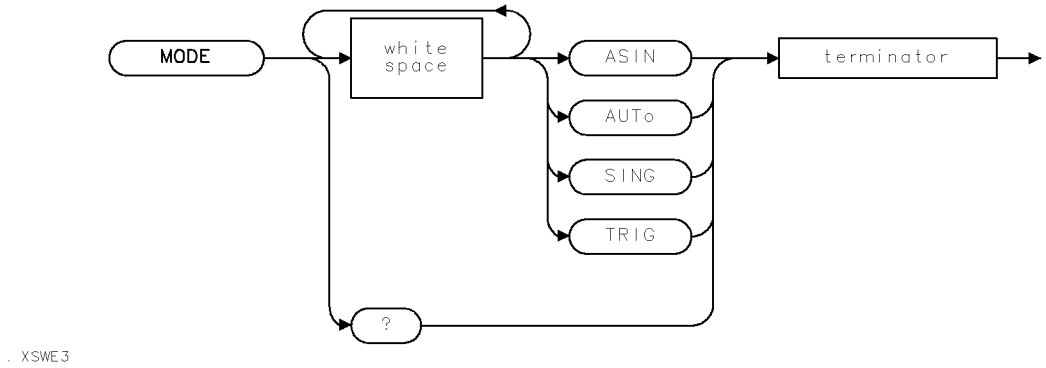
Query Example

```
OUTPUT @Mta;"SWE:FUZZ?"
ENTER @Mta;Fuzzy$
```

MODE

Specifies the sweep triggering.

Syntax



Preset State: AUTO
Related Commands: DIGitize
RUN
STOP
TRIGger:xxx

Command Example

```
OUTPUT @Mta;"SWE:MODE TRIG"
```

Description

The MODE command specifies the sweep operating mode. The various options are:

- **ASIN:** auto-triggered, single sweep
- **AUTO:** auto-triggered, continuous sweep
- **SING:** triggered single sweep
- **TRIG:** triggered continuous sweep

Triggered sweeps will wait until a trigger occurs before making a measurement (analogous to “normal” trigger in conventional oscilloscopes). Auto-triggered sweeps will wait a certain amount of time for a trigger. If the trigger occurs within this amount of time, the measurement begins the same as the triggered sweeps.

If the trigger has not occurred within this amount of time, the measurement will begin without the trigger (analogous to ’auto’ trigger in conventional oscilloscopes).

The TRIGGER subsystem defines what constitutes a trigger.

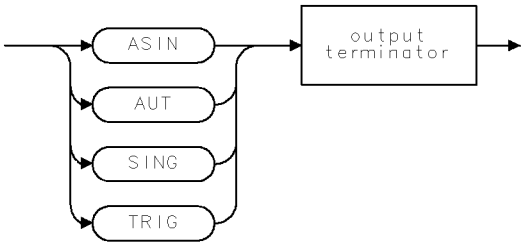
The MODE command will start a measurement (that is, sending SWEEP:MODE:xxx will start a measurement, even for ASIN and SING).

To avoid starting a measurement, send `SWEEP:MODE:xxx; :STOP;`.

To start the measurement after sending STOP, send `RUN`.

The query returns the current state of MODE.

Query Response



QSWE3

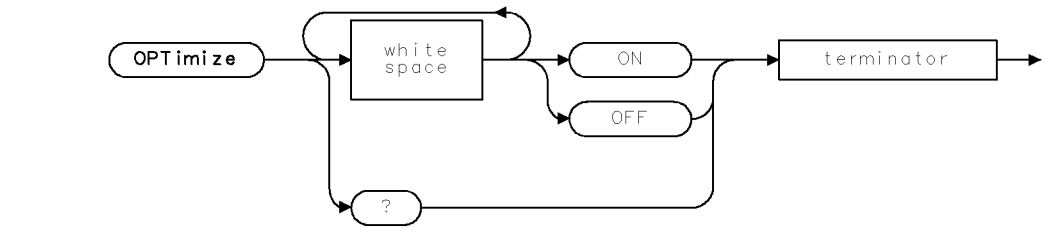
Query Example

```
OUTPUT @Mta;"SWE:MODE?"
ENTER @Mta;Mode$
```

OPTimize

Sets frequency sweep optimization for pulsed RF measurements.

Syntax



Preset State: ON

Command Example

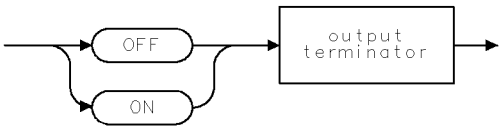
```
OUTPUT @Mta;"SWE:OPT ON"
```

Description

The OPTIMIZE command changes the frequency span in pulsed frequency sweeps so that the frequency per point will cause measurements to be taken at optimal points in the IF. Turning OPTIMIZE OFF restores the original frequency span. In non-pulsed frequency sweeps, the OPTIMIZE command has no effect other than using the optimized span if optimizing had been active during a pulsed sweep. If no pulsed sweep optimization had been done, the span remains the same whether OPTIMIZE is on or off.

The query returns the current state of the OPTIMIZE flag.

Query Response



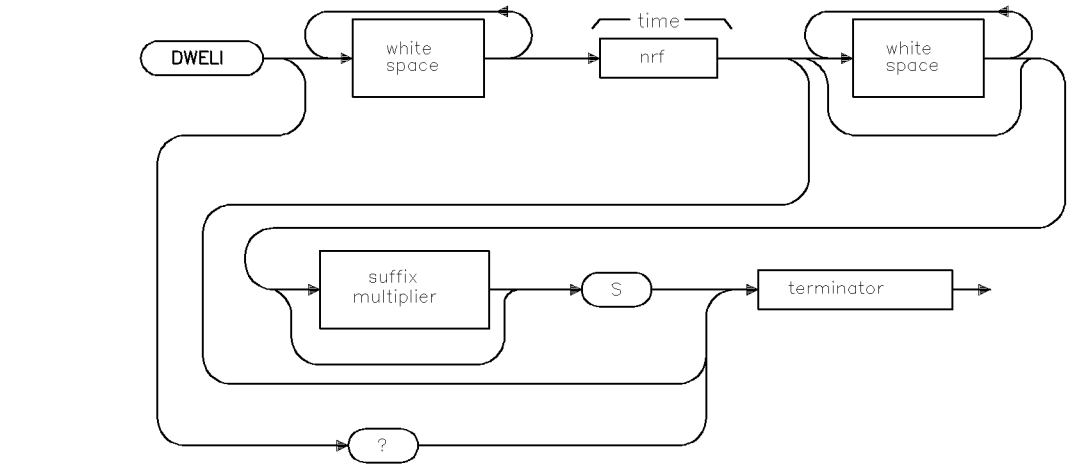
Query Example

```
OUTPUT @Mta;"SWE:OPT?"
ENTER @Mta;Optimize$
```

POWer:DWELl

Sets the dwell time.

Syntax



Preset State: 100 ms
Parameter Range: time = 0 through 1 second (with 10 ms resolution)
Fundamental Unit: seconds

Command Example

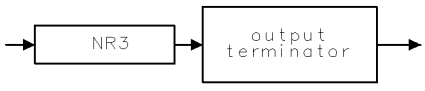
```
OUTPUT @Mta;"SWE:POW:DWEL 100 ms"
```

Description

The POWer:DWELL command specifies the minimum amount of time between programming the synthesizer and the start of a measurement. The query returns the current setting.

The command and query can also be accessed via SWEEP:FREQUENCY:DWELL.

Query Response



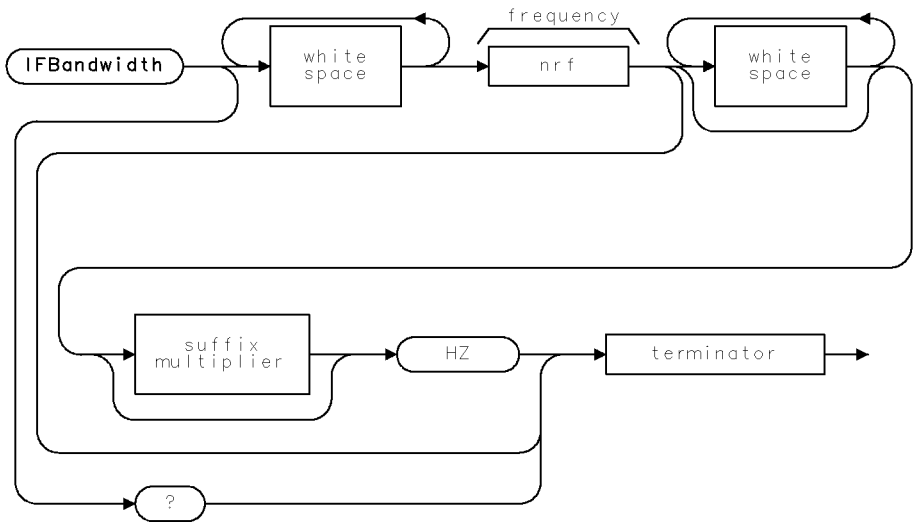
Query Example

```
OUTPUT @Mta;"SWE:POW:DWEL?"
ENTER @Mta;Dwell_time
```

POWer:IFBandwidth

Specifies the approximate resolution bandwidth used for frequency and power sweeps.

Syntax



XSWE12

Preset State: 1 kHz
Parameter Range: frequency = 10 Hz through 50 kHz
Fundamental Unit: hertz

Command Example

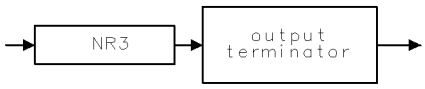
```
OUTPUT @Mta;"SWE:POW:IFB 1 kHz"
```

Description

The POWer:IFBANDWIDTH command specifies the approximate resolution bandwidth used for frequency and power sweeps. The query returns the current setting.

The command and query can also be accessed via SWEEP:FREQuency:IFBANDWIDTH.

Query Response



QGENNR3

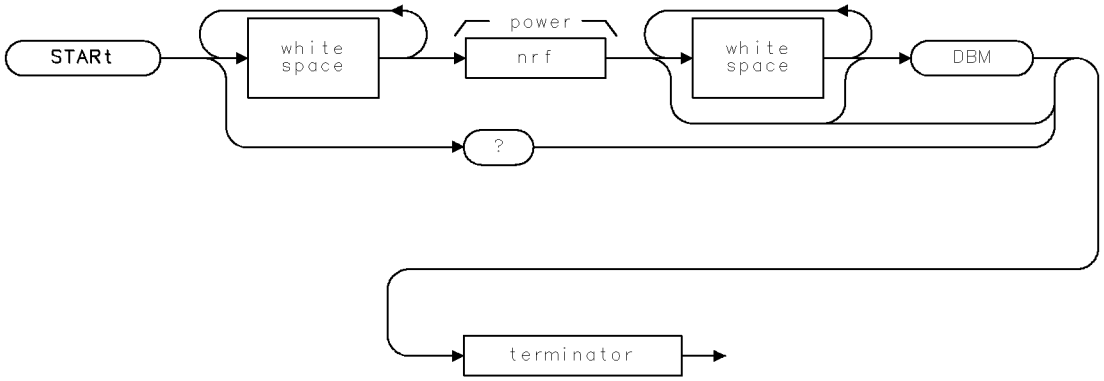
Query Example

```
OUTPUT @Mta;"SWE:POW:IFB?"
ENTER @Mta;If_bandwidth
```

POWer:STARt

Specifies the start power used for power sweeps.

Syntax



XSWE17

Preset State: −60 dBm
Parameter Range: depends on selected synthesizer
Fundamental Unit: dBm

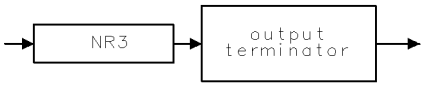
Command Example

```
OUTPUT @Mta;"SWE:POW:STAR -50 dBm"
```

Description

The POWER:START command specifies the start power used for power sweeps. The query returns the current setting.

Query Response



QGENNR3

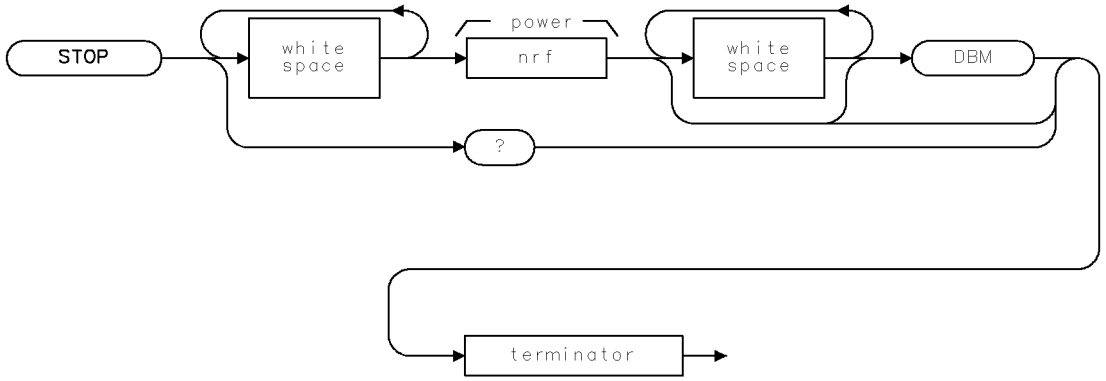
Query Example

```
OUTPUT @Mta;"SWE:POW:STAR?"
ENTER @Mta;Start_power
```

POWer:STOP

Specifies the stop power used for power sweeps.

Syntax



XSWE18

Preset State: −20 dBm
Parameter Range: depends on selected synthesizer
Fundamental Unit: dBm

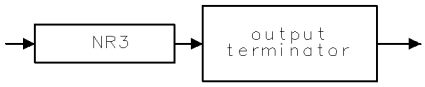
Command Example

```
OUTPUT @Mta;"SWE:POW:STOP -10 dBm"
```

Description

The POWER:STOP command specifies the stop power used for power sweeps. The query returns the current setting.

Query Response



QGENNR3

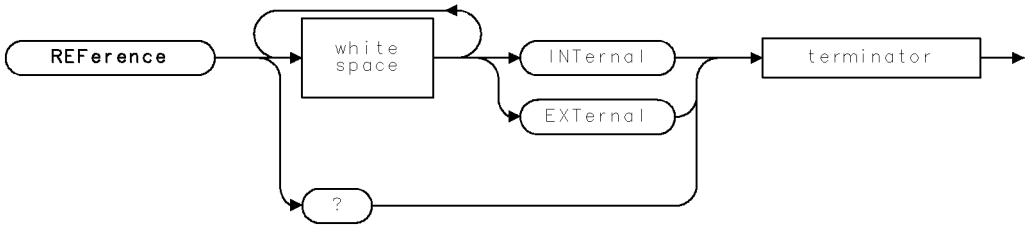
Query Example

```
OUTPUT @Mta;"SWE:POW:STOP?"
ENTER @Mta;Stop_power
```

REFerence

Selects the 10 MHz reference.

Syntax



Command Example

```
OUTPUT @Mta;"SWE:REF INT"
```

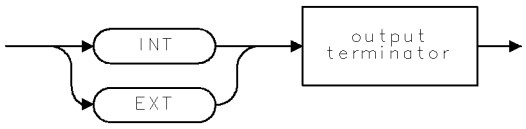
Description

The REFERENCE command specifies whether the microwave transition analyzer is to use its internal 10 MHz reference or the external 10 MHz reference.

The query returns the current state of REFERENCE.

If the specified reference is different from the current reference, this command will abort any measurement that is in progress, then switch to the specified reference. The command will always check for an unlock condition and, if the system is unlocked, report it as an error.

Query Response



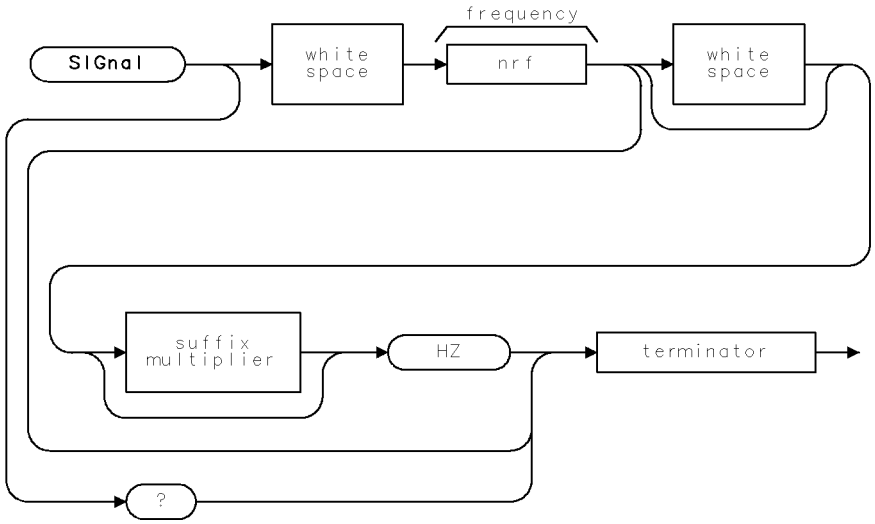
Query Example

```
OUTPUT @Mta;"SWE:REF?"
ENTER @Mta;Reference$
```

SIGnal

Defines the frequency of the active signal.

Syntax



Preset State:	one signal at 53.1787322 MHz if there is no synthesizer present; otherwise, the signal is dependent upon the synthesizer configuration.
Parameter Range:	frequency = 0.1 Hz to 100 GHz
Fundamental Unit:	hertz
Related Commands:	AUTo SWEep:SLISt

Command Example

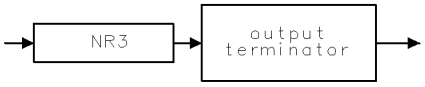
```
OUTPUT @Mta;"SWE:SIG 100 MHz"
```

Description

The SIGNAL command is used to define the frequency of the "active" signal (that is, any horizontal setup that requires the signal frequency will use this value).

The query returns the currently "active" signal.

Query Response



QGENNR3

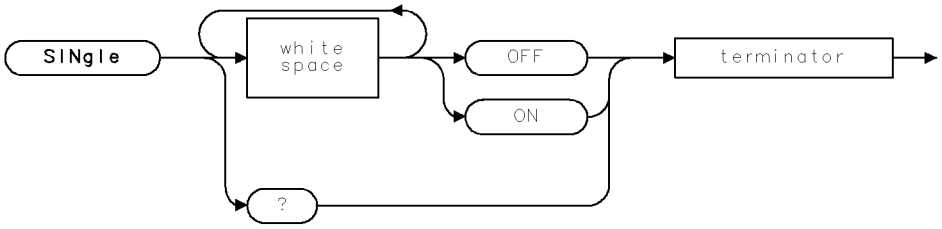
Query Example

```
OUTPUT @Mta;"SWE:SIG?"
ENTER @Mta;Freq
```

SINgle

Sets the state of the single shot mode.

Syntax



Preset State: OFF

Command Example

```
OUTPUT @Mta;"SWE:SIN ON"
```

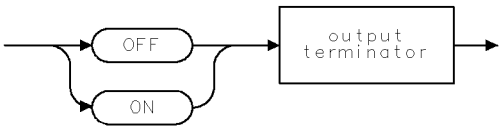
Description

The SINGLE command sets the state of the single shot mode. The query returns the current state of SINGLE.

The single shot mode acts as a modifier of the sweep type as follows:

- **TIME SWEEP**
  - SINGLE OFF: normal repetitive operation
  - SINGLE ON: single shot time acquisition, time per point is limited to greater than or equal to 50 ns, trigger type is limited to EDGE, PHASE, or OFF.
- **POWER SWEEP**
  - SINGLE xxx: normal stepped power sweep
- **FREQUENCY SWEEP**
  - SINGLE xxx: normal stepped frequency sweep

Query Response



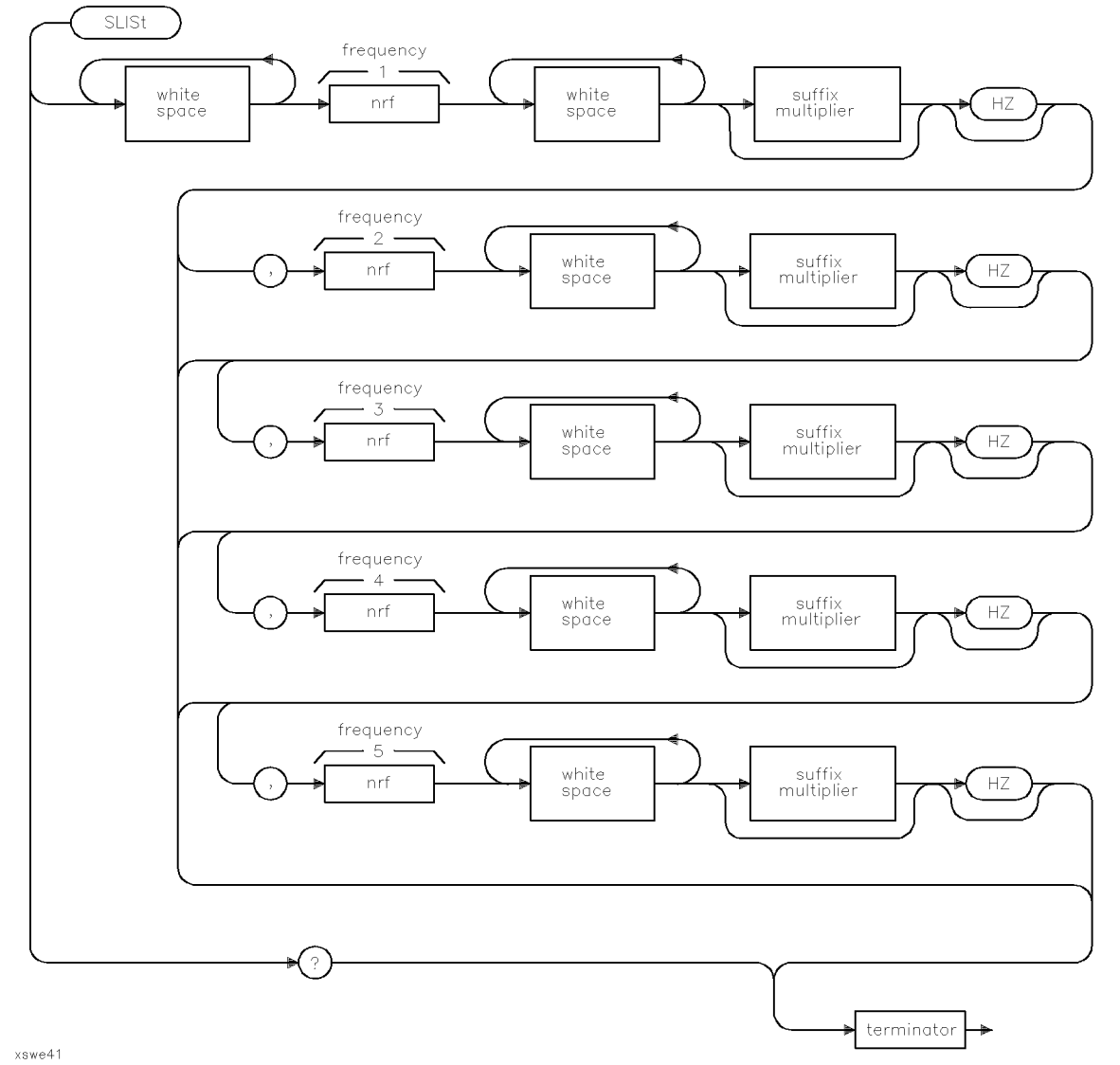
Query Example

```
OUTPUT @Mta;"SWE:SIN?"
ENTER @Mta;Single_shot$
```

SLISt

Defines the frequency list.

Syntax



Preset State:	one signal at 53.1787322 MHz if there is no synthesizer present; otherwise, the signal is dependent upon the synthesizer configuration.
Parameter Range:	frequency = 0.1 Hz to 100 GHz
Fundamental Unit:	hertz
Related Commands:	AUTO SWEep:SIGnal

Command Example

```
OUTPUT @Mta;"SWE:SLIS 100 MHz, 1 GHz"
```

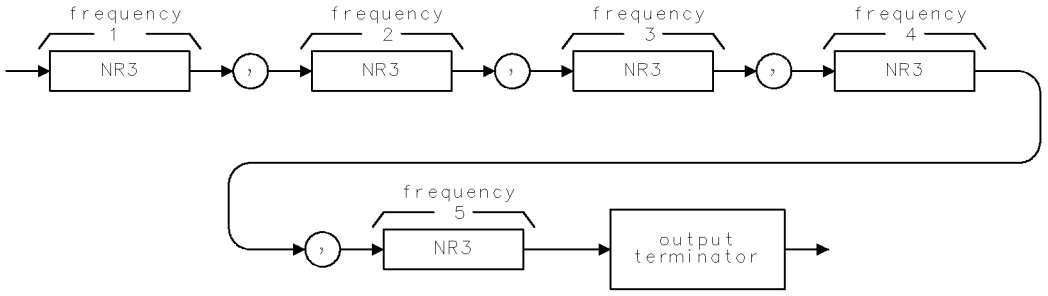
Description

The SLIST command is used to define the frequency of the signal or signals that the microwave transition analyzer knows about.

The SLIST command makes the first signal in this list the "active" signal (that is, any horizontal setup that requires the signal frequency will use this value in this list). The last four parameters are optional and omitting a value indicates that no signal is present. For example, sending `SWEEP:SLIST 100 MHz, 1GHz;` defines two signals. This list is also used as the fundamentals for the tabular frequency measurements.

The query returns the currently defined signals with a value of −1 indicating that no signal is present.

Query Response



QSWE41

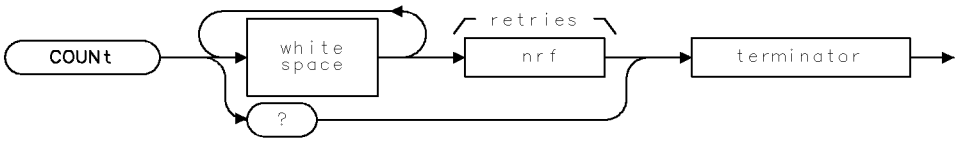
Query Example

```
OUTPUT @Mta;"SWE:SIG?"
ENTER @Mta;F(1),F(2),F(3),F(4),F(5)
FOR J=1 TO 5
  IF F(J)>0 THEN PRINT "Signal = ";F(J)
NEXT J
```

STRack:COUNt (Signal Track Count)

Controls the number of retries to acquire valid data.

Syntax



Preset State: 2
Parameter Range: retries = 0 through 8

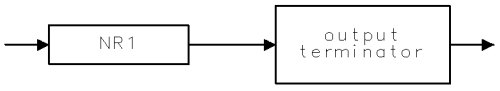
Command Example

```
OUTPUT @Mta;"SWE:STR:COUN 0"
```

Description

The STRACK:COUNT command controls the number of retries the HP 70820A will make to acquire valid data. If SWEEP:STRACK:STATE is ON and if, after the specified number of retries, signal track has identified a changed signal, then error −231 (questionable data) will be reported. The query returns the current state of STACK:COUNT.

Query Response



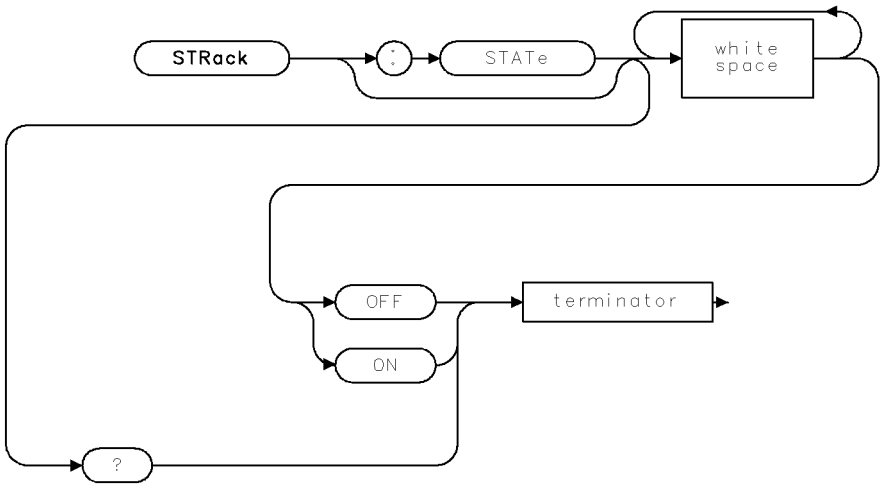
Query Example

```
OUTPUT @Mta;"SWE:STR:COUN?"
ENTER @Mta;Retries
```

STRack:STATe (Signal Track State)

Enables or disables signal track mode.

Syntax



Preset State: OFF

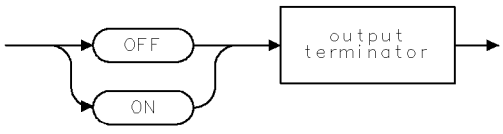
Command Example

```
OUTPUT @Mta;"SWE:STR ON"
```

Description

The STRACK:STATE command enables or disables the microwave transition analyzer signal track mode. The query returns the current state of STRACK.

Query Response



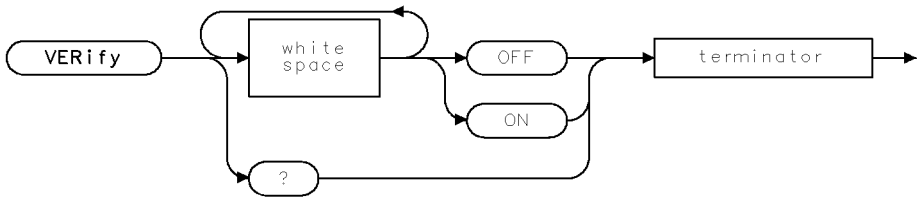
Query Example

```
OUTPUT @Mta;"SWE:STR?"
ENTER @Mta;Track$
```

STRack:VERify (Signal Track Verify)

Enables or disables signal track verification mode.

Syntax



Preset State: ON

Command Example

```
OUTPUT @Mta;"SWE:STR:VER OFF"
```

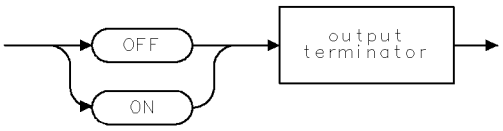
Description

The STRACK:VERIFY (signal track verify) command enables or disables the HP 70820A signal track verification mode.

If STRACK:STATE is ON when STRACK:VERIFY is set to ON, then the HP 70820A will make a measurement to verify the signal frequency. This verification measurement will be made on the sweep following the VERIFY command and every eight sweeps thereafter until VERIFY is turned OFF. This measurement will be made with a different sample rate than the normal signal track to identify cases where the signal has jumped by the sample frequency.

The query returns the current state of STRACK:VERIFY.

Query Response



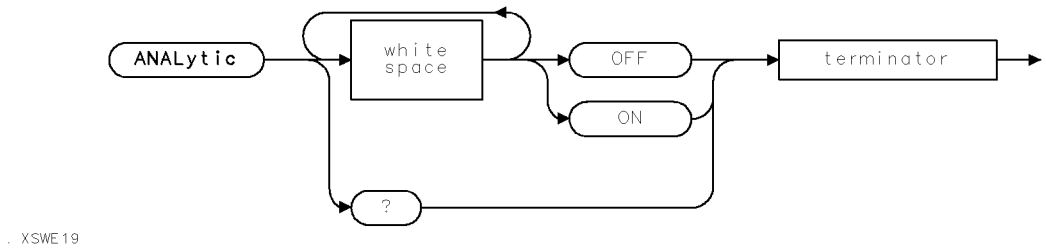
Query Example

```
OUTPUT @Mta;"SWE:STR:VER?"
ENTER @Mta;Verify$
```

TIME:ANALytic

Enables the analytic operator.

Syntax



Preset State ON

Command Example

```
OUTPUT @Mta;"SWE:TIME:ANALYTIC OFF"
```

Description

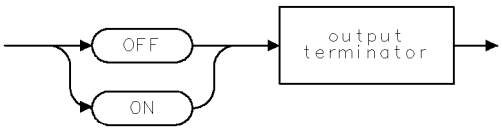
The ANALYTIC command sets a mode that converts the raw data for time sweeps into an analytic format—that is, the result is the Hilbert Transform pair of the input data. The query returns the current state of ANALYTIC.

The advantage of having ANALYTIC ON is various functions are immediately available via trace formatting. For example, the MAGNITUDE format is equivalent to AM demodulation. The PHASE format is equivalent to PM demodulation.

The disadvantage of having ANALYTIC ON is that more processing is being done which slows measurements down.

Note With ANALYTIC set to ON, the value for WAV:VRANGE? may not be the same as CHANx:RANGE.

Query Response



QGENON

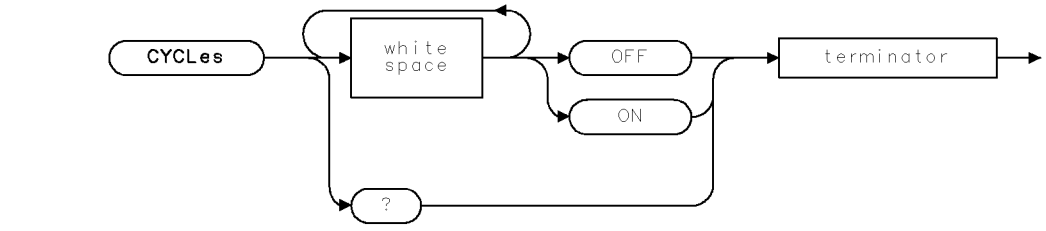
Query Example

```
OUTPUT @Mta;"SWE:TIME:ANALYTIC?"
ENTER @Mta;Analytic$
```

TIME:CYCLes

Enables or disables the “cycles” mode of operation.

Syntax



Preset State: OFF

Command Example

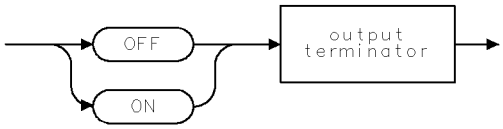
```
OUTPUT @Mta;"SWE:TIME:CYCL ON"
```

Description

The CYCLES command enables or disables the “cycles” mode of operation. If OFF, when the signal frequency is changed the measurement setup is recomputed to maintain a constant number of seconds in the time range. If ON, when the signal frequency is changed the measurement setup is recomputed to maintain a constant number of cycles in the time range.

The query returns the current state of CYCLES.

Query Response



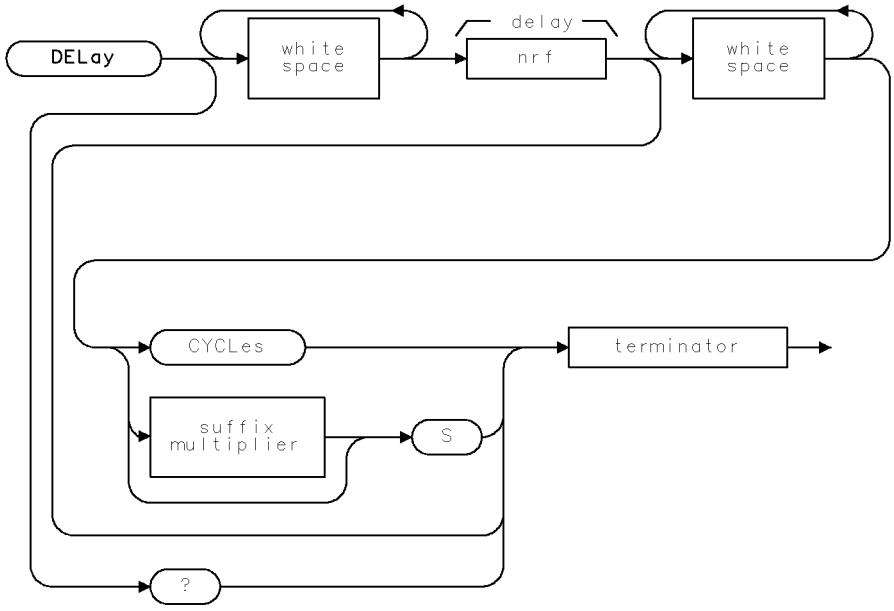
Query Example

```
OUTPUT @Mta;"SWE:TIME:CYCL?"
ENTER @Mta;Cycles$
```

TIME:DELay

Sets the timebase delay.

Syntax



XSWE21

Preset State: 0 s
Parameter Range: −time range/2 ≤ delay ≤ 1022 × time range
Fundamental Unit: seconds

Command Example

```
OUTPUT @Mta;"SWE:TIME:DEL 500 ns"
OUTPUT @Mta;"SWE:TIME:DEL 1 cycl"
```

Description

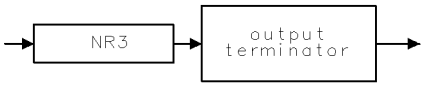
The DELAY command sets the timebase delay. This delay is the time interval between the trigger event and the delay reference point (center of the measurement).

The DELAY command will accept either seconds or cycles as a valid suffix. If the terminator is cycles, the module will determine the time delay required such that the next measurement will be offset from the delay reference (that is, center of the measurement) by the specified number of cycles.

Note For the DELAY to be accurate when specified with cycles or seconds, the microwave transition analyzer normally needs to know the correct signal frequency.

The query returns the current delay time value in seconds.

Query Response



QGENNR3

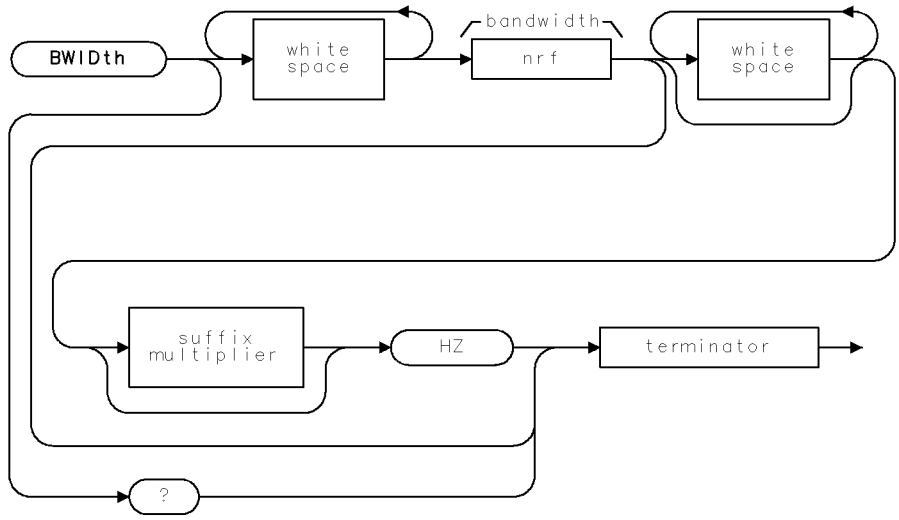
Query Example

```
OUTPUT @Mta;"SWE:TIME:DEL?"
ENTER @Mta;Delay
```

TIME:FILter:BWIDth

Controls the resolution bandwidth of a software filter.

Syntax



XSWE9

Preset State:	1 MHz
Parameter Range:	bandwidth = 62.5 kHz through 2 MHz
Fundamental Unit:	hertz
Related Commands:	MODulator:COUPle MODulator:STATe SWEep:FILter SWEep:TIME:FILter:STATe SWEep:TIME:FILter:VIDeo SWEep:TYPE

Command Example

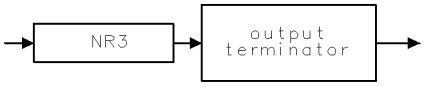
```
OUTPUT @Mta;"SWE:TIME:FIL:BWID 2.0E6"
```

Description

The BWIDTH command controls the resolution bandwidth of a software filter. This filter is used only for time domain sweeps (that is, SWEEP:TYPE is TIME) and when the modulation source is turned on (that is, MODULATOR:STATE is MANUAL or INTERNAL).

The query returns the current setting of BWIDTH.

Query Response



QGENNR3

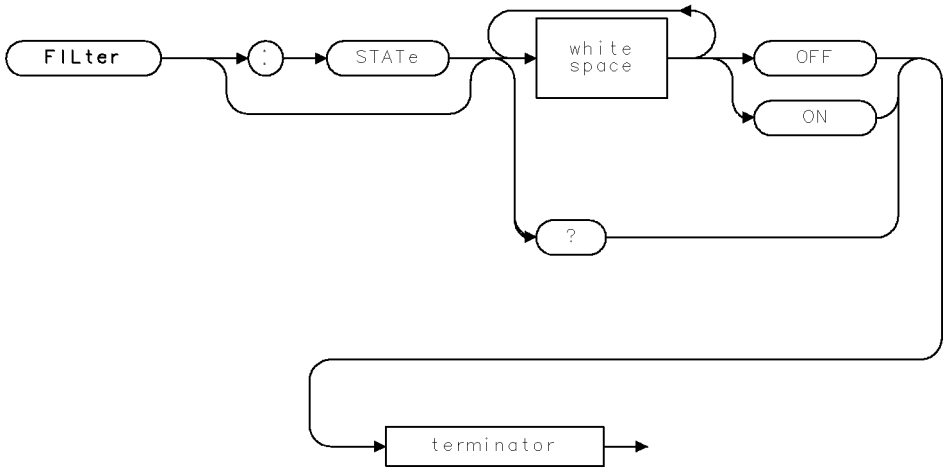
Query Example

```
OUTPUT @Mta;"SWE:TIM:FIL:BWID?"
ENTER @Mta;Filter$
```

TIME:FILter:STATe

Sets the state of an IF filtering process.

Syntax



Related Commands: MODulator:COUPle
MODulator:STATe
SWEep:FILter
SWEep:TIME:FILter:BWIDth
SWEep:TIME:FILter:VIDeo
SWEep:TYPE

Command Example

```
OUTPUT @Mta;"SWE:TIM:FIL ON"
OUTPUT @Mta;"SWE:TIM:FIL:STAT ON"
```

Description

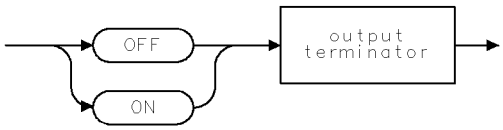
The STATE command sets the state of an IF filtering process. The query returns the current state of STATE.

The STATE command does not correspond directly to the state of a hardware filter but is interpreted based on the state of the instrument as follows:

- **TIME SWEEP:**
  - SINGLE SHOT: corresponds to hardware filter
  - REPETITIVE:
    - Modulation source off: corresponds to hardware filter
    - Modulation source on: corresponds to a software filter
- **FREQUENCY SWEEP**
  - SINGLE xxx: normal stepped frequency sweep

- **POWER SWEEP:** no effect; the hardware filter is automatically used if the signal position in the IF and the setting of resolution bandwidth warrants it.

Query Response



QGENON

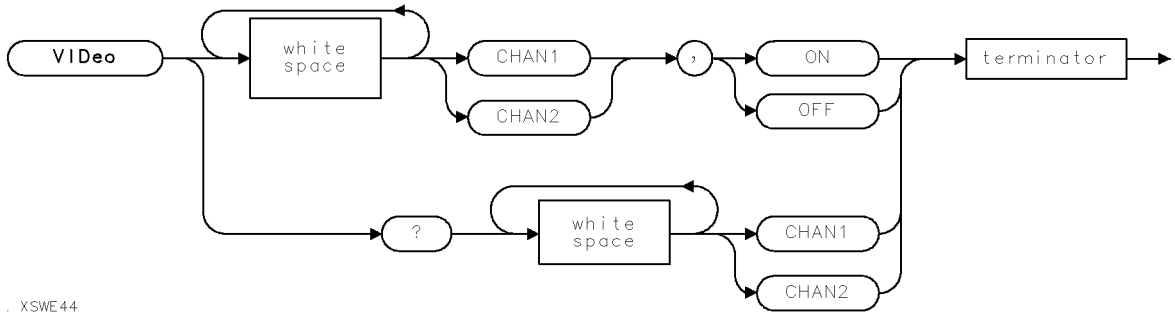
Query Example

```
OUTPUT @Mta;"SWE:TIME:FIL?"
OUTPUT @Mta;"SWE:TIME:FIL:STAT?"
ENTER @Mta;Filter$
```

TIME:FILter:VIDeo

Determines what portion of the signal will be used in pulsed RF measurements.

Syntax



Preset State: OFF
Related Commands: MODulator:COUPle
MODulator:STATe
SWEep:FILter
SWEep:TIME:FILter:BWIDth
SWEep:TIME:FILter:STATe
SWEep:TYPE

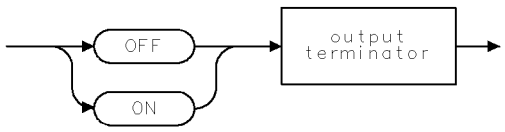
Command Example

```
OUTPUT @Mta;"SWE:TIME:FIL:VID CHAN1, ON"
```

Description

The VIDEO command determines what portion of the signal will be used in pulsed RF measurements with the filter on. When VIDEO is OFF, the carrier will be used (that is, the normal RF pulse measurement). When VIDEO is ON, the video feedthru of the signal is used.

Query Response



Query Example

```
OUTPUT @Mta;"SWE:TIME:FIL:VID? CHAN2"
ENTER @Mta:Video$
```

OVERSWEEP Commands

The SWEEP:TIME:OVERSWEEP: commands allow the user to acquire up to 250000 points of data in a single measurement for single shot data (for example, SWEeep:SINgle is ON). The data is available for the other subsystems (that is, FUNCTION, MARKER, WAVEFORM, and so on) a block at a time (where "a block" is up to 1024 points, specified by the ACQuire:POINts command sent before the measurement).

The measurement setup is controlled by the POINts command in addition to the "normal" setup commands (for example, ACQuire:POINts, SWEep:SINgle, SWEep:TIME:SRATe, and so on). After the data has been acquired, it can be manipulated with the DECimate, DETect, and OFFSet commands.

The DECimate and OFFSet commands provide a means of selecting various sub-portions of the raw data. For example, if the raw measurement is 100,000 points, setting DECimate to 10 and OFFSet to 25,000 would examine every 10th point centered at point 75,000 (note that an OFFSet of 0 corresponds to the center of the raw data).

The DETect commands provide a method of manipulating the data to provide more information. For example, if the raw measurement is 100,000 points and you would like to examine the entire data set for "spikes" that are only 5 or 6 points wide, you could use one of the following 2 approaches:

- The hard way:
  - set the DECimate to 1
  - set the OFFSet to -50,000
  - examine the data
  - increment the offset
  - repeat b, c, and d until all the data has been examined
- The easy way:
  - set the DECimate to 100
  - set the OFFSet to 0
  - set the DETect to MAXimum
  - examine the data
  - adjust DECimate and OFFSet to "zoom in" on any event of interest

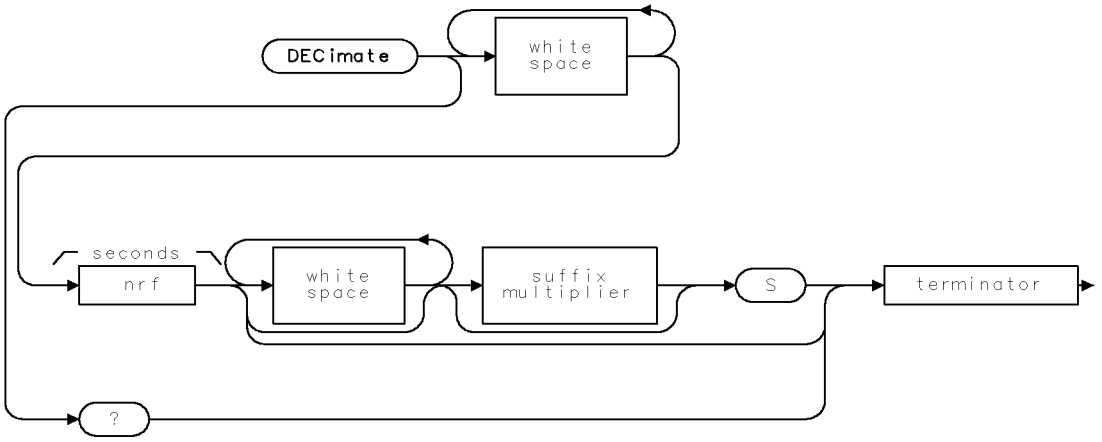
The following program fragment is an example of how to set up an oversweep measurement.

```
10  ASSIGN @Mta to 711
20  OUTPUT @Mta;"*RST"
30  OUTPUT @Mta;"SWE:MODE ASIN"             ! Synchronization is easier
                                              in single sweep mode
40  OUTPUT @Mta;"RUN;*OPC?"
50  ENTER @Mta;A$
60  OUTPUT @Mta;"ACQ:POIN 1024"             ! Set size of "blocks"
70  OUTPUT @Mta;"SWE:SIN ON"                ! Single shot must be on
80  OUTPUT @Mta;"SWE:TIME:SRAT 19.9 MHz"    ! Choose the sample rate
90  OUTPUT @Mta;"SWE:TIME:OVER:STAT ON"     ! Turn OVERSWEEP on
100 OUTPUT @Mta;"SWE:TIME:OVER:POIN 100000" ! Measurement size
110 OUTPUT @Mta;"RUN;*OPC?"                 ! Invoke the measurement
120 ENTER @Mta;A$
130 OUTPUT @Mta;"SWE:TIME:OVER:DEC 10"      ! Look at every 10th point
140 OUTPUT @Mta;"SWE:TIME:OVER:OFFS 10000"  ! Position the "block"
150                                         ! The rest is up to you!
```

TIME:OVERsweep:DECimate

Specifies the portion of the oversweep data to make available to the normal trace processing.

Syntax



Preset State: 1
Range: 1 ≤ decimate ≤ INT(SWE:TIM:OVER:POIN / ACQ:POIN)
Fundamental Unit: seconds

Command Example

```
OUTPUT @Mta;"SWE:TIME:OVER:DEC 5"
OUTPUT @Mta;"SWE:TIME:OVER:DEC 10 ms"
```

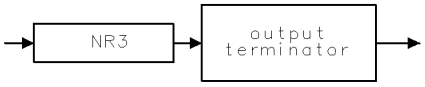
Description

The DECIMATE command specifies the portion of the oversweep data to make available to the normal trace processing. Since the DECIMATE command manipulates existing oversweep data, it should only be used after the data has been acquired (e.g. after the RUN;*OPC? sequence). The query returns the current value of DECIMATE.

If a measurement has been made with SWEEP:TIME:RANGE set to 1 us, a value of 1 for DECIMATE would result in a 1 us interval, a value of 5 would result in a 5 us interval, and so on.

If a seconds suffix is present, the value will be interpreted as a time range otherwise the value will be interpreted as a decimation factor (for example, keep 1 out of value points).

Query Response



QGENNR3

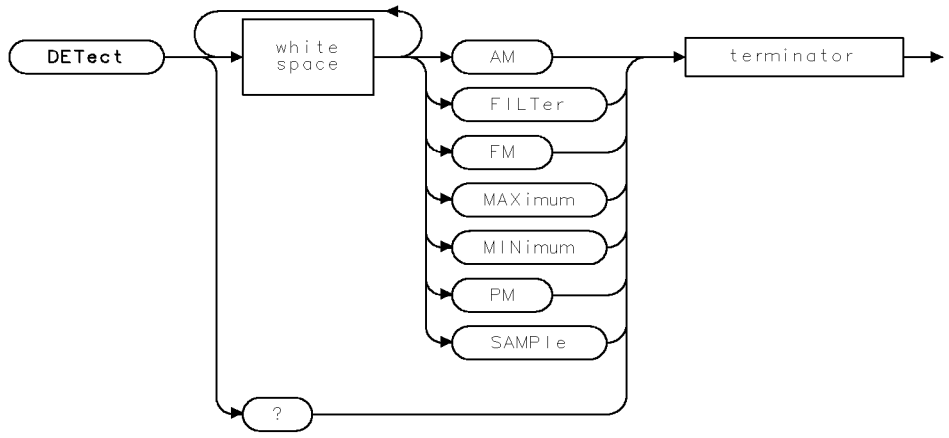
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:DEC?"
ENTER @Mta;Decimation
```

TIME:OVERsweep:DETect

Provides some preprocessing of the raw oversweep data before it is passed to any other subsystem.

Syntax



Preset State: SAMPle

Command Example

```
OUTPUT @Mta;"SWE:TIME:OVER:DET MAX"
```

Description

The DETECT command provides some preprocessing of the raw oversweep data before it is passed to any other subsystem. The detection modes available are:

AM

Detector returns the magnitude (for example, amplitude demodulate) of the data. The carrier should not be near DC or Fnyquist of SWEEP:TIME:SRATE. Because the detector operates before any decimation, you do not need to worry about how the carrier "mixes" in the effective sample rate (for example, SRATE / DECIMATE).

FILTER

Detector enables a mode that gives the user a larger degree of control over the preprocessing operation. Please see the SWEEP:TIME:OVERSWEEP:FILTER subsystem for details.

FM

The ability to resolve frequency is affected by the setting of SWEEP:TIME:OVERSWEEP:FILTER:APERTURE. This will attempt to track the largest signal present, so if multiple varying amplitude signals are present the result may

be invalid. Note that if the signal passes through DC or Fnyquist of SWEEP:TIME:SRATE that the resultant data will be invalid. By using the FILTER detector, you can achieve the same results as FM but with additional control over various parameters.

MAXIMUM

Detector returns the point with the maximum value from the subset determined by the DECIMATE command. For example, if DECIMATE is set to 10 the largest point out of each group of 10 points is returned. Note that if DECIMATE is set to 1 this is effectively a no-operation.

MINIMUM

Detector returns the point with the minimum value from the subset determined by the DECIMATE command. For example, if DECIMATE is set to 10 the smallest point out of each group of 10 points is returned. Note that if DECIMATE is set to 1 this is effectively a no-operation.

PM

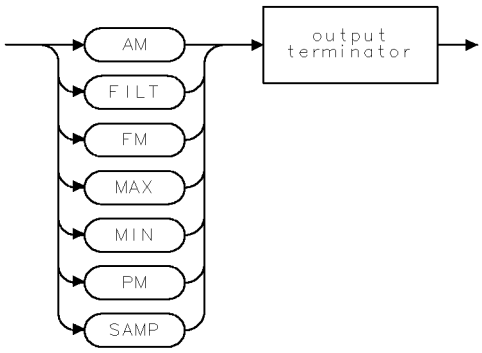
Detector returns the phase of the data. The carrier should not be near DC or Fnyquist of SWEEP:TIME:SRATE. Because the detector operates before any decimation, you do not need to worry about how the carrier "mixes" in the effective sample rate (for example, SRATE / DECIMATE).

SAMPLE

Detector does no processing. The points determined by the OFFSET and DECIMATE commands are returned.

The query returns the current state of DETECT.

Query Response



qswe56

Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:DET?"
ENTER @Mta;Detector$
```

SWEEP:TIME:OVERSWEEP:FILTER Commands

The SWEEP:TIME:OVERSWEEP:FILTER commands control a software filter applied to oversweep data. In general, this can be thought of as performing an FFT on a subset of the raw data and processing the resultant FFT in some way. For example, given a noisy sinusoidal input, the signal can be filtered by setting:

- FILTER:STATE to ON
- FILTER:APERTURE to 256 (this is the size of the FFT)
- FILTER:CHAN1:TRACK to CHAN1
- FILTER:CHAN1:FORMAT to SAMPLE

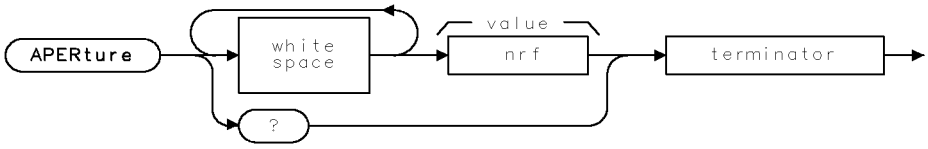
The result of this is to get the noise processing gain of a 256 point FFT at every point in the resulting trace. Setting TRACK to CHAN1 specifies to look for the largest signal which eliminates the need to know the input frequency. Setting FORMAT to SAMPLE specifies how to process the FFT data, in this case the results "look" the same as "normal" time domain data.

Note that the filtering (for example, the FFT) is done before the decimation specified by SWEEP:TIME:OVERSWEEP:DECIMATE. Depending on the value of DECIMATE, the filtering for adjacent points in the output may or may not use some of the same input points (for example, overlapping FFTs).

TIME:OVERsweep:FILTer:APERture

Sets the number of input points to use for filtering operations.

Syntax



Preset State: 64
Parameter Rrange: 16 through 256

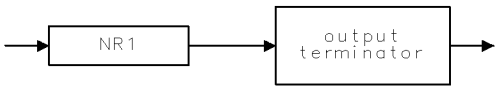
Command Example

```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:APER 256"
```

Description

The APERATURE command sets the number of input points to use for filtering operations. The value is always rounded to a power of 2. The query returns the current value of APERATURE.

Query Response



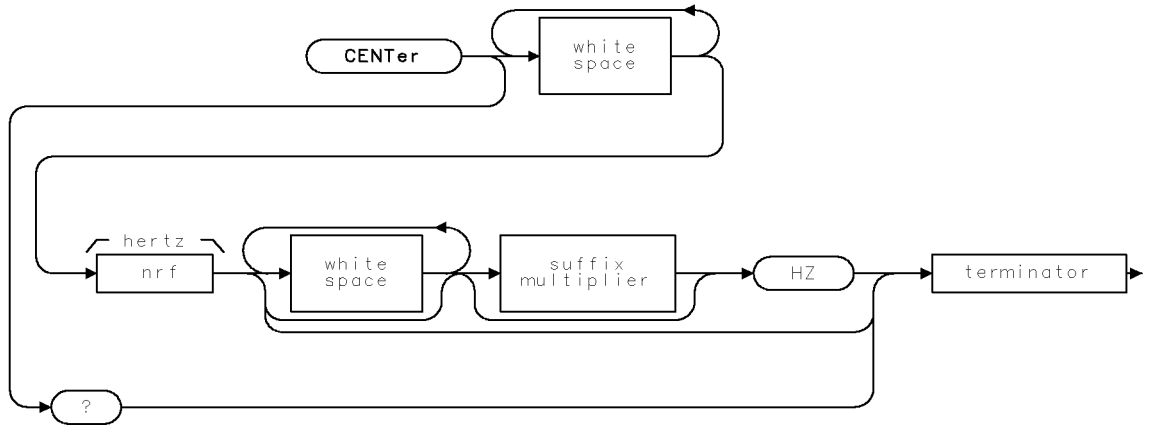
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:APER?"
ENTER @Mta;Aperture
```

TIME:OVERsweep:FILTer:CHANx:CENTer

Specifies the frequency value to use for the indicated channel.

Syntax



Preset State: The signal frequency at *RST, IP, or PON. This is dependent on the configuration of the system.
Parameter Range: 0 through 100 GHz
Fundamental Unit: hertz

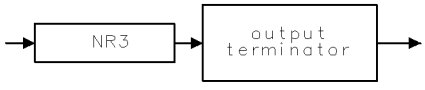
Command Example

```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:CHAN1:CENT 1 GHz"
```

Description

The CHANX:CENTER command specifies the frequency value to use for the indicated channel when CHANX:TRACK is set to MANUAL. The query returns the current value of CHANX:CENTER.

Query Response



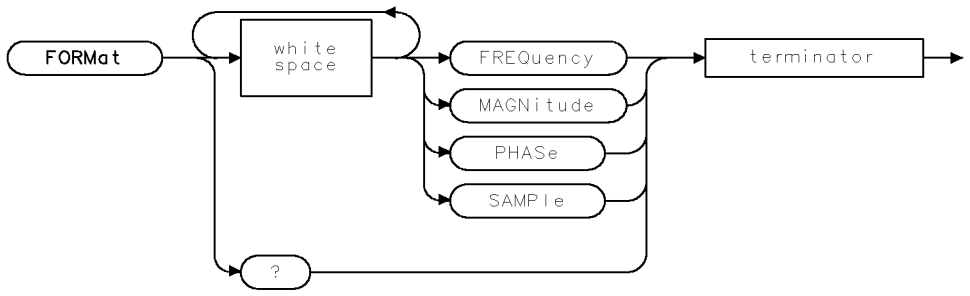
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:CHAN1:CENT?"
ENTER @Mta;Frequency
```

TIME:OVERsweep:FILTer:CHANx:FORMat

Specifies how to process the output of the filtering operation.

Syntax



Preset State: SAMPle

Command Example

```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:CHAN1:FORM FREQ"
```

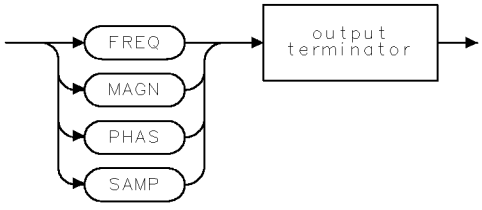
Description

The CHANX:FORMAT command specifies how to process the output of the filtering operation. The query returns the current value of CHANX:FORMAT.

- FREQUENCY returns the horizontal index into the FFT (for example, frequency) of the selected point.
- MAGNITUDE returns the magnitude of the selected point.
- PHASE returns the phase of the selected point.
- SAMPLE returns the complex value of the selected point.

"The selected point" is controlled by CHANX:TRACK and CHANX:CENTER.

Query Response



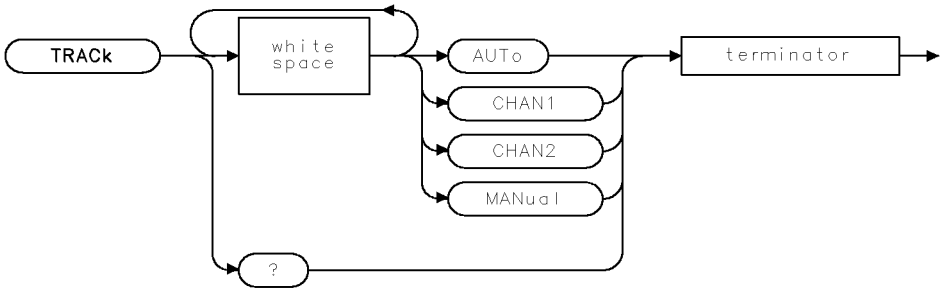
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:CHAN1:FORM?"
ENTER @Mta;Format$
```

TIME:OVERsweep:FILter:CHANx:TRACk

Specifies how to set the center frequency of the filtering operation.

Syntax



Preset State: AUTo

Command Example

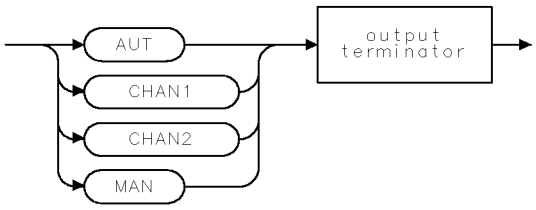
```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:CHAN1:TRAC CHAN1"
```

Description

The CHANX:TRACK command specifies how to set the center frequency of the filtering operation (for example, how to choose the output point from the FET). The query returns the current value of CHANX:TRACK.

- **AUTO** sets the center frequency of the filter to the currently defined signal frequency.
- **CHANX** chooses the largest amplitude signal on the channel. Note that channel 2 can be specified to track the largest signal on channel 1 and visa versa.
- **MANUAL** sets the center frequency of the filter to the value specified by SWEEP:TIME:OVERSWEEP:FILTER:CHANX:CENTER.

Query Response



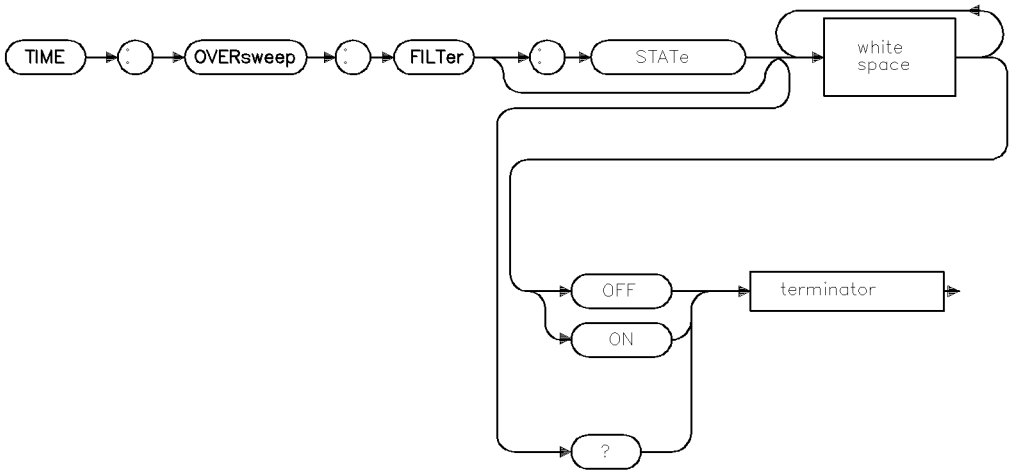
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:CHAN1:TRAC?"
ENTER @Mta;Track$
```

TIME:OVERsweep:FILTer:STATe

Turns the filter processing of oversweep data on or off.

Syntax



Preset State: ON

Command Example

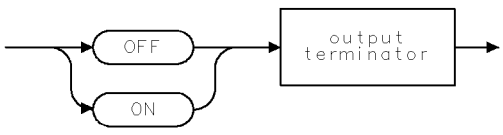
```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:STAT ON"
```

Description

The STATE command turns the filter processing of oversweep data on or off. The query returns the current value of STATE.

The value of STATE is coupled to SWEEP:TIME:OVERSWEEP:DETECT. When DETECT is set to FM, STATE is set to ON. When DETECT is set to any other value than FM, STATE is set to OFF. In other words, set the value of STATE after setting the value of DETECT.

Query Response



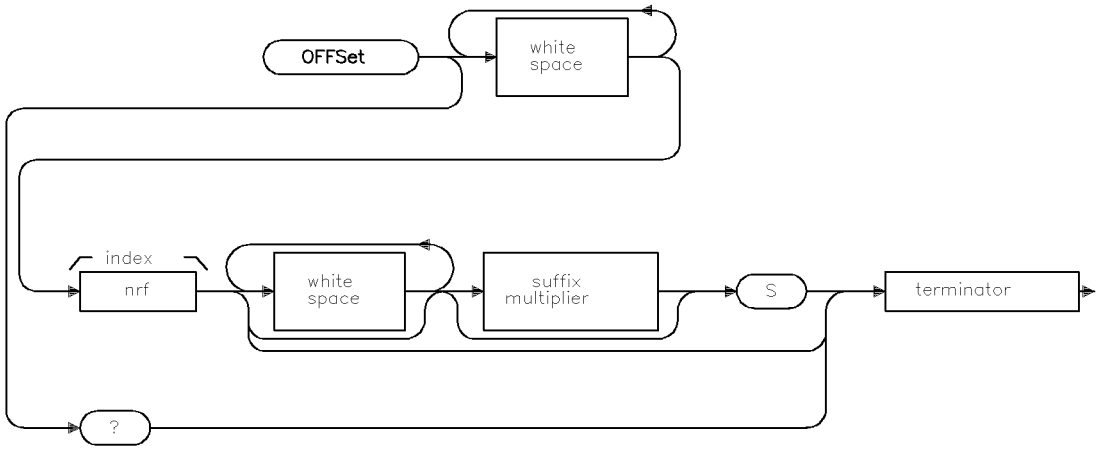
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:FILT:STAT?"
ENTER @Mta;State$
```

TIME:OVERsweep:OFFSet

Specifies the index into the oversweep data made available to the normal trace processing.

Syntax



Preset State: 0
Parameter Range: abs(value) ≤ (SWEep:TIME:OVERsweep:POINts-ACQuire:POINts)/2
Fundatmental Unit: seconds

Command Example

```
OUTPUT @Mta;"SWE:TIME:OVER:OFFS 5"
OUTPUT @Mta;"SWE:TIME:OVER:OFFS 10 ms"
```

Description

The OFFSET command specifies the index into the oversweep data made available to the normal trace processing. Since the OFFSET command manipulates existing oversweep data, it should only be used after the data has been acquired (for example, after the RUN;*OPC? sequence).

The query returns the current value of OFFSET.

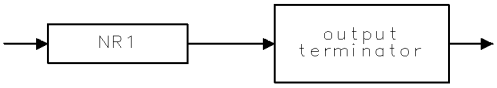
If a measurement has been made with

- ACQUIRE:POINTS set to 1000
- SWEEP:TIME:RANGE set to 1 us
- SWEEP:TIME:DELAY set to 0 us
- SWEEP:TIME:OVERSWEEP:DECIMATE set to 1

a value of 0 for OFFSET would result in a horizontal axis of -.5 us to +.5 us, a value of 500 would result in 0 us to 1.0 us, and so on.

If a seconds suffix is present, the value will be interpreted as a time offset otherwise the value will be interpreted as a point offset.

Query Response



. QGENN1

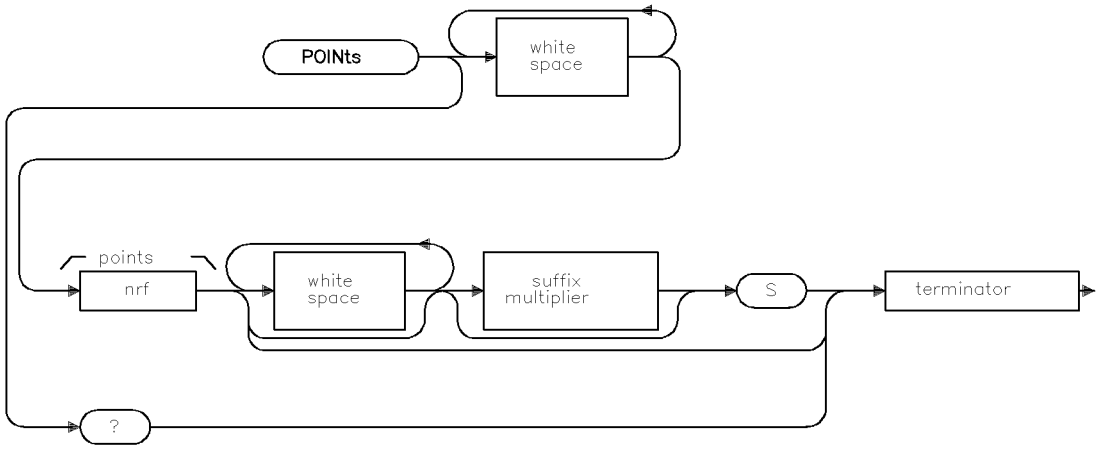
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:OFFS?"
ENTER @Mta;Offset
```

TIME:OVERsweep:POINts

Sets the number of data points that will be collected when the oversweep mode of operation is enabled.

Syntax



Preset State: 1024
Parameter Range: value1 = to 1024 through 250000

Command Example

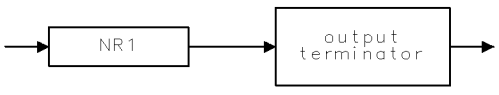
```
OUTPUT @Mta;"SWE:TIME:OVER:POIN 100000"
OUTPUT @Mta;"SWE:TIME:OVER:POIN 10 ms"
```

Description

The POINTS command sets the number of data points that will be collected when the oversweep mode of operation is enabled. The query returns the current value of POINTS.

If a seconds suffix is present, the value will be interpreted as a time range instead of as number of points.

Query Response



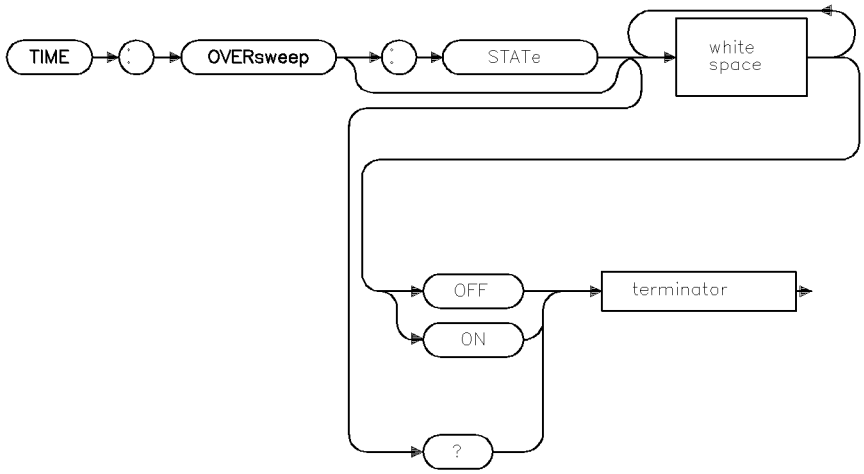
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER:POIN?"
ENTER @Mta;Points
```

TIME:OVERsweep:STATe

Enables and disables the oversweep mode of operation.

Syntax



Preset State: OFF

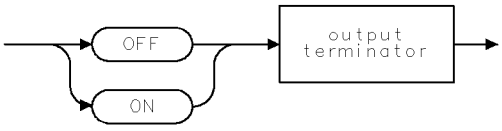
Command Example

```
OUTPUT @Mta;"SWE:TIME:OVER ON"
```

Description

The STATE command enables and disables the oversweep mode of operation. The query returns the current value of STATE.

Query Response



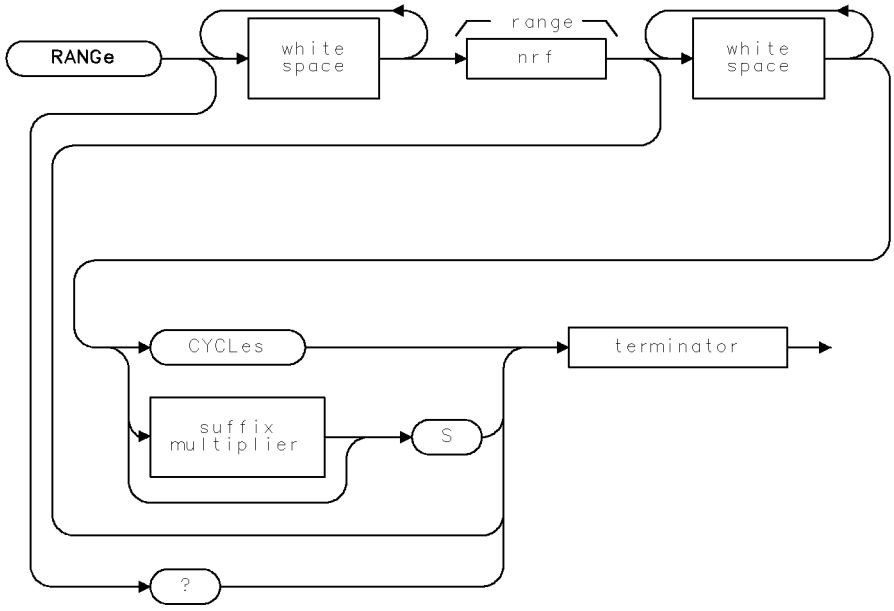
Query Example

```
OUTPUT @Mta;"SWE:TIME:OVER?"
ENTER @Mta;State$
```

TIME:RANGe

Sets the time range.

Syntax



Preset State:	100 nS
Parameter Range:	single shot: time range/trace length = 50 ns through 1.0 second all other modes: time range = 50 ps through 1024 seconds
Fundamental Unit:	seconds

Command Example

```
OUTPUT @Mta;"SWE:TIME:RANG 50 us"
OUTPUT @Mta;"SWE:TIME:RANG 5 cycl"
```

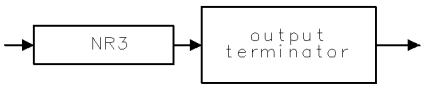
Description

The RANGE command sets the time range. (Refer to the TIME:SRATE command in the Sweep Subsystem.) The query returns the current time RANGE in seconds.

The RANGE command will accept either seconds or cycles as a valid suffix. If the terminator is cycles, the module will determine the time range required such that the next measurement will contain the specified number of cycles.

Note For the RANGE to be accurate when specified with cycles or seconds, the microwave transition analyzer normally needs to know the correct signal frequency.

Query Response



QGENNR3

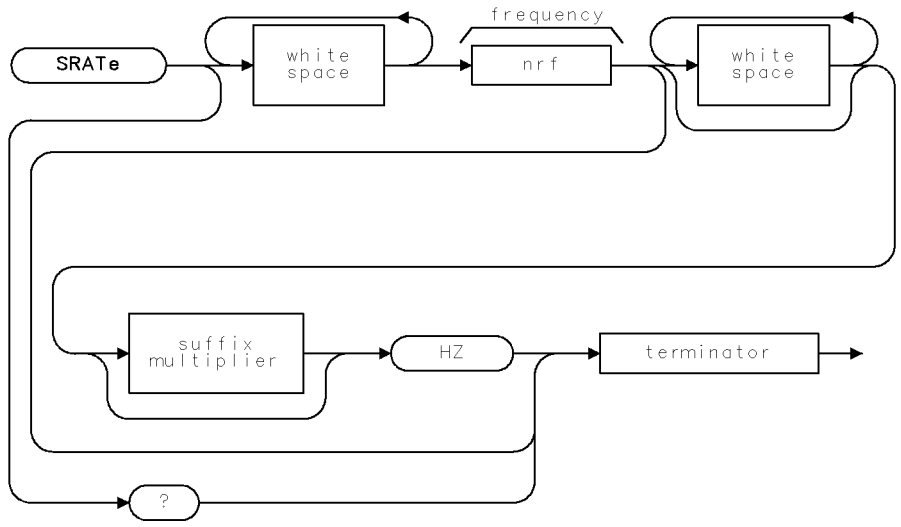
Query Example

```
OUTPUT @Mta;"SWE:TIME:RANG?"
ENTER @Mta;Time_range
```

TIME:SRATe (Sample Rate)

Allows the user to set the effective sampling rate of the module when SWEEP:SINGLE is ON.

Syntax



XSWE23

Preset State: frequency = 17.687 MHz if synthesizer is not present (otherwise, dependent on system configuration)
Parameter Range: frequency = 1 Hz through 20 MHz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"SWE:TIME:SRAT 15 MHz"
```

Description

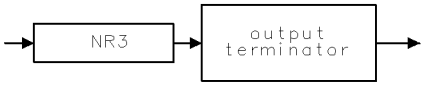
The SRATE command allows the user to set the effective sampling rate of the module when SWEEP:SINGLE is ON. The time range is affected according to:

time range = trace length/sample rate

Note that the inverse of the above equation does not apply unless the SRATE command has been sent. The reason for this is there are several operating modes which can cause multiple values of time range to map to the same sample rate.

The query returns the current value of SRATE.

Query Response



QGENNR3

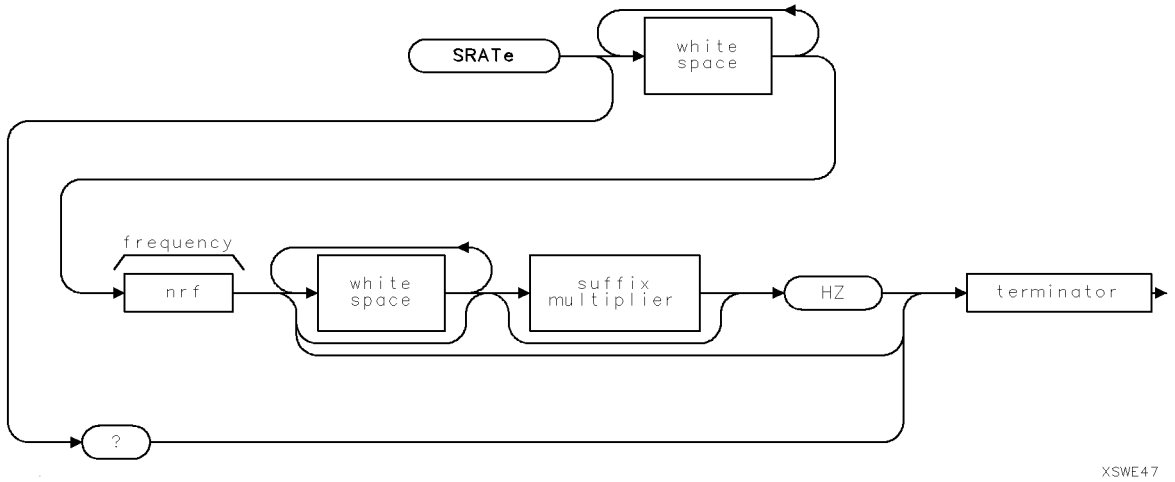
Query Example

```
OUTPUT @Mta;"SWE:TIME:SRAT?"
ENTER @Mta;Freq
```

TIME:TARGet:SRATe

Sets the desired sample frequency for the RF sampler.

Syntax



Preset State: 19.975 MHz
Parameter Range: frequency = 10 MHz through 20 MHz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"SWE:TIME:TARG:SRAT 17 MHz"
```

Description

The SRATE command sets the desired sample frequency for the RF sampler (that is, the sample rate before any decimation occurs). The query returns the current setting of SRATE.

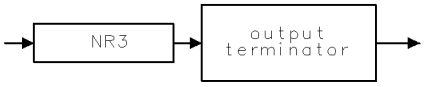
This value will be used in the single shot mode when the noise filter is enabled and SWEEP:TIME:TARGET:STATE is set to MANUAL. The decimation and sample rates will be determined as:

$$decimation = int(Ftarget / Feffective)$$
$$Fsampler = decimation \times Feffective$$

where

- Ftarget is the value specified by SWEEP:TIME:TARGET:SRATE
- Fsampler is the sample rate the hardware will run at
- Feffective is the value specified by SWEEP:TIME:SRATE

Query Response



QGENNR3

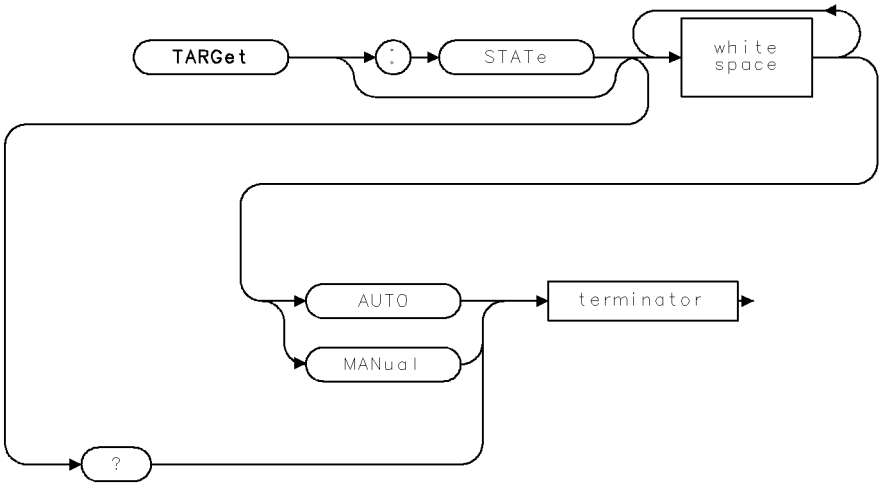
Query Example

```
OUTPUT @Mta;"SWE:TIME:TARG:SRAT?"
ENTER @Mta;Fss
```

TIME:TARGet:STATe

Determines how the hardware sample rate and decimation are chosen.

Syntax



Preset State: MANUAL

Command Example

```
OUTPUT @Mta;"SWE:TIME:TARG:STAT AUTO"
```

Description

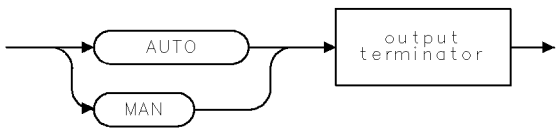
The STATE command determines how the hardware sample rate and decimation are chosen to achieve a desired sample rate when the signal shot is enabled and the noise filter is ON.

In the MANUAL mode, the hardware sample rate and decimation will be chosen as described in the SWEEP:TIME:TARGET:SRATE command.

In the AUTO mode, the hardware sample rate and decimation will be chosen to place the signal (or carrier frequency) at a reasonable position in the IF.

The query returns the current value of STATE.

Query Response



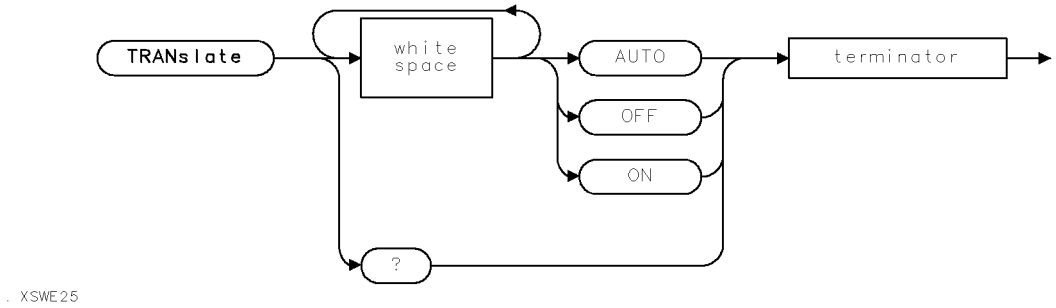
Query Example

```
OUTPUT @Mta;"SWE:TIME:TARG:STAT?"
ENTER @Mta;Target$
```

TIME:TRANslate

Enables or disables translate mode.

Syntax



Preset State: AUTO
Related Commands: CALibrate:CORRect:RF
MODulator:COUPle
MODulator:STATe
SWEep:SINgle
SWEep:TYPE
USER:STATe

Command Example

```
OUTPUT @Mta;"SWE:TIME:TRAN OFF"
```

Description

The TRANSLATE command enables or disables the microwave transition analyzer translate mode. The query returns the current state of TRANSLATE.

The translate mode, which is applicable only for time sweeps (that is, SWEEP:TYPE is TIME), controls the frequency axis interpretation of the data (that is, the start and stop frequencies if an FFT were performed).

When TRANSLATE is **OFF:**

$start\ frequency = 0$

$stop\ frequency = 0.5 \times trace\ length/time\ range$

The OFF state of TRANSLATE is generally used for wideband signals (such as a 1 GHz square wave).

When TRANSLATE is **ON:**

If MODULATOR:COUPLE is RFMOD and MODULATOR:STATE is not OFF

The start and stop frequencies are computed to provide a translated span about the carrier frequency. The span is determined by:

$$span = 0.5 \times trace\ length/time\ range$$

If MODULATOR:COUPLE is not RFMOD or MODULATOR:STATE is OFF

The start and stop frequencies are computed to provide a translated span about the signal frequency. The span is determined by the sample rate of the data being saved and processed (which involves both the actual sample rate of the RF samplers and any hardware/software decimation that follows).

Note In this mode, the frequency span computed by

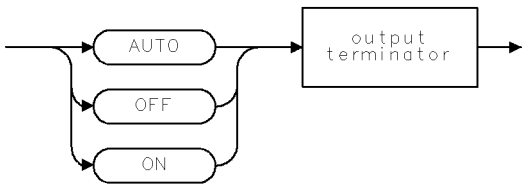
$$frequency\ span = 0.5 \times trace\ length/time\ range$$

will not match the frequency span determined for the frequency domain data unless SWEEP:SINGLE is ON. This can cause certain operations (such as frequency shifts) to behave differently in the time domain versus the frequency domain.

The ON state of TRANSLATE is generally used for either narrowband signals (such as a 1 GHz sine wave or a carrier with narrowband modulation) or for pulsed RF signals.

Note The MODULATOR:COUPLE and MODULATOR:STATE commands couple to the state of TRANSLATE. If MODULATOR:COUPLE = RFMOD and MODULATOR:STATE is not OFF, TRANSLATE is set to ON; otherwise, TRANSLATE is set to OFF.

Query Response



QSWE25

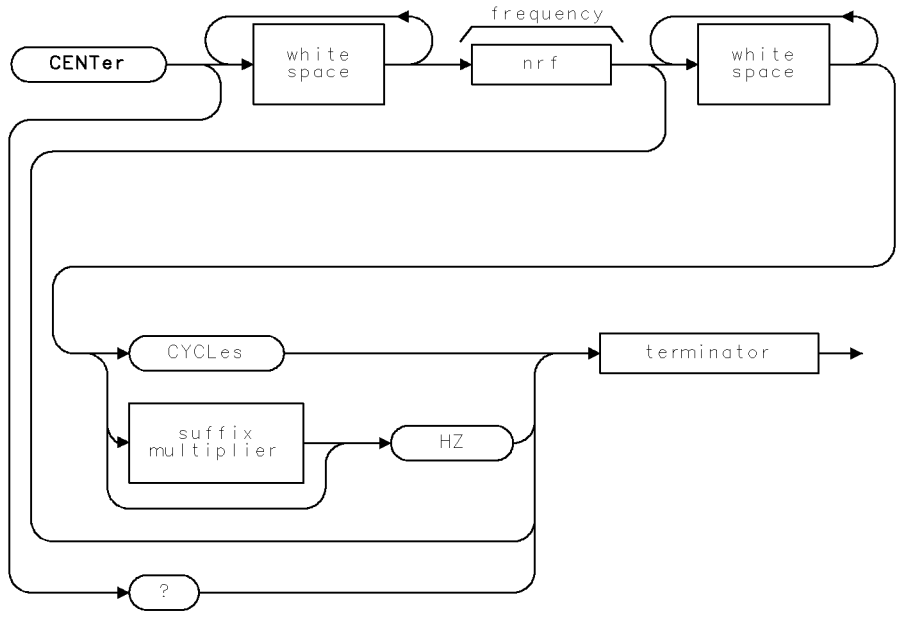
Query Example

```
OUTPUT @Mta;"SWE:TIME:TRAN?"
ENTER @Mta;Translate$
```

TIME:ZOOM:CENTer

Specifies the center frequency used for the zoom transform.

Syntax



XSWE26

Preset State: 50 MHz
Parameter Range: frequency = 0 through 1000 GHz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:CENT 5 GHz"
```

Description

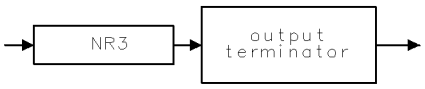
The TIME:ZOOM:CENTER command specifies the center frequency used for the zoom transform. The query returns the current setting.

The CENTER command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting of a measurement independent of the specific signal frequency. Internally, the module will convert cycles to hertz by:

$$hertz = cycles \times signal\ frequency.$$

The query always returns the value in hertz.

Query Response



QGENNR3

Query Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:CENT?"
ENTER @Mta;Center_freq
```

TIME:ZOOM:REZoom

Recomputes the zoom transform.

Syntax

xswe50

Command Example

```
! Set up a 1 MHz span centered at 5 MHz
OUTPUT @Mta;"SWE:TIME:ZOOM:STAT ON"
OUTPUT @Mta;"SWE:TIME:ZOOM:SPAN 1MHz"
OUTPUT @Mta;"SWE:TIME:ZOOM:CENTER 5MHz"
! Take the data
OUTPUT @Mta;"RUN;*OPC?"
ENTER @Mta;A$
! Reading the data goes here...
!
! Change the zoom to 1 MHz span centered at 1 MHz
OUTPUT @Mta;"SWE:TIME:ZOOM:CENTER 1MHz"
OUTPUT @Mta;"SWE:TIME:ZOOM:REZOOM"
! Reading the data goes here...
```

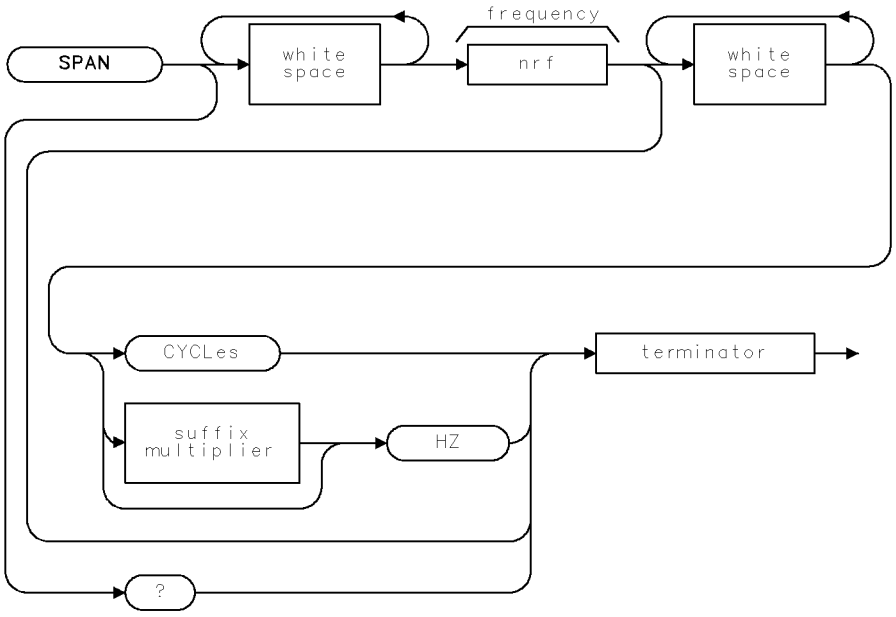
Description

The REZOOM command recomputes the zoom transform of the input. Only the center frequency may be changed between successive "re-zooms" (for example, STOP-START must be a constant). This command allows the user to make a single measurement and look at various portions of the spectrum with the zoom transform.

TIME:ZOOM:SPAN

Specifies the frequency span used for the zoom transform.

Syntax



XSWE27

Preset State: 100 MHz
Parameter Range: frequency = 0 through 1000 GHz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:SPAN 10 GHz"
```

Description

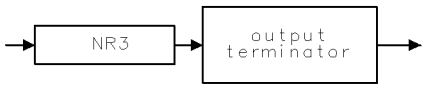
The TIME:ZOOM:SPAN command specifies the frequency span used for the zoom transform. The query returns the current setting.

The SPAN command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting of a measurement independent of the specific signal frequency. Internally, the module will convert cycles to hertz by:

$$hertz = cycles \times signal\ frequency.$$

The query always returns the value in hertz.

Query Response



QGENNR3

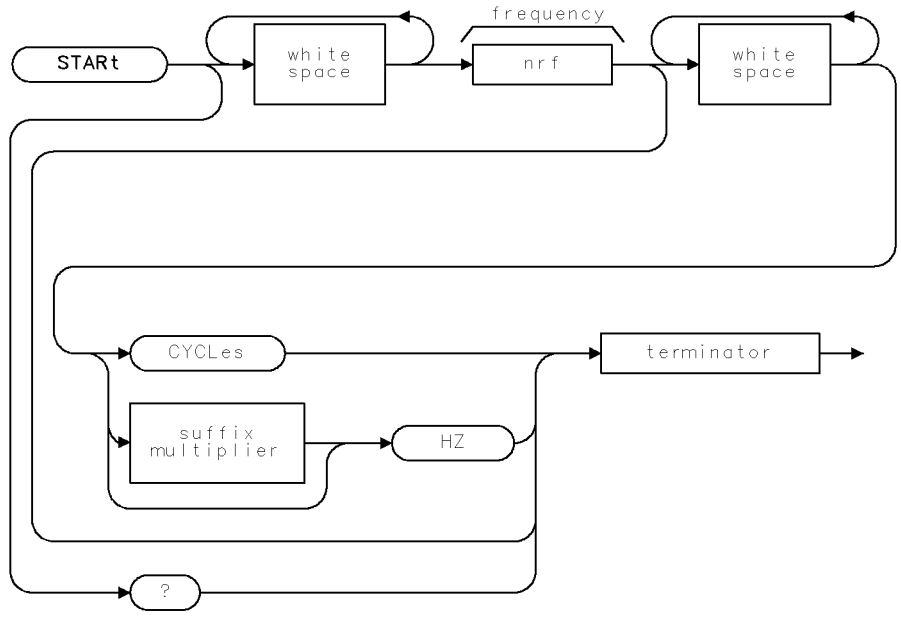
Query Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:SPAN?"
ENTER @Mta;Span
```

TIME:ZOOM:STARt

Specifies the start frequency used for the zoom transform.

Syntax



XSWE28

Preset State:	0
Parameter Range:	frequency = 0 through 1000 GHz
Fundamental Unit:	hertz
Related Commands:	CHANx\:ZOOM:STARt FUNCx:ZOOM:STARt

Command Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:STAR 100 MHz"
```

Description

The TIME:ZOOM:START command specifies the start frequency used for the zoom transform. The query returns the current setting.

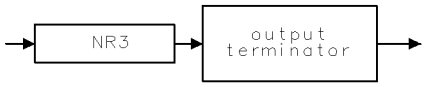
The START command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting of a measurement independent of the specific signal frequency. Internally, the module will convert cycles to hertz by:

$$hertz = cycles \times signal\ frequency.$$

The query always returns the value in hertz.

Note This command modifies the same value as the CHANx:ZOOM:START and FUNCx:ZOOM:START commands.

Query Response



QGENNR3

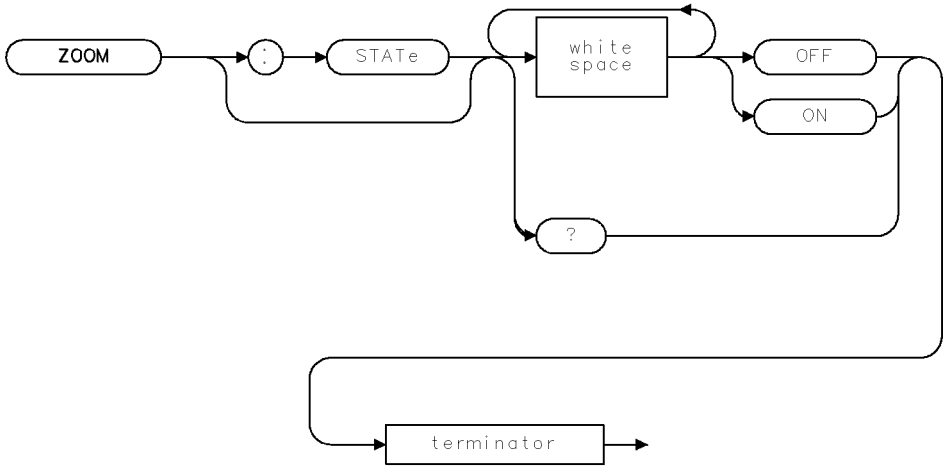
Query Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:STAR?"
ENTER @Mta;Start_freq
```

TIME:ZOOM:STATe

Turns the zoom transform on and off.

Syntax



Preset State: OFF

Command Example

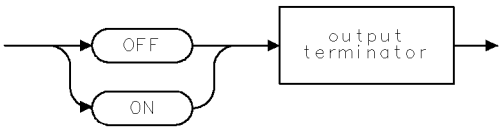
```
OUTPUT @Mta;"SWE:TIME:ZOOM:STAT ON"
```

Description

The ZOOM:STATE command is used to turn the zoom transform on and off. The query returns the current state of ZOOM:STATE.

When ZOOM:STATE is on, the module will take enough extra data to perform a zoom transform with the start and stop frequencies specified by TIME:ZOOM:START and TIME:ZOOM:STOP. The zoom data is available when the channel is displayed in the frequency domain or via the functions when the functions have zoom data selected.

Query Response



QGENON

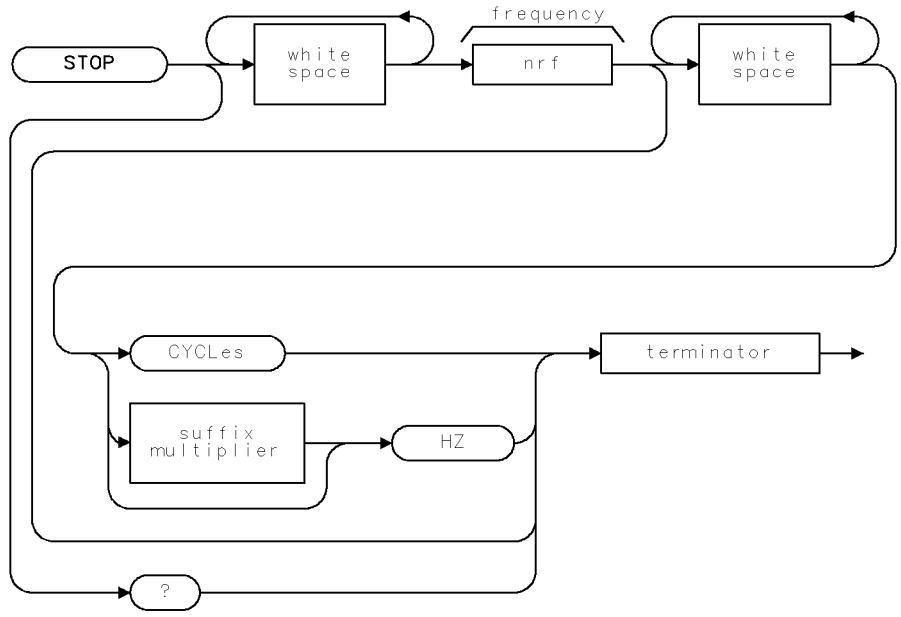
Query Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:STAT?"
ENTER @Mta;Zoom_xform$
```

TIME:ZOOM:STOP

Specifies the stop frequency used for the zoom transform.

Syntax



XSWE30

Preset State:	100 MHz
Parameter Range:	frequency = 0 through 1000 GHz
Fundamental Unit:	hertz
Related Commands:	CHANx:ZOOM:STARt FUNCx:ZOOM:STARt

Command Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:STOP 10 GHz"
```

Description

The TIME:ZOOM:STOP command specifies the stop frequency used for the zoom transform. The query returns the current setting.

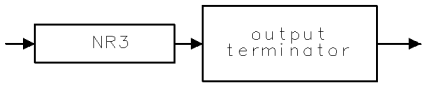
The STOP command will accept either hertz or cycles as a valid suffix. The cycles suffix is intended to provide a method of setting of a measurement independent of the specific signal frequency. Internally, the module will convert cycles to hertz by

$$hertz = cycles \times signal\ frequency.$$

The query always returns the value in hertz.

Note This command modifies the same value as the CHANx:ZOOM:STOP and FUNCx:ZOOM:STOP commands.

Query Response



QGENNR3

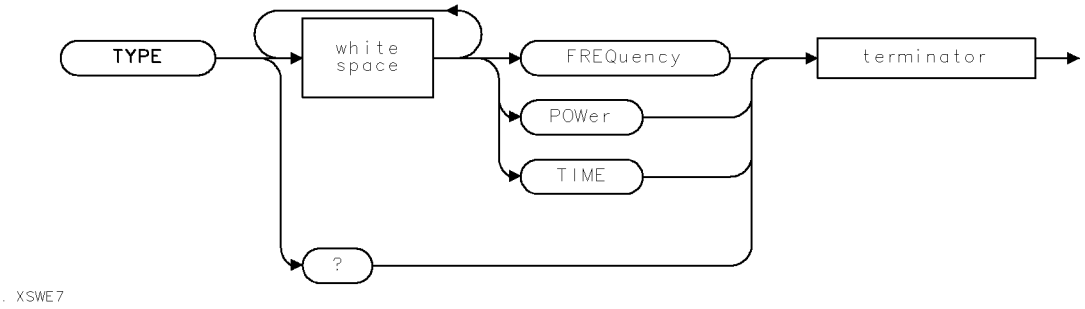
Query Example

```
OUTPUT @Mta;"SWE:TIME:ZOOM:STOP?"
ENTER @Mta;Stop_freq
```

TYPE

Sets the sweep type.

Syntax



Preset State: TIME

Command Example

```
OUTPUT @Mta;"SWE:TYPE POW"
```

Description

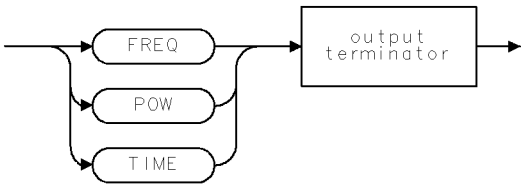
The TYPE command specifies how the next sweep will be made (that is, time sweep, frequency sweep, or power sweep). The query returns the current state.

When the sweep type is changed (for example, TIME to FREQUENCY) this command will couple the sweep type to the WAVEFORM:DOMAIN and WAVEFORM:MAGNITUDE parameters for the channels and functions as follows:

- **FREQUENCY or POWER to TIME:**
  - DOMain = REAL,OFF
  - MAGNitude = LINear, oldvalue
- **POWER or TIME to FREQUENCY:**
  - DOMain = OFF,MAGN
  - MAGNitude = oldvalue,LOG
- **FREQUENCY or TIME to POWER:**
  - DOMain = OFF,MAGN
  - MAGNitude = oldvalue,LOG

Note that if the sweep type is unchanged, no coupling occurs. Because of this coupling, it is generally a good idea to set the DOMAIN and MAGNITUDE values for traces after setting the sweep type.

Query Response



. QSWE7

Query Example

```
OUTPUT @Mta;"SWE:TYPE?"
ENTER @Mta;Sweep_type$
```

27

System Subsystem

The SYSTEM subsystem commands are used to return the state of the microwave transition analyzer system. See the following list for a command syntax summary of the SYSTEM subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:SYSTem		
:PRESet		
[:EXECute]		[no query]
:SAVE		[no query]
:TYPE	FACTory|USER	
:REFerence	AUTO|EXTernal|INTernal	
:STATus?		[query only]

PRESet:EXECute

Performs an instrument preset.

Syntax

xsys2

Command Example

```
OUTPUT @Mta;"SYST:PRES"
```

Description

The PRESET:EXECUTE command performs the same operation as the front panel IP key. Note that if SYSTEM:PRESET:TYPE is set to USER, the state of the instrument after PRESET:EXECUTE is user defined.

PRESet:SAVE

Saves the current state as a user defined instrument preset.

Syntax

xsys3

Command Example

```
OUTPUT @Mta;"SYST:PRES:SAVE"
```

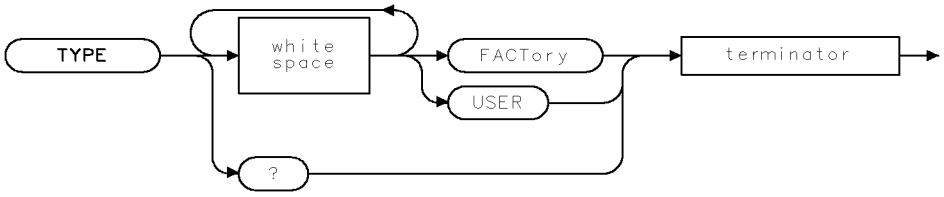
Description

The PRESET:SAVE command saves the current state as a user defined instrument preset.

PRESet:TYPE

Selects either a factory defined preset or a user defined preset.

Syntax



Preset State: Unaffected by PON, IP, or *RST

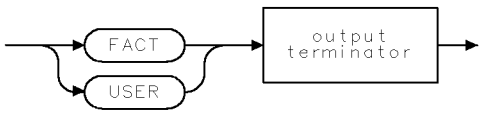
Command Example

```
OUTPUT @Mta;"SYST:PRES:TYPE USER"
```

Description

The PRESET:TYPE command selects either a factory defined preset or a user defined preset state when the IP key is pressed or the SYSTEM:PRESET:EXECUTE command is received. The query returns the current state of PRESET:TYPE.

Query Response



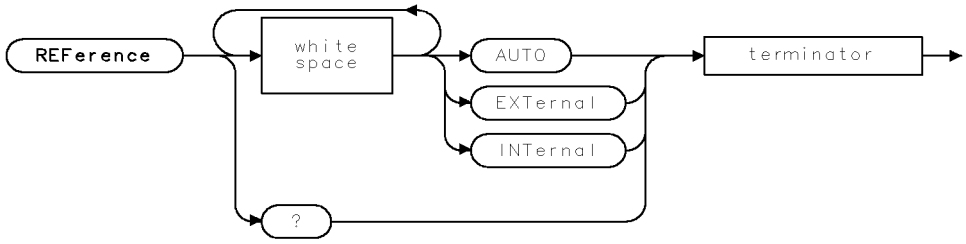
Query Example

```
OUTPUT @Mta;"SYST:PRES:TYPE?"
ENTER @Mta;Preset$
```

REFerence

Selects 10 MHz reference.

Syntax



Preset State: Unaffected by PON, IP, or *RST

Command Example

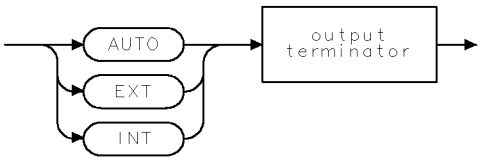
```
OUTPUT @Mta;"SYST:REF AUTO"
```

Description

The REFERENCE command determines how the 10 MHz reference will be set at power on (PON) and instrument preset (IP). The query returns the current state of REFERENCE.

- AUTO: the module will check for an external reference. If one is present, SWEEP:REFERENCE will be set to EXTERNAL. If one is not present, SWEEP:REFERENCE will be set to INTERNAL.
- EXTERNAL: SWEEP:REFERENCE will be set to EXTERNAL.
- INTERNAL: SWEEP:REFERENCE will be set to INTERNAL.

Query Response



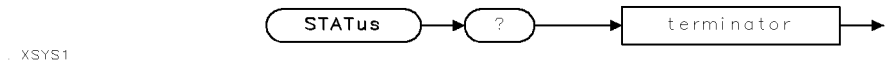
Query Example

```
OUTPUT @Mta;"SYST:REF?"
ENTER @Mta;Reference$
```

STATus

Returns time sweep values.

Syntax

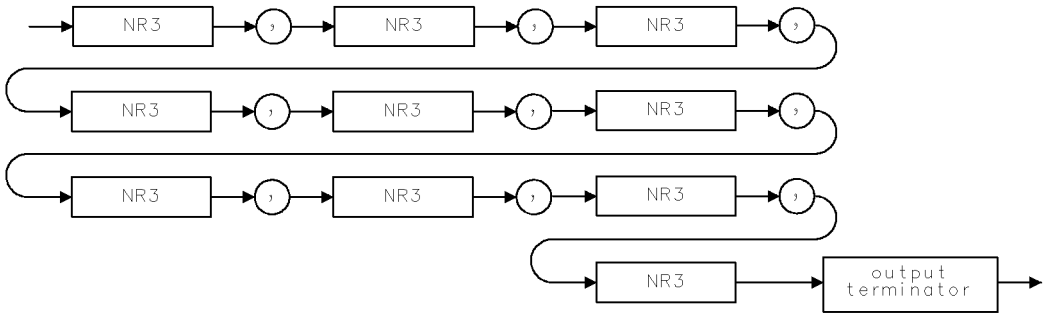


Description

The STATUS query returns various internal state values for time sweeps (for example, SWEEP:TYPE is TIME). The results are invalid for FREQUENCY and POWER sweeps. The parameters are returned in the following order:

- SAMPLE FREQUENCY: the sample rate of the front-end RF sampler.
- DECIMATION: hardware decimation of the sampled data.
- SIGNAL FREQUENCY: the expected signal frequency.
- SIGNAL I.F.: the placement of the signal frequency in the IF filters.
- CARRIER FREQUENCY: the expected carrier frequency (valid for pulsed RF measurements only).
- CARRIER I.F.: the placement of the carrier frequency in the IF filters (valid for pulsed RF measurements only).
- DSP SAMPLE FREQUENCY: the effective sample rate used for pulsed RF measurements and for signals <10 MHz (except single shot).
- DSP SIGNAL I.F.: the placement of the signal frequency with respect to DSP SAMPLE FREQUENCY for pulsed RF measurements and for signals <10 MHz (except single shot).
- DSP CARRIER I.F.: the placement of the carrier frequency with respect to DSP SAMPLE FREQUENCY for pulsed RF measurements.
- EFFECTIVE SAMPLE FREQUENCY: the effective sample frequency including the effects of all decimation and processing.

Query Response



. QSYS1

Query Example

```
OUTPUT @Mta;"SYST:STAT?"
ENTER @Mta;S1,S2,S3,S4,S5,S6,S7,S8,S9,S10
```

28

Table Subsystem

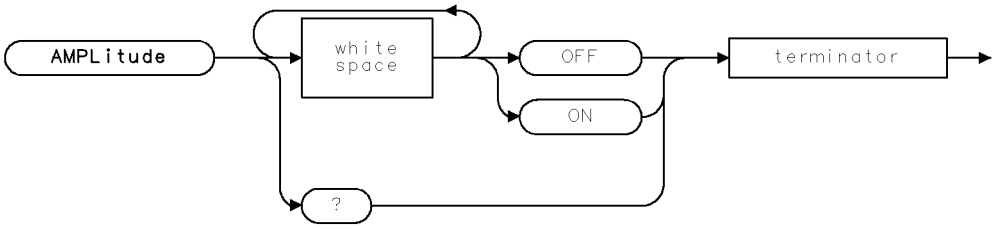
The TABLE subsystem allows the user access to the tabular data that can be generated in the microwave transition analyzer. See the following list for a command syntax summary of the TABLE subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:TABLe		
:AMPLitude	OFF|ON	
:CLEAr		[no query]
:COUNt	<nrf>	
:DELTa	OFF|ON	
:FILTer	OFF|ON	
:FIND		[no query]
:HARMonics	<nrf>	
:INCLude	OFF|ON	
:MEASure	ONE|ALL	
:MODE	ABSolute|RELative	
:PHASe	OFF|ON	
:REFerence	<nrf>,<nrf>	
:RESults?	<nrf>,<nrf>	[query only]
:SETReference		[no query]
:SOURce	CH1CH2|CH2CH1|CHAN1|CHAN2|OFF	
[:TYPE]	AVERage|CONTinuous|SINGle	
:VALid?		[query only]

AMPLitude

Turns the display of amplitude information on or off.

Syntax



Preset State: ON

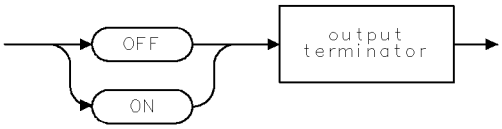
Command Example

```
OUTPUT @Mta;"TABL:AMPL ON"
```

Description

The AMPLITUDE command allows the display of amplitude information to be turned on or off. This does not affect remote operation.

Query Response



Query Example

```
OUTPUT @Mta;"TABL:AMPL?"
ENTER @Mta;A$
```

CLEAr

Terminates tabular mode operation.

Syntax

Command Example

```
OUTPUT @Mta;"TABL:CLEA"
```

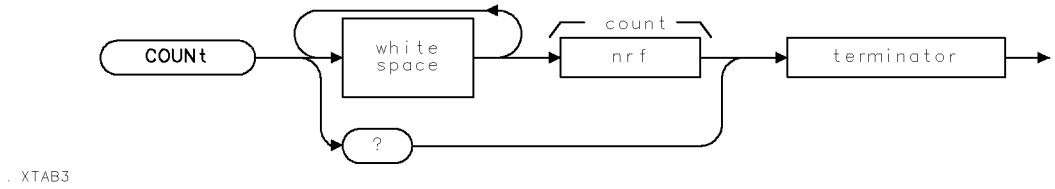
Description

The CLEAR command is used to terminate table operation. If the tabular type is in CONTINUOUS update, the updating will stop and the display is returned to the normal formatting. The table source is switched to OFF (see TABLE:SOURCE).

COUNt

Sets the number of averages to use.

Syntax



Preset State: 16
Parameter Range: count = 1 through 1024

Command Example

```
OUTPUT @Mta;"TABL:COUN 16"
```

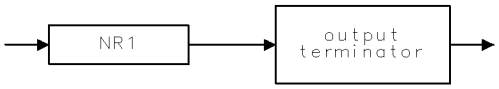
Description

The COUNT command sets the number of averages to use when in the tabular mode of operation. The averaging operation is implemented as a moving average, that is:

$$new\ value = \frac{((averages - 1) \times old\ value + new\ value)}{averages}$$

The query returns the last value set for the COUNT command.

Query Response



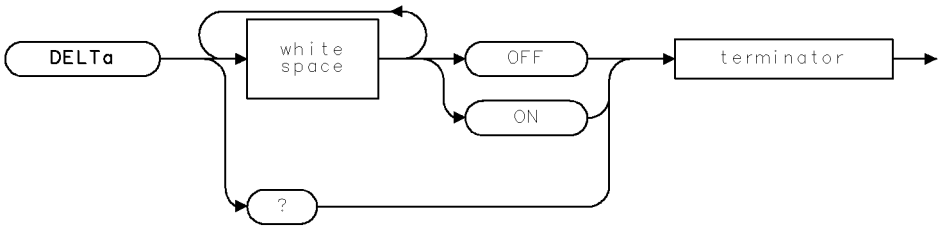
Query Example

```
OUTPUT @Mta;"TABL:COUN?"
ENTER @Mta;Count
```

DELTa

Selects the tabular delta mode.

Syntax



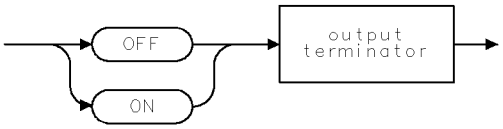
Command Example

```
OUTPUT @Mta;"TABL:DELT ON"
```

Description

The DELTA command is used to subtract a reference table (see TABLE:SETREFERENCE) from the current tabular measurement before display. If DELTA is ON, the reference table will be subtracted. If a valid reference table does not exist, FAILVALUE (=1.0E38) is returned.

Query Response



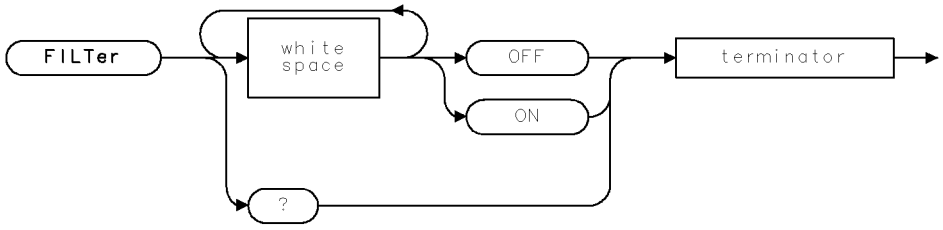
Query Example

```
OUTPUT @Mta;"TABL:DELT?"
ENTER @Mta;A$
```

FILTer

Selects an internal narrowband filter.

Syntax



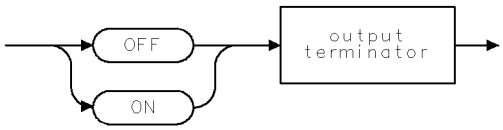
Command Example

```
OUTPUT @Mta;"TABL:FILT ON"
```

Description

The FILTER command determines whether an internal narrowband filter is used for the tabular measurement. Use of the filter generally results in increased measurement accuracy on stable signals.

Query Response



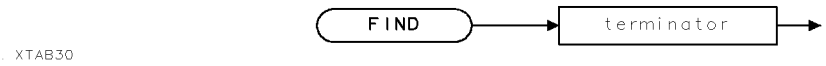
Query Example

```
OUTPUT @Mta;"TABL:FILT?"
ENTER @Mta;A$
```

FIND

Determines what signals are present.

Syntax



Command Example

```
OUTPUT @Mta;"TABL:FIND"
OUTPUT @Mta;"*OPC?"
ENTER @Mta;A$
```

Description

The FIND command is very similar to the AUTO command in the Top Level Command set. However, the FIND command recognizes the current setting of TABLE:INCLUDE whereas the AUTO command does not. (The AUTO command sets TABLE:INCLUDE to ON.) See the description for TABLE:INCLUDE to see how this variable affects the operation of the FIND command.

The FIND command performs a measurement function that automatically selects the input, vertical sensitivity, vertical offset, trigger level, and sweep speed for a display of an input signal. Generally it is a good idea to wait until the find-frequencies operation is finished before performing any other action (see *OPC or *WAI in the Common Command set).

The input selection operates as follows:

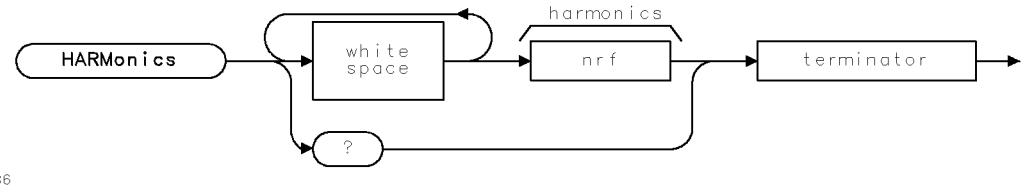
- If a signal is found on both inputs or no signal is found on either input, the input specified by the trigger source will be selected.
- If a signal is found on one input but not the other input, the input with the signal will be selected.

If the find-frequencies operation fails, an error is declared.

HARMonics

Sets the number of harmonics examined.

Syntax



Preset State: 4
Parameter Range: harmonics = 1 through 16

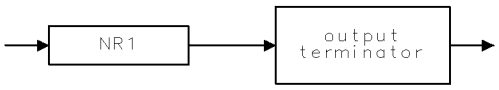
Command Example

```
OUTPUT @Mta;"TABL:HARM 4"
```

Description

The HARMONICS command sets the number of harmonics examined in the tabular mode of operation. The query returns the currently set value of harmonics.

Query Response



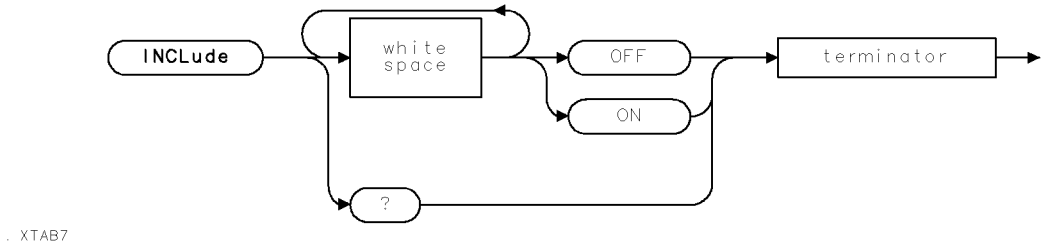
Query Example

```
OUTPUT @Mta;"TABL:HARM?"
ENTER @Mta;Harmonics
```

INCLude

Indicates whether the signals are to be treated as individual tones, or as fundamentals with harmonics.

Syntax



Preset State: ON

Command Example

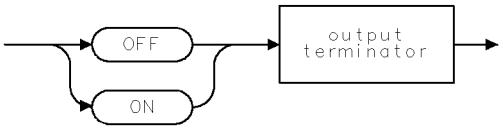
```
OUTPUT @Mta;"TABL:INCL ON"
```

Description

The INCLUDE command indicates whether the signals to be measured in the tabular mode are to be treated as individual tones, or as fundamentals with harmonics. If INCLUDE (harmonics) is ON, the signals are treated as fundamentals with harmonics in the tabular display. The number of displayed harmonics is set by the TABLE:HARMONICS command. If INCLUDE is OFF, the number of harmonics is ignored.

When the signal acquisition routine is invoked, either explicitly (see TABLE:FIND) or via signal track, the state of INCLUDE (harmonics) affects the resultant signal list. If INCLUDE is ON, harmonically related tones are not considered as separate signals, and only the fundamentals are returned in the signal list. If INCLUDE is OFF, there is no discrimination between harmonics and fundamentals.

Query Response



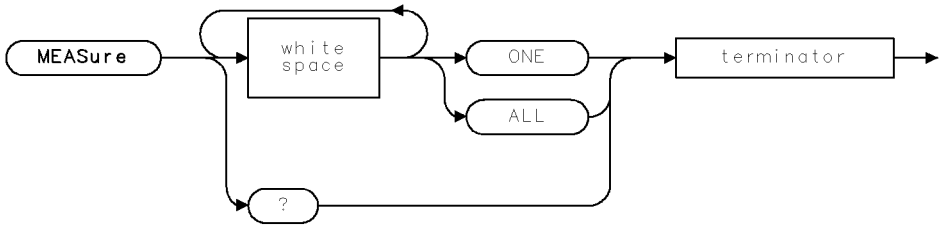
Query Example

```
OUTPUT @Mta;"TABL:INCL?"
ENTER @Mta;A$
```

MEASure

Performs a table measurement.

Syntax



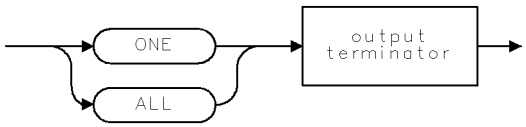
Command Example

```
OUTPUT @Mta;"TABL:MEAS ONE"
```

Description

The MEASURE command is used in the tabular measurement mode of operation to indicate whether the entire signal list should be measured or only the currently active signal. If MEASURE is ONE, only the active signal is measured. The query returns the current state of MEASURE.

Query Response



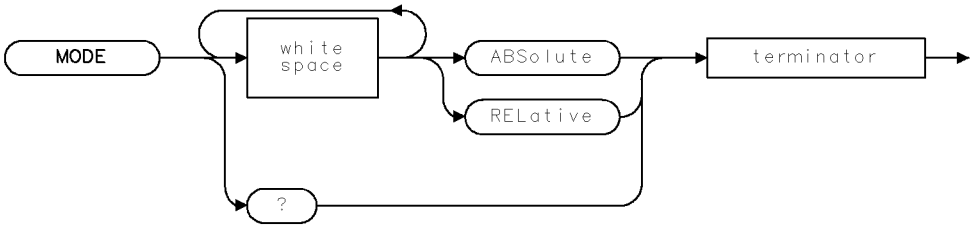
Query Example

```
OUTPUT @Mta;"TABL:MEAS?"
ENTER @Mta;A$
```

MODE

Sets the table to display ABSOLUTE or RELATIVE values.

Syntax



XTAB9

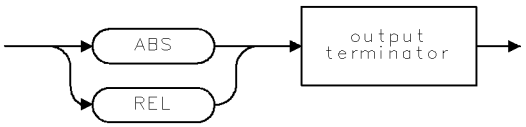
Command Example

```
OUTPUT @Mta;"TABL:MODE REL"
```

Description

The MODE command will set the table to display ABSOLUTE or RELATIVE values. In the absolute mode, fundamentals are measured in dBm and set to 0 degrees and the harmonics are measured relative to the fundamental (dBc and degrees). In the relative mode, the fundamental or harmonic specified by TABLE:REFERENCE is defined as 0 dB at 0 degrees and all other fundamentals and harmonics are measured relative to this. The query returns the current setting of MODE.

Query Response



QTAB9

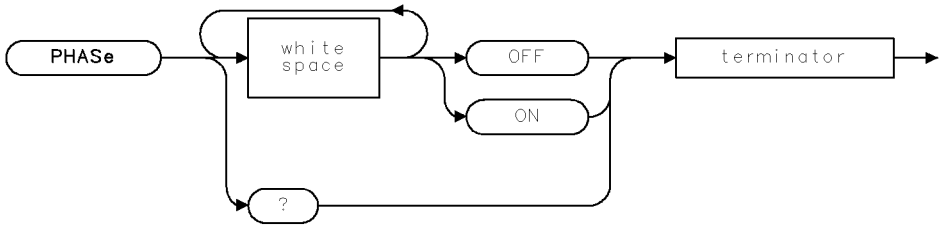
Query Example

```
OUTPUT @Mta;"TABL:MODE?"
ENTER @Mta;A$
```

PHASe

Turns the phase display on or off.

Syntax



Command Example

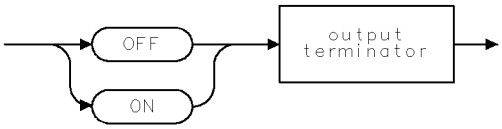
```
OUTPUT @Mta;"TABL:PHAS ON"
```

Description

The PHASE command allows the phase display to be turned on or off. This does not affect remote operation.

The query returns the current setting of PHASE.

Query Response



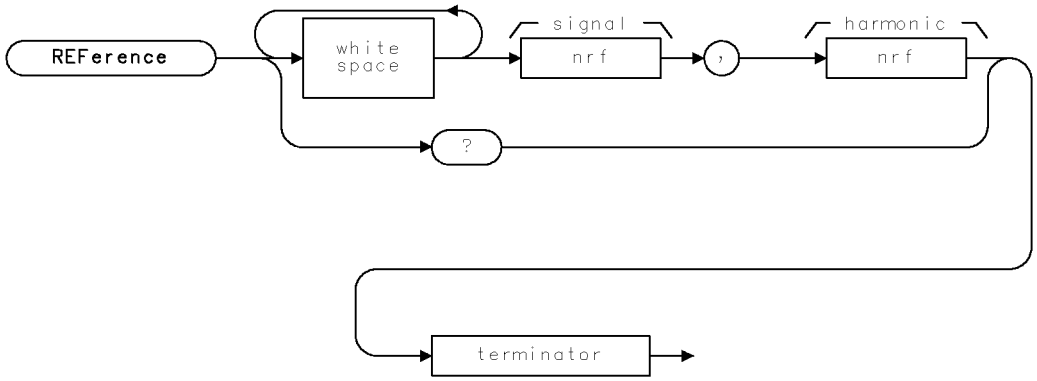
Query Example

```
OUTPUT @Mta;"TABL:PHAS?"
ENTER @Mta;A$
```

REFerence

Specifies which signal component to use as the reference signal.

Syntax



Preset State: signal = 1
harmonic = 1 (fundamental)

Parameter Range: signal = 1 through current number of known signals
harmonic = 1 through HARMONICS (see TABLe:HARMonics)

Command Example

```
OUTPUT @Mta;"TABL:REF 2,3"            ! signal 2, 3rd harmonic
```

Description

The REFERENCE command allows the user to specify which signal component to use as the reference signal when using the RELATIVE mode (see TABLE:MODE). The first parameter refers to the signal number and the second parameter refers to the harmonic number of that signal. The query returns the current state of REFERENCE.

Query Response

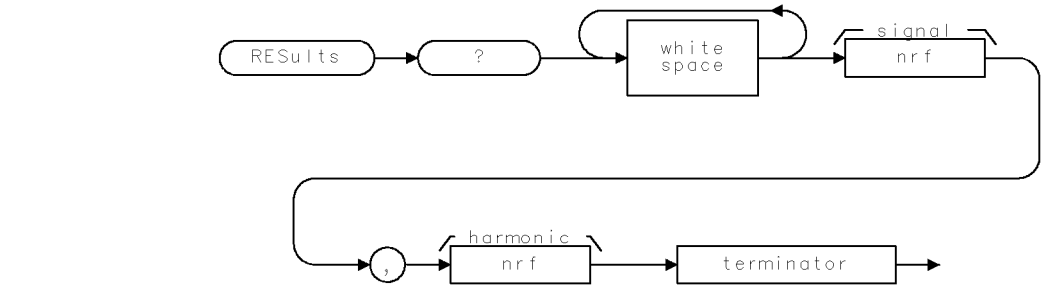
Query Example

```
OUTPUT @Mta;"TABL:REF?"
ENTER @Mta;Signal,Harmonic
```

RESults

Returns the frequency, amplitude, and phase of the specified signal component.

Syntax

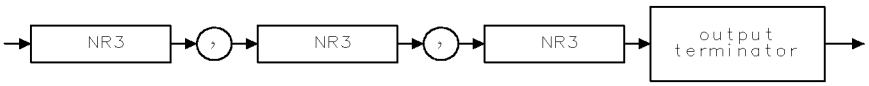


Parameter Range: signal = 1 through current number of known signals
harmonic = 1 through HARMONICS (see TABLe:HARMonics)

Description

The RESULTS query returns the frequency, amplitude, and phase of the specified signal component in the table. The first parameter refers to the signal number and the second parameter refers to the harmonic number of that signal.

Query Response



Query Example

```
OUTPUT @Mta;"TABL:RES? 2,3"
ENTER @Mta;Freq,Magn,Phase
```

SETReference

Copies the current table into a reference table.

Syntax

Command Example

```
OUTPUT @Mta;"TABL:SETR"
```

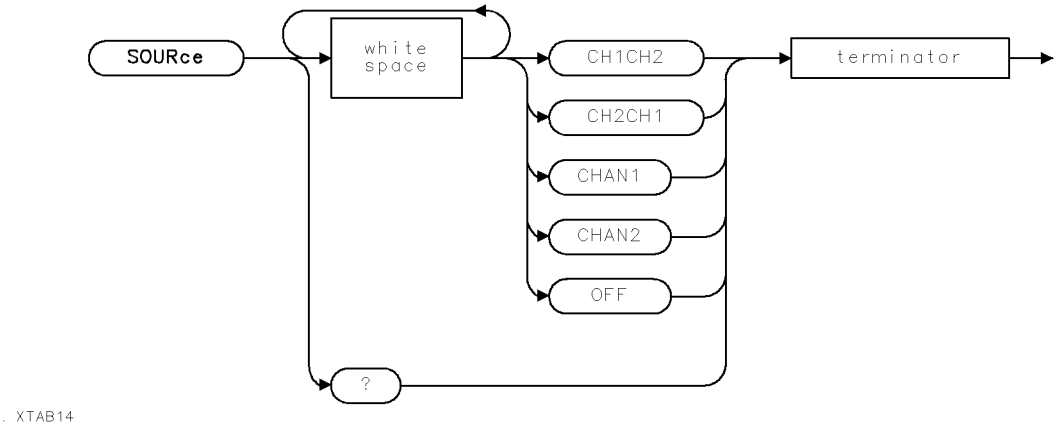
Description

The SETREFERENCE command is used in the tabular mode of operation to copy the current table into a reference table, which is then subtracted from subsequent table updates. Invoking the SETREFERENCE command sets the table delta to ON if it is not on already (see TABLE:DELTA).

SOURce

Specifies the input for table measurements.

Syntax



Preset State: OFF
Parameters: CHAN1 source is channel 1
CHAN2 source is channel 2
CH1CH2 source is channel 1 / channel 2
CH2CH1 source is channel 2 / channel 1
OFF source is off

Command Example

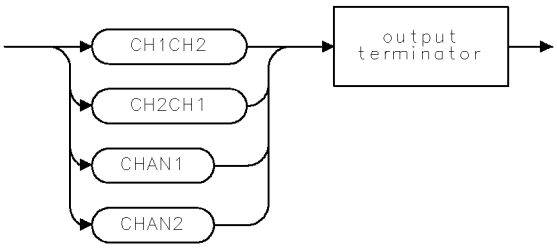
```
OUTPUT @Mta;"TABL:SOUR CH1CH2"
```

Description

The SOURCE command will specify the input channel(s) and operations to be used in table measurements. The choices are CHAN1, CHAN2, CHAN1 / CHAN2, CHAN2 / CHAN1 or OFF. Setting SOURCE to OFF exits the tabular mode of operation.

The query returns the current state of SOURCE.

Query Response



QTAB14

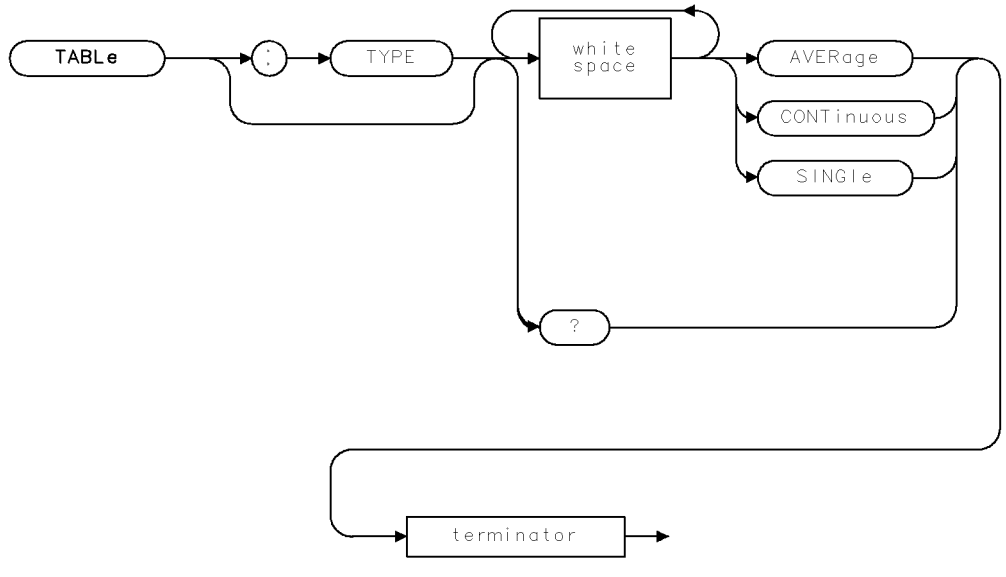
Query Example

```
OUTPUT @Mta;"TABL:SOUR?"
ENTER @Mta;A$
```

TYPE

Specifies tabular acquisition mode.

Syntax



Command Example

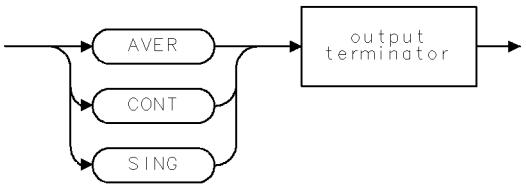
```
OUTPUT @Mta;"TABL:TYPE CONT"
```

Description

The TYPE command allows the user to specify the tabular acquisition mode. The choices are SINGLE, CONTINUOUS, and AVERAGE. If the table source is OFF (see TABLE:SOURCE) and a TYPE command is received, the source is set to the trigger source.

The query returns the current state of TYPE.

Query Response



. QTAB15

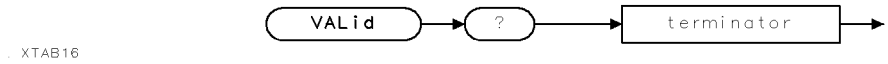
Query Example

```
OUTPUT @Mta;"TABL:TYPE?"
ENTER @Mta;A$
```

VALid

Determines valid data.

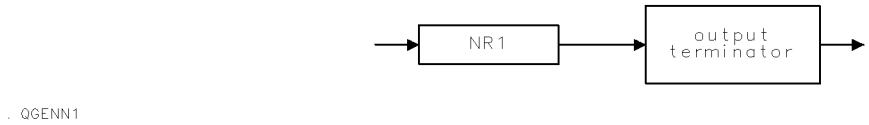
Syntax



Description

The VALID query is used to determine whether valid data is available in the frequency table. One (1) is returned if there is valid data. Zero (0) is returned if there is no valid data.

Query Response



Query Example

```
OUTPUT @Mta;"TABL:VAL?"
ENTER  @Mta;Valid
IF Valid=1 THEN
  PRINT "There is valid data"
ELSE
  PRINT "There is not valid data"
END IF
```

29

Trigger Subsystem

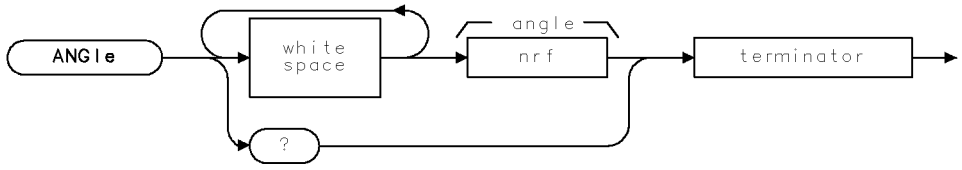
The TRIGGER subsystem commands are used to define the conditions for a trigger. See the following list for a command syntax summary of the TRIGGER subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:TRIGger:		
:ANGle	<nrf>	
:CENTer		[no query]
:FREQuency		
:STATe	AUTO|MANual	
[:VALue]	<nrf>[[<suffix multiplier>]HZ]	
HYSTeresis	<nrf>[PCT]	
:LEVel	<nrf>[DBM|[<suffix multiplier>]V]	
:POWer		
:STATe	AUTO|MANual	
[:VALue]	<nrf>[DBM|[<suffix multiplier>]DBM]	
:QUALifier	POSitive|NEGative	
:RELative		
:LEVel	<nrf>[PCT|DBC]	
[:STATe]	OFF|ON	
:SOURce	CHAN1|CHAN2|GATE1|GATE2|EXTernal	
:TYPE	EGDE|OFF|PHASe|PULSe	
:WIDTh	NARRow|WIDE	

ANGle

Sets the phase triggering angle.

Syntax



Preset State: 0
Parameter Range: angle = −360 through +360 degrees
(resolution = 1 degree)

Note Negative values will be "rotated" to positive values (for example, −45 becomes 315).

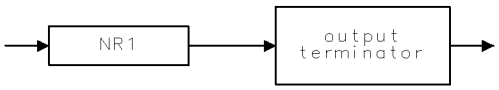
Command Example

```
OUTPUT @Mta;"TRIG:TYPE PHAS"
OUTPUT @Mta;"TRIG:ANG 45"
```

Description

The ANGLE command sets the angle used for phase triggering. This parameter is only used when TRIGGER:TYPE is set to PHASE. The query returns the current setting of ANGLE.

Query Response



Query Example

```
OUTPUT @Mta;"TRIG:ANG?"
ENTER @Mta;Angle
```

CENTer

Sets the trigger level to the midpoint.

Syntax

Command Example

```
OUTPUT @Mta;"TRIG:CENT"
```

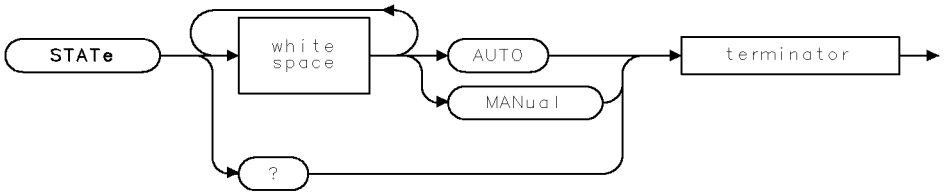
Description

The CENTER command sets the trigger level to the current midpoint of the vertical range for video triggering (which is equal to the offset defined by CHANx:OFFSET).

FREQuency:STATe

Determines how to choose the frequency to trigger a RF pulsed frequency sweep.

Syntax



Preset Value: MANual

Command Example

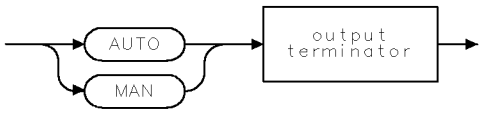
```
OUTPUT @Mta;"TRIG:FREQ:STAT AUTO"
```

Description

The FREQUENCY:STATE command determines how to choose the frequency to trigger a RF pulsed frequency sweep. The query returns the current value of FREQUENCY:STATE.

- AUTO: in this mode the trigger frequency is set to the center frequency of the span.
- MANUAL: n this mode the trigger frequency is set to the value of TRIGger: FREQuency[:VALue].

Query Response



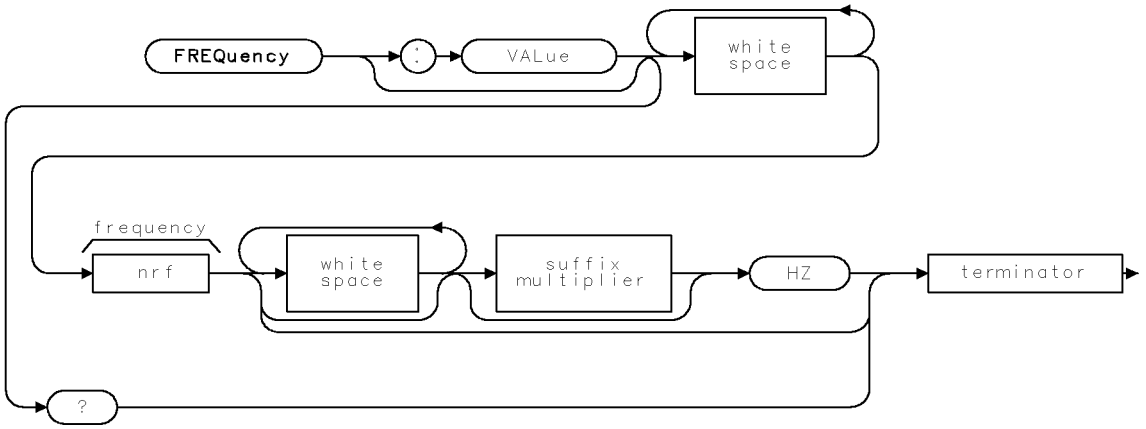
Query Example

```
OUTPUT @Mta;"TRIG:FREQ:STAT?"
ENTER @Mta;State$
```

FREQuency:VALue

Specifies the synthesizer frequency.

Syntax



Preset Value: 3.123456789 GHz
Parameter Range: 1 Hz through 40 GHz
Related Commands: TRIGger:FREQuency:STATe

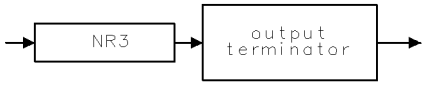
Command Example

```
OUTPUT @Mta;"TRIG:FREQ 1GHZ"
```

Description

The FREQUENCY:VALUE command specifies the synthesizer frequency to be used when triggering for stepped frequency sweeps with RF pulse input. This command will set the value of TRIGGER:FREQUENCY:STATE to MANUAL. The query returns the current setting of FREQUENCY:VALUE.

Query Response



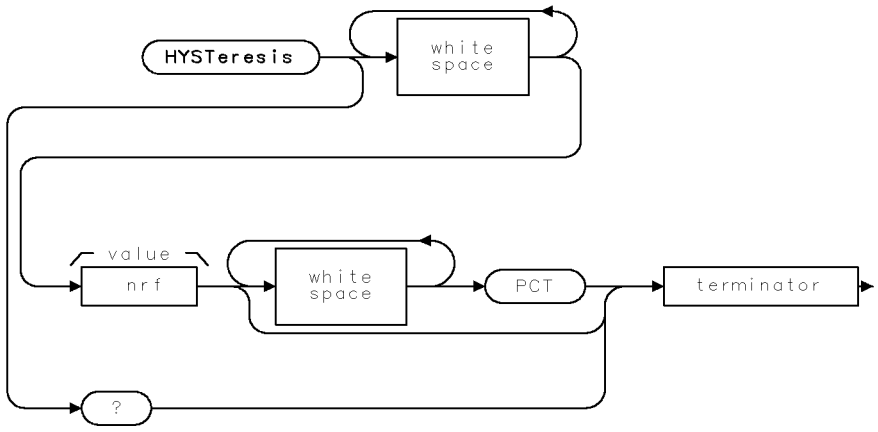
Query Example

```
OUTPUT @Mta;"TRIG:FREQ?"
ENTER @Mta;Frequency
```

HYSTeresis

Sets the value (in percent) of the trigger hysteresis for video triggering.

Syntax



Preset State: 10
Related Commands: CHANx:AUTorange
CHANx:RANGe
TRIGger:LEVel
Parameter Range: hysteresis = 1 through 100 percent

Command Example

```
OUTPUT @Mta;"TRIG:HYST 20 PCT"
```

Description

The HYSTERESIS command sets the value (in percent) of the trigger hysteresis for video triggering. The hysteresis is a percentage of the current hardware range of the trigger channel (CHANx:RANGE). Hysteresis is a method of making the triggering less sensitive to noise. For example, for positive-edge triggering the signal must go below the value of:

$$TRIGGER:LEVEL - \frac{CHANx:RANGE \times TRIGGER:HYSTERESIS}{100}$$

then above the value of:

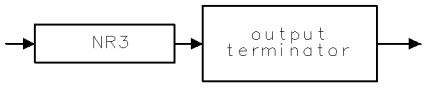
$$TRIGGER:LEVEL$$

to satisfy the trigger. The value of HYSTERESIS can be set so that noise on the signal cannot cause triggering.

The query returns the current value of HYSTERESIS in volts.

Note If hardware autoranging is ON, the actual voltage value of the trigger:hysteresis may change from sweep to sweep.

Query Response



QGENNR3

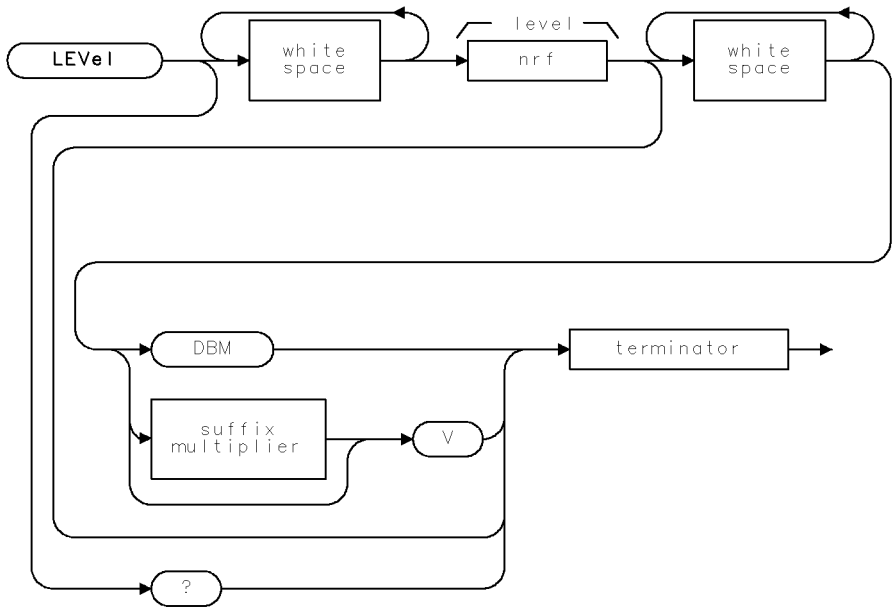
Query Example

```
OUTPUT @Mta;"TRIG:HYST?"
ENTER @Mta;Hysteresis
```

LEVel

Sets the value of the trigger level for video triggering.

Syntax



Preset State:	0
Parameter Range:	(OFFSET − RANGE/2) ≤ level (in volts) ≤ (OFFSET + RANGE/2)
Fundamental Unit:	volts or dBm
Related Commands:	TRIGger:RELative:STATe

Command Example

```
OUTPUT @Mta;"TRIG:TYPE EDGE"
OUTPUT @Mta;"TRIG:LEV 50 uV"
```

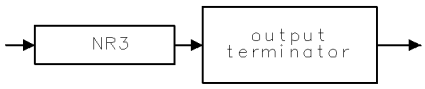
Description

The LEVEL command sets the value (in volts or dBm) of the trigger level for video triggering. The LEVEL command also sets TRIGGER:RELATIVE:STATE to OFF. The query returns the current value of LEVEL (always in volts).

When the trigger level is specified in dBm, it is internally converted to peak volts using the following equation:

$$sqrt(2 \times 50 \times 0.001 \times 10^{level/10})$$

Query Response



QGENNR3

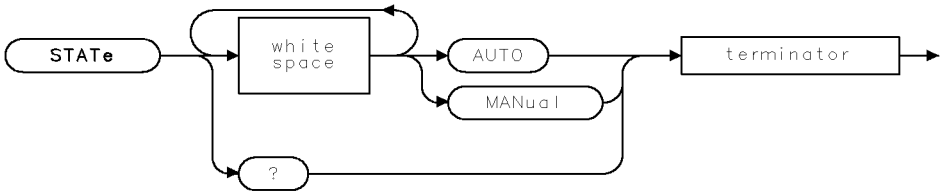
Query Example

```
OUTPUT @Mta;"TRIG:LEV?"
ENTER @Mta;Level
```

POWer:STATe

Determines how to choose the power to trigger a RF pulsed power sweep.

Syntax



Preset Value: MANual

Command Example

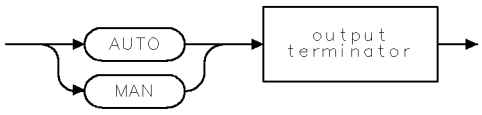
```
OUTPUT @Mta;"TRIG:POW:STAT AUTO"
```

Description

The POWER:STATE command determines how to choose the power to trigger a pulsed power sweep. The query returns the current value of POWER:STATE.

- AUTO: in this mode the trigger power is set to the center power of the span.
- MANUAL: in this mode the trigger power is set to the value of TRIGGER: POWER[:VALUE].

Query Response



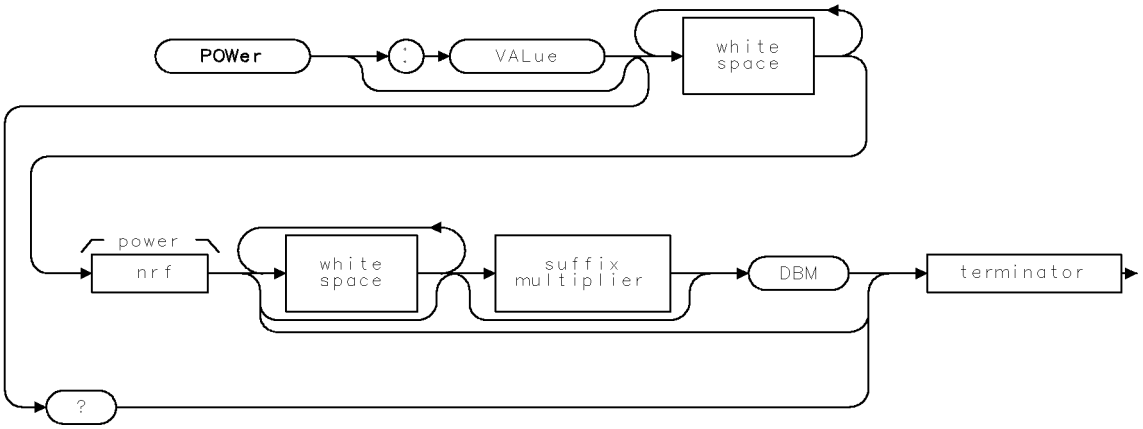
Query Example

```
OUTPUT @Mta;"TRIG:POW:STAT?"
ENTER @Mta;State$
```

POWer:VALue

Specifies the synthesizer power to be used when triggering for stepped power sweeps.

Syntax



Parameter Range: = −70 dBm through +10 dBm
Related Commands: TRIGger:POWer:STATe

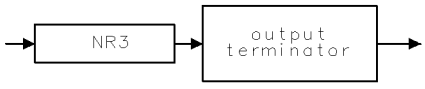
Command Example

```
OUTPUT @Mta;"TRIG:POW -10 dBm"
```

Description

The POWER:VALUE command specifes the synthesizer power to be used when triggering for stepped power sweeps with RF pulse input. This command will set the value of TRIGGER:POWER:STATE to MANUAL. The query returns the current setting of POWER:VALUE.

Query Response



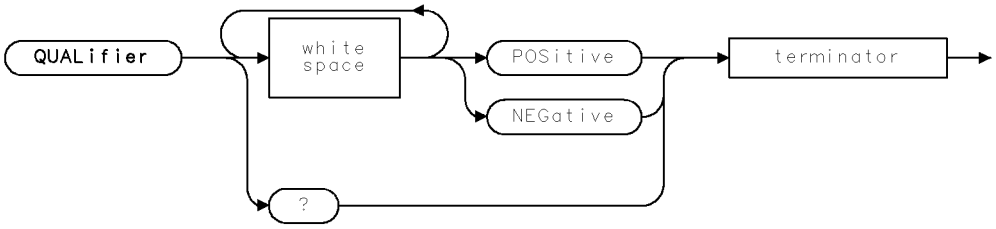
Query Example

```
OUTPUT @Mta;"TRIG:POW[:VALue]?"
ENTER @Mta;Power
```

QUALifier

Specifies the trigger qualifier used for video level triggering.

Syntax



Preset State: POSitive

Command Example

```
OUTPUT @Mta;"TRIG:QUAL POS"
```

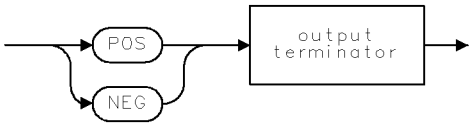
Description

The QUALIFIER command specifies the trigger qualifier used for video level triggering. The options are:

- **POSitive** edge selects the positive slope of the trigger source for triggering.
- **NEGative** edge selects the negative slope of the trigger source for triggering.

The query returns the current trigger qualifier status.

Query Response



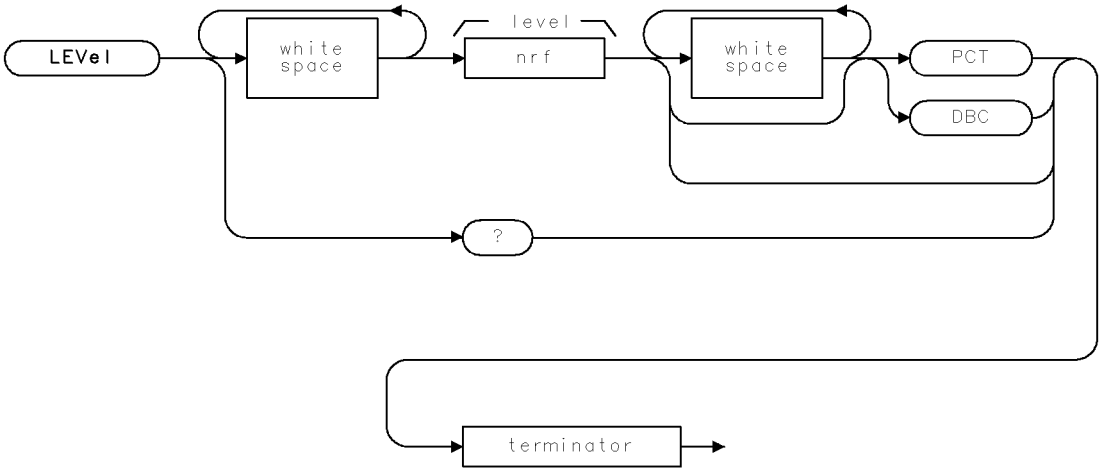
Query Example

```
OUTPUT @Mta;"TRIG:QUAL?"
ENTER @Mta;Qualifier$
```

RELative:LEVel

Specifies the trigger level as a percentage of the source amplitude.

Syntax



XTRI10

Preset State:	0 %
Parameter Range:	level = −200% through +200%;
	level = −100 dBc through 12.04 dBc
Fundamental Unit:	percent or dBc
Related Commands:	CHANx:AUTorange
	SOURce:POWer:LEVel
	TRIGger:LEVel
	TRIGger:RELative:STATe

Command Example

```
OUTPUT @Mta;"TRIG:REL:LEV 10%;STAT ON"
```

Description

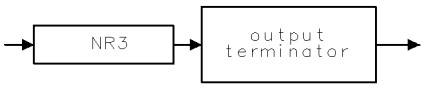
The RELATIVE:LEVEL command specifies the trigger level as a percentage (or dBc) of the source amplitude. The RELATIVE:LEVEL command also sets TRIGGER:RELATIVE:STATE to ON. For example, if the source amplitude is 100 mV (−10 dBm), RELATIVE:LEVEL is set to 50% (or −6 dBc), then the trigger level will be set to 50 mV. Changes in the source amplitude will couple to the trigger level as long as RELATIVE:STATE is ON.

The query returns the current state of RELATIVE:LEVEL (always in percent).

The operation of this command is influenced by the state of autoranging for the trigger channel. If autoranging is on, the channel range and offset will, at minimum, include the

trigger level. If autoranging is off, the value of the relative trigger level must be within the current measurement range.

Query Response



QGENNR3

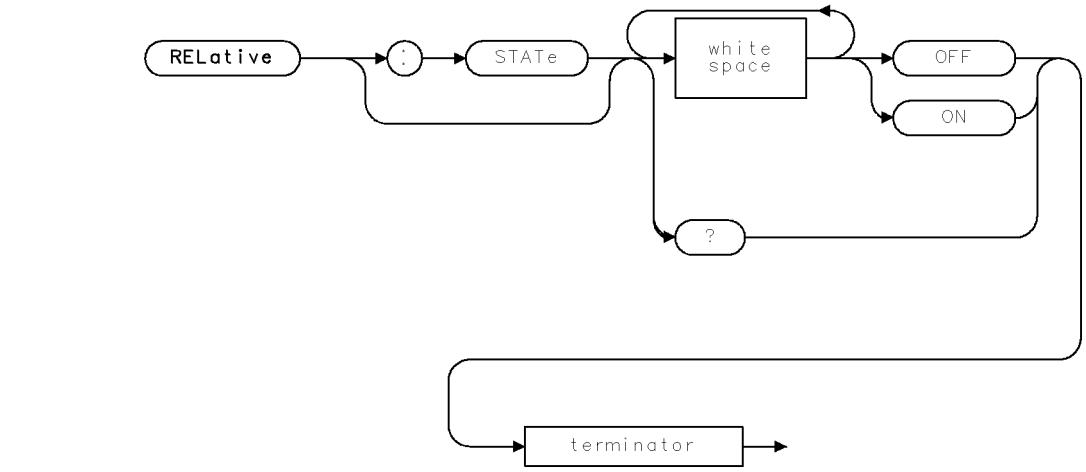
Query Example

```
OUTPUT @Mta;"TRIG:REL:LEV?"
ENTER @Mta;Level
```

RELative:STATe

Enables and disables the coupling between the source amplitude and trigger level.

Syntax



Preset State: OFF
Related Commands: SOURce:POWer:LEVel
TRIGger:LEVel
TRIGger:RELative:LEVel
TRIGger:SOURce
TRIGger:TYPE

Command Example

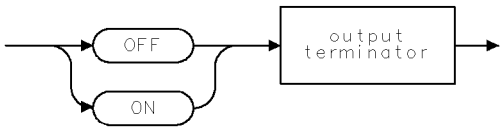
```
OUTPUT @Mta;"TRIG:REL:LEV 10%;STAT ON"
```

Description

The RELATIVE:STATE command enables and disables the coupling between the source amplitude and trigger level. The query returns the current state of RELATIVE:STATE.

Note that this mode is only applicable if TRIGGER:TYPE is EDGE or PULSE and TRIGGER:SOURCE is CHAN1, CHAN2, GATE1, or GATE2.

Query Response



QGENON

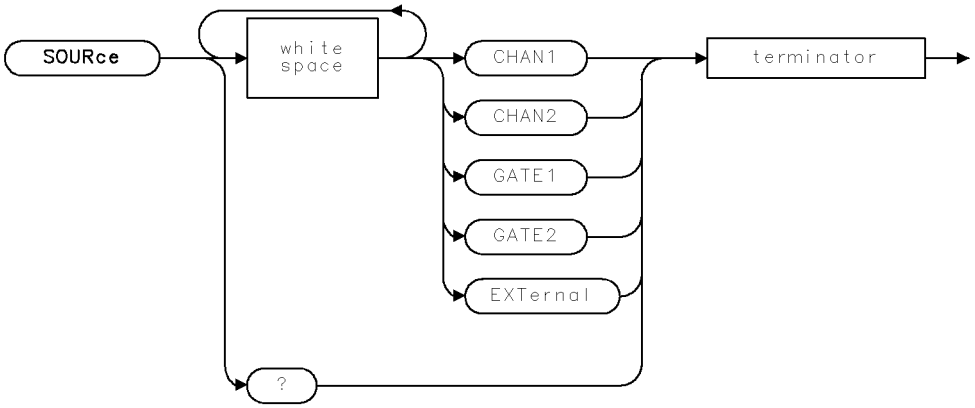
Query Example

```
OUTPUT @Mta;"TRIG:REL:STAT?"
ENTER @Mta;State$
```

SOURce

Specifies the trigger source.

Syntax



Preset State: CHAN1

Command Example

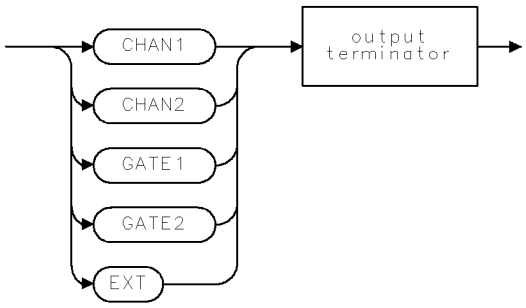
```
OUTPUT @Mta;"TRIG:SOUR GATE1"
```

Description

The SOURCE command specifies the trigger source. The query returns the current trigger source status. The options are:

- **CHAN1** selects channel 1 as the trigger source.
- **CHAN2** selects channel 2 as the trigger source.
- **GATE1** selects channel 1 qualified by the sync input as the trigger source.
- **GATE2** selects channel 2 qualified by the sync input as the trigger source.
- **EXT** selects the sync input as the trigger source.

Query Response



QTRI7

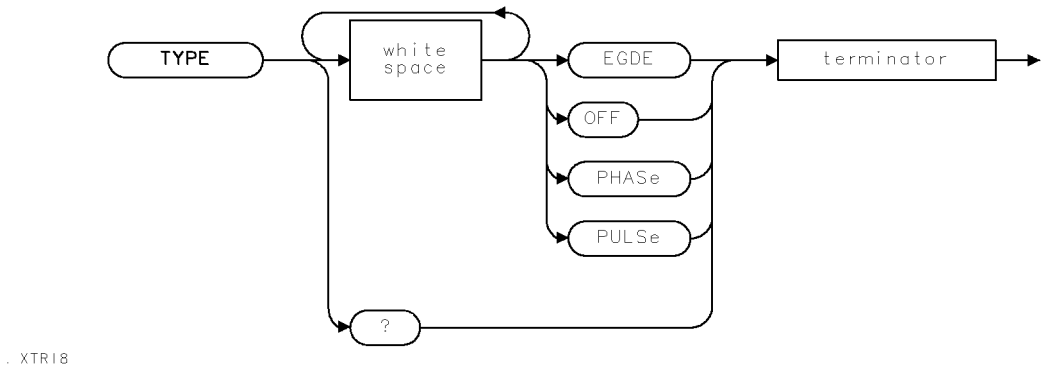
Query Example

```
OUTPUT @Mta;"TRIG:SOUR?"
ENTER @Mta;Source$
```

TYPE

Selects the triggering mode.

Syntax



Preset State: EDGE

Command Example

```
OUTPUT @Mta;"TRIG:TYPE PHAS"
```

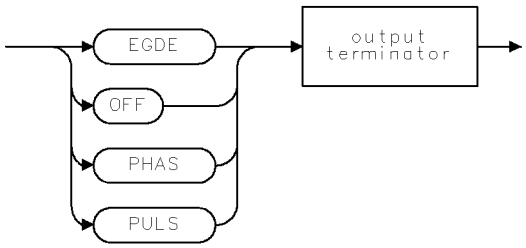
Description

The TYPE command selects the triggering mode. The options are:

- **OFF** No triggering is done (that is, free run).
- **EDGE** Video level triggering is done. The hysteresis on the trigger level is approximately 10 percent.
- **PHASE** The waveform is positioned so the phase of the fundamental of the signal is constant.
- **PULSE** AM (amplitude demodulation) is performed on the signal and video level triggering is performed on the result. This mode is only available for signals less than 10 MHz.

The query returns the current state of TYPE.

Query Response



. QTRI8

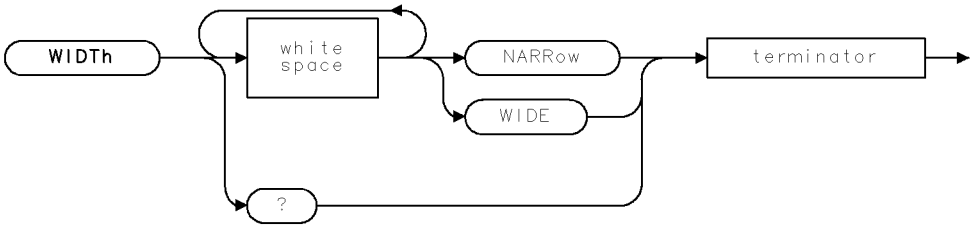
Query Example

```
OUTPUT @Mta;"TRIG:TYPE?"
ENTER @Mta;Type$
```

WIDTh

Specifies what type of trigger search to perform.

Syntax



Preset State: WIDE

Command Example

```
OUTPUT @Mta;"TRIG:WIDT NARR"
```

Description

The WIDTH command specifies what type of trigger search to perform for certain types of measurements (see below).

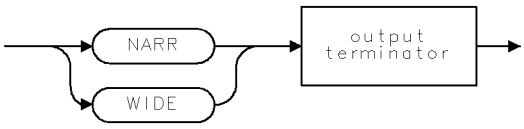
- **NARROW** The trigger search is performed with approximately 1 ns resolution.
- **WIDE** The trigger search is performed with approximately 50 ns resolution.

This command only affects the following measurements:

- pulsed power and frequency sweeps
- repetitive time sweeps with input frequency <10 MHz

The query returns the current state of WIDTH.

Query Response



Query Example

```
OUTPUT @Mta;"TRIG:WIDT?"
ENTER @Mta;Width$
```

30

Waveform Subsystem

The WAVEFORM subsystem in the microwave transition analyzer is used to send trace data to the user as well as determine various trace parameters such as trace size, time per point, voltage range, and so on. In some instances, notably using the HP 70820A as an array processor, the waveform subsystem is also used to receive trace data and the associated parameters from the user.

The waveform subsystem generally refers to a single trace at a time. That trace is determined by the WAVEFORM:SOURCE command.

Many of the commands allow the setting of trace parameters. This should only be done as part of the array processor mode since arbitrarily changing these parameters can "confuse" the HP 70820A. For example, telling the HP 70820A that there are 1024 points in a trace when there are only 512 will cause various functions to "misbehave". Also, for the traces CHAN1, CHAN2, FUNC1, FUNC2, FUNC3, and FUNC4 most of these parameters are overwritten when ever a sweep is performed. With the exception of the WAVEFORM:DOMAIN and WAVEFORM:MAGNITUDE commands, there is generally no reason to set any trace parameters except in the array processor mode.

When setting trace parameters, many parameters depend on others. Because of this, it is generally a good idea to send the independent parameters first. For example: WAVEFORM:DTYPE ... WAVEFORM:POINTS ... followed by all other parameters.

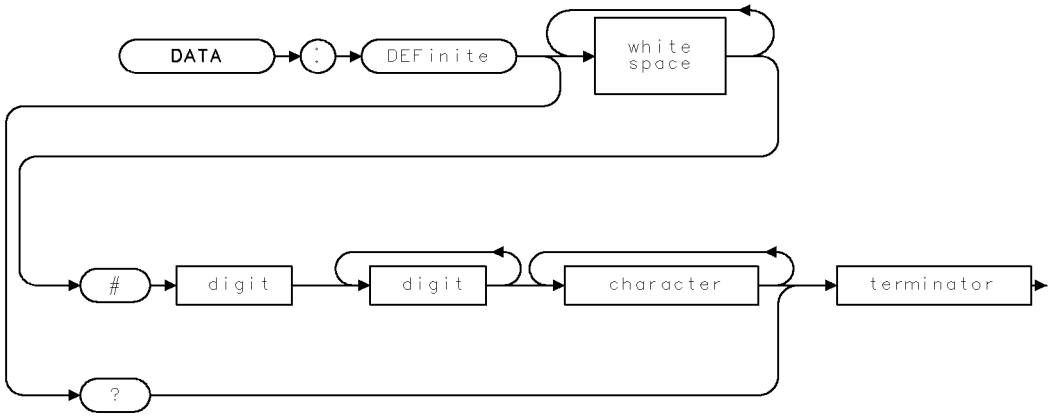
See the following list for a command syntax summary of the WAVEFORM subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:WAVeform		
:DATA		
:DEFinite	#<digit>{<digit>}{<character>}	
[:INDefinite]	#<digit>{<digit>}{<character>}	
:ASCii	<nrf>[{,<nrf>}]	
:DELay	<nrf>[[<suffix multiplier>]S]	
:DOMain	REAL,OFF|IMAGinary,OFF|COMPlex,OFF| MAGNitude,OFF|PHASe,OFF| POLar,OFF| OFF,REAL|OFF,IMAGinary|OFF,COMPlex| OFF,MAGNitude|OFF,PHASe|OFF,POLar)	
:DTYPe	CFReq|CTIMe|RFReq|RTIMe	
:FINCrement	<nrf>[[<suffix multiplier>]HZ]	
:FMODe	OFF|ON	
:FORigin	<nrf>[[<suffix multiplier>]HZ]	
:FORMat	INT16|INT32	
:FRANge	<nrf>[[<suffix multiplier>]HZ]	
:FSHift	<nrf>[[<suffix multiplier>]HZ]	
:INTerleave	OFF|ON	
:MAGNitude	LINear|LOGarithmic,LINear|LOGarithmic	
:PINCrement	<nrf>[DB]	
:POINts	<nrf>	
:PORigin	<nrf>[DBM]	
:PRANge	<nrf>[DB]	
:PREamble	{<character>}	
:PSLope	<nrf>[DEG]	
:PSPectrum	OFF|ON	
:RLEVel	<nrf>[DBM]	
:SOURce	CHAN<channel_number>|WMEM<memory_number> |FUNC<function_number>	
:TINCrement	<nrf>[[<suffix multiplier>]S]	
:TORigin	<nrf>[[<suffix multiplier>]S]	
:TRANge	<nrf>[[<suffix multiplier>]S]	
:TREFerence?		[query only]
:TYPE	AVERage|FAVerage|HOLD|MAXhold |MINhold|NORMal	
:UNITs	DEGrees|HERTz|NONE|SEConds|VOLTs|WATTs, DEGrees|HERTz|NONE|SEConds|VOLTs|WATTs	
:VALid?		[query only]
:VINCrement	<nrf>[[<suffix multiplier>]V]	
:VORigin	<nrf>[[<suffix multiplier>]V]	
:VRANge	<nrf>[[<suffix multiplier>]V]	
:YBOTtom?		[query only]
:YREFerence?		[query only]
:YTOP?		[query only]
:ZPHase		[no query]

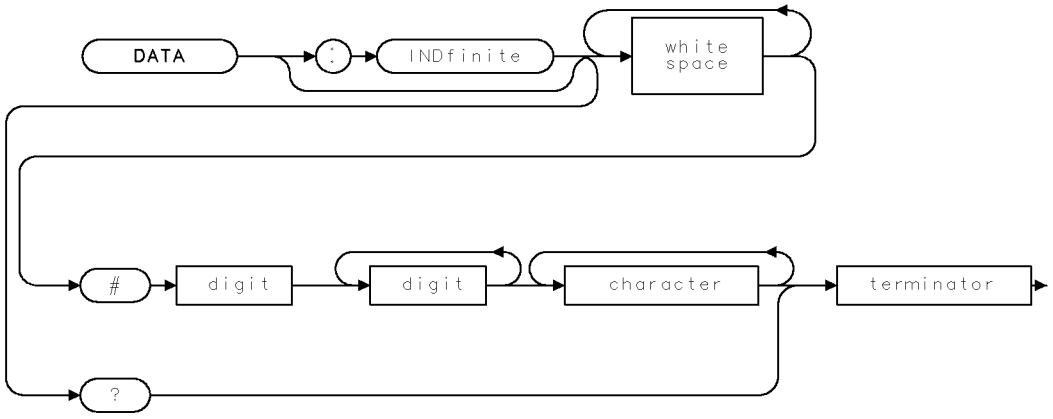
DATA

Sends or receives trace data.

Syntax



xwav1a



xwav1b

Command Example

See the query example below.

Description

The DATA command is used to send or receive trace data.

The interpretation of the data depends on various parameters that can be determined from other WAVEFORM commands. A quick summary is given below:

- **WAVeform:DOMain** determines the domain (for example, amplitude vs time, amplitude vs frequency, amplitude and phase vs frequency ...) for the trace data.

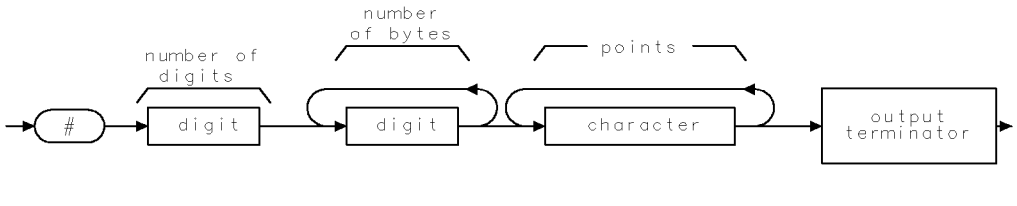
- **WAVeform:FORMat** determines the numeric format of the data transfer. Currently only 16- and 32-bit binary are supported.
- **WAVeform:INTerleave** determines whether complex data will be output as interleaved values (that is, magn1,phase1,magn2,phase2 ...) or as blocks of data (that is, magn1..magnN,phase1..phaseN).
- **WAVeform:POINts** determines the number of points that will be transferred.

Note A complex point (for example, magnitude and phase) counts as a single point.

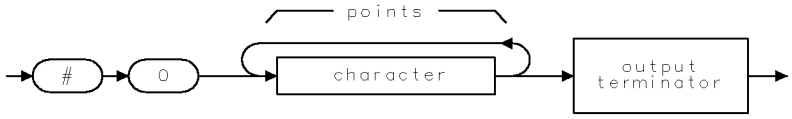
- When the returned data is in dB, the scaling is 0.01 dB per count for 16-bit data, 0.0001 dB per count for 32-bit data and the offset is determined by WAVEFORM:RLEVEL.
- When the returned data is a phase, the scaling is 0.01 degrees per count for 16-bit data, 0.0001 for 32-bit data.
- When the returned data is not in dB and not in phase, the resolution is determined by WAVEFORM:VINCREMENT.
- When the returned data is not in dB and not in phase, an offset is determined by WAVEFORM:VORIGIN.

Query Response

DEFinite



INDefinite



The query response is defined as

```
<header><point1>...<pointn><term>
```

<header> = #<length><number of bytes> for DEFINITE

= #0 for INDEFINITE

<length> = an ASCII digit indicating how many digits are in <N>

<number of bytes>	= an ASCII number indicating the number of bytes (8 bit) values of data following

<point>	if WAVeform:FORMat = INT16 (16 bit binary value):
	HP-IB: MSB first, LSB last
	MSIB: sent in one 16 bit packet
	if WAVeform:FORMat = INT32 (32 bit binary value):
	HP-IB: MSB first, LSB last
	MSIB: sent in two 16 bit packets; MSW first, LSW last

<term>	HP-IB: line feed with EOI asserted MSIB: END immediate command

Query Example

Data input example

```
OUTPUT @Mta;"WAV:SOUR CHAN2"  ! Ask for trace data from CHAN2
OUTPUT @Mta;"WAV:POIN?"
ENTER @Mta;N
OUTPUT @Mta;"WAV:DATA?"
ENTER @Mta USING "#,2A";A$
IF A$<>"#0" THEN
  PRINT "Header error"
  PAUSE
END IF
FOR J=0 TO N-1
  ENTER @Mta USING "#,W";Values(J)
NEXT J
ENTER @Mta USING "#,B";Linefeed
IF Linefeed<> 10 THEN
  PRINT "Terminator error"
  PAUSE
END IF
```

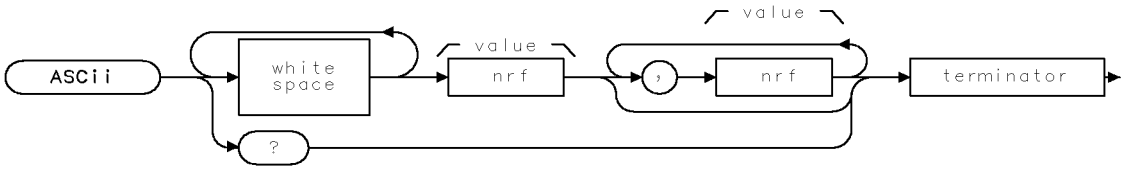
Data output example

```
Linefeed = 10
N = 256
Header$=VAL$(2*N)                    ! Convert words to bytes
Header$=VAL$(LEN(Header$))&Header$
OUTPUT @Mta;"WAV:SOUR WMEM1"         ! Download data into mem 1
OUTPUT @Mta USING "#,K,K";"WAV:DATA #",Header$
FOR J=0 TO N-1
  OUTPUT @Mta USING "#,W";Values(J)
NEXT J
OUTPUT @Mta USING "#,B";10 END       ! Send linefeed with EOI
asserted
```

DATA:ASCii

Loads the trace.

Syntax



Command Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:DTYP RTIM"
OUTPUT @Mta;"WAV:TOR 0;TINC 1 ns"
N=100
ALLOCATE Wmem1(N-1)
FOR J=0 TO N-1
  Wmem1(J)=SIN(2*PI*5*J/N)
NEXT J
OUTPUT @Mta;"WAV:DATA:ASCII ";Wmem1(*)
DEALLOCATE Wmem1(*)
```

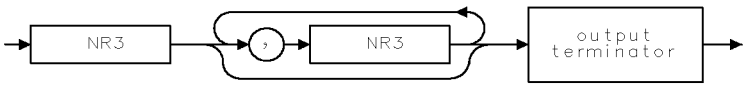
Description

The DATA:ASCII command loads the trace specified by WAVEFORM:SOURCE. The input is ASCII numeric (for example, 1.2,3.4,5.6 ...). The data will be interpreted according to the current settings of WAVEFORM:DTYPE (that is, real, imaginary or complex), WAVEFORM:FORMAT (that is, 16 or 32 bits) and WAVEFORM:INTERLEAVE.

The DATA:ASCII query returns the trace data as ASCII numbers. The data will be formatting according to the current settings of WAVEFORM:DOMAIN, WAVEFORM:MAGNITUDE and WAVEFORM:INTERLEAVE.

When using DATA:ASCII, the pertinent values of VORIGIN, VINCREMENT, FORMAT, and RLEVEL (that is, the vertical axis) are already incorporated into the data. When appropriate, you must still set or read the values for the horizontal axis (for example, TORIGIN, TINCREMENT, FORIGIN ...).

Query Response



qgenn7

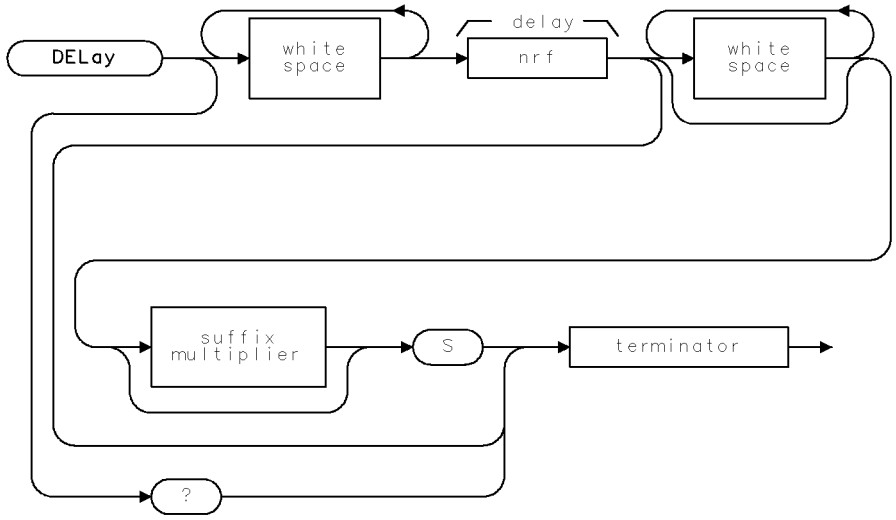
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:DOM REAL,OFF;MAGN LIN,LOG"
OUTPUT @Mta;"WAV:POIN?"
ENTER @Mta;N
ALLOCATE Wmem1(N-1)
OUTPUT @Mta;"WAV:DATA:ASCII?"
ENTER @Mta;Wmem1(*)
PRINT "The first point is:";Wmem1(0)
DEALLOCATE Wmem1(*)
```

DELay

Specifies the phase slope to apply to the trace.

Syntax



Preset State:	0
Parameter Range:	delay = −1.0E12 through +1.0E12
Fundamental Unit:	seconds
Related Commands:	WAVeform:FSHift
	WAVeform:PSLope

Command Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:DEL 5 ns"
```

Description

The DELAY command specifies the phase slope to apply to the trace specified by the WAVEFORM:SOURCE command in terms of time. The query returns the current setting.

The WAVEFORM:DELAY, WAVEFORM:FSHIFT, and WAVEFORM:PSLOPE commands all affect the same parameter. When the trace data has a horizontal axis of time, the phase slope should be set by WAVEFORM:FSHIFT and is interpreted as a frequency shift and the parameter is scaled by:

$$phase = 360 \times phase\ slope \times time\ range$$

Also, when the horizontal axis is time, an optional frequency offset is applied to *slope* depending on the trace parameters. If the start frequency of the trace is nonzero (this can occur when in the translate mode, see SWEEP:TIME:TRANSLATE) and *start frequency +*

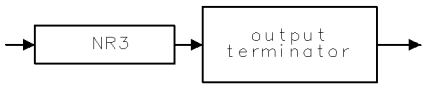
slope is less than 0, then the value used for *slope* is *start frequency + slope*. This mode is mainly intended for use with pulsed RF signals when it is desired to shift the carrier to bucket 0. This mode can be enabled or disabled by the WAVEFORM:FMODE command.

When the trace data has a horizontal axis of frequency, the phase slope should be set by WAVEFORM:DELAY and is interpreted as a time delay and the parameter is scaled by:

$$phase = 360 \times phase\ slope \times (stop\ frequency - start\ frequency)$$

When the trace data has a horizontal axis of neither time nor frequency, the phase slope should be set by WAVEFORM:PSLOPE and not be scaled.

Query Response



QGENNR3

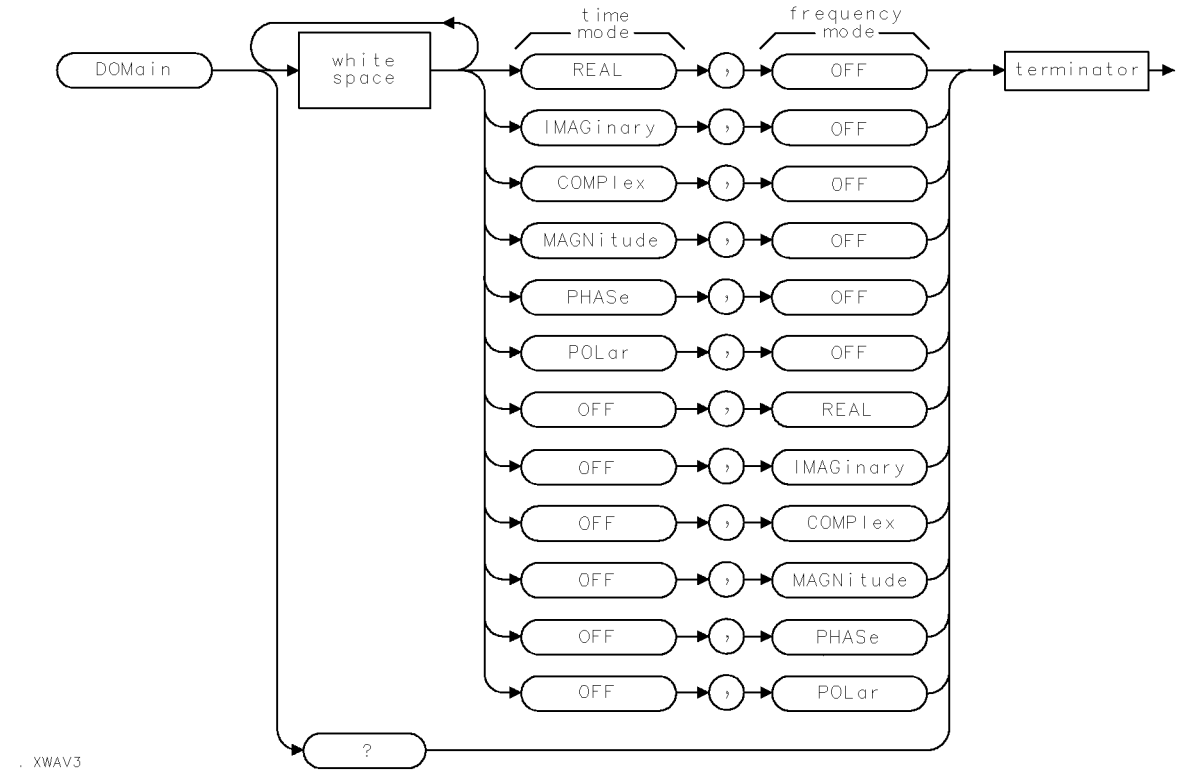
Query Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:DEL?"
ENTER @Mta;Delay
```

DOMain

Specifies how the trace data is to be displayed or output.

Syntax



Preset State: REAL,OFF (see SWEep:TYPE)

Command Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:DOM OFF,POL"
```

Description

This command specifies how the trace data is to be displayed or output. The query returns the current setting.

The valid combinations are:

- REAL,OFF..... use only the real (R) data array of the time domain.
- IMAG,OFF..... use only the imaginary (I) data array of the time domain.
- COMPLEX,OFF.. use both the real and imaginary data arrays of the time domain
- MAGN,OFF..... use the magnitude ($=\sqrt{R+I\times I}$) of the time domain.

- PHASE,OFF.... use the phase (= arctan(I/R)) of the time domain.
- POLAR,OFF.... use both the magnitude and phase of the time domain.
- OFF,REAL..... use only the real (R) data array of the frequency domain.
- OFF,IMAG..... use only the imaginary (I) data array of the frequency domain.
- OFF,COMPLEX.. use both the real and imaginary data arrays of the frequency domain.
- OFF,MAGN..... use the magnitude of the frequency domain.
- OFF,PHASE.... use the phase of the frequency domain.
- OFF,POLAR.... use both the magnitude and phase of the frequency domain.

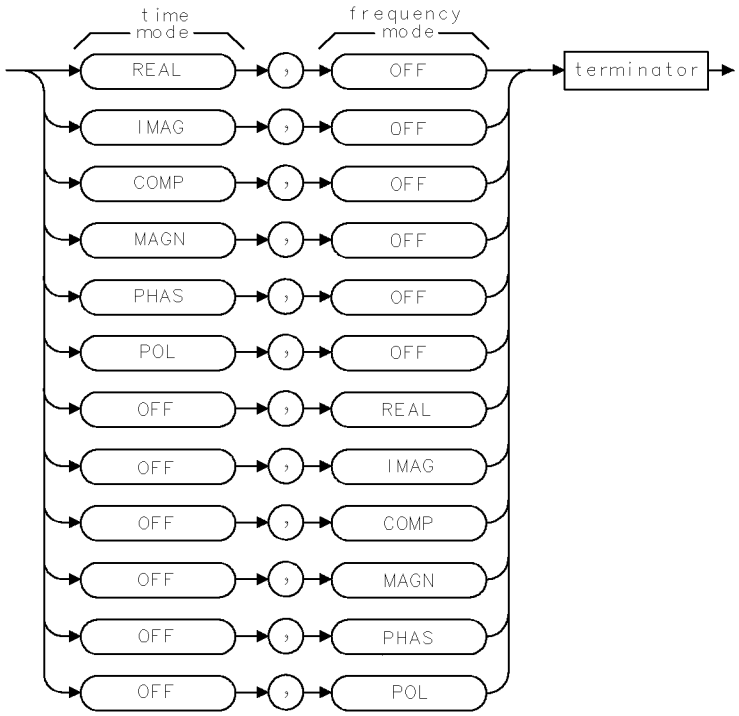
When COMPLEX or POLAR is selected and trace data is output, the WAVEFORM:INTERLEAVE command determines whether the data is sent as alternating pairs (for example, real1,imag1,..,realN,imagN) or as blocks of data (for example, real1,..realN, imag1..imagN)

This command allows the user to perform FFTs or IFFTs on any trace in the HP 70820A. If a trace type is time (see WAVEFORM:DTYPE) a forward FFT can be done by specifying the DOMAIN to be OFF,MAGN. If a trace type is freq (see WAVEFORM:DTYPE) an inverse FFT can be done by specifying the DOMAIN to be REAL,OFF.

The parameter time mode refers to the trace data in the time domain. If the data is not already in the time domain, an inverse FFT will be performed.

The parameter frequency mode refers to the trace data in the frequency domain. If the data is not already in the frequency domain, an FFT will be performed.

Query Response



. QWAV3

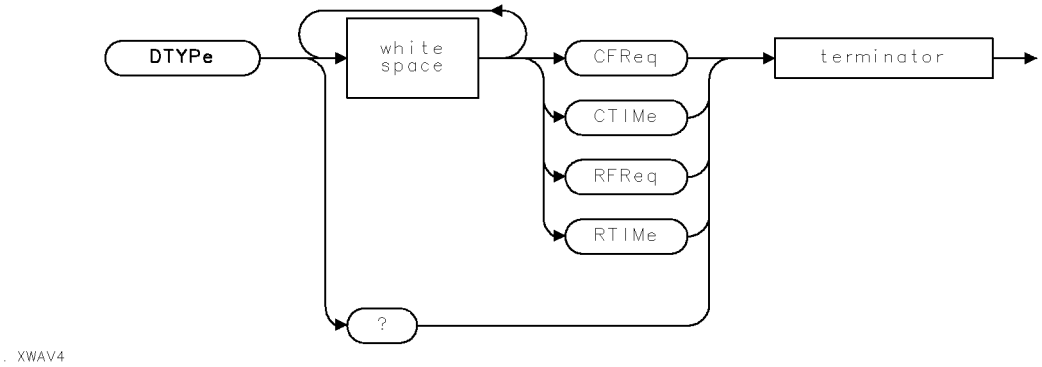
Query Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:DOM?"
ENTER @Mta;Domain$
```

DTYPe

Sets or determines the internal data format for a trace.

Syntax



Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:DTYP RTIM"
```

Description

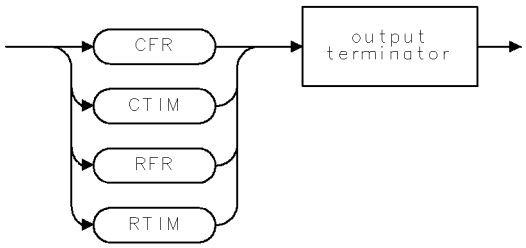
This command is used to set or determine the internal data format for a trace.

The various options are listed below:

- **CFReq:** The data is in the frequency domain and both positive and negative frequencies are explicitly stored. The format is the most negative frequency first and the most positive frequency last. The positive frequencies are points 0 through N/2. The negative frequencies are points N/(2 + 1) through N − 1. Both real and imaginary arrays are required.
- **CTIMe:** The data is in the time domain and is composed of both a real and imaginary array.
- **RFReq:** The data is in the frequency domain and only the positive frequencies (but both real and imaginary arrays) are stored. When required, the negative frequencies are generated assuming even symmetry for the real array and odd symmetry for the imaginary array.
- **RTIMe:** The data is in the time domain and is composed of only a real array. The imaginary array is assumed to be 0. Because data from frequency sweeps cn have negative start frquencies, the data for these sweeps is CFReq rather than RFReq.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data when using the HP 70820A as an array processor.

Query Response



QWAV4

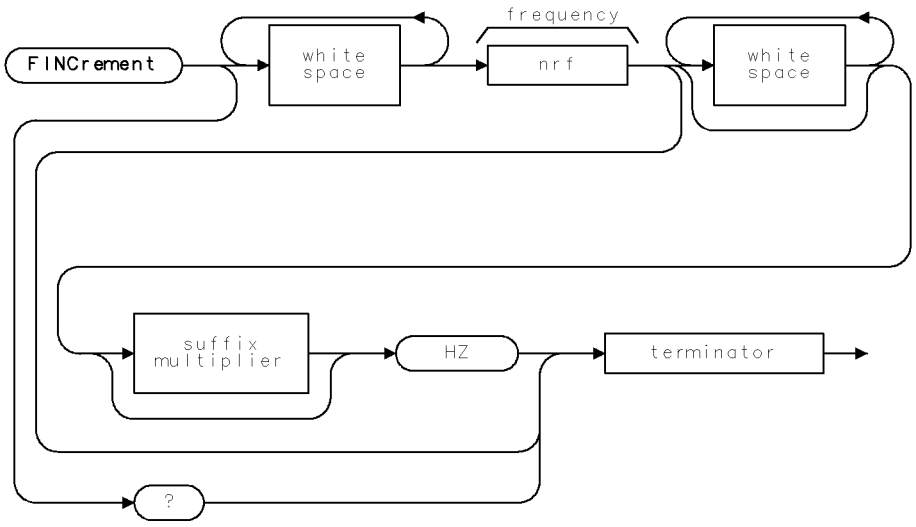
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:DTYP?"
ENTER @Mta;Dtype$
```

FINCrement (Frequency Increment)

Sets or determines the frequency increment between adjacent points.

Syntax



Parameter Range: abs((length − 1)frequency − FORigin) ≤ 1.0E20 Hz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:FINC 1KHz"
```

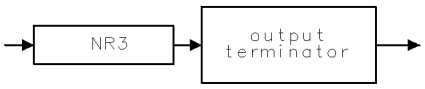
Description

The FINCREMENT command sets or determines the frequency increment between adjacent points in the frequency domain. The frequency increment is related to the sampling frequency by: frequency increment = sampling frequency/length, where length is the number of frequency domain points (including both positive and negative frequencies).

This sets the same parameter as the WAVEFORM:FRANGE command.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. The interpretation of this command depends on WAVEFORM:POINTS and WAVEFORM:DTYPE

Query Response



QGENNR3

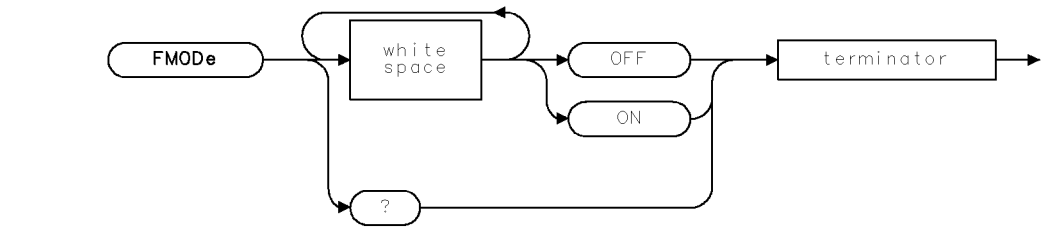
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"    ! Determine the frequency delta
OUTPUT @Mta;"WAV:FINC?"         ! for memory 1
ENTER  @Mta;Finc
```

FMODe

Enables or disables a frequency offset when a frequency shift of trace data is done.

Syntax



Preset State: ON
Related Commands: WAVeform:DELay
WAVeform:FSHift
WAVeform:PSLope

Command Example

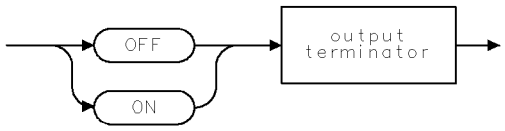
```
OUTPUT @Mta;"WAV:FMOD OFF"
```

Description

The FMODE command enables or disables a frequency offset when a frequency shift of trace data is done. See WAVEFORM:FSHIFT for a more complete description of how this frequency offset is applied.

The query returns the current state of FMODE.

Query Response



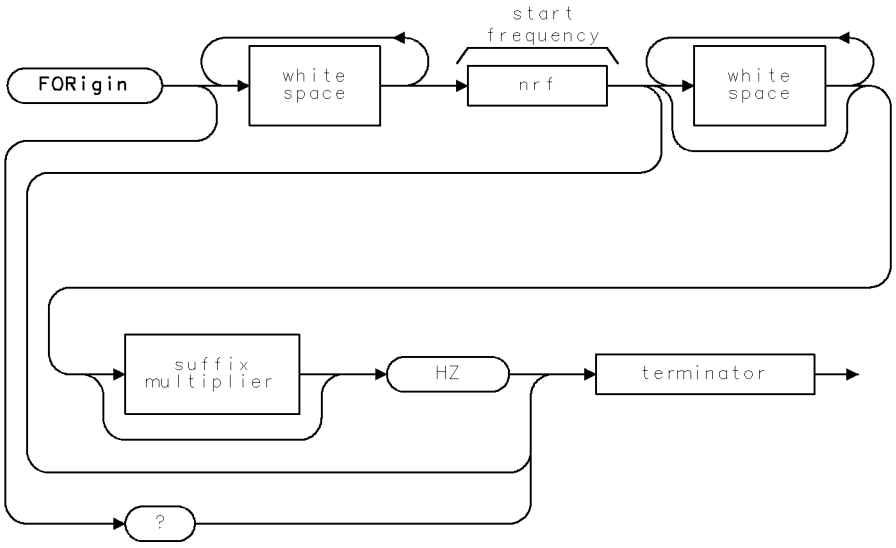
Query Example

```
OUTPUT @Mta;"WAV:FMOD?"
ENTER  @Mta;State$
```

FORigin

Sets or determines the starting frequency of the data in the frequency domain.

Syntax



Parameter Range: start frequency = 0 through 1.0E20 Hz
Fundamental Unit: hertz

Command Example

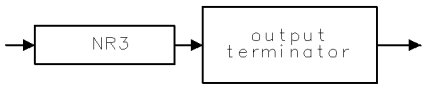
```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:FOR 0"
```

Description

This command sets or determines the starting frequency of the data in the frequency domain.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. To set the start frequency for data acquisition use SWEEP:FREQ:START.

Query Response



QGENNR3

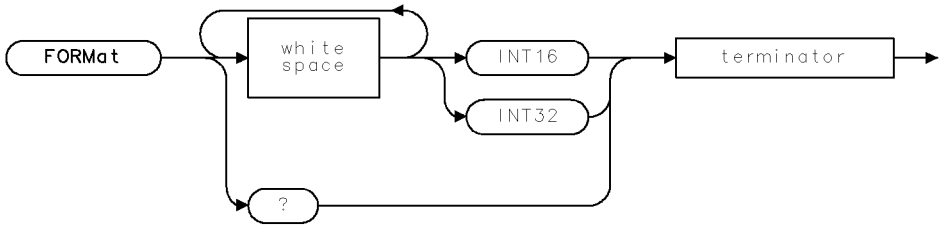
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"       ! Determine the starting
OUTPUT @Mta;"WAV:FOR?"             ! frequency for memory 1
ENTER  @Mta;Fstart
```

FORMat

Determines the internal data size for the trace selected.

Syntax



Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:FORM INT32"
```

Description

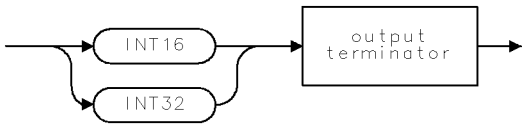
This command determines the internal data size for the trace selected with the WAVEFORM:SOURCE command. The possibilities are listed below.

- INT16: twos complement 16 bit. Full scale is $\pm 2^{14}$
- INT32: twos complement 32 bit. Full scale is $\pm 2^{30}$

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data.

The query returns the current state of the selected trace.

Query Response



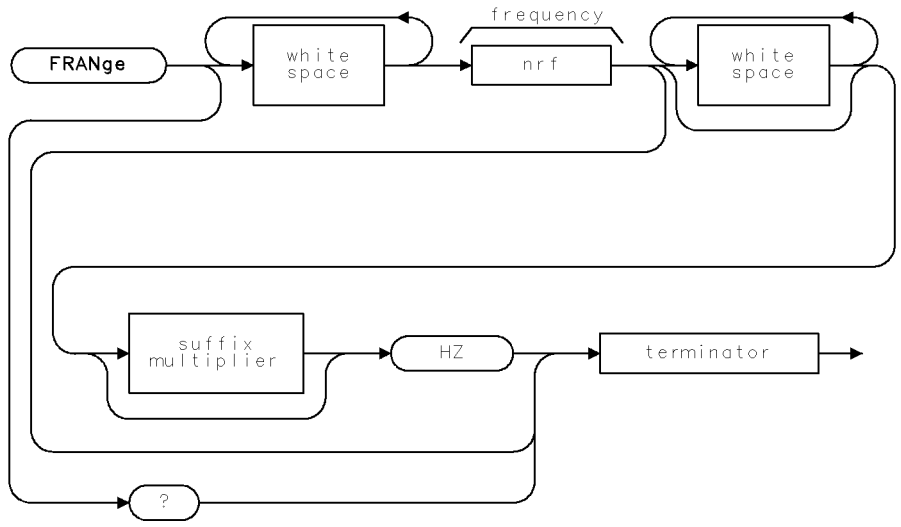
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"        ! Determine the data size
OUTPUT @Mta;"WAV:FORM?"             ! for channel 1
ENTER @Mta;Format$
```

FRANge (Frequency Range)

Sets or determines the frequency range of the frequency domain data.

Syntax



Parameter Range: abs(frequency − FORigin)≤1.0E20 Hz
Fundamental Unit: hertz

Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:FRAN 1 GHZ"
```

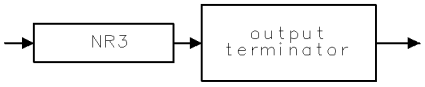
Description

This command sets or determines the frequency range of the frequency domain data.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. The interpretation of this command depends on WAVEFORM:POINTS and WAVEFORM:DTYPE

This specifies the same parameter as the WAVEFORM:FINCREMENT command.

Query Response



QGENNR3

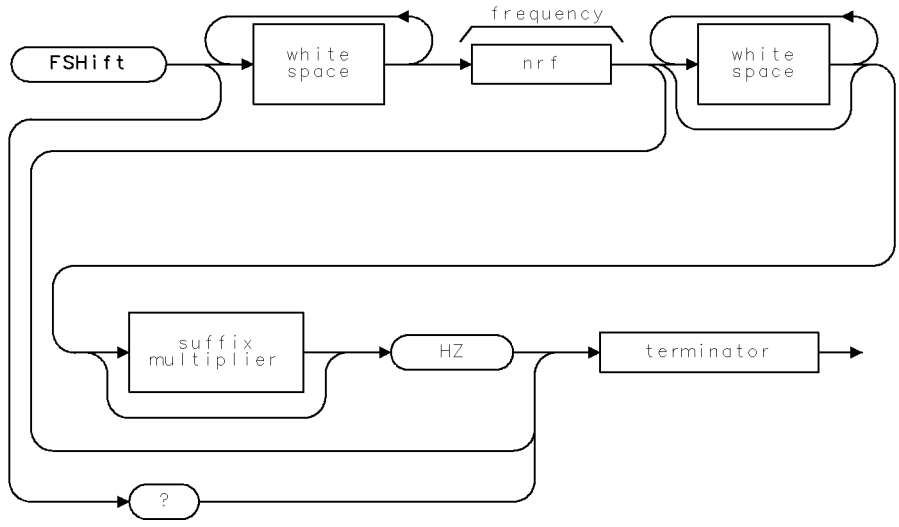
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"       ! Determine the sampling
OUTPUT @Mta;"WAV:FRAN?"            ! frequency for memory 1
ENTER  @Mta;Finc
```

FSHift

Specifies the phase slope to apply to the trace.

Syntax



XWAV10

Preset State: 0
Parameter Range: frequency = −1.0E12 through +1.0E12
Fundamental Unit: hertz
Related Commands: WAVeform:DELay
WAVeform:PSLope

Command Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:FSH 20 MHz"
```

Description

The FSHIFT command specifies the phase slope to apply to the trace specified by the WAVEFORM:SOURCE command in terms of frequency.

The WAVEFORM:DELAY, WAVEFORM:FSHIFT, and WAVEFORM:PSLOPE commands all affect the same parameter. When the trace data has a horizontal axis of time, the phase slope should be set by WAVEFORM:FSHIFT and is interpreted as a frequency shift and the parameter is scaled by:

$$phase = 360 \times phase\ slope \times time\ range$$

Also, when the horizontal axis is time, an optional frequency offset is applied to *slope* depending on the trace parameters. If the start frequency of the trace is nonzero (this can occur when in the translate mode, see SWEEP:TIME:TRANSLATE) and *start frequency +*

slope is less than 0, then the value used for *slope* is *start frequency + slope*. This mode is mainly intended for use with pulsed RF signals when it is desired to shift the carrier to bucket 0. This mode can be enabled or disabled by the WAVEFORM:FMODE command.

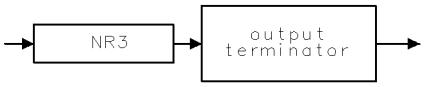
When the trace data has a horizontal axis of frequency, the phase slope should be set by WAVEFORM:DELAY and is interpreted as a time delay and the parameter is scaled by:

$$phase = 360 \times phase\ slope \times (stop\ frequency - start\ frequency)$$

When the trace data has a horizontal axis of neither time nor frequency, the phase slope should be set by WAVEFORM:PSLOPE and is not scaled.

The query returns the current setting.

Query Response



QGENNR3

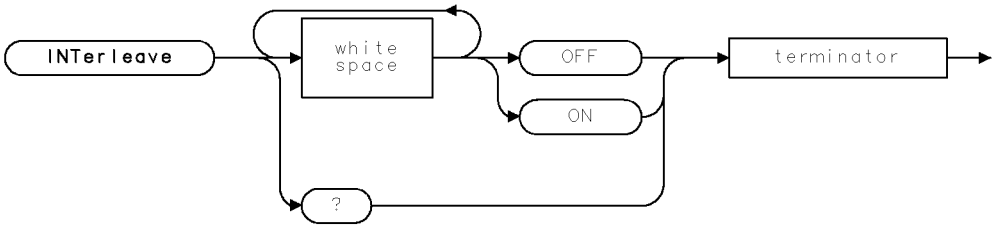
Query Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:FSH?"
ENTER @Mta;Shift
```

INTerleave

Sets or determines whether trace data is sent or received in a blocked or interleaved format.

Syntax



Command Example

```
OUTPUT @Mta;"WAV:INT OFF"       ! Turn interleave off
OUTPUT @Mta;"WAV:SOUR CHAN1"    ! Read trace data from channel 1
OUTPUT @Mta;"WAV:DOM OFF,POL"   ! as magnitude and phase in the
OUTPUT @Mta;"WAV:DATA?"         ! frequency domain
ENTER @Mta USING "#,2A";Header$
FOR J=0 TO N-1
  ENTER @Mta USING "#,W";Magn(J)
NEXT J
FOR J=0 TO N-1
  ENTER @Mta USING "#,W";Phase(J)
NEXT J
ENTER @Mta USING "#,B";Linefeed
OUTPUT @Mta;"WAV:INT ON"         ! Turn interleave on
OUTPUT @Mta;"WAV:DATA?"          ! Reread the same data
ENTER @Mta USING "#,2A";Header$
FOR J=0 TO N-1
  ENTER @Mta USING "#,W,W";Magn(J),Phase(J)
NEXT J
ENTER @Mta USING "#,B";Linefeed
```

Description

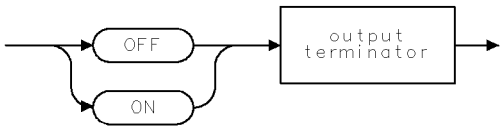
This command sets or determines whether trace data is sent or received in a blocked or interleaved format. This command is not associated with any specific trace, but instead sets the transfer mode for all traces accessed from this point on. When reading traces with WAVEFORM:DOMAIN set to IMAGINARY, MAGNITUDE, PHASE, or REAL, the state of INTERLEAVE does not affect the trace transfer.

The order of the data for the various modes is:

```
INTerleave OFF:
   <real1>...<realN><imag1>...<imagN>
  or
   <magn1>...<magnN><phase1>...<phaseN>

INTerleave ON:
   <real1><imag1>...<realN><imagN>
  or
   <magn1><phase1>...<magnN><phaseN>
```

Query Response



QGENON

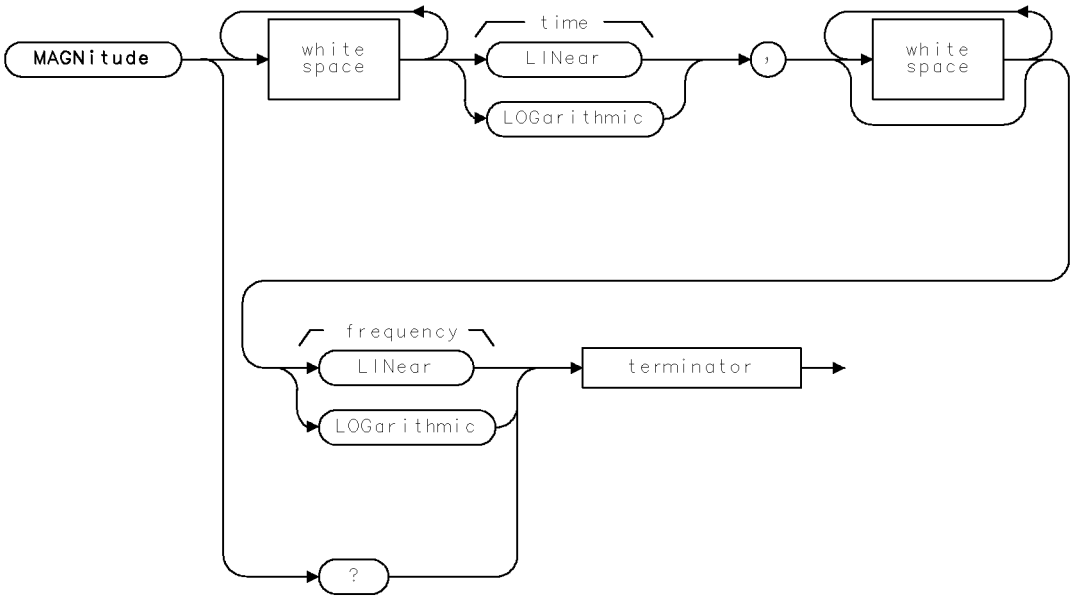
Query Example

```
OUTPUT @Mta;"WAV:INT?"
ENTER @Mta;Interleave$
```

MAGNitude

Specifies whether the amplitude or magnitude of the trace is to be logged before being displayed or output.

Syntax



Preset State: LINear,LOGarithmic (see SWEep:TYPE)

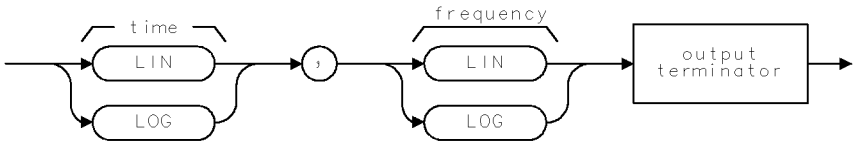
Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"    ! Set channel 1 to linear
OUTPUT @Mta;"WAV:MAGN LIN,LOG"  ! amplitude for time domain and
                                ! log amplitude for the frequency
                                ! domain
```

Description

This command specifies whether the amplitude or magnitude of the trace specified by the WAVEFORM:SOURCE command is to be logged before being displayed or output. The resolution of the logger is 0.01 dB for a 16-bit trace and 0.0001 dB for a 32-bit trace.

Query Response



. QWAV12

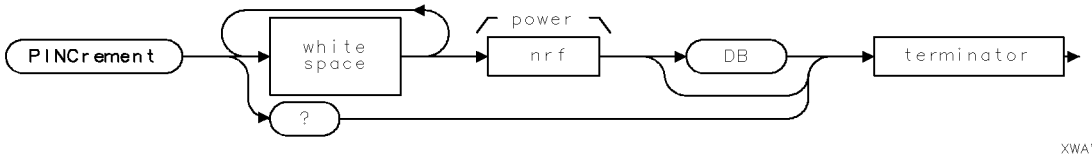
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:MAGN?"
ENTER @Mta;Magnitude$
```

PINCrement

Sets and determines the power increment between adjacent points.

Syntax



Parameter Range: -1000 dBm $\leq$ ((length-1)power $-$PORigin) $\leq$1000 dBm

Command Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:PINC 1 dB"
```

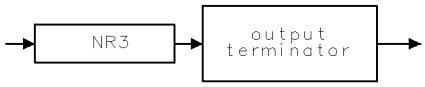
Description

The PINCREMENT command sets and determines the power increment between adjacent points in the frequency domain. The query returns the value of PINCREMENT.

This command sets the same parameter as the WAVEFORM:PRANGE command.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or, for FUNCX, whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data.

Query Response



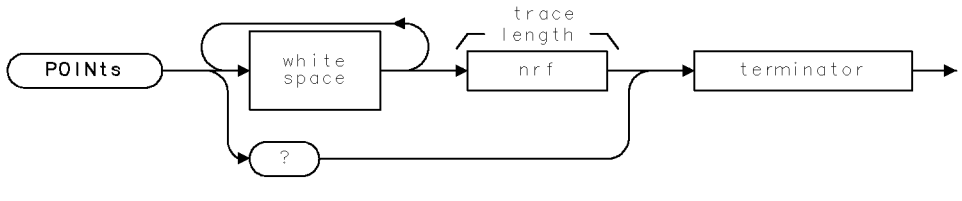
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"       ! Determine the frequency delta
OUTPUT @Mta;"WAV:PINC?"            ! for memory 1
ENTER  @Mta;Pinc
```

POINts

Sets the number of points in the trace specified.

Syntax



Parameter Range: trace length = 4 through 1024
Related Commands: ACQuire:POINts

Command Example

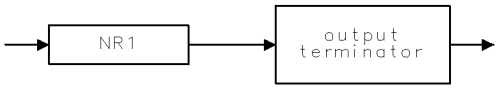
```
OUTPUT @Mta;"WAV:SOUR WMEM2"
OUTPUT @Mta;"WAV:POIN 512"
```

Description

The POINTS command sets the number of points in the trace specified by the WAVEFORM:SOURCE command. The query returns the number of points in the trace.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. It is also overwritten by the WAVEFORM:DATA command but still needs to be used if the trace parameters are loaded before the trace data is sent. To set the number of points acquired during a measurement, use the ACQUIRE:POINTS command.

Query Response



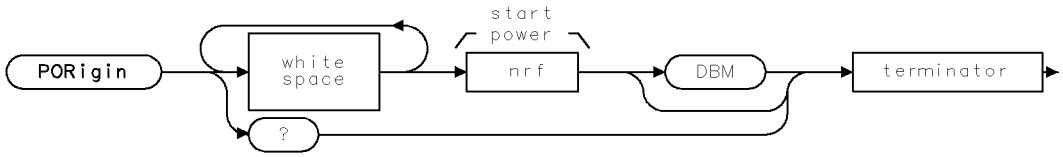
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM2"        ! Determine the number of
OUTPUT @Mta;"WAV:POIN?"             ! points in memory 2
ENTER @Mta;Length
```

PORigin

Sets and determines the starting power of the data.

Syntax



Parameter Range: start power = −1000 dBm through +1000 dBm

Command Example

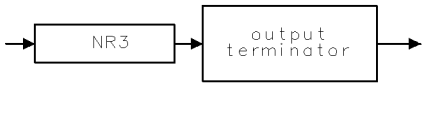
```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:POR -10 dBm"
```

Description

The PORIGIN command sets and determines the starting power of the data in the frequency domain. Query returns the value of PORIGIN.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or, for FUNCx, whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. To set the start frequency for data acquisition use SWEEP:POWER:START.

Query Response



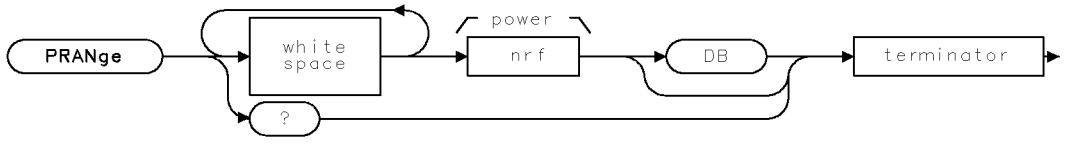
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:POR?"
ENTER  @Mta;Pstart
```

PRANge

Sets and determines the power range of the frequency domain data.

Syntax



Parameter Range: power = (PORigin – 1000 dBm) through (PORigin + 1000 dBm)

Command Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:PRAN 50 dB"
```

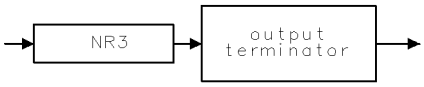
Description

The PRANGE command sets and determines the power range of the frequency domain data. The query returns the value of PRANGE.

This command specifies the same parameter as the WAVEFORM:PINCREMENT command.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or, for FUNCx, whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data.

Query Response



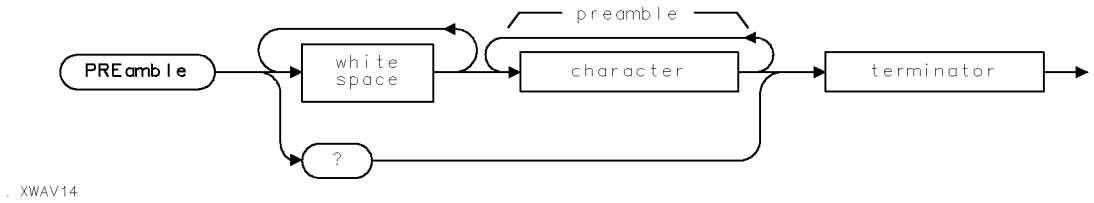
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:PRAN?"
ENTER  @Mta;Prange
```

PREamble

Sets or returns the values of the trace parameters.

Syntax



Command Example

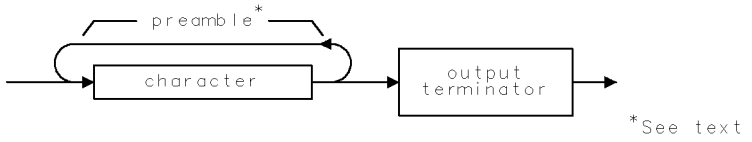
```
DIM Preamble$[100]
OUTPUT @Mta;"WAV:SOUR CHAN1"
Preamble$= "REAL,OFF,RTIM,SEC,VOLT,INT16,NORM"
Preamble$=Preamble$&"256,1E-12,0,1E-4,0,2E8,0,0"
OUTPUT @Mta;"WAV:PRE "&Preamble$
```

Description

This command sets or returns the values of the trace parameters. Only one form of each parameter is returned (that is, since VINCREMENT is returned, VRANGE and RLEVEL are not). The fields are listed below:

```
DOMain       :      see WAVeform:DOMain
DTYPe        :      see WAVeform:DTYPe
UNITs        :      see WAVeform:UNITs
FORMat       :      see WAVeform:FORMat
TYPE         :      see WAVeform:TYPE
POINts       :      see WAVeform:POINts
TINCrement   :      see WAVeform:TINCrement
TORigin      :      see WAVeform:TORigin
VINCrement   :      see WAVeform:VINCrement
VORigin      :      see WAVeform:VORigin
FINCrement   :      see WAVeform:FINCrement
FORigin      :      see WAVeform:FORigin
YREFerence   :      see WAVeform:YREFerence
```

Query Response



. QWAV14

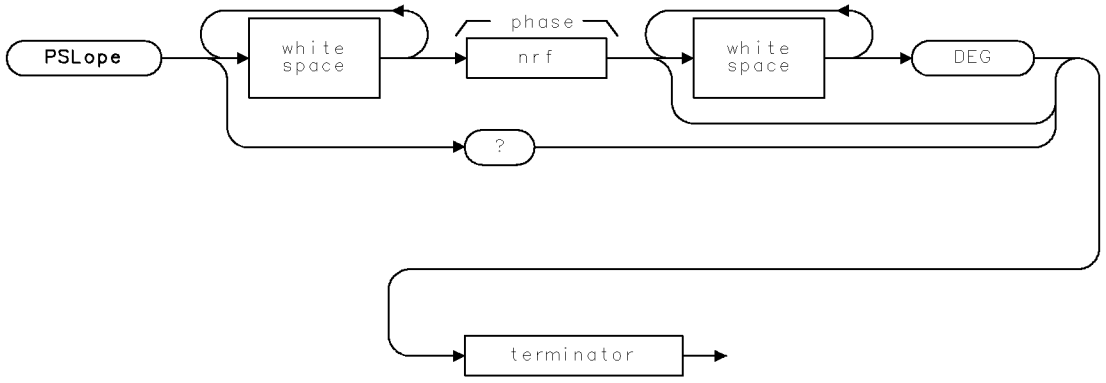
Query Example

```
DIM Preamble$[500]
OUTPUT @Mta;"WAV:SOUR WMEM1"   ! Read the preamble for memory 1
OUTPUT @Mta;"WAV:PRE?"
ENTER @Mta;Preamble$
```

PSLope

Specifies the phase slope to apply to the trace.

Syntax



XWAV15

Preset State: 0
Parameter Range: phase = −1.0E12 through +1.0E12
Fundamental Unit: hertz
Related Commands: WAVeform:DELay
WAVeform:FSHift

Command Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:PSL 45 deg"
```

Description

The PSLOPE command specifies the phase slope to apply to the trace specified by the WAVEFORM:SOURCE command in terms of degrees. The query returns the current setting.

The WAVEFORM:DELAY, WAVEFORM:FSHIFT, and WAVEFORM:PSLOPE commands all affect the same parameter. When the trace data has a horizontal axis of time, the phase slope should be set by WAVEFORM:FSHIFT and is interpreted as a frequency shift and the parameter is scaled by:

$$phase = 360 \times phase\ slope \times time\ range$$

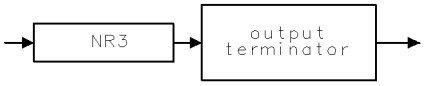
Also, when the horizontal axis is time, an optional frequency offset is applied to *slope* depending on the trace parameters. If the start frequency of the trace is nonzero (this can occur when in the translate mode, see SWEEP:TIME:TRANSLATE) and *start frequency + slope* is less than 0, then the value used for *slope* is *start frequency + slope*. This mode is mainly intended for use with pulsed RF signals when it is desired to shift the carrier to bucket 0. This mode can be enabled or disabled by the WAVEFORM:FMODE command.

When the trace data has a horizontal axis of frequency, the phase slope should be set by WAVEFORM:DELAY and is interpreted as a time delay and the parameter is scaled by:

$$phase = 360 \times phase\ slope \times (stop\ frequency - start\ frequency)$$

When the trace data has a horizontal axis of neither time nor frequency the phase slope should be set by WAVEFORM:PSLOPE and is not scaled.

Query Response



QGENNR3

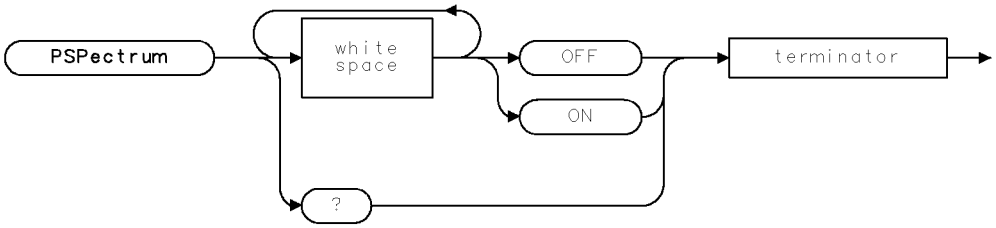
Query Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:PSL?"
ENTER @Mta;Phase_slope
```

PSPectrum

Determines how much data of an FFT is used for the trace.

Syntax



Preset State: ON
Related Commands: WAVeform:DATA
WAVeform:FINCrement
WAVeform:POINts
WAVeform:SOURce

Command Example

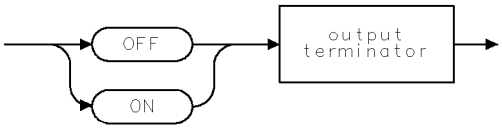
```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:PSP OFF"
```

Description

The PSPECTRUM command determines how much data of an FFT is used for the trace specified by WAVEFORM:SOURCE. When PSPECTRUM is ON, only the half of the data corresponding to the positive frequencies are used. When PSPECTRUM is OFF, all of the data from the FFT is used.

The query returns the current setting.

Query Response



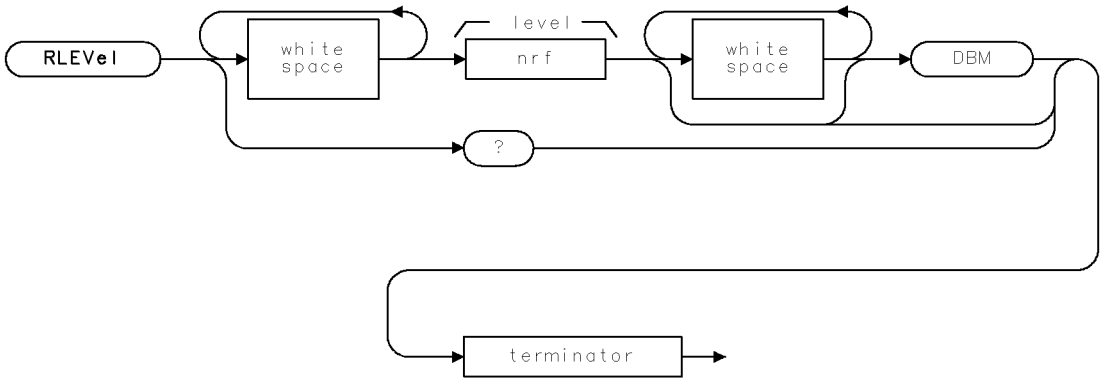
Query Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:PSP?"
ENTER @Mta;Spectrum$
```

RLEVel (Reference Level)

Sets or determines the reference level for the trace.

Syntax



Preset State: −10 dBm
Parameter Range: 2.0E–20 ≤ linear range ≤ 2.0E20 V
−400 dB ≤ logged reference level ≤ 400 dB
−390 dBm ≤ logged reference level ≤ 410 dBm
Fundamental Unit: dBm

Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:RLEV -20 dBm"
```

Description

This command sets or determines the reference level for the trace specified by the WAVEFORM:SOURCE command. This specifies the same parameter as the WAVEFORM:VRANGE command and uses the relationship:

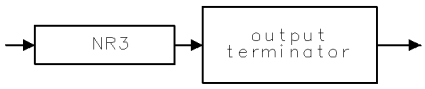
$$reference\ level(dBm) = 10 \times log_{10}((voltage\ range/2/\sqrt{2})^2/50/0.001)$$

$$reference\ level(dB) = 20 \times log_{10}(0.5 \times voltage\ range)$$

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. To set the reference level for data acquisition, use the CHANx:RLEVEL command.

The query returns the current reference level setting.

Query Response



QGENNR3

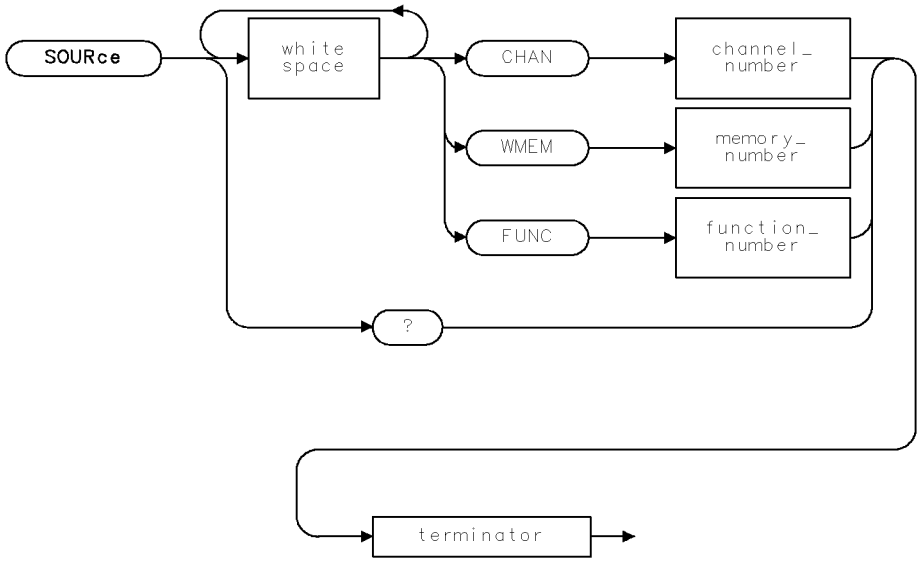
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"        ! Determine the reference
OUTPUT @Mta;"WAV:RLEV?"             ! level for channel 1
ENTER @Mta;Ref_level
```

SOURce

Sets or determines the trace that most of the other WAVEFORM commands will operate on.

Syntax



Command Example

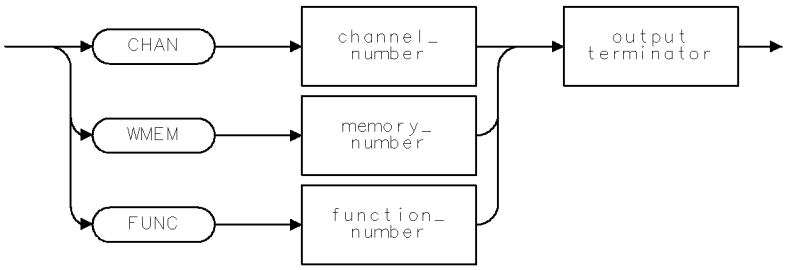
```
OUTPUT @Mta;"WAV:SOUR WMEM1"
```

Description

This command sets or determines the trace that most of the other WAVEFORM commands will operate on.

Note In the HP 70820A, all traces TR1 through TR4 are accessed via FUNC1 through FUNC4.

Query Response



. QWAV17

x = 1,2 for channels
1,4 for functions
1 through 4 for memories

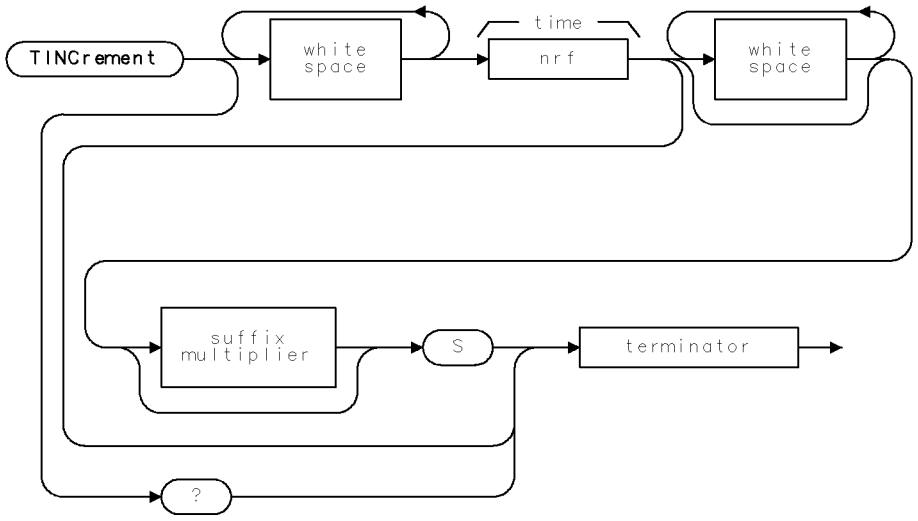
Query Example

```
OUTPUT @Mta;"WAV:SOUR?"        ! Determine the currently
ENTER @Mta;Source$             ! selected trace
```

TINCrement (Time Increment)

Sets or determines the time per point in the time domain.

Syntax



XWAV18

Parameter Range: time = 0 through 1 seconds
Fundamental Unit: seconds

Command Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:TINC 1 ns"
```

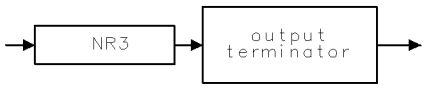
Description

This command sets or determines the time per point in the time domain. The actual parameter stored is time range = trace length × time per point.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. The interpretation of this command depends on WAVEFORM:POINTS and WAVEFORM:DTYPE.

This specifies the same parameter as the WAVEFORM:TRANGE command.

Query Response



QGENNR3

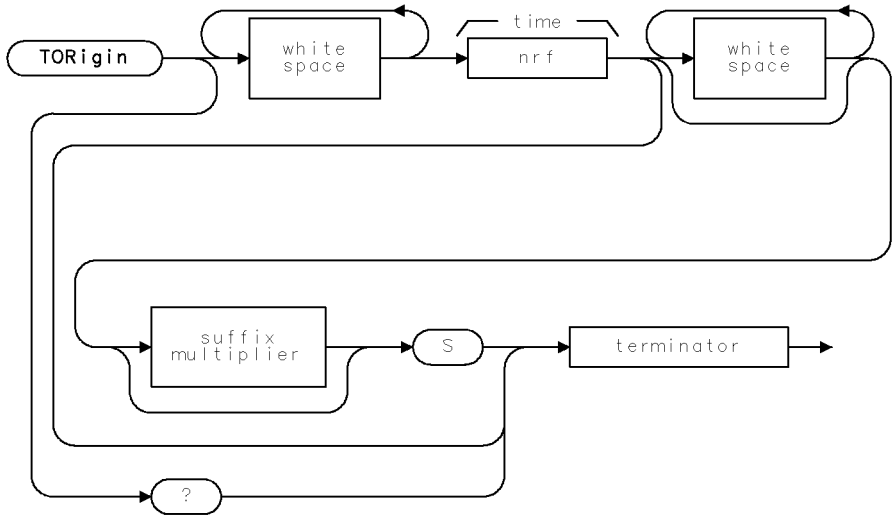
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"        ! Determine the time
OUTPUT @Mta;"WAV:TINC?"             ! increment for memory 1
ENTER  @Mta;Tinc
```

TORigin

Sets or determines the time of the first point in a trace relative to the trigger.

Syntax



XWAV19

Parameter Range: time = −1.0E6 through +1.0E6
Fundamental Unit: seconds

Command Example

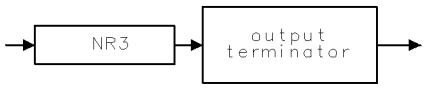
```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:TOR 0"
```

Description

This command sets or determines the time of the first point in a trace relative to the trigger.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. To set the starting time of the trace for data acquisition, use the TIMEBASE:DELAY command.

Query Response



QGENNR3

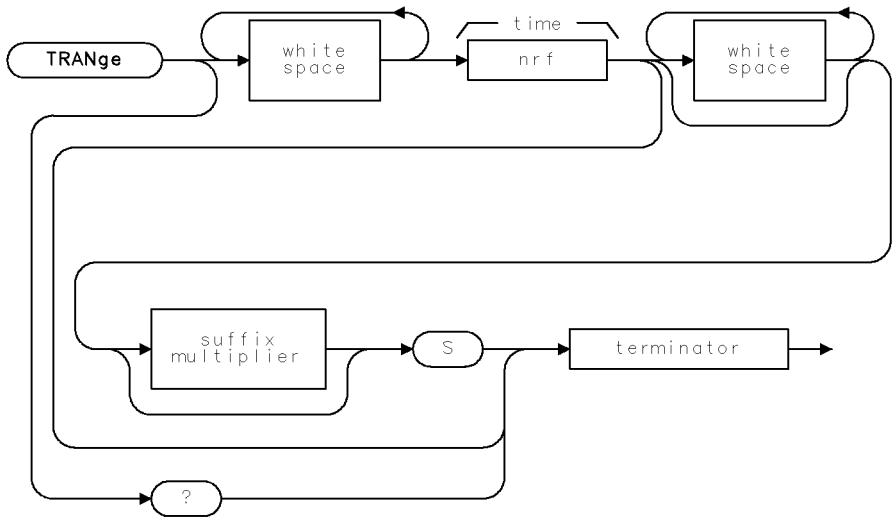
Query Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"          ! Determine the starting
OUTPUT @Mta;"WAV:TOR?"                ! time for function 1
ENTER @Mta;Start_time
```

TRANge

Sets or determines the time range for a trace in the time domain.

Syntax



Parameter Range: time = 0 through (trace length) seconds
Fundamental Unit: seconds

Command Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"
OUTPUT @Mta;"WAV:TRAN 1S"
```

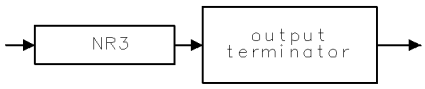
Description

The TRANGE command sets or determines the time range for a trace in the time domain.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. The interpretation of this command depends on WAVEFORM:POINTS and WAVEFORM:DTYPE.

This specifies the same parameter as the WAVEFORM:FRANGE, WAVEFORM:FINCREMENT, and WAVEFORM:TINCREMENT commands.

Query Response



QGENNR3

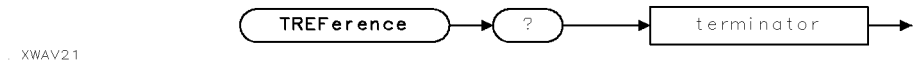
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"        ! Determine the time range
OUTPUT @Mta;"WAV:TRAN?"             ! for memory 1
ENTER  @Mta;Trange
```

TREFerence

Returns the time at the trigger point.

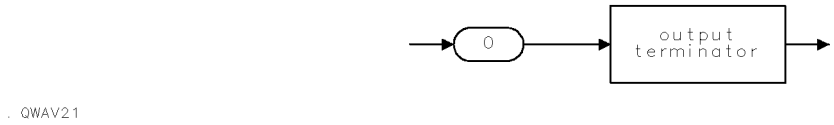
Syntax



Description

This query returns the time at the trigger point. Currently this is always set to 0.

Query Response



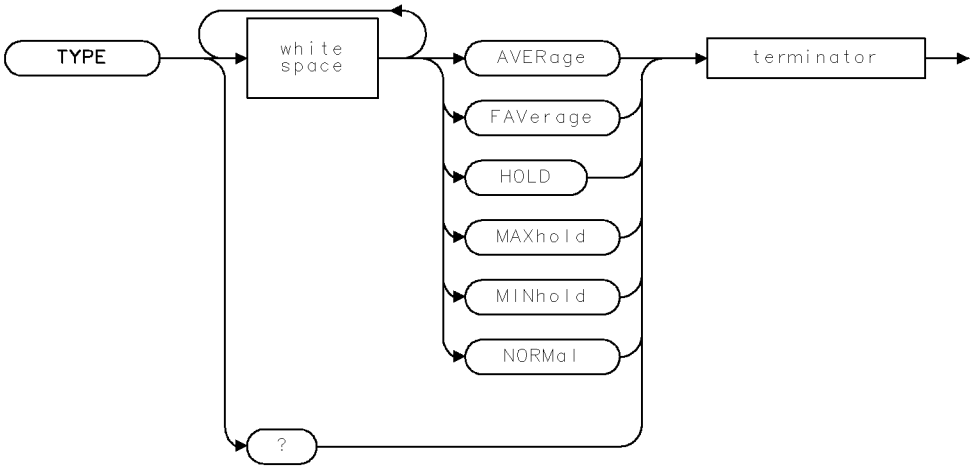
Query Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1"
OUTPUT @Mta;"WAV:TREF?"
ENTER @Mta;Tref
```

TYPE

Sets the post-processing mode for either channels or functions.

Syntax



Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:TYPE AVER"
```

Description

The command form of TYPE sets the post-processing mode for either channels or functions. An error will be declared if the current value of WAVEFORM:SOURCE is a memory. The query returns the post-processing mode that was applied after a data acquisition.

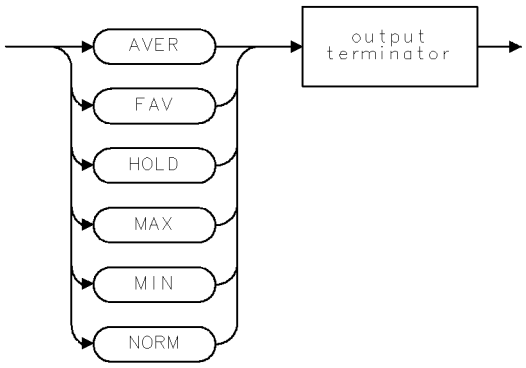
The possible values are listed below.

- **NORMAL:** No post processing is performed.
- **AVERAGE:** The averaging method described for ACQUIRE:COUNT is applied to the raw—that is, unformatted—data.
- **FAVERAGE:** The averaging method described for ACQUIRE:COUNT is applied to the formatted data.
- **HOLD:** A minimum or maximum hold operation is performed on the formatted data. The odd points of the result (that is, the first, third ...) contain the maximum values. The even points of the result (that is, the second, fourth ...) contain the minimum values. The trace size is not affected.
- **MAXHOLD:** A maximum hold operation is performed on the formatted data.
- **MINHOLD:** A minimum hold operation is performed on the formatted data.

Note that "WAVEFORM:SOURCE CHANx;TYPE xxx" is functionally equivalent to "CHANx:TYPE xxx" and "WAVEFORM:SOURCE FUNCx;TYPE xxx" is functionally

equivalent to "FUNCx:TYPE xxx". See also ACQUIRE:TYPE, CHANx:TYPE, and FUNCx:TYPE.

Query Response



QWAV22

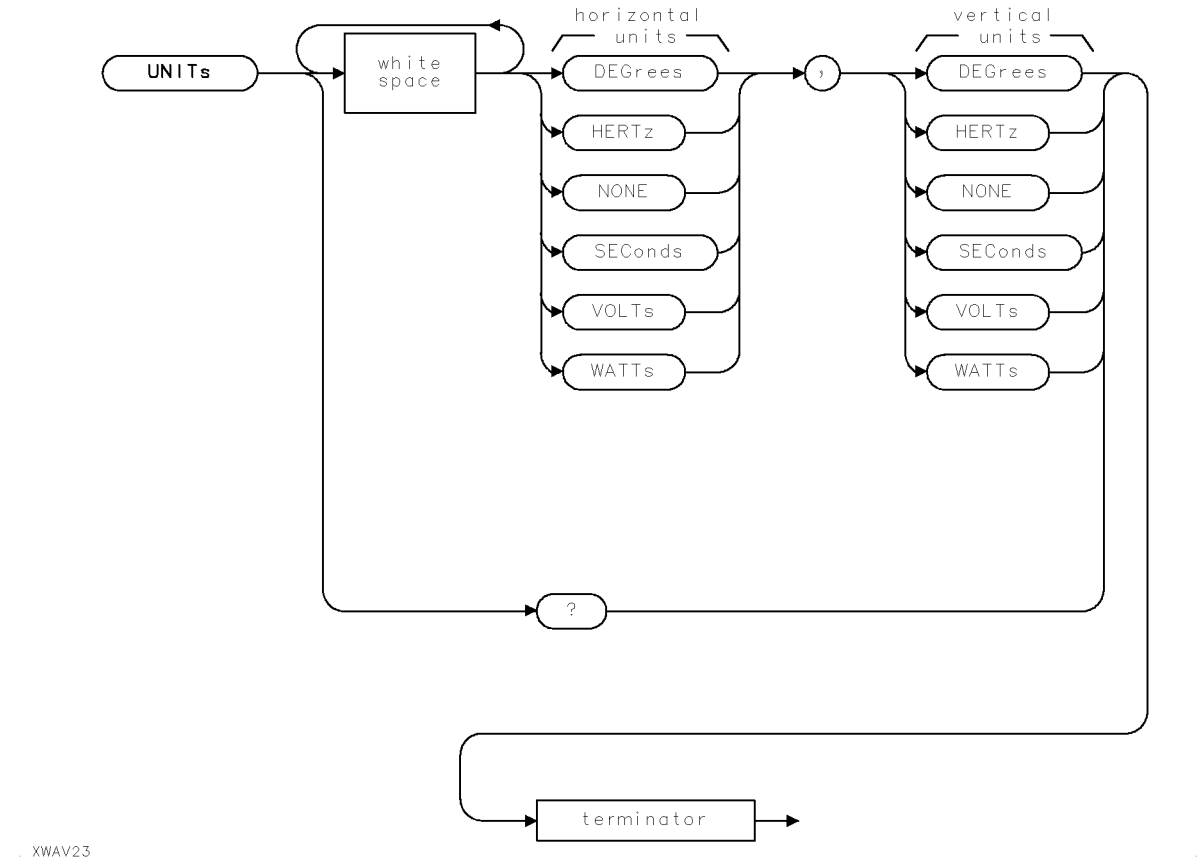
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"  ! Determine the post-processing
OUTPUT @Mta;"WAV:TYPE?"       ! applied to channel 1
ENTER @Mta;Post_process$
```

UNITs

Defines the horizontal and vertical units for the raw data format.

Syntax



Preset State: SEConds, VOLTs

Command Example

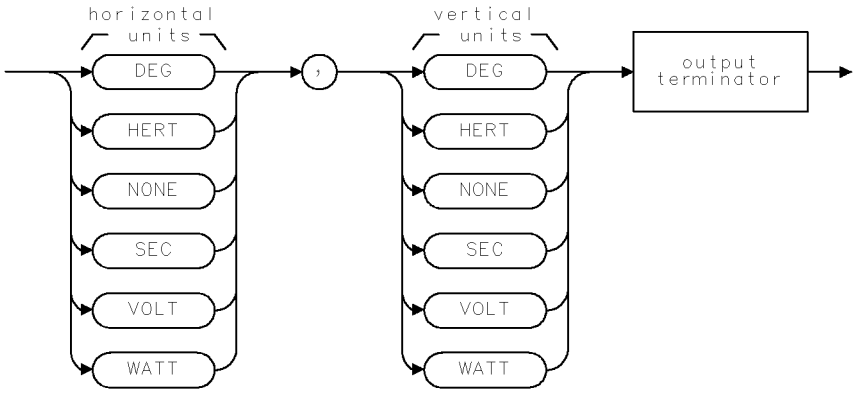
```
OUTPUT @Mta;"WAV:SOUR CHAN1;"
OUTPUT @Mta;"WAV:UNIT NONE,NONE;"
```

Description

The UNITS command defines the horizontal and vertical units for the raw data format (set by WAVEFORM:DTYPE). The query returns the current settings.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. The interpretation of this command depends on WAVEFORM:DTYPE.

Query Response



QWAV23

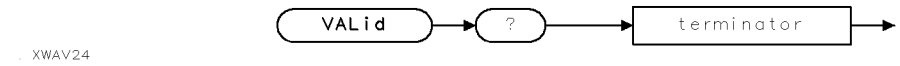
Query Example

```
OUTPUT @Mta;"WAV:SOUR FUNC1;"
OUTPUT @Mta;"WAV:UNIT?"
ENTER @Mta;Unit$
```

VALid

Returns a data valid flag for the trace specified.

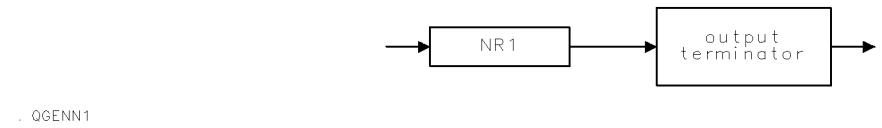
Syntax



Description

This command returns a data valid flag for the trace specified by WAVEFORM:SOURCE. Possible reasons for invalid data would be: a trace acquisition is in progress, the function is undefined, no data is stored in the memory, and so forth. Zero (0) is returned for invalid data, one (1) returned for valid.

Query Response



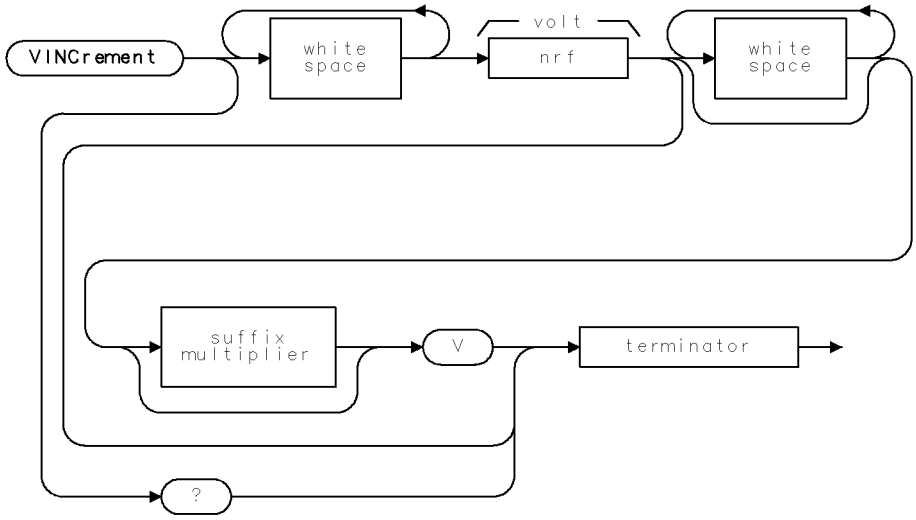
Query Example

```
OUTPUT @Mta;"WAV:SOUR WMEM1"      ! Determine if memory 1 has
OUTPUT @Mta;"WAV:VAL?"            ! valid data
ENTER @Mta;Valid
IF Valid = 1 THEN
   PRINT "Memory 1 is valid"
ELSE
   PRINT "Memory 1 is invalid"
END IF
```

VINCrement

Sets or determines the value of one LSB of the trace specified.

Syntax



Parameter Range:	2.0E–20 ≤ linear range ≤ 2.0E20 V
	−400 dB ≤ logged reference level ≤ 400 dB
	−390 dBm ≤ logged reference level ≤ 410 dBm
Fundamental Unit:	volts
Related Commands:	WAVeform:RLEVel
	WAVeform:VRANge

Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:VINC 1 mV"
```

Description

This command sets or determines the value of one LSB of the trace specified by the WAVEFORM:SOURCE command. This sets the same parameter as the WAVEFORM:RLEVEL and WAVEFORM:VRANGE commands.

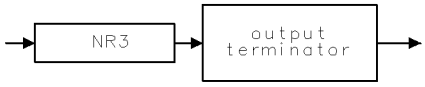
The relationship is:

$$1\ LSB = \frac{voltage\ range}{(WAVeform:YTOP - WAVeform:YBOTtom + 1)}$$

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. To set the voltage range for data acquisition, use the CHANx:RANGE command.

Note This parameter will not necessarily correspond to the value set with CHANx:RANGE or FUNCx:RANGE due to the processing done on the data (for example, corrections, FFTs, and so on).

Query Response



QGENNR3

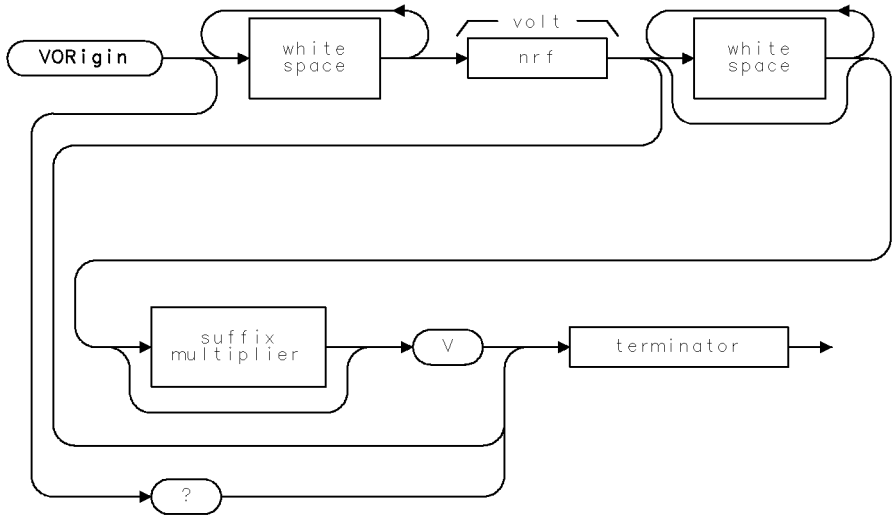
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"      ! Determine the voltage range
OUTPUT @Mta;"WAV:VINC?"           ! for channel 1
ENTER @Mta;Volts
```

VORigin

Sets or determines the center of the voltage range for the trace specified.

Syntax



XWAV26

Parameter Range: volt = −1.0E6 through +1.0E6 V
Fundamental Unit: volts

Command Example

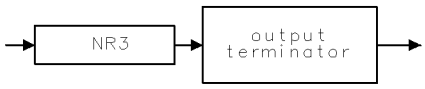
```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:VOR 0 V"
```

Description

This command sets or determines the center of the voltage range for the trace specified by the WAVEFORM:SOURCE command. For example, with a voltage range of 20 mV and WAVEFORM:VORIGIN set to 10 mV, the measurement range will cover from 0 V to 20 mV.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated It is intended to be used in conjunction with downloading trace data. To set the voltage offset for data acquisition, use the CHANx:OFFSET command.

Query Response



QGENNR3

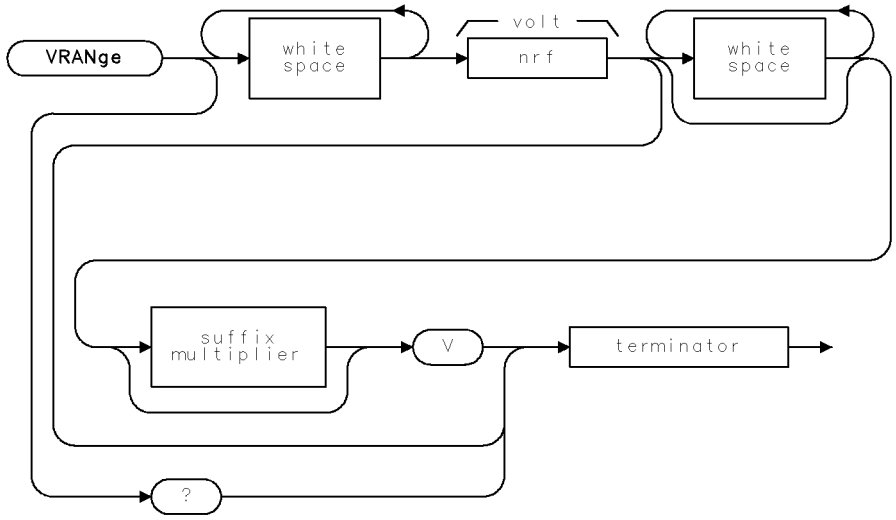
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"      ! Determine the voltage origin
OUTPUT @Mta;"WAV:VOR?"            ! of channel 1
ENTER @Mta;Vorigin
```

VRANge

Sets or determines the voltage range for the trace specified.

Syntax



XWAV27

Parameter Range: 2.0E–20 ≤ linear range ≤ 2.0E20 V
−400 dB ≤ logged reference level ≤ 400 dB
−390 dBm ≤ logged reference level ≤ 410 dBm

Fundamental Unit: volts

Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:VRAN 200 mV"
```

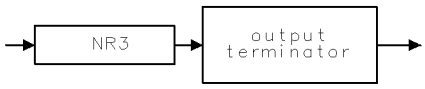
Description

This command sets or determines the voltage range for the trace specified by the WAVEFORM:SOURCE command. This sets the same parameter as the WAVEFORM:RLEVEL and WAVEFORM:VINCREMENT commands.

Note This parameter is overwritten for CHANx and FUNCx whenever a sweep is taken or for FUNCx whenever the function is reevaluated. It is intended to be used in conjunction with downloading trace data. To set the voltage range for data acquisition, use the CHANx:RANGE command.

Note This parameter will not necessarily correspond to the value set with CHANx:RANGE or FUNCx:RANGE due to the processing done on the data (for example, corrections, FFTs, and so on).

Query Response



QGENNR3

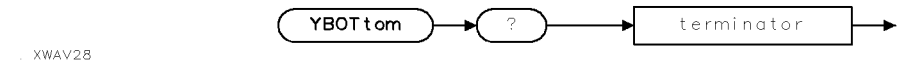
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"      ! Determine the voltage range
OUTPUT @Mta;"WAV:VRAN?"           ! for channel 1
ENTER @Mta;Volts
```

YBOTtom

Returns the value that maps the bottom of the voltage range for the trace specified.

Syntax



Description

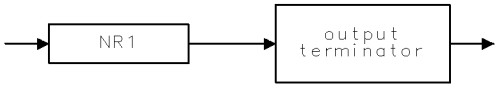
The YBOTTOM command returns the value that maps the bottom of the voltage range for the trace specified by WAVEFORM:SOURCE. Currently there are two possible values:

```
-16384 (    -4000H) for WAVeform:FORMat = INT16

-1073741824 (-40000000H) for WAVeform:FORMat = INT32
```

For example, if WAVEFORM:VRANGE returns 20 mV and WAVEFORM:VORIGIN returns 0 V, the measurement range covers −10 mV to 10 mV, and a trace value of −16384 (for INT16) corresponds to −10 mV.

Query Response



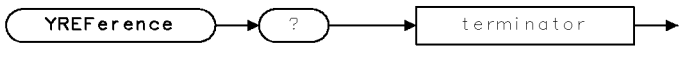
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"   ! Determine the value for
OUTPUT @Mta;"WAV:YBOT?"        ! YBOTtom for channel 1
ENTER @Mta;Ybottom
```

YREFerence

Returns the value that maps the center of the voltage range for the trace specified.

Syntax

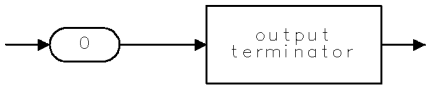


XWAV29

Description

The YREFERENCE command returns the value that maps the center of the voltage range for the trace specified by WAVEFORM:SOURCE. Currently there is only one possible value: 0.

Query Response



QWAV29

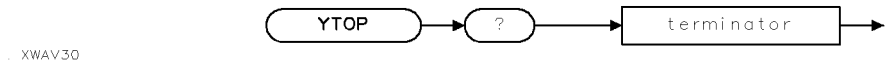
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:YREF?"
ENTER @Mta;Yreference
```

YTOP

Returns the value that maps the top of the voltage range for the trace specified.

Syntax



Description

The YTOP command returns the value that maps the top of the voltage range for the trace specified by WAVEFORM:SOURCE.

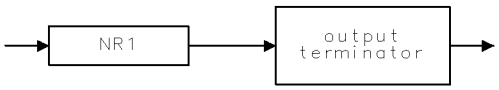
Currently there are two possible values:

```
      16383 (    3FFFH) for WAVeform:FORMat = INT16

1073741823 (3FFFFFFFH) for WAVeform:FORMat = INT32
```

For example, if WAVEFORM:VRANGE returns 20 mV and WAVEFORM:VORIGIN returns 0 V, the measurement range covers −10 mV to 10 mV, and a trace value of 16383 (for INT16) corresponds to 10 mV.

Query Response



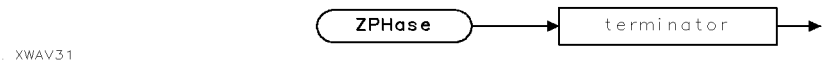
Query Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"          ! Determine the value for
OUTPUT @Mta;"WAV:YTOP?"               ! YTOP for channel 1
ENTER @Mta;Ytop
```

ZPHase
(Zero Phase)

Measures the phase slope of the trace and sets it to zero.

Syntax



Related Commands: WAVeform:DELay
WAVeform:FSHift
WAVeform:PSLope

Command Example

```
OUTPUT @Mta;"WAV:SOUR CHAN1"
OUTPUT @Mta;"WAV:ZPH"
OUTPUT @Mta;"WAV:PSLOPE?"
ENTER @Mta;Phase_slope
```

Description

The ZeroPHASE command measures the phase slope of the trace specified by the WAVEFORM:SOURCE command and sets it to zero (that is, it sets the parameter for WAVEFORM:DELAY, WAVEFORM:FSHIFT, or WAVEFORM:PSLOPE to the measured value). The results can be obtained by querying either WAVEFORM:DELAY, WAVEFORM:FSHIFT, or WAVEFORM:PSLOPE.

31

Window Subsystem

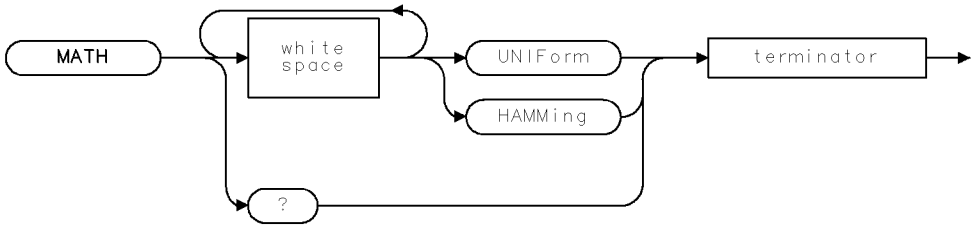
The WINDOW subsystem commands are used to specify the window to be used by the fast Fourier transform (FFT) algorithm when translating data from the time domain to the frequency domain. See the following list for a command syntax summary of the WINDOW subsystem commands.

KEYWORD	PARAMETER FORM	NOTES
:WINDow		
:MATH	UNIForm|HAMMing	
[:TYPE]	UNIForm|HANNing|FLATtop|USER	
:USER	<nrf>{,<nrf>}	

MATH

Selects the window for math windowing operations.

Syntax



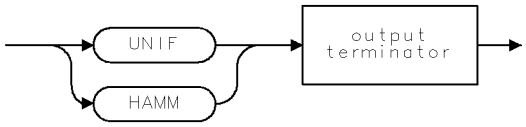
Command Example

```
OUTPUT @Mta;"WIND:MATH UNIF"
```

Description

The MATH command selects the window to be used for any windowing operations in the math subsystem (see FUNCx:DEFINE). The query returns the currently specified window.

Query Response



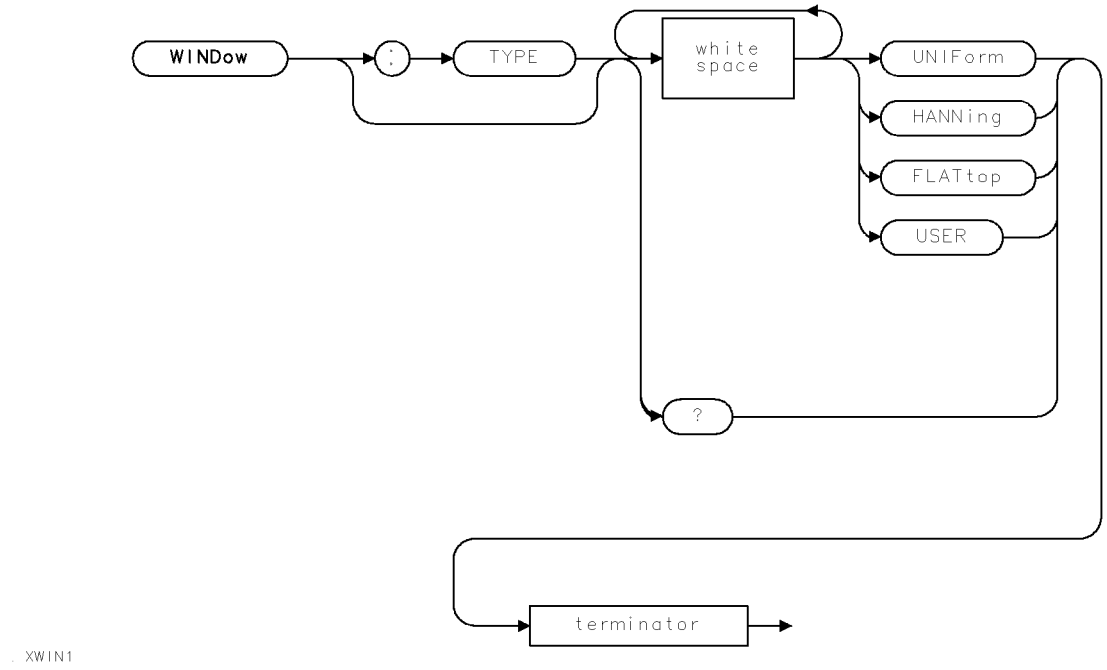
Query Example

```
OUTPUT @Mta;"WIND:MATH?"
ENTER @Mta;Window$
```

TYPE

Selects the window for formatting operations.

Syntax



Preset State: HANNing

Command Example

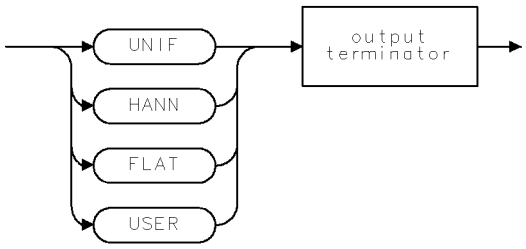
```
OUTPUT @Mta;"WIND:TYPE UNIF"
```

Description

The TYPE command selects the window to be used on the specified source trace when displaying data in the frequency domain.

The query returns the currently specified window.

Query Response



QWIN1

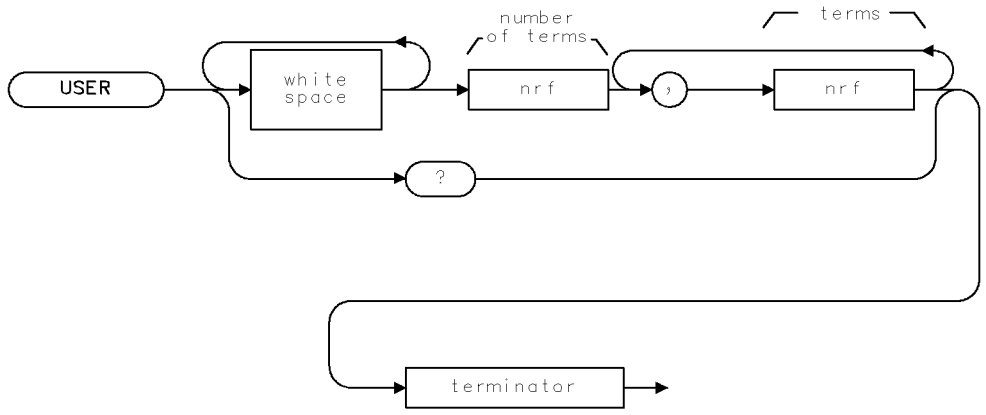
Query Example

```
OUTPUT @Mta;"WIND:TYPE?"
ENTER @Mta;Window$
```

USER

Defines a window in terms of a cosine power series.

Syntax



Preset State: terms = 3
gain = 0.424619564
A(0) = 0.346106052
A(1) = −0.497339606
A(2) = 0.156553984
A(3), A(4), A(5), A(6), A(7) = 0

Parameter Range: terms = 1 through 8
gain = 0.05 through 1.0
$-1 \leq$ an $\leq +1$

Command Example

```
OUTPUT @Mta;"WIND:USER 2,.5,-.5"              ! hanning window
```

Description

The USER command defines a window in terms of a cosine power series. This window is selected with the WINDOW:TYPE:USER command. The parameters are:

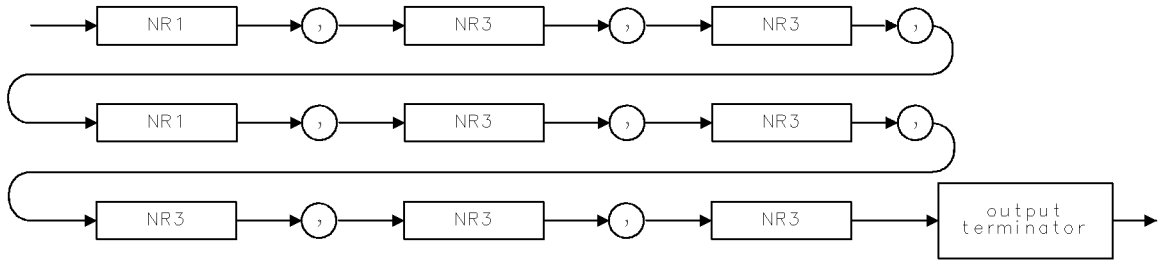
- number of coefficients
- window gain
- 1 to 8 coefficients

The query returns the currently defined user window.

The equation for the window is:

$$window(x) = a0 + a1 \times cos(x) + a2 \times cos^2(x)...$$

Query Response



QWIN2

Query Example

```
OUTPUT @Mta;"WIND:USER?"
ENTER @Mta;Terms,Gain,A(0),A(1),A(2),A(3),A(4),A(5),A(6),A(7)
```

32

Xline, Yline Subsystem

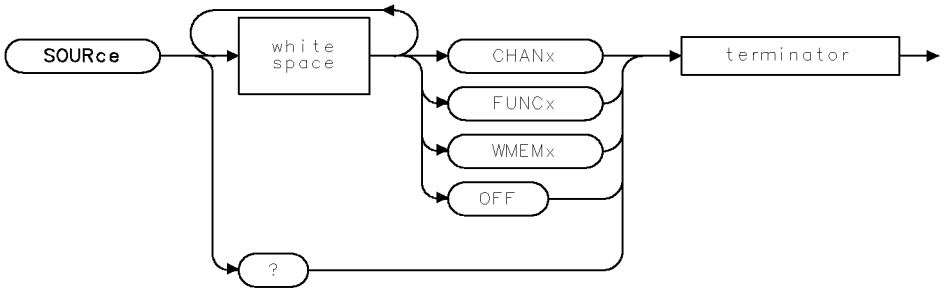
The Line commands (XLINE1, XLINE2, YLINE1, and YLINE2) access line markers. The intent of these commands is to make it easy for the user to place a vertical or horizontal line on the display with the scaling appropriate for a specified trace. See the following list for a command syntax summary of the XLINE, YLINE Subsystem.

KEYWORD	PARAMATER FORM	NOTES
:XLINE |		
:YLINE		
:SOURce	CHANx|FUNCx|WMEMx|OFF	
[:VALue]	<nrf>[[<suffix multiplier>]HZ]	

SOURce

Selects which trace the line markers use to determine display scaling and limits.

Syntax



xline1

Preset State: off

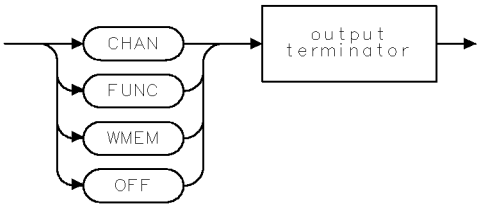
Command Example

```
OUTPUT @Mta;"XLINE1:SOUR CHAN1"
```

Description

The SOURCE command selects which trace the line markers use to determine display scaling and limits. The query returns the current setting of SOURCE.

Query Response



qline1

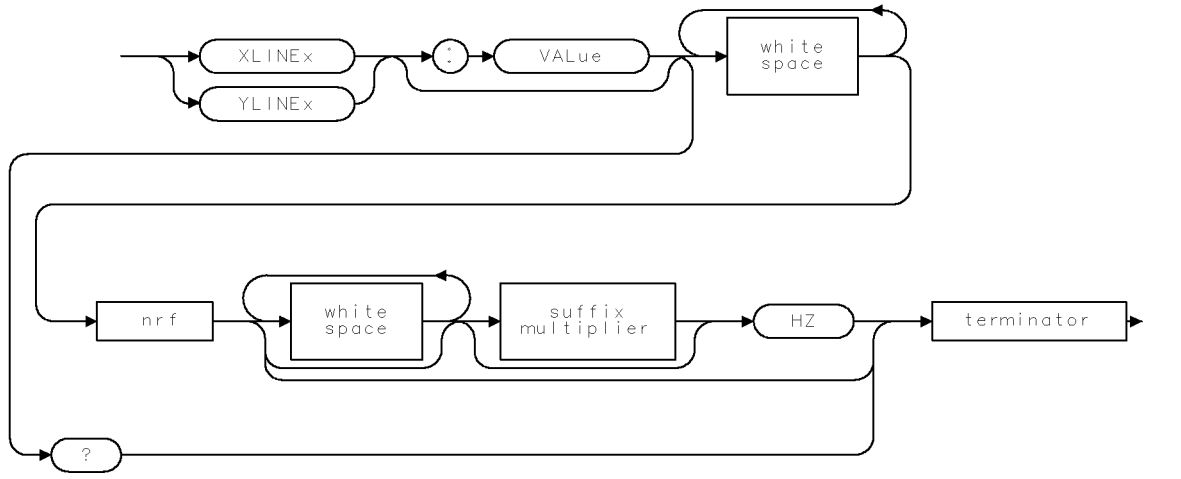
Query Example

```
OUTPUT @Mta;"YLINE2:SOUR?"
ENTER @Mta;Source$
```

VALue

Sets the value (position) of the specified line marker.

Syntax



Preset State: 0
Parameter Range: value is = to −9.9E37 through 9.9E37

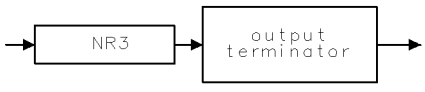
Command Example

```
OUTPUT @Mta;"XLINE1:VAL 1 MHz"
```

Description

The VALUE command sets the value (position) of the specified line marker. Any suffix recognized by the module is accepted. The query returns the current setting of VALUE.

Query Response



Query Example

```
OUTPUT @Mta;"YLINE2:VAL?"
ENTER @Mta;Value
```

33

Error Messages

This chapter defines all possible error messages displayed on the HP 71500A's screen. These error messages can be the result of incorrect operating procedures, illegal programming commands, or hardware failures. Normally, the microwave transmission analyzer removes error messages from the screen as soon as the error conditions are corrected.

In this chapter, all error messages are listed in numerical order.

Error messages are organized using the following categories:

- Series −100: Program Command Errors
- Series −200: Program Performance Errors
- Series −300: PON and Self Test Hardware Errors
- Series −400: Program Query/Response Errors
- Series 6000 and 7000: Hardware, Operation, and Mass Memory Errors
- Series 8000: Floating Point Errors
- Series 9000: Hardware Errors

Program Command Errors

The microwave transition analyzer reports -1XX errors when it cannot decipher a remote programming message.

-100

`command error`

This is the generic syntax error for devices that cannot detect more specific errors. This code indicates only that a "command error" as defined in IEEE 488.2 (11.5.1.1.4) has occurred.

-101

`invalid char`

A syntactic element contains a character which is invalid for that type; for example, a header containing an ampersand, SETUP&. This error might be used in place of errors -114, -121, -141, and perhaps some others.

-102

`syntax error`

An unrecognized command or data type was encountered; for example, a string was received when the device does not accept strings.

-103

`invalid separator`

The parser was expecting a separator and encountered an illegal character; for example, the semicolon was omitted after a program message unit, `*EMC 1:CH1:VOLTS 5`.

-104

`data type error`

The parser recognized a data element different than one allowed; for example, numeric or string data was expected but block data was encountered.

-105

`GET not allowed`

A Group Execute Trigger was received within a program message (see IEEE 488.2, 7.7)

-108

`parameter not allowed`

More parameters were received than expected for the header; for example, the *EMC common command only accepts one parameter, so receiving `*EMC 0,1` is not allowed.

-109

`missing parameter`

Fewer parameters were received than required for the header; for example, the *EMC common command requires one parameter, so receiving `*EMC` is not allowed.

-110

`command header`

An error was detected in the header. This error message should be used when the device cannot detect the more specific errors described for errors -111 through -119.

-111

`header separator`

A character which is not a legal header separator was encountered while parsing the header; for example, no white space followed the header, thus *GMC“MACRO” is an error.

-112

`program mnemonic too long`

The header contains more than twelve characters (see IEEE 488.2, 7.6.1.4.1).

-113

`undefined header`

The header is syntactically correct, but it is undefined for this specific device; for example, *XYZ is not defined for any device.

-120

`numeric data error`

This error, as well as errors -121 through -129, are generated when parsing a data element which appears to be numeric, including the nondecimal numeric types. This particular error message should be used if the device cannot detect a more specific error.

-121

`invalid char in number`

An invalid character for the data type being parsed was encountered; for example, an alpha in a decimal numeric or a "9" in octal data.

-123

`numeric overflow`

The magnitude of the exponent was larger than 32000 (see IEEE 488.2, 7.7.2.4.1).

-124

`too many digits`

The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros (see IEEE 488.2, 7.7.2.4.1).

-128

`numeric data not allowed`

A legal numeric data element was received, but the device does not accept one in this position for the header.

-130

`suffix error`

This error, as well as errors -131 through -139, are generated when parsing a suffix. This particular error message should be used if the device cannot detect a more specific error.

-131

`invalid suffix`

The suffix does not follow the syntax described in IEEE 488.2 (7.7.3.2) or the suffix is in appropriate for this device.

-134

`suffix too long`

The suffix contains more than 12 characters. See IEEE 488.2, 7.7.3.4.

-138

`suffix not allowed`

A suffix was encountered after a numeric element which does not allow suffixes.

-140

`char data error`

This error, as well as errors -141 through -149, are generated when parsing a character data element. This particular error message should be used if the device cannot detect a more specific error.

-141

`invalid char data`

Either the character data element contains an invalid character of the particular element received is not valid for the header.

-144

`char data too long`

The character data element contains more than 12 characters. See IEEE 488.2, 7.7.1.4.

-148

`char data not allowed`

A legal character data element was encountered where prohibited by the device.

-150

`string data error`

This error, as well as errors -151 through -159, are generated when parsing a string data element. This particular error message should be used if the device cannot detect a more specific error.

-151

`invalid string data`

A string data element was expected, but was invalid for some reason (see IEEE 488.2, 7.7.5.2); for example, an END message was received before the terminal quote character.

-158

`string data not allowed`

A string data element was encountered but was not allowed by the device at this point in parsing.

-160

`block data error`

This error, as well as errors -161 through -169, are generated when parsing a block data element. This particular error message should be used if the device cannot detect a more specific error.

-161

`invalid block data`

A block data element was expected, but was invalid for some reason (see IEEE 488.2, 7.7.6.2); for example, an END message was received before the length was satisfied.

-168

`block data not allowed`

A legal block data element was encountered but was not allowed by the device at this point in parsing.

-170

`expression error`

This error, as well as errors -171 through -179, are generated when parsing an expression data element. This particular error message should be used if the device cannot detect a more specific error.

-171

`invalid expression`

The expression data element was invalid (see IEEE 488.2, 7.7.7.2); for example, unmatched parentheses or an illegal character.

-174

`expression too long/complex`

The expression data element contains too many characters or involves too many operations.

-178

`expression data not allowed`

A legal expression data was encountered but was not allowed by the device at this point in parsing.

-180

`macro error`

This error, as well as errors -181 through -189, are generated when defining a macro or executing a macro. This particular error message should be used if the device cannot detect a more specific error.

-181

`invalid outside macro`

Indicates that a macro parameter placeholder ($<number) was encountered outside of a macro definition.

-183

`invalid inside macro`

Indicates that the program message unit sequence, sent with *DDT or *DMC command, is syntactically invalid. See IEEE 488.2, 10.7.6.3.

-184

`macro parameter error`

Indicates that a command inside the macro definition had the wrong number or type of parameters.

Program Performance Errors

The microwave transition analyzer reports -2XX errors when it is unable to perform a valid programming command.

-200

`execution error`

This is the generic syntax error for devices that cannot detect more specific errors. This code indicates only that an Execution Error as defined in IEEE 488.2, 11.5.1.1.5 has occurred.

-201

`remote only`

Indicates that a command is not executable while the device is in local due to a hard local control (see IEEE 488.2, 5.6.1.5); for example, a device with a rotary switch receives a message which would change the switches state, but the device is in local so the message cannot be executed.

-202

`settings lost`

Indicates that a setting associated with a hard local control (see IEEE 488.2, 5.6.1.5) was lost when the device changed to LOCS from REMS or to LWLS from RWLS.

-211

`trigger ignored`

Indicates that a GET, *TRG, or triggering signal was received and recognized by the device but was ignored because of device timing considerations; for example, the device was not ready to respond.

Note A DT0 device always ignores GET and treats *TRG as a Command Error.

-220

`parameter error`

Indicates that a program data element related error occurred. This error message should be used when the device cannot detect the more specific errors described for errors -221 through -229.

-221

`settings conflict`

Indicates that a legal program data element was parsed but could not be executed due to the current device state. See IEEE 488.2, 6.4.5.3 and 11.5.1.1.5.

-222

`data out of range`

Indicates that a legal program data element was parsed but could not be executed because the interpreted value was outside the legal range as defined by the device. See IEEE 488.2, 11.5.1.1.5.

-223

`too much data`

Indicates that a legal program data element of block, expression, or string type was received that contained more data than the device could handle due to memory or related device-specific requirements.

-230

`data corrupt`

Possibly invalid data; new reading started but not completed since last access.

-231

`data questionable`

Indicates that measurement accuracy is suspect.

-270

`macro error`

Indicates that a macro-related execution error occurred. This error message should be used when the device cannot detect the more specific errors described for errors -271 through -279.

-271

`macro syntax error`

Indicates that a syntactically legal macro program data sequence, according to IEEE 488.2, 10.7.2, could not be executed due to a syntax error within the macro definition. See IEEE 488.2, 10.7.6.3.

-272

`macro execute error`

Indicates that a syntactically legal macro program data sequence could not be executed due to some error in the macro definition. See IEEE 488.2, 10.7.6.3.

-273

`illegal macro label`

Indicates that the macro label defined in the *DMC command was a legal string syntax, but could not be accepted by the device (see IEEE 488.2, 10.7.3 and 10.7.6.2); for example, the label was too long, the same as a common command header, or contained invalid header syntax.

-274

`macro parameter error`

Indicates that the macro definition improperly used a macro parameter placeholder. See IEEE 488.2, 10.7.3.

-275

`macro definition too long`

Indicates that a syntactically legal macro program data sequence could not be executed because the string or block contents were too long for the device to handle. See IEEE 488.2, 10.7.6.1.

-276

`macro recursion error`

Indicates that a syntactically legal macro program data sequence could not be executed because the device found it to be recursive. See IEEE 488.2, 10.7.6.6.

-277

`macro redefinition`

Indicates that a syntactically legal macro label in the *DMC command could not be executed because the macro label was already defined. See IEEE 488.2, 10.7.6.4.

-278

`macro header not found`

Indicates that a syntactically legal macro label in the *GMC? query could not be executed because the header was not previously defined.

PON and Self Test Hardware Errors

The microwave transition analyzer reports -3XX errors when a problem with the hardware is detected during the PON and SELF TEST routines.

-300

`generic execute`

This is the generic device-dependent error for devices that cannot detect more specific errors. This code indicates only that a Device-Dependent Error as defined in IEEE 488.2, 11.5.1.1.6 has occurred.

-310

`system error`

Indicates that some error, termed "system error" by the device, has occurred. This code is device-dependent.

-311

`memory error`

Indicates that an error was detected in the device's memory. The scope of this error is device-dependent.

-312

`PUD memory lost`

Indicates that the protected user data saved by the *PUD command has been lost.

-313

`calibration memory lost`

Indicates that nonvolatile calibration data used by the *CAL? command has been lost.

-314

`save/recall memory lost`

Indicates that the nonvolatile data saved by the *SAV? command has been lost.

-315

`configuration memory lost`

Indicates that nonvolatile configuration data saved by the device has been lost. The meaning of this error is device-specific.

-330

`self test failed`

Indicates that the self-test routine has failed.

-350

`too many errors`

A specific code entered into the queue in lieu of the code that caused the error. This code indicates that there is no room in the queue and an error occurred but was not recorded.

Program Query/Response Errors

The microwave transition analyzer reports -4XX errors during remote programming. These errors result from a violation of query/response syntax.

-400

`query error`

This is the generic query error for devices that cannot detect more specific errors. This code indicates only that a Query Error as defined in IEEE 488.2, 11.5.1.1.7 and 6.3 has occurred.

-410

`query intr`

Indicates that a condition causing an INTERRUPTED Query Error occurred (see IEEE 488.2, 6.3.2.3); for example, a query followed by DAB or GET before a response was completely sent.

-420

`query unterm`

Indicates that a condition causing an UNTERMINATED Query Error occurred (see IEEE 488.2, 6.3.2.2); for example, the device was addressed to talk and an incomplete program message was received.

-430

`query deadlock`

Indicates that a condition causing a DEADLOCKED Query Error occurred (see IEEE 488.2, 6.3.1.7); for example, both input buffer and output buffer are full and the device cannot continue.

Hardware, Operation, and Mass Memory Errors

The microwave transition analyzer is capable of writing to an external disk or the RAM card. When "disk" is mentioned in this section, it is referring to an external disk or the RAM card.

6200

`chan skew too large`

This error indicates that the channel skew value is too large to perform a measurement. Increase the time span or refer to the appropriate application note.

6201

`zoom span too small`

This error indicates that the zoom processing used requires more data than can be acquired. Refer to the appropriate application note.

6202

`time span too small`

This error indicates that the time span selected was too small for a valid measurement. Check the time span.

6203

`no instrument present`

This error indicates that the microwave transition analyzer cannot find the external synthesizer on the private HP-IB or HP-MSIB system bus. Check the configuration of your instrument set up and the cable connections first. Otherwise, the problem could be with the A8 Microprocessor.

6204

`instrument does not match config or not available`

This error indicates that the external synthesizer does not match expected values or has responded with a reject link (for example, another module is using the synthesizer and has it "locked").

6205

`not active controller`

This error indicates that the HP 70820A is not the active controller. Therefore, either control needs to be passed to the HP 70820A or the system control switch needs to be changed.

6206

`edges required for meas. not found`

This error indicates that the measurement routine could not find one or more edges needed for the requested operation with the current threshold definition.

6207

`"fuzzy" adjustment limits exceeded`

This error indicates that the desired measurement cannot be correctly set up without exceeding the adjustment limits for either the synthesizer frequency, trace length, or time range.

6208

`carrier frequency mixed to poor i.f.`

This error indicates that the carrier frequency is such that its aliased position in the I.F. will probably compromise the measurement.

6209

`oven cold during measurement`

This error indicates that the oven is cold.

6210

`math overrange`

This error indicates that one of the internal DSP chips has limited a data value. this may be caused by an overrange input that has gone undetected.

6211

`channel 1 hardware overrange`

This error indicates that the input signal to channel 1 is too large. Reduce the signal, increase the voltage range, or turn on autoranging.

6212

`channel 2 hardware overrange`

This error indicates that the input signal to channel 2 is too large. Reduce the signal, increase the voltage range, or turn on autoranging.

6213

`delay too large`

This error indicates that an attempt was made to to affect a delay beyond the capabilities of the hardware.

6214

`too many files open`

This error indicates that too many files have been opened for the internal processing to keep track of.

6215

`illegal or undefined mass storage device`

This error indicates that an attempt was made to perform a mass storage operation without a valid mass storage device.

6216

`could not acquire storage link`

This error indicates that an attempt to establish a storage link (usually on HP-MSIB) failed.

6217

`file open failed`

This error indicates that an attempt to open a file has failed. Check to see if the file is valid.

6218

`catalog open failed`

This error indicates that an attempt to access an disk has failed. Check to see if it is formatted or present.

6219

`unexpected end of file`

This error is a result of encountering the physical end-of-file before the logical end-of-file.

6220

`record size too large or small`

This error indicates that the file is either corrupt or the wrong type (for example, loading a recall state into a trace).

6221

`write to file failed`

This error indicates that an attempt to write to a file has failed. Check to see if the file is valid.

6222

`purge file failed`

This error indicates that an attempt to purge a file has failed. Check to see if the file is valid.

6223

`format failed`

This error indicates that an attempt to format a disk has failed. Check to see if the disk is present and not write protected.

6224

`illegal filename`

This error is the result of trying to use a filename which is too long or contains invalid characters.

6225

`file load failed`

This error indicates that an attempt to load a file has failed. Check to see if the file is valid.

6226

`wrong type of a file`

This error is result of the file header not being the expected value for the type of operation in progress (for example, loading a recall state into a trace).

6227

`unrecognized file header`

This error is the result of the file header not matching any of the file types recognized (for example, trying to load 70900 amplitude corrections into 70820).

6327

`time ram:1`

This error indicates that the time RAM did not pass the one's test. The one's test fills RAM with logic ones and then reads the memory and detects any errors. This error probably results from a failure on the A11 Address.

6329

`time cntr,ram`

This error indicates that the time counter or RAM is not working properly. The time RAM counters were reset, and then allowed to run for one second. The time is read and compared to an expected value. This error probably results from a failure on the A11 Address.

6330

`add1`

This error indicates that the add1 line is not working. This error probably results from a failure on the A11 Address board.

6331

`1st time ram num`

This error indicates a failure on the A11 Address.

6332

`tm rm post trig n 0`

This error indicates a failure on the A11 Address.

6333

`tm rm time bet trigs wrng`

This error indicates a failure on the A11 Address.

6334

`too many trigs`

This error indicates a failure on the A11 Address.

6349

`mem addr hi bits`

This error indicates that memory addressing is not linear, high order bits. There is probably a failure on the A9 or A10 Converter or the A11 Address.

6350

`adc bit stuck low: bit`

This error indicates a failure on the A9 or A10 Converter. Check the clock input to Channel 1 and Channel 2 first.

6351

`adc bit stuck high: bit`

This error indicates a failure on the A9 or A10 Converter. Check the clock input to Channel 1 and Channel 2.

6352

`adc missing code num`

This error probably results from a failure on the A9 or A10 Converters.

6353

`mem addr lo bits`

This error indicates that the memory addressing is not linear, high order bits. Check the A9 or A10 Converter or A11 Address.

6359

`gain=3 not working`

This error indicates a failure on the A9 or A10 Converter.

6360

`gain=1 not working`

This error indicates a failure on the A9 or A10 Converter.

6363

`pos slope hyst`

This error indicates a failure on the A9 or A10 Converter. The offset DAC is used as a signal source to test this function.

6364

`pos slope trig`

This error indicates a failure on the A9 or A10 Converter. The offset DAC is used as a signal source to test this function.

6365

`neg slope hyst`

This error indicates a failure on the A9 or A10 Converter. The offset DAC is used as a signal source to test this function.

6366

`neg slope trig`

This error indicates a failure on the A9 or A10 Converter. The offset DAC is used as a signal source to test this function.

6367

`abv lvl trig n reset`

This error indicates a failure on the A9 or A10 Converter. The offset DAC is used as a signal source to test this function.

6368

`abv lvl trig n set`

This error indicates a failure on the A9 or A10 Converter. The offset DAC is used as a signal source to test this function.

6370

`bel lvl trig n reset`

This error indicates a failure on the A9 or A10 Converter. The offset DAC is used as a signal source to test this function.

6371

`bel lvl trig n set`

This error indicates a failure on the A9 or A10 Converter. The offset DAC is used as a signal source to test this function.

6384

`chx miss/broke`

This error indicates a failure on the A9 or A10 Converter as indicated by chx.

6385

`RTC set/broke`

This error indicates that the Real Time Clock may not be correctly set or is broken. Check the A8A1 Control.

6391

`wait state ROM,RAM,MSIB,HPIB,SW,CLK,DSPA,DSP`

This error indicates that the wait state generator on the A8 Microprocessor is generating the incorrect number of wait states. Check the A8A1 Control or the A8A2 ROM/RAM.

6392

`Frac N unlock`

This error indicates that the A15 Fractional N is not locked within the expected time after measurement setup. Check the A15 Fractional N or the A4 LO. Refer also to error code description 7391.

6393

`illegal sample rate`

This error code indicates that the user tried to set an illegal sample rate. Reset the sample rate.

6394

`390 Mhz loop unlocked`

This error indicates a failure on the A4 LO or a problem with the A12 Crystal Reference.

6395

`Frac N Sweep not finished`

This error indicates an internal synthesizer error. Check the A15 Fractional N.

6396

`no external ref`

This error indicates that the external reference is selected and the reference loops are unlocked. Normally this means that the external reference is not connected. If the external reference is present at the A4 LO, the failure is on the A4 LO.

6600

`Illegal Cal Signal`

This error indicates that the internal calibration signal is not within expected limits. Normally this means that it is not connected. If the internal calibration signal is not present at the calibrator output, this error indicates a failure on the A4 LO or a related assembly problem. If the signal is present, the failure may be the A5 IF, the A9 or A10 Converter, or the connecting cables.

6601

`bad factory cal`

This error indicates invalid factory calibration data. The EEROM data is corrupted past recovery. Check the A8A1 Control.

6602

`bad drive flat cal`

This error indicates invalid LO drive calibration data. Calibrate the appropriate channel.

6603

`bad IF flat cal`

This error indicates invalid calibration data. Calibrate the appropriate channel.

6604

`bad insert loss cal`

This error indicates invalid insertion loss calibration data. Calibrate the appropriate channel.

6605

`correction data clipped`

This error indicates internal correction data overflow. Minor adjustment of seconds/division, trace length, or recalibration may be required.

6606

`cal numeric ovrflw`

This error indicates that the computation in calibration data overflowed. Calibrate both channels.

6607

`adc gain cal`

This error indicates that the gain calibration failed for internal ADC. Check the A9 or A10 Converter.

6608

`adc trig cal`

This error indicates that the trigger calibration failed for internal ADC. Check the A9 or A10 Converter.

6609

`bad adc cal data`

This error indicates that the ADC calibration data is invalid. Calibrate the appropriate channel.

6610

`bad user cal data`

This error indicates invalid USER calibration data.

6611

`questionable cal data (ch1)`

This error indicates invalid calibration data for channel 1. Calibrate channel 1.

6612

`Questionable cal data (ch2)`

This error indicates invalid calibration data for channel 2. Calibrate channel 2.

6613

`empty recall reg`

This error indicates that the user attempted to recall data from an empty state register.

6614

`invalid trace`

This error indicates that an operation involving a trace without valid data has been performed.

6615

`signal not found`

This error indicates that `Find Signals` did not find any signals.

6616

`busy`

This error indicates that the microwave transition analyzer is too busy to handle the remote programming request.

6621

`rf carrier adj`

This error indicates that the RF carrier adjustment for subsync/profile failed.

6622

`low i.f.`

This error indicates that the signal mixed too low in the IF, generally a result of too narrow a time span. Change the time span.

6623

`user cal resol`

This error indicates a bad match between the user calibration resolution and the time range. Change the time range.

6624

`no histogram data`

This error indicates that an operation was requested using histogram data when none was available. Check the input.

6625

`bad zero cal data`

This error indicates invalid ZERO calibration data. Calibrate appropriate channel.

6626

`chan align cal error`

This error indicates that the two channels are not close enough in time skew. Calibrate channel 1 or channel 2.

6627

`unable to separate signals`

This error indicates that the table mode setup failed because the signal or harmonic is nearly identical to another signal or harmonic.

6628

`EEROM read waring (recovered)`

This error indicates that the EEROM had one or more recoverable 1-bit read errors.

6629

`Questionable frequency list`

This error indicates that the signal frequency found by the microwave transition analyzer is questionable. Verify the signal frequency.

6630

`freq domain format required`

This error indicates that the last operation invoked requires the source trace to be formatted in the frequency domain.

6631

`freq domain data required`

This error indicates that the last operation invoked requires the source trace to be frequency domain data (for example, a frequency sweep).

6632

`cannot perform operation`

This error indicates that a marker operation cannot be performed (usually marker to center).

6633

`invalid source`

This error indicates that the source trace is invalid for the requested operation (usually marker to center or marker is off).

Series 7000 Errors

7301

`uP bus, wr addr: bit`

This error indicates that the A8 Microprocessor tried to load and then read back the write address counter. This process failed at 'bit'. The sampler clocks from the A4 LO must be ECL (referenced to +5 volts). If they are not, this is an A4 LO. If correct, this is an A11 Address failure. The A16 Motherboard connections to the A9 or A10 Converters may also be suspect.

7302

`rd addr, inh rd: bit`

This error indicates that the A8 Microprocessor tried to set the read/write address so that inhibit read would be set. This process failed with read address at 'bit'. This error probably results from an A11 Address failure.

7303

`shld addr, inh wr: bit`

This error indicates that the A8 Microprocessor tried to set the shield address so that inhibit write would be set. This process failed with read address at 'bit'. This error probably results from an A11 Address failure.

7304

`seg size short: bit`

This error indicates that the A8 Microprocessor tried to set the segment size counters. This error probably results from an A11 Address failure.

7305

`seg size long: bit`

This error indicates that the A8 Microprocessor tried to set the segment size counters. This error probably results from an A11 Address failure.

7306

`inh rd n reset`

This error indicates that the A8 Microprocessor could not reset the inhibit read. This error probably results from an A11 Address failure.

7307

`inh wr n reset`

This error indicates that the A8 Microprocessor could not reset the inhibit write. This error probably results from an A11 Address failure.

7308

`wr addr n count`

This error indicates that the write address counters are not counting or not reaching terminal count. This error probably results from an A11 Address failure or an A9 or A10 Converter failure.

7309

`cps cntr early`

This error indicates that the clock per sample counters timed out too soon, probably resulting from an A11 Address board failure. Another possible cause is a bad clock frequency from the A8A1 Control or the A4 LO.

7310

`cps cntr late`

This error indicates that the clock per sample counters timed out too late. This error probably results from an A11 Address failure. Another possible cause is a bad clock frequency from the A8A1 Control or the A4 LO.

7311

`trig n reset @ uP`

This error indicates that the trigger signal did not reset at the A8 Microprocessor. It can be caused by a trigger signal or an A8 Microprocessor read ckt. This error probably results from an A11 Address failure or an A8A1 Control failure.

7312

`trig n set @ uP`

This error indicates that the trigger signal did not reset at the A8 Microprocessor. It can be caused by a trigger signal or an A8 Microprocessor read ckt. This error probably results from an A11 Address failure or an A8A1 Control failure.

7313

`WAT timeout early`

This error indicates that the writes after trigger counter timed out early. This error probably results from an A11 Address failure. This can also be caused by the A10 Converter or by the convertor boards being switched. See chapter 8 for major assembly locations.

7314

`WAT timeout late`

This error indicates that the writes after trigger counter timed out late. This error probably results from an A11 Address failure. This can also be caused by the A10 Converter.

7315

`wr addr n stop @ EOS`

This error indicates that the write address counter did not stop counting at the end of the segment boundary. This error probably results from an A11 Address failure.

7316

`WAT cntr n stop @ EOS`

This error indicates that the writes after trigger counter did not stop at the end of the segment boundary. This error probably results from an A11 Address failure.

7317

`cps stuck @ 1`

This error indicates that the clock per sample counter stuck at divide by 1. This error probably results from an A11 Address failure.

7318

`cps bad load`

This error indicates that the clocks per sample counter was not loaded correctly. This error probably results from an A11 Address failure.

7319

`WAT n wait for trig`

This error indicates that the writes after trigger counter started counting before the trigger signal occurred. This error probably results from an A11 Address failure.

7320

`WAT n strt by trig`

This error indicates that the writes after trigger counter was not counting after trigger occurred. This error probably results from an A11 Address failure.

7321

`WAT n reload`

This error indicates that the writes after trigger counter did not reload. This error probably results from an A11 Address failure.

7322

`wr addr, WAT lsbs`

This error indicates that there is an error in the least significant bits of the writes after trigger counter. This error probably results from an A11 Address failure.

7323

`trace n wait for trig`

This error indicates that the trace started before trigger. This error probably results from an A11 Address failure.

7324

`trace n strt by trig`

This error indicates that the trace did not start when the trigger occurred. This error probably results from an A11 Address failure.

7325

`timebase switch-early`

This error indicates that the clocks per sample counter is loaded with the alternate value earlier than expected. This error probably results from an A11 Address failure or an A10 Converter failure.

7326

`timebase switch-late`

This error indicates that the clocks per sample counter is loaded with the alternate value later than expected. This error probably results from an A11 Address failure or an A10 Converter failure.

7382

`missing adc clock`

This error indicates that the microwave transition analyzer cannot detect a clock on the internal adc(s). The clock to the A9 and A10 Converters should be ECL (referenced to +5 volts). If this is verified, check for an A9 or A10 Converter failure. Otherwise, the problem was caused by an A4 LO failure.

7383

`slow adc clock`

This error indicates that the clock for the internal adc(s) is slow. The clock to the A9 and A10 Converters should be ECL (referenced to +5 volts). If this is verified, this error is probably the result of an A9 or A10 Converter failure. Otherwise, it is an A4 LO failure.

7386

`: Memory Overflow`

This error indicates that the internal heap is too small.

7389

`HP_MSIB error`

This error indicates that the power-on check for the MSIB failed. This can be caused by a connector problem or an A8A1 Control failure.

7390

`via`

This error indicates an A8A1 Control failure.

7391

`: Frac N is not responding`

This error indicates that the A8 Microprocessor cannot talk to the A15 Fractional N. This failure may be caused by a failure with the A15 Fractional N or the A8A1 Control.

7394

`trace lengths do not match`

This error indicates that the two traces are not the same size or are not completely overlapping.

7395

`operand mismatch`

This error indicates that the two traces are not the same type.

7396

`operand domain error`

This error indicates that this type of operation on this type of trace is invalid.

7397

`unsupport transform`

This error indicates that the current code does not support the requested trace transformation.

7398

`recursive function`

This error indicates that the recursive function is defined.

7400

`dsp chip(dspx overlay 0: up/dn load)`

This error indicates that the DSP chips on the A9 and A10 Converter boards did not pass the self-test. This failure may be caused by the A9 or A10 Converters, the A8A1 Control, or connections through the A16 Motherboard.

7401

`channel 1 I.F. path bad`

This error indicates that the signal path for channel 1 is not intact.

7402

`channel 2 I.F. path bad`

This error indicates that the signal path for channel 2 is not intact.

7403

`I.F. cables reversed`

This error indicates that the cables for channel 1 and channel 2 signal paths are backwards.

7617

`Data(Battery) failure`

This error indicates that some or all of the data in the battery backed-up RAM has been lost. Replace the battery. This repair requires the same retesting on the Repair vs Rework Matrix (Table 10-2) as changing the A8A1 Control.

7618

`RAM address buss`

This error indicates an A8A1 Control failure or an A8A2 ROM/RAM failure.

7619

`RAM data Error`

This error indicates either an A8A1 Control failure or an A8A2 ROM/RAM failure.

7620

`ROM checksum Error`

This error indicates an A8A2 ROM/RAM failure.

7621

`Internal GND`

This error indicates that an internal ground is not at zero volts. This error may be caused by incorrect calibration data or by an A5 IF failure.

7622

`Write to EEROM failed`

This error indicates an A8A1 Control failure.

7624

`EEROM read failure`

This error indicates an A8A1 Control failure.

Floating Point Errors

The microwave transition analyzer generates the following floating point math errors.

8000

`zero div`

Indicates that an attempt to divide by zero was made (usually an internal firmware failure).

8001

`real ovflw/undflw`

Indicates that the result of a floating point operation exceeds the legal range for the data format.

8002

`log(0)`

Indicates that an attempt to compute log(0) was made (usually an internal firmware failure.

8003

`log(-x)`

Indicates that an attempt to compute log(x) was made, where x<0.

8004

`int ovflw`

Indicates that math operation results exceed the range for a 32-bit integer (that is, integer math overrange).

8005

`sqrt err`

Indicates that an attempt to compute sqrt(x) was made, where x<0.

8006

`x mod 0`

Indicates that an attempt to modulo by zero was made.

8007

`acs(x), abs(x) > 1`

Indicates that an attempt to compute arccos(x) was made, where abs(x)>1.

8999

`real undrflw`

Indicates that the result of a floating point operation exceeds the legal range for the data format.

Hardware Errors

Series 9000 Errors

9101

`Protocol`

This error indicates that an illegal MSIB protocol command was received. Review the command given.

9380

`Stack Overflow`

This error indicates that an internal stack overflowed its allocated size. The probable cause for this problem is a firmware failure.

9381

`Internal`

This error indicates a generic internal firmware error.

9387

`use of non-acq resource`

This error indicates an internal firmware error.

9388

`parser error`

This error indicates an internal firmware error.

9390

`illegal DSP instr`

This error indicates that the DSP overlay does not contain instruction.

9391

`instr shell error`

This error indicates an internal SHELL firmware error.

A

Trace Model

The HP 70820A, depending on the option, uses one of two different trace models for front panel operation. There are no remote programming differences between the two trace models.

- Normal Trace Model: Used for the H10 option. All traces (that is, channels, memories, and functions) are equally emphasized from the front panel. For screen annotation, functions are labeled as FN1 through FN4.
- Four Trace Model: Used for non-H10 options. Four of the traces are emphasized from the front panel, FN1 through FN4. For screen annotation, functions are labeled as TR1 through TR4.

B

Passing Control

The HP 70820A currently does not support controller capability on the "public" HP-IB (for example, the HP-IB that is used to program the HP 70820A) and hence does not support passing control on the public HP-IB.

The HP 70820A does support pass control on the "private" HP-IB (for example, the HP-IB that is used to control the synthesizers). The typical use for passing control is to share a synthesizer between the HP 70820A and another instrument or the controlling computer.

The basic sequence for passing control back and forth is:

- Define the address for the HP 70820A to pass control back to when it is finished.
- Execute the command(s) that the HP 70820A requires control of the HP-IB for
- Pass control to the HP 70820A

Note The HP 70820A will wait up to 1 second to become the active controller.

- Wait for the HP 70820A to pass control back.

The condition for the HP 70820A to pass back control is the asserting of the OPERATION COMPLETE bit in the STANDARD EVENT STATUS REGISTER (see *ESR).

The only "tricky" part is knowing when the HP 70820A requires control of the HP-IB. The commands that require control of the HP-IB after a program message terminator (that is, line feed) has been received are (that is, you may send commands between these and the program message terminator):

- SOURce:CONFigure:ADDRess
- SOURce:CONFigure:BUS
- SOURce:CONFigure:COLumn
- SOURce:CONFigure:ROW
- SOURce:CONFigure:TYPE

The commands that require control of the HP-IB when the command is executed are (that is, do not send any commands between these and the pass control):

- SOURce:FREQuency:CW
- SOURce:POWer:LEVel
- SOURce:POWer:STATe
- SOURce:PULSe:STATe

The commands that require control of the HP-IB, but at no particular time (within the 1 second timeout), are:

- DIGitize (for a frequency or power sweep)
- RUN (for a frequency or power sweep)

The following example defines the synthesizer, sets up a measurement, and executes the measurement. The public and private HP-IBs are connected together for this example.

```
10    ASSIGN @Hta TO 710              ! I/O assignments
20    ASSIGN @Cntl TO 711
30    ASSIGN @Synth TO 719
40    Sc=7
50    !
60    STATUS Sc,3;Stat_and_addr       ! Find computer's address
70    My_addr=BINAND(Stat_and_addr,31)
80                                    !
90    OUTPUT @Hta;"*RST;*OPC?"
100   ENTER @Hta;A$
110   OUTPUT @Hta;"SWEEP:MODE ASIN;"
120   OUTPUT @Hta;"STOP"
130                                   ! "Play" with the
                                        synthesizer
140   FOR Freq=100 TO 1000 STEP 50    ! (just to show it can be
                                        done)
150     OUTPUT @Synth;"FREQ:CW ";Freq;" MHZ"
160     WAIT .1
170   NEXT Freq
180                                   ! Define the synthesizer
190   OUTPUT @Hta;"PCB ";My_addr;";SOURCE:CONF:TYPE HP836XX;ADDR
      19;BUS HPIB;*OPC;"
200   Pass_control(Sc,@Cntl)
210                                   ! Turn the RF on
220   OUTPUT @Hta;"PCB ";My_addr;";*OPC;SOURCE:POWER:STATE ON"
230   Pass_control(Sc,@Cntl)
240                                   ! Set the power level
250   OUTPUT @Hta;"PCB ";My_addr;";*OPC;SOURCE:POWER:LEVEL -15 DBM"
260   Pass_control(Sc,@Cntl)
270                                   ! Set up a freq sweep and GO
280   OUTPUT @Hta;"SWEEP:TYPE FREQ;"
290   OUTPUT @Hta;"SWEEP:FREQ:START 100MHZ; STOP 200MHZ"
300   OUTPUT @Hta;"ACQ:POIN 101"
310   OUTPUT @Hta;"*OPC?"
320   ENTER @Hta;A$
330   OUTPUT @Hta;"PCB ";My_addr
340   OUTPUT @Hta;"RUN;*OPC;"
350   Pass_control(Sc,@Cntl)
360                                  ! Undefine the synthesizer
370   OUTPUT @Hta;"PCB ";My_addr;";*OPC;SOURCE:CONF:TYPE NONE;"
380   Pass_control(Sc,@Cntl)
390   END
400   !
410   !-----------------------------------------------
420   !
430   SUB Pass_control(Sc,@Addr)
440      PASS CONTROL @Addr           ! Pass active control to @Addr
450      !
460      REPEAT                       ! Wait until the computer is
470        STATUS Sc,3;Stat_and_addr  ! the active controller
                                        again.
480      UNTIL BIT(Stat_and_addr,6)=1
490   SUBEND
```

C

Vector Voltmeter Operation

Here are 2 example programs for setting up vector voltmeter style measurements.

```
10   ! ***************************************************************
20   ! The following instructions set up the same measurement
30   ! obtained with the "VECTOR VOLTAGE" key in the "States" menu
40   !
50   ASSIGN @Nta TO 711
80   OUTPUT @Nta;"*RST;*WAI;"
90   OUTPUT @Nta;"BLANK FUNC1"           ! Trace 1:      OFF
100  OUTPUT @Nta;"DISP:GRAT OFF"         ! Graticule:    OFF
110  OUTPUT @Nta;"SWE:STR ON"            ! Signal track: ON
120  OUTPUT @Nta;"SWE:TIME:CYCL ON"
130  OUTPUT @Nta;"SWE:TIME:RANGE 2 CYCLES"
140  OUTPUT @Nta;"TABL:SOUR CH1CH2"      ! Table input:      CH1/CH2
150  OUTPUT @Nta;"TABL:HARM 1"           ! Table harmonics:  1
160  OUTPUT @Nta;"TABL:INCL OFF  "       ! Include mode:     OFF
170  OUTPUT @Nta;"TABL:PHAS ON"          ! Show phase
180  OUTPUT @Nta;"TABL:AMPL ON"          ! Show amplitude
190  OUTPUT @Nta;"TRIG:SOUR CHAN2"       ! Trigger source is CH2
200  END

10   ! ***************************************************************
20   ! This routine sets up a vector voltmeter mode that used
30   ! the traces/ffts/markers.
40   !
50   ASSIGN @Nta TO 711
70   OUTPUT @Nta;"*RST;*WAI;"
80   OUTPUT @Nta;"SWE:SIN ON"            ! Single shot:   ON
90   OUTPUT @Nta;"SWE:TIME:TRAN ON"      ! Translated:    ON
100  OUTPUT @Nta;"SWE:TIME:CYCL ON"      ! Cycles mode:   ON
110  OUTPUT @Nta;"SWE:TIME:RANGE 127 CYCL"
120  OUTPUT @Nta;"FUNC1:DEF (CHAN1)"
130  OUTPUT @Nta;"FUNC2:DEF (CHAN2)"
140  OUTPUT @Nta;"WAV:SOUR FUNC1;DOM OFF,MAGN;MAGN LIN,LOG"
150  OUTPUT @Nta;"WAV:SOUR FUNC2;DOM OFF,MAGN;MAGN LIN,LOG"
160  OUTPUT @Nta;"MARK:FOR VEC"          ! Markers readout vector results
170  OUTPUT @Nta;"MARK1:SOUR FUNC1"
180  OUTPUT @Nta;"MARK2:SOUR FUNC2"
190  OUTPUT @Nta;"MARK1:POS 1 CYCL"      ! Position markers at the 1st
200  OUTPUT @Nta;"MARK2:POS 1 CYCL"      ! harmonic... Signal track and
210  OUTPUT @Nta;"MARK:DELT 0 CYCL"      ! cycles mode maintain marker pos.
220  OUTPUT @Nta;"SWE:STR ON"            ! Signal track: ON
230  END
```

D

Math Algorithms

Definitions

The components of the trace are referred to as:

- *real* - the real portion of the trace data array.
- *imag* - the imag portion of the trace data array.
- *DC* - an offset term stored in the trace header.
- real trace - a trace with an undefined imag component.

Note $real + 0 * j$ is not a real trace

- complex trace - a trace with a defined imag component

Note $x * j$ AND $0 + x * j$ are complex traces

- Scalar trace - a "normal" real trace ... that is, there is no special scaling such as that for LOGMAG or PHASE in the format menu.

Unless otherwise noted, the following "rules" will be applied:

- For real time traces, DC is considered part of real (for example, for an operation such as sqrt(x), the operation is sqrt(DC+real)).
- For real frequency trace, DC is defined as 0 so that $(DC + real)$=$(real)$.
- For complex traces, DC is considered a separate term and is either ignored (that is, left unchanged) or treated as indicated.
- In a 2 operand operation, if either of the operands is complex, the other operand is forced to be complex (by using zero fills).
- In a 2 operand operation, if either of the operands is 32 bits, the result is 32 bits.
- Unless otherwise noted, the window used for math operations is selected by the WINDow:MATH command.
- The math system only operates on real or complex data. Any data that is in another format (such as phase or magnitude) will be converted to complex data before any operations are applied.

Operations

- + (add)
  - operands are real or complex traces
    $(real1 + j * imag1) + (real2 + j * imag2)$
    $DC1 + DC2$

- − (negate)
  - operand is real or complex trace
    $-(real + j * imag)$
    $-DC$

- − (subtract)
  - operands are real or complex traces
    $(real1 + j * imag1) - (real2 + j * imag2)$
    $DC1 - DC2$

- * (multiply)
  - operands are real traces
    $(real1 + DC1) * (real2 + DC2)$
  - operands are complex traces
    $(real1 + j * imag1) * (real2 + j * imag2)$
    $DC = 0$

- / (divide)
  - operands are real traces
    $real = (real1 + DC1) / (real2 + DC2)$
  - operands are complex traces
    $(real1 + j * imag1) / (real2 + j * imag2)$
    $DC = 0$

- AC()
  - operand is real trace
    * time domain input
      DC component removed from *real*, the DC component of real is computed with a hanning window and DFT.
      $DC = 0$
    * freq domain input
      $DC = 0$
  - operand is complex trace
    $DC = 0$

Note The *DC* component is assumed to have already been removed from real.

- AM()
  - operand is real or complex trace
    $MAGN(ANALY(xxx))$

Note This function is provided for backward compatibility with previous versions of code and is not accessible from the front panel.

- ANALY()
  - operand is real trace
    - time domain input
      DC component removed from real and added to *DC* in header Hilbert transform pair generated from real using FFT with hamming window (trace length is zero padded to a power of 2).
    - operator performs a complex conjugate of the result if the input data was low side mixed.
    - freq domain input: no operation
  - operand is complex trace: no operation

Note Because an FFT is used, the best results are achieved with an input trace length that is a power of 2.

Note Although ANALY() uses the same algorithm for generating the Hilbert transform pair as the analytic switch (SWEep:TIMe:ANALytic), the two methods will not generally give the same results. The reason for this is the analytic switch increases the measurement size, generates the Hilbert transform pair, then deletes the excess data. For this reason, the analytic switch is generally preferable to the ANALY() function.

- ATAN()
  - operand is real trace
    2 quadrant arctan of *real* + *DC*
  - operand is complex trace
    2 quadrant arctan of *imag/real*

Note −PI/2 <= 2 quadrant atan() <= PI/2
This does not produce the same results as the format PHASE.

– PHASE
- scaling is 0.01 or 0.0001 deg per bit
- result is 4 quadrant arctan()

– atan()
- result is a unitless scalar real trace
- result is 2 quadrant arctan()

■ CHOP()

– operand is real trace
If (*real1 + DC1*) is less than *(real2 + DC2)*, then return 0.
Otherwise return operand 1 unchanged.

–operand is complex trace
If magnitude of operand 1 is less than the magnitude of operand 2,
then return 0. Otherwise return operand 1 unchanged.

This allows filtering out of noise and other low level components. For example:
IFFT(FFT(CH1): CHOP: 0.001)
will remove any spectral components below −60 dB.

■ CONST()

The input trace is forced to be a "constant". Namely, the time or frequency domain checks for the trace are disabled. This allows operations such as *CHAN1 + FFT(CHAN1)* which would normally result in a domain mismatch error.

■ CONV()

– operand is real trace
$(real1 + DC1)$ *conv* $(real2 + DC2)$
$DC = 0$

– operand is complex trace
$(real1 + j * imag1)$ *conv* $(real2 + j * imag2)$
$DC = 0$

The convolution is done by zero padding the smaller trace up to the size of the larger trace, FFT or DFT, multiply the results, IFFT or IDFT, unwindow.

■ CORR()

– operand is *real* trace
$(real1 + DC1)$ *corr* $(real2 + DC2)$
$DC = 0$

– operand is complex trace
$(real1 + j * imag1)$ *corr* $(real2 + j * imag2)$
$DC = 0$

The correlation is done by zero padding the smaller trace up to the size of the

larger trace, FFT or DFT, complex conjugate the second data set, multiply the results, IFFT or IDFT.

- DB()

  – operand is *real* or complex trace
  $20 * lgt(magn()$ (see MAGN())

Note This does not produce the same results as the format LOGMAG.

  – LOGMAG
    - If the units of the input trace are volts, the result is dBm. If the units of the input trace are unitless, the result is dB. -scaling is 0.01 or 0.0001 dB per bit.

  – DB()
    - ignores units of input trace
    - result is a unitless scalar real trace

- DC()

  – time domain input
    * operand is real trace
      real = DC component of real trace + DC
      (NOTE: the DC component of real is computed with a hanning window and a DFT)
      $DC = 0$

    * operand is complex trace
      real=DC
      (NOTE: the DC component is assumed to have already been removed from *real*)
      $DC = 0$

    result is a real trace

  – freq domain input: no operation

- DEG()

  – operand is real trace
    4 quadrant arctan of *real + DC*

Note This will generate just 2 values, 0 and 180.

  – operand is complex trace
    "unwrapped" 4 quadrant arctan of $real + j * imag$

Note The "unwrap" operation removes phase discontinuities by adding multiples of 360. For example the data sequence 170, 173, 176, 179, −178, −175 ... , will change to: 170, 173, 176, 179, 182, 185 ...

Note This does not produce the same results as the format PHASE.

– PHASE
- scaling is 0.01 or 0.0001 deg per bit
- result is 4 quadrant arctan()

– atan()
- result is a unitless scalar *real* trace
- result is an "unwrapped" 4 quadrant arctan()

■ DIFF()

– operand is real or complex trace
$trace[n] = (trace[n + int((a + 1)/2)] - trace[n - int(a/2)])/a$
$DC = 0$

Note A is the math aperture, set by FUNCx:APERture.

Note For j < 0 $trace[j] = trace[0]$, for j $>$ N $trace[j] = trace[N]$.

Result matches the input type and size.

■ DFT()

– operand is real or complex trace
window the data and DFT
result is a complex trace

Note DFT() will always generate a double-sided spectrum.

Note The frequency range of the result is taken from the frequency "tags" of the input rather than computed from the time range of the input. This is due to various modes of operation in with the two that are not equivalent (most notably, translated sweeps and RF pulse measurements).

Note DFT() is significantly slower than FFT().

- D/DX()

  – operand is real or complex trace
  derivative of $real + j * imag$
  $DC = 0$

  Derivative is done by: FFT, multiply by jw, IFFT

Note Because an FFT is used, the best results are achieved with an input trace length that is a power of 2.

- E

  $constant = 2.71828182$
  result is a real trace

- EXPJ()

  – operand is real trace
  $exp(j * 2 * PI * (real + DC))$
  result is a complex trace

  – operand is complex trace
  undefined, generates an error

- FFT()

  – operand is real or complex trace
  window the data, zero padded to a power of 2, FFT
  result is a complex trace

Note FFT() will always generate a double-sided spectrum.

Note The frequency range of the result is taken from the frequency "tags" of the input rather than computed from the time range of the input. This is due to various modes of operation in with the two are not equivalent (most notably, translated sweeps and RF pulse measurements).

- FM()

  – operand is *real* trace
  $real = (real[n + int((a + 1)/2)] - real[n - int(a/2)]) / dt + fstart$
  $DC = 0$

  – operand is complex trace

$temp = unwrapped\ 4\ quadrant\ arctan\ (imag/real)$
$real = (temp[n + int((a + 1)/2)] - temp[n - int(a/2)])\ /\ dt$ + fstart
$DC = 0$

result is a real trace, vert units are set to freq.

Note df = trace length / frequency span (if frequency span is 0, then df is set to a).

Note fstart is the value returned by WAVeform:FORigin.

Note a is the math aperture, set by FUNCx:APERture.

Note For j < 0 trace[j] = trace[0], for j > N trace[j] = trace[N].

Note Input trace must have a horizontal axis of time.

- FREQ

  Generates a frequency ramp. The characteristics of the frequency ramp are determined by the measurement data in CHANNEL1.

  – time sweep: a frequency ramp is generated that matches the frequency axis of a FFT of the last measurement. The length of the data will be 2^N where 2^N >= length of CHANNEL1.

  – power sweep: a flat ramp (that is, 0 slope) is generated where the value of the data is equal to the CW frequency of the source. The length of the result is set to the length of CHANNEL1.

  – frequency sweep: a frequency ramp is generated that matches the frequency axis of the data in CHANNEL1. The length of the result is set to the length of CHANNEL1.

  The result is a real trace.

- IDFT()

  – operand is real or complex trace
  IDFT performed, unwindow
  result is a complex trace

Note If the input data is single sided (for example, the results of a network sweep), the negative frequencies will be created by:

$real[N - i] = real[i]$
$imag[N - i] = -imag[i]$

Because of this, $DFT(IDFT(data)) <>$ data for single-sided spectrum.

Note The time range of the result is taken from the time "tags" of the input rather than computed from the frequency range of the input. This is due to various modes of operation in with the two are not equivalent (most notably, translated sweeps and RF pulse measurements).

Note IDFT() is significantly slower than IFFT().

- IFFT()
  - operand is real or complex trace
    zero padded to a power of 2, perform IFFT, unwindow
    result is a complex trace

Note If the input data is single sided (for example, the results of a network sweep), the negative frequencies will be created by:

$real[N - i] = real[i]$
$imag[N - i] = -imag[i]$
Because of this, $DFT(IDFT(data)) <>$ *data* for single-sided spectrum.

Note The time range of the result is taken from the time "tags" of the input rather than computed from the frequency range of the input. This is due to various modes of operation in with the two are not equivalent (most notably, translated sweeps and RF pulse measurements).

- IMAG()
  - operand is real trace
    $real = 0$
    $DC = 0$
    result is a real trace

  - operand is complex trace
    $real = imag$
    $DC = 0$
    result is a real trace

- INTEG()
  - operand is *real* trace

integration of $real + DC$
$DC = 0$

– operand is complex trace
integration of $real + j * imag$
$DC = 0$

- Integration is done by: FFT, divide by jw, IFFT

Note Because an FFT is used, the best results are achieved with an input trace length that is a power of 2.

Note real[0] is added in as a ramp after the previous process to avoid division by 0.

- J

  $constant = 0 + j$
  result is a complex trace

- MAGN()

  – operand is real trace
  $sqrt((DC + real)^2)$ ... same as $abs(DC + real)$
  $DC = 0$

  – operand is complex trace
  $sqrt(real^2 + imag\ ^2)$
  $DC = 0$

  result is a real trace with the same units as the input

- MOD

  – operand is real trace
  $(DC + real)\ modulo\ (constant)$
  $DC = 0$

  – operand is complex trace
  $(real)\ modulo\ (constant)$
  $(imag)\ modulo\ (constant)$
  $DC = 0$

Note For A MOD B, B must be a *real* constant.

- N

ramp - generates a real trace with
real[n] = n
where 0 <= n < trace length

- PI

  constant = 3.14159265
  result is a real trace

- PM()

  – operand is real or complex trace
  $DEG(ANALY(xxx))$

Note This function is provided for backward compatibility with previous versions of code and is not accessible from the front panel.

- POWER

  Generates a power ramp. The characteristics of the power ramp are determined by the measurement data in CHANNEL1.

  – time sweep: a flat ramp (that is, 0 slope) is generated where the value of the data is equal to the power level of the source. The length of the data will be 2^N where 2^N >=length of CHANNEL1.

  – power sweep: a power ramp is generated where the ramp equals the horizontal axis of CHANNEL1. The length of the result is set to the length of CHANNEL1.

  – frequency sweep: a flat ramp (that is, 0 slope) is generated where the value of the data is equal to the power level of the source. The length of the result is set to the length of CHANNEL1.

  The result is a real trace.

- REAL()

  – operand is real trace
  no operation
  – operand is complex trace *real*
  (that is, imag is "undefined") *DC* is unchanged
  result is a real trace

- SIGN()

  –operand is real trace
  If *(real + DC) > 0*, then returns +1
  If *(real + DC) < 0*, then returns -1
  If *(real + DC) = 0*, then returns 0.

  – operand is complex trace
  If *real > 0*, then returns +1

If $real < 0$, then returns -1
If $real = 0$, then returns 0.

- SQRT()
  - operand is real trace
    $sqrt(abs(DC + real))$
    result is a real trace
  - operand is complex trace
    $sqrt(magn()) * exp(j * atan(real,imag)/2)$
    $DC = 0$
    result is a complex trace

- SUM()
  - operand is real trace
    $real[n] = sum(real[i] + DC)$
    $DC = 0$
    0 <= i <= n
  - operand is complex trace
    $real[n] = sum(real[i])$
    $imag[n] = sum(imag[i])$
    0 <= i <= n

- TD()
  - operand is real trace
    $real = (real[n + int((a + 1)/2)] - real[n - int(a/2)]) / df$
    $DC = 0$
  - operand is complex trace
    $temp = unwrapped\ 4\ quadrant\ arctan\ (imag/real)$
    $real = -(temp[n + int((a + 1)/2)] - temp[n - int(a/2)]) / df$
    $DC = 0$

    result is a real trace, vert units are set to freq.

Note df = (trace length − 1) / frequency span (if frequency span is 0, then df is set to a).

Note a is the math aperture, set by FUNCx:APERture.

Note For j < 0 trace[j] = trace[0], for j > N trace[j] = trace[N].

Note Input trace must have a horizontal axis of frequency.

- TIME

  Generates a time ramp. The characteristics of the time ramp are determined by the measurement data in CHANNEL1.

  – time sweep: a time ramp is generated where the ramp equals the horizontal axis of CHANNEL1. The length of the result is set to the length of CHANNEL1. The length of the data will be 2^N where 2^N >= length of CHANNEL1.

  – power sweep: a flat ramp (that is, 0 slope) is generated where the value of the data is equal to the CW frequency of the source. The length of the data will be where 2^N 2^N >= length of CHANNEL1.

  – frequency sweep: a frequency ramp is generated where the ramp equals the horizontal axis of a FFT of CHANNEL1. The length of the data will be 2^N where 2^N >= length of CHANNEL1.

The result is a real trace.

- VS

  – operands are real or complex trace
  Interleaves the points of the two operands, that is
  real1[n] = x0,x1,x2,x3 ...
  real2[n] = y0,y1,y2,y3 ...
  results in
  real[n] = x0,y0,x1,y1,x2,y2,x3,y3 ...

  When displayed, the even points will be considered X values, the odd points will be considered Y values (for example, this produces a Y vs X display).

E

Median Smoothing

Median smoothing is a process which operates on a stored trace, producing a new trace. For a properly chosen smoothing width, the new trace retains the general shape of the original but reduces the local variation (noise). Moving average operators are also sometimes used to smooth a trace. The HP 70820 uses median smoothing instead of a moving average for the following reason. Sharp transitions in the waveform are features commonly found in, but not limited to, the display of time signals. A moving average operator distorts these transitions by reducing the slope in proportion to the smoothing width—the waveform is essentially lowpass filtered. An advantage of median smoothing is that fast transitions are preserved, even as the noise is significantly reduced. The HP 70820 implements median smoothing in the following way:

```
Let the original trace be denoted as:       in(n)      0 <= n < N
and the newly created trace as:            out(m)      0 <= m < N

Let the smoothing width be represented by the variable "w".

In general:

 out(n) = median { in(i) ; n - (w-1)/2 <= i <= n + (w-1)/2 } w odd

        = median { in(i) ; n+1 - w/2 <= i <= n + w/2 }       w even

 where median {} is computed as:

  (1) sort the values in {} from largest to smallest -- let this
      list be represented by  x(j) ; 0 <= j < w-1   where x(0) is
      the largest value.

  (2)  return the value:    x((w-1)/2)  w odd
                            x(w/2)      w even



For values of n where      n < (w-1)/2          w odd
                           n < w/2 - 1          w even

    use w' in place of w, where w' =  2n + 1


For values of n where      n > N-1 - (w-1)/2    w odd
                           n > N-1 - w/2        w even

    use w' in place of w, where w' =  2(N-n) - 1
```

F

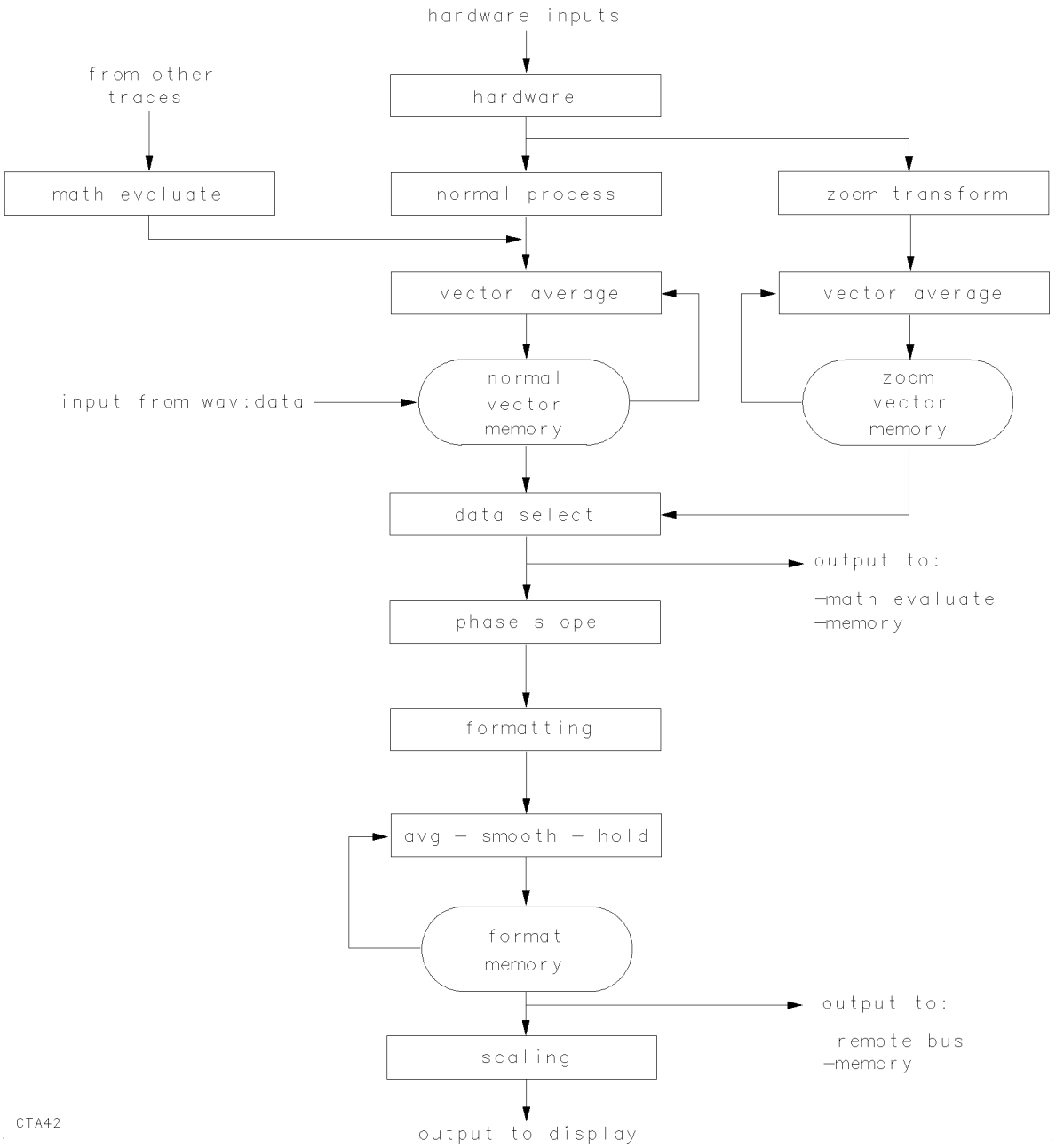
Data Flow

The figure on the following page graphically illustrates the flow of data through the HP 70820. To control the various processing operations:

- HARDWARE: use ACQuire, CHAN1, CHAN2, SWEep subsystems
- MATH EVALUATE: use FUNCx subsystem
- ZOOM TRANSFORM: use SWEep: ZOOM:xxx
- VECTOR AVERAGE: use WAVeform: TYPE, COUNt or ACQuire: TYPE, COUNt
- DATA SELECT: use CHANx | FUNCx: ZOOM: STATe, TIME, FREQuency
- PHASE SLOPE: use WAVeform: DELay, FSHift, PSLope
- FORMATTING: use WAVeform: DOMain, MAGNitude
- AVG | SMOOTH | HOLD: use WAVeform: TYPE, COUNt or ACQuire: TYPE, COUNt
- SCALING: use DISPlay: CHANx | FUNCx | WMEMx: SCALe, OFFSet

The operations and memories associated with the zoom transform exist only for CHAN1 and CHAN2.

Memory store operations copy data directly into normal vector memory. Memories do not support any averaging operations.


hardware inputs
from other traces
hardware
math evaluate
normal process
zoom transform
vector average
vector average
input from wav:data
normal vector memory
zoom vector memory
data select
output to:
–math evaluate
–memory
phase slope
formatting
avg – smooth – hold
format memory
output to:
–remote bus
–memory
scaling
output to display
CTA42

F-2 Data Flow

G

Menu Key to Remote Cross Reference

This listing summarizes the softkeys and their related programming commands found under each menu key. Italicized commands next to the softkeys indicate commands that are required to copy the key with DISPlay:MENU:COPY. In general, this is the same as the short form of the command.

All softkeys are indicated by a bullet and are listed in the order in which they appear when you step through the menus. The command(s) listed directly beneath each softkey provides the closest functionality to that softkey.

`Main` Menu Key

- `sweep:` *(SWE:TYPE)*
  SWEep:TYPE
- `SEC/DIV` *(SWE:TIME:RANG)*
  SWEep:TIME:RANGe
- `DELAY` *(SWE:TIME:DEL)*
  SWEep:TIME:DELay
- `STARt` *(SWE:FREQ:STAR, SWE:POW:STAR)*
  SWEep:FREQuency:STARt
  SWEep:POWer:STArt
- `STOP` *(SWE:FREQ:STOP, SWE:POW:STOP)*
  SWEep:FREQuency:STOP
  SWEep:POWer:STOP
- `CENTER`
  no related command
- `SPAN`
  no related command
- `signal FREQ` *(SWE:SIG)*
  SWEep:SIGnal
- `carrier FREQ` *(SOUR:FREQ:CW)*
  PULSe:CARRier
- `source FRQ/PWR` *(SOUR:FREQ:CW)*
  SOURce:FREQuency:CW
  SOURce:POWer:LEVel
- `carrier FRQ/PWR` *(SOUR:FREQ:CW)*
  SOURce:FREQuency:CW
  SOURce:POWer:LEVel
- `RF out ON/OFF` *(SOUR:POW:STAT)*

SOURce:POWer:STATe

- `noisflt ON/OFF` *(SWE:TIME:FIL:STAT)*

  SWEep:TIME:FILter:STATe
  SWEep:TIME:FILter:BWIDth
- `NOISE FILTER` *(SWE:TIME:FIL:STAT)*

  SWEep:FREQuency:IFB
  SWEep:POWer:IFB
- `sig trk ON/OFF` *(SWE:STR:STAT)*

  SWEep:STRack
- `single shot`

  - `sglshot ON/OFF` *(SWE:SIN)*

    SWEep:SINgle
    SWEep:TIME:SRATe
  - `big trc ON/OFF` *(SWE:TIME:OVER:STAT)*

    SWEep:TIME:OVERsweep:STATe
  - `detect:` *(SWE:TIME:OVER:DET)*

    SWEep:TIME:OVERsweep:DETect
  - `display SEC/DIV` *(SWE:TIME:OVER:DEC)*

    SWEep:TIME:OVERsweep:DECimate
  - `display DELAY` *(SWE:TIME:OVER:OFFS)*

    SWEep:TIME:OVERsweep:OFFset
  - `filter ON/OFF` *(SWE:TIME:OVER:FILT:STAT)*

    SWEep:TIME:OVERsweep:FILTer:STATe
- `DWELL`

  SWEep:FREQuency:DWELl
- `FIND SIGNALS` *(AUT)*

  AUTo
- `FIND CARRIER` *(AUT)*

  AUTo
- `signal list`

  - `FIND FREQS`

    TABLe:FIND
  - `INCLUDE HARMNCS`

    TABLe:INCLude
  - `EDIT`

    SWEep:SIGnal
  - `ADD`

    SWEep:SLISt
  - `DELETE`

    SWEep:SLISt
- `sweep labels` *(SWE:TIME:CYCL)*

  - `SEC/DIV, DELAY`

    SWEep:TIME:CYCLes

- `CYCLES, DELAY`
  SWEep:TIME:CYCLes
- `START, STOP`
  no related command

- `sweep options`
  - `translt ON/OFF` *(SWE:TIME:TRAN)*
    SWEep:TIME:TRANslate
  - `CHAN1 VID/RF` *(SWE:TIME:FIL:VID CHAN1)*
    SWEep:TIME:FILter:VIDeo
  - `CHAN2 VID/RF` *(SWE:TIME:FIL:VID CHAN2)*
    SWEep:TIME:FILter:VIDeo
  - `freq mult numer` *(SWE:FREQ:MULT)*
    SWEep:FREQuency:MULTiplier
  - `freq mult denom` *(SWE:FREQ:DIV)*
    SWEep:FREQuency:DIVisor
  - `freq offset` *(SWE:FREQ:OFFS)*
    SWEep:FREQuency:OFFSet

`Trigger` Menu Key

- `trg is:` *(TRIG:TYPE)*
  TRIGger:TYPE
- `input:` *(TRIG:SOUR)*
  TRIGger:SOURce
- `LEVEL` *(TRIG:LEV)*
  TRIGger:LEVel
  TRIGger:RELative:LEVel
- `ANGLE`
  TRIGger:ANGle
- `SINGLE` *(RUN)*
  SWEep:MODE
- `GROUP` *(DIG)*
  DIGitize
- `CONT/STOP` *(STOP)*
  SWEep:MODE
  RUN
  STOP
- `AUTO/TRIG'D` *(SWE:MODE)*
  SWEep:MODE
- `slope POS/NEG` *(TRIG:QUAL)*
  TRIGger:QUALifier
- `pul wid <|≥90ns` *(TRIG:WIDT)*

TRIGger:WIDTh

- `TRG FRQ AUT/MAN` *(TRIG:FREQ)*
  TRIGger:FREQuency
- `TRG POW AUT/MAN` *(TRIG:POW)*
  TRIGger:POWer
- `HYST` *(TRIG:HYST)*
  TRIGger:HYSTeresis
- `src rel ON/OFF` *(TRIG:REL:STAT)*
  TRIGger:RELative:STATe

`Traces` Menu Key

- `select:` *(WAV:SOUR)*
  DISPlay:xxx
  WAVeform:SOURce
- `display ON/OFF` *(VIEW)* This key services all traces. Use `select:` to select the trace.
  BLANk
  VIEW
- `input:`
  FUNCx:DEFine
- `format:` *(WAV:DOM)* This key services all traces. Use `select:` to select the trace.
  WAVeform:DOMain
  WAVeform:MAGNitude
- `trnsfrm cntrl`
  - `sglside ON/OFF` *(WAV:PSP)*
    WAVeform:PSPectrum
  - `zoom ON/OFF` *(SWE:TIME:ZOOM:STAT)*
    CHANx:ZOOM:STATe
    FUNCx:ZOOM:STATe
    SWEep:TIME:ZOOM:STATe
  - `CENTER/SPAN` *(SWE:TIME:ZOOM:CENT)*
    CHANx:ZOOM:CENTer
    FUNCx:ZOOM:CENTer
    SWEep:TIME:ZOOM:CENTer
    CHANx:ZOOM:SPAN
    FUNCx:ZOOM:SPAN
    SWEep:TIME:ZOOM:SPAN
  - `START/STOP` *(SWE:TIME:ZOOM:STAR)*
    CHANx:ZOOM:STARt
    FUNCx:ZOOM:STARt
    SWEep:TIME:ZOOM:STARt
    CHANx:ZOOM:STOP
    FUNCx:ZOOM:STOP
    SWEep:TIME:ZOOM:STOP

- zoom options
  - tm zoom ON/OFF
    CHANx:ZOOM:TIME
    FUNCx:ZOOM:TIME
  - fr zoom ON/OFF
    CHANx:ZOOM:FREQ
    FUNCx:ZOOM:FREQ
- REZOOM *(SWE:TIME:ZOOM:REZ)*
  SWEep:TIME:ZOOM:REZoom

- store trace
  - VEC/FMT
    STORe
  - TO MEMx
    STORe
  - to user correct
    - CHANx USR COR
      STORe
    - adaptiv ON/OFF
      CALibrate:USER:ADAPtive
      CALibrate:USER:THReshold

- avg,hld
  - OFF *(FUNC1:TYPE NORM)* This key services all traces. Use select: to select the trace.
    ACQuire:TYPE
    CHANx:TYPE
    FUNCx:TYPE
  - FORMAT AVERAGE *(FUNC1:TYPE FAV)* This key services all traces. Use select: to select the trace.
    ACQuire:TYPE
    ACQuire:COUNt
    CHANx:TYPE
    CHANx:COUNt
    FUNCx:TYPE
    FUNCx:COUNt
    WAVeform:TYPE
  - VECTOR AVERAGE *(FUNC1:TYPE AVER)* This key services all traces. Use select: to select the trace.
    ACQuire:TYPE
    ACQuire:COUNt
    CHANx:TYPE
    CHANx:COUNt
    FUNCx:TYPE
    FUNCx:COUNt
    WAVeform:TYPE

- `hold MIN/MAX` *(FUNC1:TYPE MAX)* This key services all traces. Use `select:` to select the trace.
  ACQuire:TYPE
  CHANx:TYPE
  FUNCx:TYPE
  WAVeform:TYPE
- `RESTART ALL TRC` *(RESTART)*
  ACQuire:TYPE
  CHANx:TYPE
  FUNCx:TYPE
- `SMOOTH ON/OFF` *(FUNC1:SMO:COUN)* This key services all traces. Use `select:` to select the trace.
  FUNCx:SMOoth:STATe
  FUNCx:SMOoth:COUNt

`Scale` Menu Key

- `select:` *(WAV:SOUR)*
  DISPlay:xxx
  WAVeform:SOURce
- `SCALE` *(DISP:FUNC1:RANG)* This key services all traces. Use `select:` to select the trace.
  DISPlay:xxx:RANGe
- `REF LEV/POS` *(DISP:FUNC1:OFFS)* This key services all traces. Use `select:` to select the trace.
  DISPlay:xxx:OFFSet
  DISPlay:xxx:REFerence
- `AUTOSCL` *(DISP:FUNC1:AUT)*
  DISPlay:xxx:AUTorange
- `scale track`
  - `CONT AUTOSCL` *(DISP:FUNC1:TRAC CONT)*
    DISPlay:xxx:TRACk
  - `COUP HW ON/OFF` *(DISP:FUNC1:TRAC COUP)*
    DISPlay:xxx:TRACk
- `hardware`
  - `CH1/CH2` *(CHAN1)*
    CHANx:
  - `autorng ON/OFF` *(CHAN1:AUT)* This key services both channels. Use `CH1/CH2` to select the channel.
    CHANx:AUTorange
  - `RANGE/OFFSET` *(CHAN1:RANG)* This key services both channels. Use `CH1/CH2` to select the channel.
    CHANx:RANGe
    CHANx:OFFSet

- null DC
  - input GND/IF
    CHANx:NULL
  - CURRENT GN &FLTR
    CHANx:NULL
  - CURRENT FILTER
    CHANx:NULL
  - ALL
    CHANx:NULL
- IF coup AC/DC *(CHAN1:COUP)* This key services both channels. Use CH1/CH2 to select the channel.
  CHANx:COUPling
- EXTERNL ATTEN *(CHAN1:PROB)* This key services both channels. Use CH1/CH2 to select the channel.
  CHANx:PROBe

- AUTO SHIFT *(WAV:ZPH)* This key services all traces. Use select: to select the trace.
  WAVeform:ZPHase
- AUTO DELAY *(WAV:ZPH)* This key services all traces. Use select: to select the trace.
  WAVeform:ZPHase
- FREQ SHIFT *(WAV:FSH)* This key services all traces. Use select: to select the trace.
  WAVeform:FSHift
- PHASE SLOPE *(WAV:PSL)* This key services all traces. Use select: to select the trace.
  WAVeform:PSLope
- TIME DELAY *(WAV:DEL)* This key services all traces. Use select: to select the trace.
  WAVeform:DELay
- APERTUR *(FUNC1:APER)* This key services all traces. Use select: to select the trace.
  FUNCx:APERture

Marker Menu Key

- M1(v) *(MARK1:POS)*
  MARK1:POSition
  MARK1:SOURce
- M2(^) *(MARK2:POS)*
  MARK2:POSition
  MARK2:SOURce
- MARKERS OFF *(MARK:OFF)*
  MARK:OFF
- (2-1) ON/OFF *(MARK:DELT:STAT)*
  MARK:DELTa
- MOVE BOTH *(MARK:BOTH)*
  MARK:BOTH

- `mkr->`
  - `HIGHEST PEAK` *(MARK1:PEAK)* These keys service both markers. The marker operated on depends on the currently active marker.
    MARKx:PEAK
  - `NEXT PEAK` *(MARK1:NEXT)* These keys service both markers. The marker operated on depends on the currently active marker.
    MARKx:NEXT
  - `LOCAL PEAK` *(MARK1:LPEA)* These keys service both markers. The marker operated on depends on the currently active marker.
    MARKx:LPEAk
  - `LOWEST POINT` *(MARK1:LOW)* These keys service both markers. The marker operated on depends on the currently active marker.
    MARKx:LOWest
  - `to REF LEV` *(MARK1:REF)* These keys service both markers. The marker operated on depends on the currently active marker.
    MARKx:REFerence
  - `to CENTER` *(MARK1:CENT)* These keys service both markers. The marker operated on depends on the currently active marker.
    MARKx:CENTer
- `mkr hld ON/OFF`
  MARKx:HOLD
- `mkr trk options`
  - `pk trk ON/OFF` *(MARK1:HOLD)* These keys service both markers. The marker operated on depends on the currently active marker.
    MARKx:AUTopeak
  - `pk DC ON/OFF` *(MARK:DCP)*
    MARK:DCPeak
  - `pk intp ON/OFF` *(MARK:INT)*
    MARK:INTerpolate
  - `cycles ON/OFF` *(MARK1:POS CYCL)*
    MARKx:POSition
- `readout options`
  - `ampl:` *(MARK1:READ LOG)*
    MARK:READout
  - `phase:` *(MARK1:READ DEG)*
    MARK:READout
  - `SCL/VEC` *(MARK:FOR)*
    MARK:FORmat

Config Menu Key

- TRACE POINTS *(ACQ:POIN)*
  ACQuire:POINts
- split: *(DISP:FORM)*
  DISPlay:FORMat
- place trace:
  DISPlay:xxx:POSition
- grat: *(DISP:GRAT)*
  DISPlay:GRAT
- 10 MHz INT/EXT *(SWE:REF)*
  SWEep:REFerence
- RF source
  - RF src: *(SOUR:CONF:TYPE)*
    SOURce:CONFigure:TYPE
  - HP-IB/HP-MSIB *(SOUR:CONF:BUS)*
    SOURce:CONFigure:BUS
  - HP-IB ADDR *(SOUR:CONF:ADDR)*
    SOURce:CONFigure:ADDR
  - HP-MSIB ROW *(SOUR:CONF:ROW)*
    SOURce:CONFigure:ROW
  - HP-MSIB COLUMN *(SOUR:CONF:COL)*
    SOURce:CONFigure:COLumn
  - HP-IB options
    - HP-IB PRI/SEC
      SOURce:CONFigure:ADDR
    - use SRQ ON/OFF
      SOURce:CONFigure:HANDshake
  - src cfg ON/OFF *(SOUR:COUP)*
    SOURce:COUPle
- LINES/DOTS *(DISP:CONN)*
  DISPlay:CONNect
- persist ON/OFF *(DISP:PERS)*
  DISPlay:PERSist
- stepkey incr
  no related command
- windows
  - format:
    WINDow:TYPE
  - math:
    WINDow:MATH
- preview ON/OFF

no related command

- `clock`
  - `show TIM/DAT`
    DISPlay:TIME:STATe
    DISPlay:DATE:STATe
  - `US/EURO`
    DISPlay:DATE:FORMat
  - `set clock`
    DATE
    TIME
- `secure`
  - `SECURE SCREEN`
    SECure:SCReen
  - `clear memory`
    SECure:MEMory
  - `clear state`
    SECure:STATe
  - `clear usr cor`
    SECure:UCAL
  - `CLEAR MATH EQ`
    SECure:MATH
- `status ON/OFF`
  DISPlay:STATus:STATe
- `INTERNL STATE` *(SYST:STAT)*
  SYSTem:STATus?
- `system`
  - `ROM VERSION` *(*IDN?)*
    *IDN?
    *OPT?
  - `search ON/OFF` *(CONF:SEAR)*
    CONFigure:SEARch
  - `IP REF:` *(SYST:REF)*
    SYSTem:REFerence
- `measure config`
  - `analytic ON/OFF` *(SWE:TIME:ANAL)*
    SWEep:TIME:ANALytic
  - `dither ON/OFF` *(ACQ:DITH)*
    ACQuire:DITHer
  - `tim bit 32/16` *(ACQ:FORM)*
    ACQuire:FORMat
  - `Fss TGT AUT/MAN` *(SWE:TIME:TARG:SRAT)*
    SWEep:TIME:TARGet[:STATe]

SWEep:TIME:TARGet:SRATe

- `labels`
  - `SCREEN TITLE`
    DISPlay:TITLe[:STRing]
  - `title ON/OFF`
    DISPlay:TITLe:STATe
  - `label traces`
    FUNCx:LABel
  - `label states`
    no related command

`Pulsgen` Menu Key

- `gen is:` *(MOD:STAT)*
  MODulator:STATe
- `PRI/PRF` *(MOD:PER)*
  MODulator:PERiod
  MODulator:FREQuency
- `polarty POS/NEG` *(MOD:POL)*
  MODulator:POLarity
- `WIDTH/DUT CYC` *(MOD:WID)*
  MODulator:WIDth
  MODulator:DUTY
- `RF src PUL/CW` *(SOUR:PULS:STAT)*
  SOURce:PULSe:STATe
- `use as:` *(MOD:COUP)*
  MODulator:COUPle
- `modify:` *(SWE:FUZZ)*
  SWEep:FUZZy
- `Fspan OPT/OFF` *(SWE:OPT)*
  SWEep:OPTimize

`Measure` Menu Key

- `MsrTrc:` *(MEAS:SOUR)*
  MEASure:SOURce
- `update:` *(MEAS:TYPE)*
  MEASure:TYPE
- `define:`
  - `MEASURE AVERAGE`
    MEASure:COUNt
  - `top-bas`
    MEASure:ALGorithm
  - `USR/STD`
    MEASure:MODE
  - `LOWER THRSHLD`
    MEASure:LOWer
  - `UPPER THRSHLD`
    MEASure:UPPer
- `delay:`
  - `EXECUTE`
    MEASure:DELay
  - `MsrTrc:`
    MEASure:SOURce
  - `RefTrc:`
    MEASure:REFerence
  - `thrshld Msr/Ref`
    MEASure:VSTArt
    MEASure:VSTOp
  - `MsrEdge POS/NEG`
    MEASure:ESTArt
  - `RefEdge POS/NEG`
    MEASure:ESTOp
- `MEASURE ALL` *(MEAS:ALL)*
  MEASure:ALL
- `RISE TIME` *(MEAS:RISE)*
  MEASure:RISE
- `FALL TIME` *(MEAS:FALL)*
  MEASure:FALL
- `PERIOD` *(MEAS:PER)*
  MEASure:PERiod
- `FREQ` *(MEAS:FREQ)*
  MEASure:FREQuency
- `+WIDTH` *(MEAS:PWID)*
  MEASure:PWIDth

- `-WIDTH` *(MEAS:NWID)*
  MEASure:NWIDth
- `DUTY CYCLE` *(MEAS:DUTY)*
  MEASure:DUTY
- `PK PWR (pulse)` *(MEAS:PPK)*
  MEASure:PPK
- `AVG PWR (pulse)` *(MEAS:PAVG)*
  MEASure:PAVG
- `RMS` *(MEAS:VRMS)*
  MEASure:VRMS
- `PK-PK` *(MEAS:VPP)*
  MEASure:VPP
- `AMPL (TOP-BASE)` *(MEAS:VAMP)*
  MEASure:VAMPlitude
- `OVER-SHOOT` *(MEAS:OVER)*
  MEASure:OVERshoot

`Table` Menu Key

- `input:` *(TABL:SOUR)*
  TABLe:SOURce
  TABLe:CLEAr
- `ABS/REL` *(TABL:MODE)*
  TABLe:MODE
  TABLe:REFerence
- `delta` *(TABL:DELT)*
  - `ON/OFF`
    TABLe:DELTa
  - `SET REF`
    TABLe:SETRef
- `SCROLL TABLE` *(SCROLL TABLE)*
  no related command
- `SINGLE UPDATE` *(TABL:TYPE)*
  TABLe:TYPE
- `CONT / AVG` *(TABL:COUN)*
  TABLe:TYPE
  TABLe:COUNt
- `# OF HARMNCS` *(TABL:HARM)*
  TABLe:HARMonics
- `signals ONE/ALL` *(TABL:MEAS)*
  TABLe:MEASure
- `filter ON/OFF` *(TABL:FILT)*
  TABLe:FILTer

- `sig trk ON/OFF` *(SWE:STR:STAT)*
  SWEep:STRack
- `signal list`

  - `FIND FREQS`
    TABLe:FIND
  - `INCLUDE HARMNCS`
    TABLe:INCLude
  - `EDIT`
    SWEep:SIGnal
  - `ADD`
    SWEep:SLISt
  - `DELETE`
    SWEep:SLISt
- `show MAG/PHA` *(TABL:AMPL)*
  TABLe:AMPLitude
  TABLe:PHASe

`Analyze` Menu Key

- `histogm`

  - `trace:` *(HIST:SOUR)*
    HISTogram:SOURce
  - `histog:` *(HIST:WIND X)*
    HISTogram:WINDow
    HISTogram:STATe
  - `NUMBER SAMPLES` *(HIST:SAMP)*
    HISTogram:SAMPles
  - `SINGLE ACQUIRE` *(HIST:SING)*
    HISTogram:SINGle
  - `CONT ACQUIRE` *(HIST:STOP)*
    HISTogram:STARt
  - `STOP` *(HIST:STOP)*
    HISTogram:STOP
  - `other`

    - `WINDOW MARKER1` *(HIST:WIND 1)*
      HISTogram:WINDow
    - `WINDOW MARKER2` *(HIST:WIND 2)*
      HISTogram:WINDow
    - `UPPER LIMIT` *(HIST:ULIM)*
      HISTogram:ULIMit

HISTogram:PUL?

- `LOWER LIMIT` *(HIST:LLIM)*
  HISTogram:LLIMit
  HISTogram:PLL?
- `LIMIT-> 0%-100%` *(HIST:REF)*
  HISTogram:REFerence
- `results`

  - `MEAN` *(HIST:MEAN)*
    HISTogram:MEAN
  - `STD DEV` *(HIST:SIGM)*
    HISTogram:SIGMa
  - `OFF`
    no related command

- `masks,limits`

  - `trace:` *(LIM:SOUR)*
    LIMit:SOURce
  - `display ON/OFF` *(LIM:SHOW)*
    LIMit:SHOW
  - `test ON/OFF` *(LIM:TEST)*
    LIMit:TEST
  - `end on:`

    - `#Errors ON/OFF` *(LIM:ERR:STOP)*
      LIMit:ERRor:STOP
      LIMit:ERRor:COUNt
    - `#Traces ON/OFF` *(LIM:TRAC:STOP)*
      LIMit:TRACe:STOP
      LIMit:TRACe:COUNt
  - `count TRC/DOT` *(LIM:COUN)*
    LIMit:COUNt
  - `define shapes`

    - `SELECT` *(LIM:SEL)*
      LIMit:SELect
    - `type:`
      LIMit:TYPE
    - `default shapes` *(LIM:DEF)*
      LIMit:DEFault
    - `edit`

      - `SELECT`
        LIMit:SELect
      - `ADD POINT`
        LIMit:DATA

- DELETE POINT
  LIMit:DATA
- move X/Y
  LIMit:DATA
- COPY FROM
  LIMit:COPY
- delete shapes
  LIMit:DELete

Calib Menu Key

- RF corr ON/OFF *(CAL:CORR:RF)*
  CALibrate:CORRect:RF
- user corr
  - usr cor ON/OFF *(CAL:USER:STAT)*
    CALibrate:USER:STATe
  - CH1/CH2
    CALibrate:USER:DATA
  - SEL/EDT
    CALibrate:USER:DATA
  - SLOPE/FLAT
    CALibrate:USER:DATA
  - INSERT LINE
    CALibrate:USER:DATA
  - other
    - DELETE ALL
      SECure:UCAL
    - DELETE LINE
      CALibrate:USER:DATA
    - action MPY/DIV
      CALibrate:USER:MODE
    - slope LOG/LIN
      CALibrate:USER:DATA
    - copy to CH2
      CALibrate:USER:DATA
- chan skew
  - AUTO SKEW *(CAL:ALIG)*
    CALibrate:ALIGn
  - SEC/DIV *(SWE:TIME:RANG)*
    SWEep:TIME:RANGe
  - CHAN2 SKEW *(CAL:SKEW)*

CALibrate:SKEW

- `IF calib`

  CALibrate:ALL?
- `SELF-TEST` *(*TST?)*

  *TST?
- `cal signal`

  - `cal sig ON/OFF` *(CAL:SIGN:STAT)*

    CALibrate:SIGNal:STATe
  - `FREQ` *(CAL:SIGN:FREQ)*

    CALibrate:SIGNal:FREQuency
  - `AMPL` *(CAL:SIGN:AMPL)*

    CALibrate:SIGNal:AMPLitude

`States` Menu Key

- `save state`

  *SAV
- `recall state`

  *RCL
- `FREQ`
  `&POWER`

  no related command
- `VECTOR VOLTAGE`

  no related command
- `VECTOR NETWORK`

  no related command
- `SCALAR NETWORK`

  no related command
- `TIME and FFT`

  no related command
- `PULSED TIME`

  no related command
- `mass storage`

  - `msi:`

    MMEMory:MSIS
  - `PREV PAGE` *(PREV PAGE)*

    MMEMory:CATalog
  - `NEXT PAGE` *(NEXT PAGE)*

    MMEMory:CATalog
  - `LOAD FILE` *(MMEM:LOAD)*

    MMEMory:LOAD:xxx
  - `save`

MMEMory:STORE:xxx

- □ more
  - ■ CHANGE PREFIX *(PREFIX)*
    no related command
  - ■ load to *(LOAD TO)*
    no related command
  - ■ PURGE FILE *(MMEM:DEL)*
    MMEMory:DELete
  - ■ FORMAT
    MMEMory:INITialize
- ■ preset: FAC/USR
  SYSTem:PRESet:TYPE
- ■ SAV USR PRESET
  SYSTem:PRESet:SAVE

USER Key (HP Instrument BASIC) to Remote Cross Reference

The softkeys and their related commands listed here are located under the (USER) key.

Control Menu Key

- ■ RESET
  PROGram:SELected:STATe
  PROGram:EXPLicit:STATe
- ■ PAUSE
  PROGram:SELected:STATe
  PROGram:EXPLicit:STATe
- ■ RUN
  PROGram:SELected:STATe
  PROGram:EXPLicit:STATe
- ■ CONT
  PROGram:SELected:STATe
  PROGram:EXPLicit:STATe
- ■ STEP
  no related command
- ■ LAST ERROR
  PROGram:SELected:STRing
  PROGram:EXPLicit:STRing
- ■ ENTER
  no related command

`Utility` Menu Key

- `memory`
  PROGram:SELected:MALLocate
  PROGram:EXPLicit:MALLocate
- `scratch`
  - `SCRATCH`
    PROGram:SELected:DELete
    PROGram:EXPLicit:DELete
  - `SCRATCH ALL`
    PROGram:SELected:DELete
    PROGram:EXPLicit:DELete
  - `SCRATCH COM`
    PROGram:SELected:EXECute
    PROGram:EXPLicit:EXECute
- `renumbr`
  - `START LINE #`
    no related command
  - `INCR`
    no related command
  - `PERFORM RENUMBR`
    PROGram:SELected:EXECute
    PROGram:EXPLicit:EXECute
- `secure`
  - `START LINE #`
    no related command
  - `END LINE #`
    no related command
  - `PERFORM SECURE`
    PROGram:SELected:EXECute
    PROGram:EXPLicit:EXECute
- `copy, move`
  - `START LINE #`
    no related command
  - `END LINE #`
    no related command
  - `TO LINE #`
    no related command
  - `PERFORM COPY`
    PROGram:SELected:EXECute
    PROGram:EXPLicit:EXECute
  - `PERFORM MOVE`

PROGram:SELected:EXECute
PROGram:EXPLicit:EXECute

- `typing aid`
  - `ENTER`
    no related command
  - `SELECT CHAR`
    no related command
  - `INSERT REPLACE`
    no related command
  - `SPACE`
    no related command
  - `-->`
    no related command
  - `DELETE CHAR`
    no related command
  - `EDIT`
    no related command
  - `GOTO LINE`
    no related command
  - `PAUSE`
    PROGram:SELected:STATe
    PROGram:EXPLicit:STATe
  - `INSERT LINE`
    no related command
  - `DELETE LINE`
    no related command
  - `RECALL LINE`
    no related command

`Miscel` Menu Key

- `key log`
  no related command
- `screen:`
  DISPlay:PROGram
- `suspend`
  BASic:SUSPend

`Disk` Menu Key

- `MSI`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `CAT`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `GET`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `RE-SAVE`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `PURGE`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `typing aid`
  no related command
- `ENTER`
  no related command

`Print` Menu Key

- `PRNT IS`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `LIST`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `PRINT`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `DISP`
  PROGram:SELected:EXECute
  PROGram:EXPLicit:EXECute
- `typing aid`
  no related command
- `ENTER`
  no related command

`Help` Menu Key

- `GENERAL INFO`
  no related command
- `I/O INFO`
  no related command
- `MSIB INFO`
  no related command
- `INTERNAL INFO`
  none
- `MENU KEYS`
  no related command

H

Cross Reference

The Remote Command to Menu (and User) Key Cross-Reference has been provided for your convenience in showing the remote command with its related softkey.

The cross-reference table has been set up such that the paths to access either the front-panel softkey or the remote command are both shown.

The three columns of the cross-reference table are described below:

Subsystem — This column lists the subsystem and the complete path for each command. However, there is no subsystem path for either the Common Command Set or the Top-Level Commands. Common Commands are designated with asterisks (*) as the first character of the mnemonic (for example, *RST) and Top-Level Commands are one-word mnemonics (for example, DIGitize).

If a subsystem command has multiple related softkeys, then all softkey access paths are listed from the menu key.

Menu Key — This column lists the menu key (located on the left-hand side of the display) under which the related softkey resides in; the number following the menu key indicates the right-hand side page on which the related softkey is located. (No page number indicates the first page of softkeys.)

Softkey — This column lists the softkey and its path, if the softkey happens to be nested beneath another softkey.

Table H-1. Remote Command to Menu Key Cross Reference

Common Command Set	Menu Key	Softkey
*CAL?	—	—
*CLS	—	—
*ESE	—	—
*ESR?	—	—
*IDN?	Config (2)	ROM VERSION
*LRN?	—	—
*OPC	—	—
*OPT?	Config (2)	ROM VERSION
*RCL	States	recall state
*RST	—	—
*SAV	States	save state
*SRE	—	—
*STB?	—	—
*TST?	Calib	SELF-TEST
*WAI	—	—

Top-Level Command Set	Menu Key	Softkey
ADVisory?	—	—
AMASk	—	—
AUTo	Main (2)	FIND SIGNALS, FINE CARRIER
BLANk	Traces	display ON|OFF
DATE	Config (2)	set clock, clock
DIGitize	Trigger	GROUP
ERRor?	—	—
PCB	—	—
READy	—	—
RUN	Trigger	CONT | STOP
STOP	Trigger	CONT | STOP
STORe	Traces	store trace, TO|MEMx
	Traces	store trace, VEC|FMT
	Traces	store trace, to user correct, CHANx USR COR
TIME	Config (2)	set clock, clock
VIEW	Traces	display ON|OFF

ACQuire Subsystem	Menu Key	Softkey
ACQuire:COUNt	Traces	avg,hld, FORMAT AVERAGE
	Traces	avg,hld, VECTOR AVERAGE
ACQuire:DITHer	Config (3)	measure config, dither ON|OFF
ACQuire:FORMat	Config (3)	measure config, tim bit 32|16
ACQuire:POINts	Config	TRACE POINTS
ACQuire:TYPE	Traces	avg,hld, OFF
	Traces	avg,hld, FORMAT AVERAGE
	Traces	avg,hld, RESTART ALL TRC
	Traces	avg,hld, VECTOR AVERAGE
	Traces	avg,hld, hold MIN|MAX

CALibrate Subsystem	Menu Key	Softkey
CALibrate:ALIGn	Calib	chan AUTO skew, SKEW
CALibrate:ALL?	Calib	IF calib
CALibrate:CORRect:RF	Calib	RF corr ON|OFF
CALibrate:DATA	—	—
CALibrate:SIGNal:AMPLitude	Calib	cal signal, AMPL
CALibrate:SIGNal:FREQuency	Calib	cal signal, FREQ
CALibrate:SIGNal[:STATe]	Calib	cal cal sig signal, ON|OFF
CALibrate:SKEW	Calib	chan CHAN2 skew, SKEW
CALibrate:USER:ADAPtive	Traces	store to user adaptiv trace, correct, ON|OFF
CALibrate:USER:DATA	Calib	user corr, CH1|CH2
	Calib	user INSERT corr, LINE
	Calib	user corr, SEL|EDT
	Calib	user SLOPE | corr, FLAT
	Calib	user copy corr, other, toCH2
	Calib	user DELETE corr, other, LINE
	Calib	user slope corr, other, LOG|LIN
CALibrate:USER:MODE	Calib	user action corr, other, MPY|DIV
CALibrate:USER:STATe	Calib	user usr cor corr, ON|OFF
CALibrate:USER:THReshold	Traces	store to user adaptiv trace, correct, ON|OFF

CHANnel Subsystem	Menu Key	Softkey
CHANx:AUTorange	Scale	hardwre, autorng ON|OFF
CHANx:COUNt	Traces	avg,hld, FORMAT AVERAGE
	Traces	avg,hld, VECTOR AVERAGE
CHANx:COUPling	Scale	hardwre, IF coup AC|DC
CHANx:NULL?	Scale	hardwre, null DC, input GND|IF
	Scale	hardwre, null DC, ALL
	Scale	hardwre, null DC, CURRENT FILTER
	Scale	hardwre, null DC, CURRENT GN&FLTR
CHANx:OFFSet	Scale	hardwre, RANGE | OFFSET
CHANx:PROBe	Scale	hardwre, EXTERNL ATTEN
CHANx:RANGe	Scale	hardwre, RANGE | OFFSET
CHANx:RLEVel	—	—
CHANx:SMOoth:COUNt	Traces	avg,hld, smooth ON|OFF
CHANx:SMOoth[:STATe]	Traces	avg,hld, smooth ON|OFF
CHANx:TYPE	Traces	avg,hld, OFF
	Traces	avg,hld, FORMAT AVERAGE
	Traces	avg,hld, RESTART ALL TRC
	Traces	avg,hld, VECTOR AVERAGE
	Traces	avg,hld, hold MIN|MAX

CHANnel Subsystem	Menu Key	Softkey
CHANx:ZOOM:CENTer	Traces	trnsfrm cntrl, CENTER SPAN
CHANx:ZOOM:FREQ	Traces	trnsfrm cntrl, zoom options, fr zoom ON|OFF
CHANx:ZOOM:SPAN	Traces	trnsfrm cntrl, CENTER SPAN
CHANx:ZOOM:STARt	Traces	trnsfrm cntrl, START STOP
CHANx:ZOOM[:STATe]	Traces	trnsfrm cntrl, zoom ON|OFF
CHANx:ZOOM:STOP	Traces	trnsfrm cntrl, START STOP
CHANx:ZOOM:TIME	Traces	trnsfrm cntrl, zoom options, tm zoom ON|OFF

CONFigure Subsystem	Menu Key	Softkey
CONFigure:SEARch	Config (3)	search ON|OFF

DAC Subsystem	Menu Key	Softkey
DAC:MODE	—	—
DAC:VALue	—	—

DIAGnostics Subsystem	Menu Key	Softkey
DIAGnostics:PASSthru:ADDRess	—	—
DIAGnostics:PASSthru:STRing	—	—
DIAGnostics:PASSthru:TIMEout	—	—

DISPlay Subsystem	Menu Key	Softkey
DISPlay:CONNect	Config (2)	LINES | DOTS
DISPlay:DATE:FORMat	Config (2)	clock , US|EURO
DISPlay:DATE[:STATe]	Config (2)	show clock , TIM|DAT
DISPlay:FORMat	Config	split:
DISPlay:GRATicule	Config	grat:
DISPlay:MENU:ABORt	—	—
DISPlay:MENU:CHANged?	—	—
DISPlay:MENU:CLEAr	—	—
DISPlay:MENU:COPY	—	—
DISPlay:MENU:EXCLude	—	—
DISPlay:MENU:HARDkeys	—	—
DISPlay:MENU:KEY	—	—
DISPlay:MENU:LABel	—	—
DISPlay:MENU:LIST	—	—
DISPlay:MENU:MAXimum	—	—
DISPlay:MENU:MINimum	—	—
DISPlay:MENU:PRESet	—	—
DISPlay:MENU:PROMpt	—	—
DISPlay:MENU:STATe	—	—
DISPlay:MENU:STEP	—	—
DISPlay:MENU:TERMinator	—	—
DISPlay:MENU:TEXT	—	—
DISPlay:MENU:TYPE	—	—
DISPlay:MENU:VALue	—	—
DISPlay:PERSist	Config (2)	persist ON|OFF
DISPlay:PROGram	—	—
DISPlay:REGister[:LABel]	—	—
DISPlay:STATus[:STATe]	Config (3)	status ON|OFF
DISPlay:STRing	—	—
DISPlay:TEXT:CLEar	—	—
DISPlay:TEXT:COLor	—	—
DISPlay:TEXT:CSIZe	—	—
DISPlay:TEXT[:DATA]	—	—
DISPlay:TEXT:PAGE	—	—
DISPlay:TEXT:STATe	—	—

DISPlay Subsystem	Menu Key	Softkey
DISPlay:TIME[:STATe]	Config (2)	show clock, TIM|DAT
DISPlay:TITLe:STATe	Config (3)	title labels, ON|OFF
DISPlay:TITLe[:STRing]	Config (3)	screen labels, title
DISPlay:xxx:AUToscale	Scale	AUTO-SCALE
DISPlay:xxx:OFFSet	Scale	REF LEV|POS
DISPlay:xxx:POSition	Config	place trace:
DISPlay:xxx:RANGe	Scale	SCALE
DISPlay:xxx:REFerence	Scale	REF LEV|POS
DISPlay:xxx:TRACk	Scale	scale track, CONT AUTOSCL
	Scale	scale track, COUP HW ON|OFF

FUNCtion Subsystem	Menu Key	Softkey
FUNCx:APERture	Scale (2)	APERTUR
FUNCx:AUTorange	—	—
FUNCx:COUNt	Traces	avg,hld, FORMAT AVERAGE
	Traces	avg,hld, VECTOR AVERAGE
FUNCx:DEFine	Traces	input:
FUNCx:FREeze	—	—
FUNCx:LABel	Config (3)	labels, label traces
FUNCx:OFFSet	—	—
FUNCx:RANGe	—	—
FUNCx:RECursion	—	—
FUNCx:RLEVel	—	—
FUNCx:SMOoth:COUNt	Traces	avg,hld, SMOOTH ON|OFF
FUNCx:SMOoth[:STATe]	Traces	avg,hld, SMOOTH ON|OFF
FUNCx:TYPE	Traces	avg,hld, OFF
	Traces	avg,hld, FORMAT AVERAGE
	Traces	avg,hld, RESTART ALL TRC
	Traces	avg,hld, VECTOR AVERAGE
	Traces	avg,hld, hold MIN|MAX
FUNCx:XSCale	—	—
FUNCx:ZOOM:CENTer	Traces	trnsfrm cntrl, CENTER SPAN
FUNCx:ZOOM:FREQ	Traces	trnsfrm cntrl, zoom options, frzoom ON|OFF
FUNCx:ZOOM:SPAN	Traces	trnsfrm cntrl, CENTER SPAN
FUNCx:ZOOM:STARt	Traces	trnsfrm cntrl, START STOP
FUNCx:ZOOM[:STATe]	Traces	trnsfrm cntrl, zoom ON|OFF
FUNCx:ZOOM:STOP	Traces	trnsfrm cntrl, START STOP
FUNCx:ZOOM:TIME	Traces	trnsfrm cntrl, zoom options, tm zoom ON|OFF

HISTogram Subsystem	Menu Key	Softkey
HISTogram:DATA?	—	—
HISTogram:EXTRemes?	—	—
HISTogram:LLIMit	Analyze	histogm, other, LOWER LIMIT
HISTogram:MEAN?	Analyze	histogm, other, results, MEAN
HISTogram:PEAK?	—	—
HISTogram:PLL?	—	LOWER LIMIT
HISTogram:PUL?	Analyze	histogm, other, LOWER LIMIT
	Analyze	histogm, other, UPPER LIMIT
HISTogram:REFerence	Analyze	histogm, other, LIMIT-> 0%-100%
HISTogram:SAMPles	Analyze	histogm, NUMBER SAMPLES
HISTogram:SIGMa?	Analyze	histogm, other, results, STD DEV
HISTogram:SINGle	Analyze	histogm, SINGLE ACQUIRE
HISTogram:SOURce	Analyze	histogm, histog:
	Analyze	histogm, trace:
HISTogram:STARt	Analyze	histogm, CONT ACQUIRE
HISTogram:STATe	—	—
HISTogram:STOP	Analyze	histogm, STOP ACQUIRE
HISTogram:ULIMit	Analyze	histogm, other, UPPER LIMIT
HISTogram:WINDow	Analyze	histogm, histog:
	Analyze	histogm, other, WINDOW MARKER1
	Analyze	histogm, other, WINDOW MARKER2

LIMit Subsystem	Menu Key	Softkey
LIMit:COPY	Analyze	mask, limits, define shapes, edit, COPY FROM
LIMit:COUNt	Analyze	mask, limits, count TRC|DOT
LIMit:DATA	Analyze	mask, limits, define shapes, edit, ADD POINT
	Analyze	mask, limits, define shapes, edit, DELETE POINT
	Analyze	mask, limits, define shapes, edit, move X|Y
LIMIT:DATA:ASCii	—	—
LIMit:DEFault	Analyze	mask, limit, define shapes, default shapes
LIMit:DELete	Analyze	mask, limits, define shapes, edit, delete shapes
LIMit:ERRor:COUNt	Analyze	mask, limits, end on, # errors ON|OFF
LIMit:ERRor:STOP	Analyze	mask, limits, end on, # errors ON|OFF
LIMit:ERRor:TRACe	—	—
LIMit:FAIL?	—	—
LIMit:FCOunt?	—	—
LIMit:RESCale	—	—
LIMit:SELect	Analyze	masks, limits, define shapes, edit, SELECT
LIMit:SHOW	Analyze	mask, limits, display ON|OFF
LIMit:SOURce	Analyze	mask, limits, trace:
LIMit:TEST	Analyze	mask, limits, test ON|OFF
LIMit:TRACe:COUNt	Analyze	mask, limits, end on, # Traces ON|OFF
LIMit:TRACe:STOP	Analyze	mask, limits, end on, # Traces ON|OFF
LIMit:TYPE:	—	—
LIMit:X:OFFSet	—	—
LIMit:X:SCALe	Analyze	mask, limits, define shapes, scale X|Y
LIMit:XY:SCALe	—	—
LIMit:Y:OFFSet	—	—
LIMit:Y:SCALe	Analyze	mask, limits, define shapes, scale X|Y

MARKer Subsystem	Menu Key	Softkey
MARK:BOTH	Markers (2)	MOVE mkr-->, BOTH
MARK:DCPeak	Markers (2)	mkr trk pk DC options, ON|OFF
MARK:DELTa	Markers	Δ (2-1) ON|OFF
MARK:DELTa[:POSition]	—	—
MARK:DELTa:STATe	—	—
MARK:FORmat	Markers (2)	readout options, SCL|VEC
MARK:INTerpolate	Markers (2)	readout options, SCL|VEC
MARK:OFF	Markers	MARKERS OFF
MARK:READout	Markers (2)	readout options, ampl:
	Markers (2)	readout options, phase:
MARKx:AUTopeak	Markers (2)	mkr trk pktrk options, ON|OFF
MARKx:CENTer	Markers	to mkr-->, CENTER
MARKx:HOLD	Markers (2)	mkr hld ON|OFF
MARKx:LOWest?	Markers	mkr-->, LOWEST
MARKx:LPEAk?	Markers	LOCAL mkr-->, PEAK
MARKx:NEXT?	Markers	NEXT or mkr-->, LOWEST
MARKx:PEAK?	Markers	HIGHEST mkr-->, PEAK
MARKx[:POSition]	Markers	M(|) (off)
	Markers	M(↑) (off)
	Markers (2)	mkr trk cycles options, ON|OFF
MARKx:REFerence	Markers	to mkr-->, REF LEV
MARKx:SOURce	Markers	M1(|) (off) or M2(↑) off

Mass MEMory Subsystem	Menu Key	Softkey
MMEMory:CATalog?	States (2)	mass storage, PREV PAGE
	States (2)	mass storage, NEXT PAGE
MMEMory:DELete	States (2)	mass storage, more, PURGE FILE
MMEMory:INITialize	States (2)	mass storage, more, FORMAT
MMEMory:LOAD:CALibration	States (2)	mass storage, LOAD FILE
MMEMory:LOAD:CUSTom	States (2)	mass storage, LOAD FILE
MMEMory:LOAD:HISTogram	States (2)	mass storage, LOAD FILE
MMEMory:LOAD:PROGram	—	—
MMEMory:LOAD:STATe	States (2)	mass storage, LOAD FILE
MMEMory:LOAD:TRACe	States (2)	mass storage, LOAD FILE
MMEMory:LOAD:USER	States (2)	mass storage, LOAD FILE
MMEMory:MSIS	States (2)	mass storage, msi:
MMEMory:STORe:CALibration	States (2)	mass storage, save
MMEMory:STORe:CUSTom	States (2)	mass storage, save
MMEMory:STORe:FTRace	States (2)	mass storage, save
MMEMory:STORe:HISTogram	States (2)	mass storage, save
MMEMory:STORe:STATe	States (2)	mass storage, save
MMEMory:STORe:TRACe	States (2)	mass storage, save
MMEMory:STORe:USER	States (2)	mass storage, save

MEASure Subsystem	Menu Key	Softkey
MEASure:ALGorithm	Measure	define, top-bas AUTO
MEASure:ALL	Measure	MEASURE ALL
MEASure:COUNt	Measure	define, MEASURE AVERAGE
MEASure:CURSor?	—	—
MEASure:DELay	Measure (2)	DELAY
MEASure:DUTy	Measure (2)	DUTY CYCLE
MEASure:ESTArt	Measure	delay:, MsrEdge POS|NEG
MEASure:ESTOp	Measure	delay:, RefEdge POS|NEG
MEASure:FALLtime	Measure	FALL TIME
MEASure:FREQuency	Measure (2)	FREQ
MEASure:LOWer	Measure	define, LOWER THRSHOLD
MEASure:MODE	Measure	define, USR|STD 10%,90%
MEASure:NWIDth	Measure (2)	-WIDTH
MEASure:OVERshoot	Measure (3)	OVER- SHOOT
MEASure:PAVG	Measure (3)	AVG PWR (pulse)
MEASure:PERiod	Measure (2)	PERIOD
MEASure:PIP	—	—
MEASure:PPK	Measure (3)	PK PWR (pulse)
MEASure:PREShoot	—	—
MEASure:PRMS	—	—
MEASure:PTIMe?	—	—
MEASure:PVOLt	—	—
MEASure:PWIDth	Measure (2)	+WIDTH
MEASure:REFerence	Measure	delay:, RefTrc:
MEASure:RISEtime	Measure	RISE TIME
MEASure:SOURce	Measure	MsrTrc:
MEASure:TDELta?	—	—

MEASure Subsystem	Menu Key	Softkey
MEASure:TMAX?	—	—
MEASure:TMIN?	—	—
MEASure:TPOint?	—	—
MEASure:TSTArt	—	—
MEASure:TSTOp	—	—
MEASure:TVOLt	—	—
MEASure:TYPE	Measure	update
MEASure:UNIT	—	—
MEASure:UPPer	Measure	define, UPPER THRSHOLD
MEASure:VAMPlitude	Measure (3)	AMPL= TOP-BASE
MEASure:VAVerage	—	—
MEASure:VBASe	—	—
MEASure:VDELta?	—	—
MEASure:VFIFty	—	—
MEASure:VMAX	—	—
MEASure:VMIN	—	—
MEASure:VPOint?	—	—
MEASure:VPP	Measure (3)	PK-PK
MEASure:VREL	—	—
MEASure:VRMS	Measure (3)	RMS
MEASure:VSTArt	Measure	delay:, thrshld Msr|Ref
MEASure:VSTOp	Measure	delay:, thrshld Msr|Ref
MEASure:VTIMe?	—	—
MEASure:VTOP	—	—

MODulator Subsystem	Menu Key	Softkey
MODulator:COUPle	Pulsgen	use as:
MODulator:DUTY	Pulsgen	WIDTH | DUT CYC
MODulator:FREQuency	Pulsgen	PRI|PRF
MODulator:PERiod	Pulsgen	PRI|PRF
MODulator:POLarity	Pulsgen	polarty POS|NEG
MODulator:STATe	Pulsgen	gen is:
MODulator:WIDTh	Pulsgen	WIDTH | DUT CYC

Peak POWer Subsystem	Menu Key	Softkey
PPOWer:AUTocount	—	—
PPOWer:COUNt	—	—
PPOWer:MEASure	—	—
PPOWer:RESults?	—	—
PPOWer:SRATe	—	—
PPOWer:SRESolution	—	—
PPOWer:TYPE	—	—
PPOWer:WIDth	—	—

PULSe Subsystem	Menu Key	Softkey
PULSe:CARRier	Main	carrier FREQ
PULSe:MCARrier	Main (2)	FIND CARRIER

SECure Subsystem	Menu Key	Softkey
SECure:MATH	Config (3)	secure, CLEAR MATH EQ
SECure:MEMory	Config (3)	secure, clear memory
SECure:SCReen	Config (3)	secure, SECURE SCREEN
SECure:STATe	Config (3)	secure, clear state
SECure:UCAL	Config (3)	secure, clear usr cor
SECure:UCAL	Calib	user corr, other, DELETE ALL

SOURce Subsystem	Menu Key	Softkey
SOURce:BERT:CLOCk	Main	BERT setup, CLOCK RATe
SOURce:BERT:DIVisor	Main	BERT setup, DIVIDE RATE
SOURce:BERT:READ:CLOCk	—	—
SOURce:BERT:READ:DIVisor	—	—
SOURce:BERT:READ[:STATe]	Main	BERT setup, READ BERT
SOURce:BERT:SIGNal	Main	BERT setup, signal
SOURce:CONFigure:ADDRess	Config	RF source, HP-IB ADDR
	Config	RF source, HP-IB options, HP-IB PRI|SEC
SOURce:CONFigure:BUS	Config	RF source, HP-IB| HP-MSIB
SOURce:CONFigure:COLumn	Config	RF source, HP-MSIB COLUMN
SOURce:CONFigure:CUSTom	—	—
SOURce:CONFigure:GENeric	—	—
SOURce:CONFigure:HANDshake	Config	RF source, HP-IB options, use SRQ ON|OFF

SOURce Subsystem	Menu Key	Softkey
SOURce:CONFigure:HP8510:ADDRess	Config	RF source, HP-IB options, HP8510 ADDR
SOURce:CONFigure:HP8510:DEVice	Config	RF source, HP-IB options, HP8510 Device
SOURce:CONFigure:HP8510[:STATe]	Config	RF source, HP-IB options, PASSTHRU ON|OFF
SOURce:CONFigure:RESet	—	—
SOURce:CONFigure:ROW	Config	RF source, HP-MSIB ROW
SOURce:CONFigure:TYPE	Config	RF source, RF src:
SOURce:COUPle	Config	RF source, src cfg ON|OFF
SOURce:FREQuency:CW	Main	source FRQ|PWR
	Main	carrier FRQ|PWR
SOURce:POWer:LEVel	Main	source FRQ|PWR
	Main	carrier FRQ|PWR
SOURce:POWer:STATe	Main	RF out ON|OFF
SOURce:PULSe:STATe	Pulsgen	RF src PUL|CW
SOURce:USER	—	—

STATus Subsystem	Menu Key	Softkey
STATus:OPERation:CONDition?	—	—
STATus:OPERation:ENABle	—	—
STATus:OPERation[:EVENt]?	—	—
STATus:OPERation:NTRansition	—	—
STATus:OPERation:PTRansition	—	—
STATus:PRESet	—	—

SWEep Subsystem	Menu Key	Softkey
SWEep:FILter	—	—
SWEep:FREQuency:DIVisor	Main	sweep options, freq mult denom.
SWEep:FREQuency:DWELl	Main	DWELL
SWEep:FREQuency:IFBandwidth	Main	NOISE FILTER
SWEep:FREQuency:MULTiplier	Main (2)	sweep options, freq mult numer.
SWEep:FREQuency:OFFSet	Main (2)	sweep options, freq offset
SWEep:FREQuency:STARt	Main	STARt
SWEep:FREQuency:STOP	Main	STOP
SWEep:FUZZy	Pulsgen (2)	modify
SWEep:MODE	Trigger	CONT | SINGLE or STOP
	Trigger (2)	AUTO | TRIG'D
SWEep:OPTimize	Pulsgen (2)	Fspan OPT|OFF
SWEep:POWer:DWELl	—	—
SWEep:POWer:IFBandwidth	Main	NOISE FILTER
SWEep:POWer:STARt	Main	STARt
SWEep:POWer:STOP	Main	STOP
SWEep:REFerence	Config	10 MHz INT|EXT
SWEep:SIGnal	Main	signal FREQ
	Table (2)	signal list, EDIT
	Main (2)	signal list, EDIT
SWEep:SINgle	Main (2)	single sglshot shot, ON|OFF
SWEep:SLISt	Main (2)	signal list, DELETE
	Main (2)	signal list, ADD
	Table (2)	signal list, ADD
	Table (2)	signal list, DELETE

SWEep Subsystem	Menu Key	Softkey
SWEep:STRack	Main (2)	sig trk
	Table (2)	sig trk
SWEep:STRack:COUNt	Main (2)	sig trk ON|OFF
SWEep:STRack[:STATe]	Table (2)	sig trk ON|OFF
SWEep:STRack:VERify	—	—
SWEep:TIME:ANALytic	Config (3)	measure analytc config , ON|OFF
SWEep:TIME:CYCLes	Main (2)	sweep labels , CYCLES, DELAY
	Main (2)	sweep SEC|DIV labels , DELAY
SWEep:TIME:DELay	Main	DELAY
SWEep:TIME:FILter:BWIDth	Main	noisflt ON|OFF
SWEep:TIME:FILter[:STATe]	Main	noisflt ON|OFF
SWEep:TIME:FILter:VIDeo	Main (2)	sweep CHAN1 options , VID|RF
	Main (2)	sweep CHAN2 options , VID|RF
SWEep:TIME:OVERsweep:DECimate	Main (2)	single display shot , SEC/DIV
SWEep:TIME:OVERsweep:DETect	Main (2)	single shot , detect:
SWEep:TIME:OVERsweep:FILTer:APERture	—	—
SWEep:TIME:OVERsweep:FILTer:CHANx:CENTer	—	—
SWEep:TIME:OVERsweep:FILTer:CHANx:FORMat	—	—
SWEep:TIME:OVERsweep:FILTer:CHANx:TRACk	—	—
SWEep:TIME:OVERsweep:FILTer[:STATe]	Main (2)	single filter shot , ON|OFF
SWEep:TIME:OVERsweep:OFFSet	Main (2)	single display shot , DELAY
SWEep:TIME:OVERsweep:POINTs	—	—
SWEep:TIME:OVERsweep[:STATe]	Main (2)	single shot , big trc
SWEep:TIME:RANGe	Main	SEC/DIV
	Calib	chan skew , SEC|DIV

SWEep Subsystem	Menu Key	Softkey
SWEep:TIME:SRATe	Main (2)	single sglshot shot, ON|OFF
SWEep:TIME:TARGet:SRATe	Config (3)	measure Fse TGT config, AUT|MAN
SWEep:TIME:TARGet[:STATe]	Config (3)	measure Fss TGT config, AUT|MAN
SWEep:TIME:TRANslate	Main (2)	sweep translt options, ON|OFF
SWEep:TIME:ZOOM:CENTer	Traces	trnsfrm CENTER cntrl, SPAN
SWEep:TIME:ZOOM:REZoom	Traces	trnsfrm cntrl, REZOOM
SWEep:TIME:ZOOM:SPAN	Traces	trnsfrm CENTER cntrl, SPAN
SWEep:TIME:ZOOM:STARt	Traces	trnsfrm START cntrl, STOP
SWEep:TIME:ZOOM[:STATe]	Traces	trnsfrm zoom cntrl, ON|OFF
SWEep:TIME:ZOOM:STOP	Traces	trnsfrm START cntrl, STOP
SWEep:TYPE	Main	sweep

SYSTem Subsystem	Menu Key	Softkey
SYSTem:PRESet[:EXECute]	—	—
SYSTem:PRESet:SAVE	States (2)	mass storage, more, SAV USR PRESET
SYSTem:PRESet:TYPE	States (2)	mass storage, more, preset FAC|USR
SYSTem:REFErence	—	—
SYSTem:STATus?	Config (3)	INTERNL STATE

TABLe Subsystem	Menu Key	Softkey
TABLe:AMPLitude	Table (2)	show MAG|PHA
TABLe:CLEAr	Table	input:
TABLe:COUNt	Table	CONT | AVG
TABLe:DELTa	Table	delta, ON|OFF
TABLe:FILTer	Table (2)	filter ON|OFF
TABLe:FIND	Main (2)	signal list, FIND FREQS
	Table (2)	signal list, FIND FREQS
TABLe:HARMonics	Table (2)	# OF HARMNCS
TABLe:INCLude	Table (2)	signal list, INCLUDE HARMNCS
	Main (2)	signal list, INCLUDE HARMNCS
TABLe:MEASure	Table (2)	signals ONE|ALL
TABLe:MODE	Table	ABS|REL
TABLe:PHASe	Table (2)	show MAG|PHA
TABLe:REFerence	Table	ABS|REL
TABLe:RESults?	—	—
TABLe:SETReference	Table	delta, SET REF
TABLe:SOURce	Table	input:
TABLe[:TYPE]	Table	SINGLE CONT | UPDATE or AVG
TABLe:VALid?	—	—

TRIGger Subsystem	Menu Key	Softkey
TRIGger:ANGle	Trigger	ANGLE
TRIGger:CENTer	—	—
TRIGger:FREQuency	Trigger (2)	TRG FRQ AUT|MAN
TRIGger:FREQuency:STATe	—	—
TRIGger:FREQuency[:VALue]	—	—
TRIGger:HYSTeresis	Trigger (2)	HYST
TRIGger:LEVel	Trigger	LEVEL
TRIGger:POWer	Trigger (2)	TRG POW AUT|MAN
TRIGger:POWer:STATe	—	—
TRIGger:POWer[:VALue]	—	—
TRIGger:QUALifier	Trigger (2)	slope POS|NEG
TRIGger:RELative:LEVel	Trigger	LEVEL
TRIGger:RELative[:STATe]	Trigger (2)	src rel ON|OFF
TRIGger:SOURce	Trigger	input:
TRIGger:TYPE	Trigger	trg is:
TRIGger:WIDth	Trigger (2)	pulse <|>90ns

WAVeform Subsystem	Menu Key	Softkey
WAVeform:DATA	—	—
WAVeform:DATA:DEFinite	—	—
WAVeform:DATA[:INDefinite]	—	—
WAVeform:DATA:ASCii	—	—
WAVeform:DELay	Scale (2)	`TIME` `DELAY`
WAVeform:DOMain	Traces	`format:`
WAVeform:DTYPe	—	—
WAVeform:FINCrement	—	—
WAVeform:FMODe	—	—
WAVeform:FORigin	—	—
WAVeform:FORMat	—	—
WAVeform:FRANge	—	—
WAVeform:FSHift	Scale (2)	`FREQ` `SHIFT`
WAVeform:INTerleave	—	—
WAVeform:MAGNitude	Traces	`format:`
WAVeform:PINCrement	—	—
WAVeform:POINts	—	—
WAVeform:PORigin	—	—
WAVeform:PRANge	—	—
WAVeform:PREamble	—	—
WAVeform:PSLope	Scale (2)	`PHASE` `SLOPE`
WAVeform:PSPectrum	Traces	`trnsfrm` `sglside` `control` , `ON|OFF`
WAVeform:RLEVel	—	—
WAVeform:SOURce	Scale	`select:`
	Traces	`select:`
WAVeform:TINCrement	—	—
WAVeform:TORigin	—	—
WAVeform:TRANge	—	—
WAVeform:TREFerence?	—	—

WAVeform Subsystem	Menu Key	Softkey
WAVeform:TYPE	Traces	FORMAT avg,hld, AVERAGE
	Traces	VECTOR avg,hld, AVERAGE
	Traces	hold avg,hld, MIN|MAX
WAVeform:UNITs	—	—
WAVeform:VALid?	—	—
WAVeform:VINCrement	—	—
WAVeform:VORigin	—	—
WAVeform:VRANge	—	—
WAVeform:YBOTtom?	—	—
WAVeform:YREFerence?	—	—
WAVeform:YTOP?	—	—
WAVeform:ZPHase	Scale (2)	AUTO SHIFT
	Scale (2)	AUTO DELAY

WINDow Subsystem	Menu Key	Softkey
WINDow:MATH	Config (2)	windows, math:
WINDow[:TYPE]	Config (2)	windows, format:
WINDow:USER	—	—

XLINE, YLINE Subsystem	Menu Key	Softkey
XLINE | YLINE:SOURce	—	—
XLINE | YLINE[:VALue]	—	—

Table H-2. Remote Command to User Key Cross Reference

Subsystem	User Key	Softkey
BASic:SUSPend	Miscel	`suspend`
DISPlay:PROGram	Miscel	`screen`
PROGram:EXPLicit:DELete	Utility	`scratch`, `SCRATCH COM`
	Utility	`scratch`, `SCRATCH ALL`
	Utility	`scratch`, `SCRATCH`
PROGram:EXPLicit:EXECute	Disk	`CAT`
	Disk	`GET`
	Disk	`MSI`
	Disk	`PURGE`
	Disk	`RE-SAVE`
	Print	`DISP`
	Print	`LIST`
	Print	`PRINT`
	Print	`PRNT IS`
	Utility	`copy, move`, `PERFORM COPY`
	Utility	`copy, move`, `PERFORM MOVE`
	Utility	`renumbr`, `PERFORM RENUMBR`
	Utility	`secure`, `PERFORM SECURE`
PROGram:EXPLicit:MALLocate	Utility	`memory`
	Control	`CONT`
	Control	`PAUSE`
	Control	`RESET`
	Control	`RUN`
	Control	`STRING`

Table H-2. Remote Command to User Key Cross Reference (continued)

Subsystem	User Key	Softkey
PROGram:EXPLicit:STATe	Utility	typing aid, PAUSE
PROGram:SELected:DELete	Utility	scratch, SCRATCH
	Utility	scratch, SCRATCH ALL
	Utility	scratch, SCRATCH COM
	Disk	CAT
PROGram:SELected:EXECute	Disk	GET
	Disk	MSI
	Disk	PURGE
	Disk	RE-SAVE
	Print	DISP
	Print	LIST
	Print	PRINT
	Print	PRNT IS
	Utility	copy, PERFORM move, COPY
	Utility	copy, PERFORM move, MOVE
	Utility	PERFORM renumbr, RENUMBR
	Utility	PERFORM secure, SECURE
PROGram:SELected:MALLocate	Utility	memory
PROGram:SELected:STATe	Control	CONT
	Control	CONT
	Control	CONT
	Control	PAUSE
	Control	RESET
	Control	RUN
	Control	STRING
	Utility	typing aid, PAUSE

I

New Commands and Math Operators

This appendix lists new commands and math operators documented in this book. These additions were not included in the original edition. Some additions apply only to HP 70820A modules having the firmware revision number indicated. To view the revision number, press one of the following softkey paths: (Which path you use depends on the firmware version.)

- `Config`, `more 1 of 3`, `more 2 of 3`, `system`, `ROM VERSION`.
- `Config`, `more 1 of 2`, `ROM VERSION`.

Revision 1.0 Commands

These commands are available in all modules, but were not documented in the original manual.

MEASure:ALGorithm
MEASure:COUNt
MEASure:TYPE
SOURce:CONFigure:HP8510:xxx
SOURce:USER
SWEep:STRack:COUNt
SWEep:STRack:VERify

Revision 1.01 Commands

CALibrate:USER:ADAPtive
CALibrate:USER:THREShold
DISPlay:TITLe:STATe
DISPlay:TITLe:STRing
FUNCx:FREEze
FUNCx:RECursion
MMEMory:xxx
SECure:MATH
SOURce:BERT:xxx
SOURce:GENeric
SWEEP:TIME:TARGet:SRATe
SWEEP:TIME:TARGet:STATe
TRIGger:HYSTeresis
WAVeform:PINCrement
WAVeform:PORigin

WAVeform:PRANge

Revision 1.01 Math Operations

CHOP()
CONST()
SIGN()

Revision 1.1 Commands

BASic:SUSPend
CALibrate:USER:COPY
DIAGnostics:PASSthru:xxx
DISPlay:PROGram
DISPlay:REGister:LABel
DISPlay:MENU:xxx
DISPlay:TEXT:xxx
FUNCx:TD
HISTogram:STATe
HIST:EXTRemes
HIST:PEAK
LIM:xxx
MARK:DELTa:POSition
MARK:DELTa:STATe
MASK:xxx
MMEMory:LOAD:PROGram
MMEMory:STORe:PROGram
SYSTem:PRESet:xxx
SYSTem:REFerence
SWEep:TIME:OVERsweep:xxx
SWEep:TIME:ZOOM:REZoom
TRIGger:FREQuency:STATe
TRIGger:FREQuency:VALue
TRIGger:POWer:STATe
TRIGger:POWer:VALue
WAVeform:DATA:ASCii
XLINEx:xxx
YLINEx:xxx

Revision 1.1 Math Operations

Fcar
Fsig
SHIFT
[] (subrange operator)