DL716
Digital Scope
Communication Interface
USER'S MANUAL

YOKOGAWA
Yokogawa Electric Corporation

IM 701830-11E
1st Edition

Introduction

Thank you for purchasing YOKOGAWA's DL716 Digital Scope.
This Communication Interface User's Manual describes the functions and commands of the GP-IB and RS-232 interfaces. To ensure proper use of the GP-IB/RS-232 interfaces, please read this manual thoroughly.
Keep the manual in a safe place for quick reference whenever a question arises.
Three manuals are provided with the DL716 including this Communication Interface User's Manual.

Manual Name	Manual No.	Description
DL716 User's Manual	IM 701830-01E	Describes all functions except for the communications functions and operation procedures of the instrument.
DL716 Communication User's Manual	IM 701830-11E	Describes the communications functions of the GP-IB/RS-232 interface.
DL716 Operation Guide	IM701830-02E	Explains basic operations only.

Note

- The contents of this manual are subject to change without prior notice as a result of improvements in instrument's performance and functions.
- Every effort has been made in the preparation of this manual to ensure the accuracy of its contents. However, should you have any questions or find any errors, please contact your nearest YOKOGAWA representative listed on the back cover of this manual.
- Copying or reproduction of all or any part of the contents of this manual without YOKOGAWA's permission is strictly prohibited.

Trademarks

- HP-GL is a registered trademark of Hewlett-Packard Company.
- IBM PC/AT is a registered trademark of International Business Machines Corporation.
- Other product names are trademarks or registered trademarks of their respective holders.

Revisions

1st Edition: January 1999

Disk No. DL24
1st Edition: January 1999 (YK)

How to Use this Manual

Structure of this Manual

This User's Manual consists of five chapters, an Appendix and an Index as described below.

Chapter 1 Overview of the GP-IB Interface

Describes the functions and specifications of GP-IB.

Chapter 2 Overview of the RS-232 Interface

Describes the functions and specifications of RS-232.

Chapter 3 Before Programming

Describes formats used when sending a command.

Chapter 4 Command

Describes each command.

Chapter 5 Status Report

Describes the status byte, various registers and queues.

Chapter 6 Sample Programs

Sample programs, written in Visual BASIC, for MS-DOS/V machines equipped with the following GP-IB board: AT-GPIB/TNT IEEE-488.2, from National Instruments.

Appendix

Contains references including the ASCII character code table.

Index

Provides an alphabetically ordered index.

Conventions Used in this Manual

- **Symbols used for Notes and Keys**

Type	Symbol	Description
Unit	k K	1000 e.g.: 100 kS/s (sample rate) 1024 e.g.: 640 KB (floppy disk memory capacity)
Note	***Note***	Provides information that is necessary for proper operation of the instrument.
Key	"**Probe**"	Refers to a soft key displayed on the screen.

- **Symbols used in syntax descriptions**

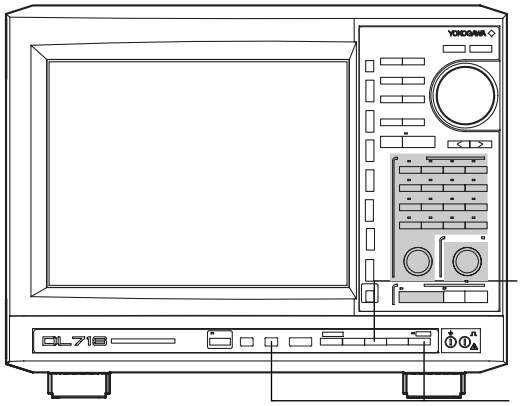
  Symbols which are used in the syntax descriptions in Chapter 4 are shown below. These symbols are referred to as BNF notation (Backus-Naur Form). For detailed information, refer to page 3-5.

Symbol	Description	Example	Example of Input
<>	Defined value	`CHANnel <x> <x>=1 to 4`	`→CHANNEL2`
{} |	One of the options in {} is selected. Exclusive OR	`COUPling {AC|DC|GND}`	`→COUPLING AC`
[]	Abbreviated	`TRIGger [:SIMPle]:SLOPe`	`→TRIGger:SLOPe`

Contents

1.1 Names of the Parts and Their Function

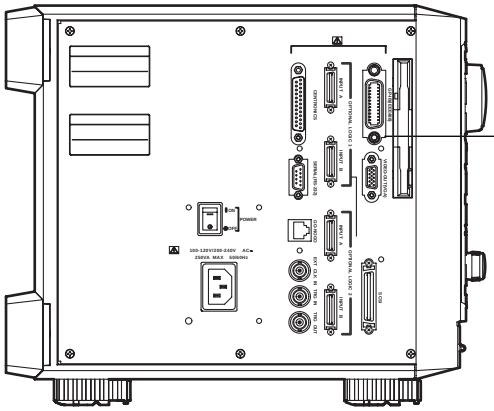
Front Panel



MISC key (page 2-8)
Press to enter the communication settings such as the address and the timeout.

SHIFT+CLEAR key
Press to switch from remote mode to local mode which allows key operation.
However, this is not possible if Local Lockout has been set by the controller (refer to page 1-8).

Rear Panel



GP-IB connector
This connector is for connecting the controller (such as a PC), the plotter or etc. with the GP-IB cable. For information on how to connect the GP-IB cable, refer to the following page.

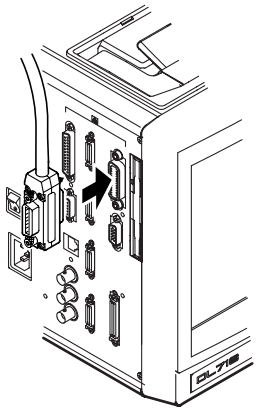
1.2 Connecting the GP-IB Cable

GP-IB Cable

The GP-IB connector on the side panel of the DL716 is a 24-pin connector that conforms to IEEE Standard 488-1978. Use a GP-IB cable that also conforms to IEEE Standard 488-1978.

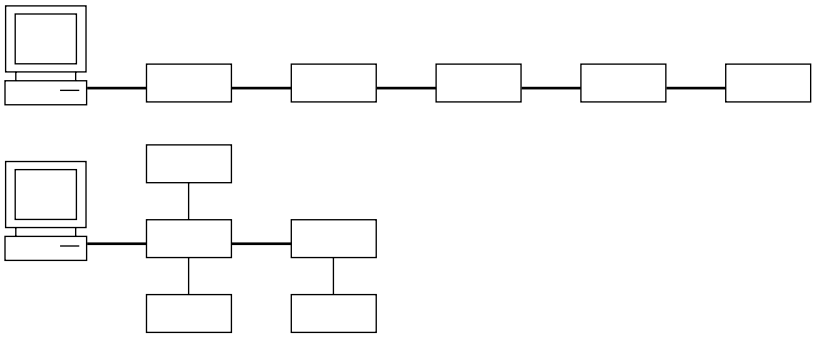
Connection Method

Connect the GP-IB cable as shown below.



Connection Precautions

- Be sure to tighten the screws on the GP-IB cable connector firmly.
- The instrument can be connected to more than one item of equipment (e.g. a personal computer) if more than one GP-IB cable is used. However, it is not possible to connect more than 15 items of equipment (including the controller) to a single bus.
- If you connect the instrument to more than one item of equipment, make sure that a different address is used for each item.
- Each connecting cable must be 2 m or less in length.
- The total length of all the cables must not exceed 20 m.
- While communications are in progress, more than two-thirds of the connected equipment items must be turned ON.
- When connecting more than one item of equipment, connect them so that the connection route forms a star or linear configuration. Loop or parallel wiring is not allowed.



CAUTION

Be sure to switch off power to both your PC and the oscilloscope before connecting or disconnecting cables. Failure to switch power off may cause internal circuit failure or improper operation.

1.3 GP-IB Interface Functions

GP-IB Interface Functions

Listener function

- Allows you to make the settings which you can make using the panel keys on the instrument, except for the power ON/OFF and GP-IB communications settings.
- Receives commands from a controller requesting output of set-up and waveform data. Also receives status report commands.

Talker function

- Outputs set-up and waveform data.

Talk-only function

- Outputs the screen data in various formats without using a controller. This is useful when you want to output the screen data to a plotter which conforms to the HP-GL requirements. For details, refer to Section 10.5 of the User's Manual (IM 701830-01E). Talk-only mode is entered automatically when output to a plotter is performed.
- Outputs waveform data to a YOKOGAWA AG Series arbitrary waveform generator, without using a controller. For more detailed information, refer to page 1-6.

Note

The listen-only and controller functions are not available on this instrument.

Switching between Remote and Local Modes

When switched from Local to Remote Mode

Remote mode is activated when a REN (Remote Enable) message is received from a controller while local mode is active

- REMOTE is displayed on the lower right of the screen.
- All front panel keys except the SHIFT+CLEAR key can no longer be operated any more.
- Settings entered in local mode are retained.

When switched from Remote to Local Mode

Pressing the SHIFT+CLEAR key in remote mode puts the instrument in local mode. However, this is not possible if Local Lockout has been set by the controller (page 1-8).

- The REMOTE indicator on the lower right of the screen is turned off.
- All front panel keys are operative.
- Settings entered in remote mode are retained.

1.4 GP-IB Interface Specifications

GP-IB Interface Specifications

Electrical and mechanical specifications	:	Conforms to IEEE Standard 488-1978.
Interface functions	:	Refer to the table below.
Protocol	:	Conforms to IEEE Standard 488.2-1992.
Code	:	ISO (ASCII) code
Mode	:	Addressable mode / Talk-only mode (switched automatically)
Address setting	:	Addresses 0 to 30 can be selected from the GP-IB setting screen, displayed when you press the MISC key.
Remote mode clear	:	Remote mode can be cleared by pressing the SHIF+CLEAR key. However, this is not possible if Local Lockout has been set by the controller.

Interface functions

Function	Subset Name	Description
Source handshaking	SH1	Full source handshaking capability
Acceptor handshaking	AH1	Full acceptor handshaking capability
Talker	T5	Basic talker capability, serial polling, untalk on MLA (My Listen Address), talk-only capability
Listener	L4	Basic listener capability, unlisten on MTA (My Talk Address), no listen-only capability
Service request	SR1	Full service request capability
Remote local	RL1	Full remote / local capability
Parallel poll	PP0	No parallel polling capability
Device clear	DC1	Full device clear capability
Device trigger	DT0	No device trigger capability
Controller	C0	No controller function
Electrical characteristic	E1	Open collector

Data Transfer Rate

The table below shows approximate response times for output of waveform data, assuming the following configuration.

Model	: 701830/M3(4MW/CH), Module701851
Controller	: DELL GXMT 5133
GB-IB Board	: NI AT-GPIB/TNT(PNP)
Programming language	: Borland C++

Data volume	Byte format	Word format	ASCII format
1002	about 25 ms	about 30 ms	about 650 ms
10020	about 75 ms	about 120 ms	about 6.4 s
100200	about 580 ms	about 1 s	about 63 s
1002000	about 5.6 s	about 9.9 s	about 622 s

1.5 Setting Addressable Mode

Before You Begin

When you make settings which can be made using the front panel keys of the instrument or when you output set-up data or waveform data using the controller, the following settings must be made.

Setting the address

This function allows you to set the instrument's address for addressable mode within the range of 0 to 30. Each item of equipment connected via a GP-IB interface has its own address, by which it can be identified. Care must be taken to ensure that all interconnected devices are assigned unique addresses.

Turning the display of the menu and setting parameters ON/OFF

This function allows you to select whether or not to display the menu and setting parameters on the right side of screen when the setting parameters of the instrument are being altered by communication interface. If this setting is set to ON, the new setting parameters will be updated on the screen. If this setting is set to OFF, and a command is received, the menu and setting parameters will be deleted from the screen. If this function is set to ON, however, the execution speed of commands is reduced.

Note

Do not change the address while the GP-IB interface is being used by the controller.

Operationg Procedure

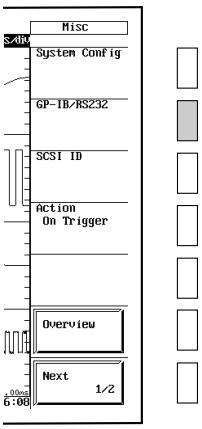
Displaying the GP-IB menu

1. Press the ***MISC*** key.
2. Press the "**GP-IB/RS232**" soft key.
3. Press the "**Comm Device**" soft key to select "**GPIB**."

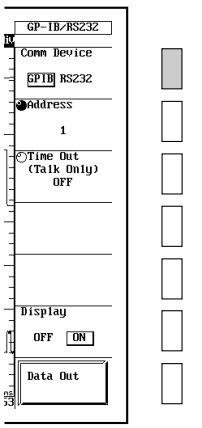
Setting the address and turning menu display ON/OFF

4. After step 3 is performed, press the "**Address**" soft key.
   The jog shuttle is now operative for setting the address.
5. Turn the jog shuttle to set the desired address.
6. Press the "**Display**" soft key to select either "**ON**" or "**OFF**."

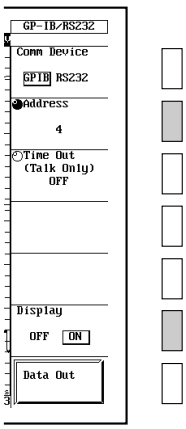
Step 1, 2



Step 3



Step 4, 5, 6



1.6 Setting Talk-Only Mode (Data Output to the AG Series)

Before You Begin

When you output waveform data to a YOKOGAWA AG series Arbitrary Waveform Generator without using the controller, the following settings must be made.

Setting the transmission time out period

This function allows you to set the instrument's transmission time out period within the following range. If this function is set to OFF, the transmission time out period will not be set, thus no time out will occur.

OFF, 10 s, 20 s, 30 s, 40 s, 50 s, 1 min, 2 min, 3 min, 4 min, 5 min, 6 min, 7 min, 8 min, 9 min, 10 min, 20 min, 30 min

When the GP-IB interface is used, data transmission will be stopped if the transmission data is not read by the receiver within the specified time. This specified time is called the time out period.

Outputting data

This function allows output of waveform data.

Note

The purpose of the transmission time out period is to ensure that time out does not occur before the specified time elapses.

Operating Procedure

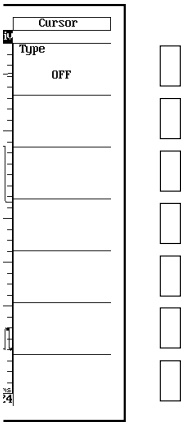
Connecting to the AG

1. Connect the instrument to the AG using the GP-IB cable.
2. Put the AG in listen-only mode. (Refer to the User's Manual for the AG.)

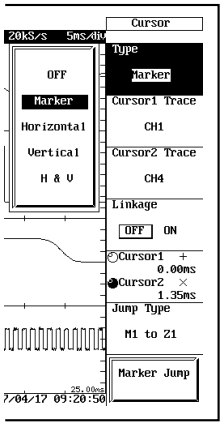
Selecting the waveform to be sent

3. Press the ***CURSOR*** key.
4. Press the "**Type**" soft key to select "**Marker**."
5. Select the desired channels for "**Cursor1 Trace**" and "**Cursor2 Trace**."
6. Locate the two marker cursors so that the portion of the waveform that is to be sent is enclosed by the cursors.

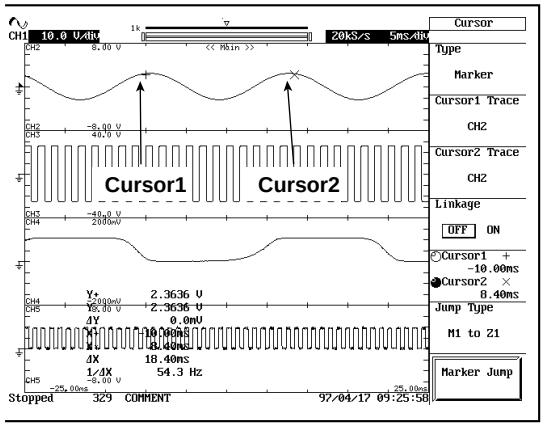
Step 3



Step 4



Step 5, 6



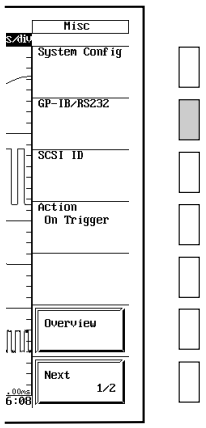
Displaying the GP-IB menu

7. Press the ***MISC*** key.
8. Press the "**GP-IB/RS232**" soft key.
9. Press the "**Comm Device**" soft key to select "**GPIB**."

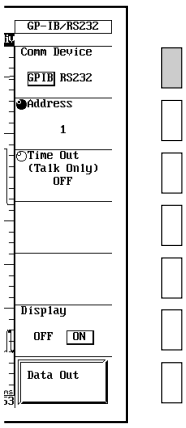
Setting the transmission time out period and outputting the data (in talk-only mode)

10. After step 9 is performed, press the "**Time Out**" soft key.
    The jog shuttle can now be used to set the time out period.
11. Turn the jog shuttle to set the desired time out period.
12. Press the "**Data Out**" soft key to output waveform data.

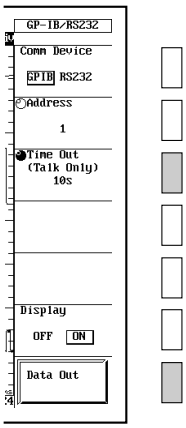
Step 7, 8



Step 9



Step 10, 11, 12



Note

Never press the "**Data Out**" soft key when equipment other than the AG is connected.
"**Cursor1 Trace**" and "**Cursor2 Trace**" should be set to the same waveform.
You do not need to set the cursors for a computed waveform, since all points are always sent.

1.7 Response to Interface Messages

Response to Interface Message

Response to a uni-line message

IFC (Interface Clear)

Clears the talker and listener. Stops output if data is being output.

REN (Remote Enable)

Switches between remote and local modes.

IDY (Identify) is not supported.

Response to a multi-line message (address command)

GTL (Go To Local)

Switches to local mode.

SDC (Selected Device Clear)

Clears the program message (command) which is currently being output. Also clears the output queue (page 5-5).
`*OPC` and `*OPC?` will be disabled if they are currently being executed.
`*WAI` and `COMMunicate:WAIT` will be stopped immediately.

PPC (Parallel Poll Configure), GET (Group Execute Trigger) and TCT (Take Control) are not supported.

Response to a multi-line message (universal command)

LLO (Local Lockout)

Invalidates the ***SHIFT+CLEAR*** key on the front panel to disable switching to local mode.

DCL (Device Clear)

Same as SDC.

SPE (Serial Poll Enable)

Sets the talker function to serial poll mode for all equipment connected to the communications bus. The controller performs polling on equipment sequentially.

SPD (Serial Poll Disable)

Clears serial poll mode as the talker function for all equipment connected to the communications bus.

PPU (Parallel Poll Unconfigure) is not supported.

What is an Interface Message?

An interface message is also called an interface command or bus command, and is issued by the controller. Interface messages are classified as follows.

Uni-line messages

Messages are transferred through a single control line. The following three types of uni-line message are available.
IFC (Interface Clear)
REN (Remote Enable)
IDY (Identify)

Multi-line message

Eight data lines are used to transmit a message. Multi-line messages are classified as follows.

Address commands

Valid when the equipment is designated as a listener or a talker. The following five address commands are available.

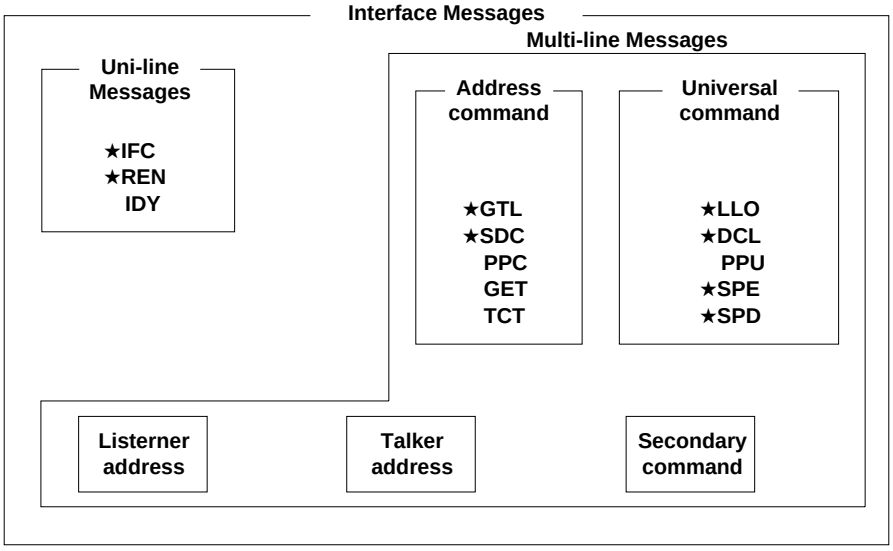
Commands valid for equipment designated as a listener;
GTL (Go To Local)
SDC (Selected Device Clear)
PPC (Parallel Poll Configure)
GET (Group Execute Trigger)

Command valid for equipment designated as a talker;
TCT (Take Control)

Universal commands

Valid for any item of equipment, irrespective of whether the item is designated as a listener or a talker. The following five universal commands are available.
LLO (Local Lockout)
DCL (Device Clear)
PPU(Parallel Poll Unconfigure)
SPE (Serial Poll Enable)
SPD (Serial Poll Disable)

In addition to the above commands, a listener address, talker address on secondary command can be sent in an interface message.



Messages marked with a "★" are interface messages supported by the DL716.

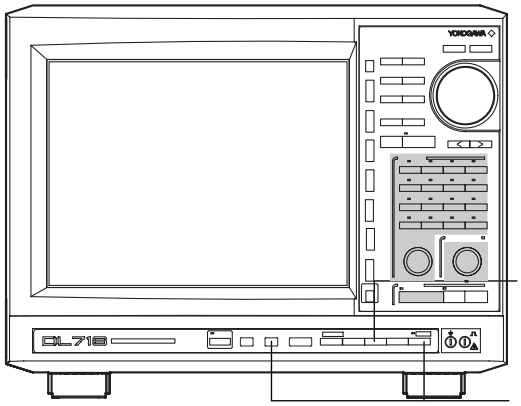
Note

Differences between SDC and DCL

The SDC command is an address command and requires that both the talker and listener be designated; however DCL is a universal command and does not require that the talker and listener be designated. Therefore, SDC is used for particular items of equipment, while DCL can be used for any equipment connected to the communications bus.

2.1 Names of the Parts and Their Function

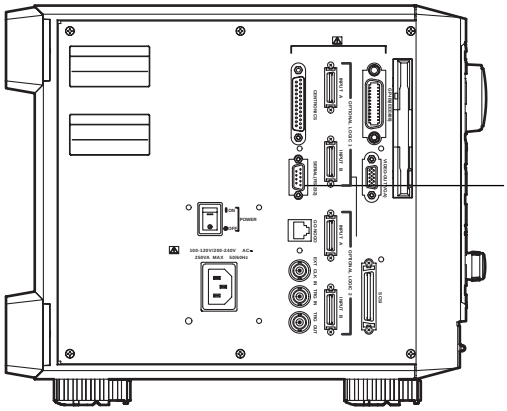
Front Panel



MISC key (page 2-8)
Press to enter the communication settings such as the baud rate, data format, and the handshaking method.

SHIFT+CLEAR key
Press to switch from remote mode to local mode which allows key operation. However, this is not possible if Local Lockout has been set by the controller (refer to page 1-8).

Side Panel



RS-232 connector
This connector is for connecting the controller (such as a PC), the plotter or etc. with the RS-232 cable. For information on how to connect the RS-232 cable, refer to the following page.

2.2 RS-232 Interface Functions and Specifications

Receiving Function

It is possible to make the same settings via the RS-232 interface as can be made using the front panel keys.
Measured / computed data, panel set-up information and error codes can be received.

Sending Function

Measured / computed data can be output.
Panel set-up information and the status byte can be output.
Error codes which have occurred can be output.

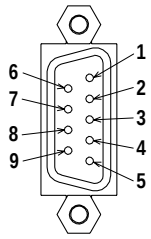
RS-232 Interface Specifications

Electrical characteristics	: Conforms to EIA RS-232.
Connection	: Point-to-point
Communications	: Full-duplex
Synchronization	: Start-stop system
Baud rate	: 1200, 2400, 4800, 9600, 19200
Start bit	: 1 bit (fixed)
Data Length	: 7 or 8 bits
Parity	: Even, odd or no parity
Stop Bit	: 1 or 2 bits
Connector	: DELC-J9PAF-13L6 (JAE or equivalent)
Hardware handshaking	: User can select whether CA or CB signals will always be True, or will be used for control.
Software Handshaking	: User can select whether to control only transmission or both transmission and reception using X-on and X-off signals. X-on (ASCII 11H) X-off (ASCII 13H)
Receive buffer size	: 256 bytes

2.3 Connecting the RS-232 Interface Cable

When connecting this instrument to a computer, make sure that the handshaking method, data transmission rate and data format selected for the instrument match those selected for the computer.
For details, refer to the following pages. Also make sure that the correct interface cable is used.

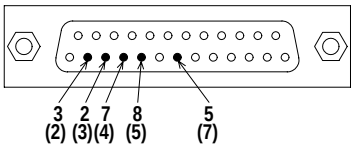
Connector and Signal Names



2. RD (Received Data) : Data received from personal computer.
   Signal direction...Input.
3. SD (Send Data) : Data transmitted to a personal computer.
   Signal direction...Output.
5. SG (Signal Ground) : Ground for signals.
7. RS (Request to Send) : Signal used for handshaking when receiving data from a personal computer.
   Signal direction...Output.
8. CS (Clear to Send) : Signal used for handshaking when transmitting data to a personal computer.
   Signal direction...Input.

Pin Nos. 1, 4, 6, and 9 are not used.

9-25 Pin Connector



The number between brackets refer to the pin Nos. of the 25-pin connector.

Signal Direction

The figure below shows the direction of the signals used by the RS-232 interface.

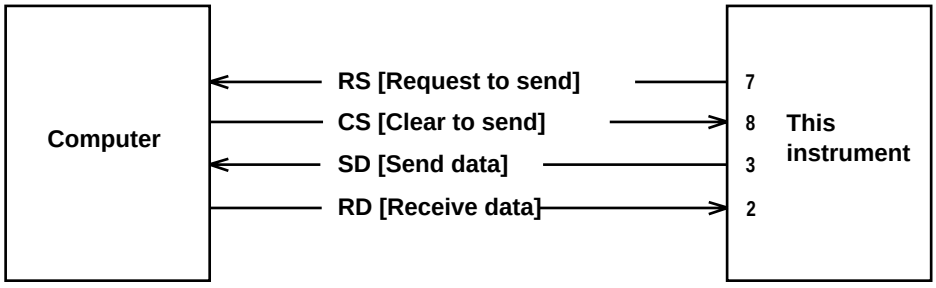


Table of RS-232 Standard Signals and their JIS and CCITT Abbreviations

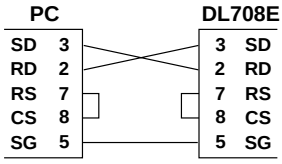
Pin No. (9-pin connector)	Abbreviation			Description
	RS-232	CCITT	JIS	
5	AB (GND)	102	SG	Signal ground
3	BA (TXD)	103	SD	Transmitted data
2	BB (RXD)	104	RD	Received data
7	CA (RTS)	105	RS	Request to send
8	CB (CTS)	106	CS	Clear to send

Signal line connection examples

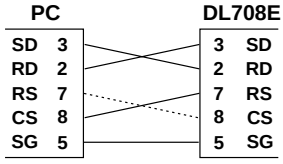
The pin numbers shown are that of 9-pin connectors.

In general, use a cross cable.

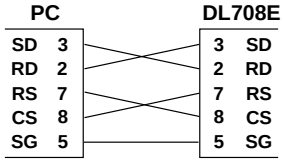
•OFF-OFF / XON-XON



•XON-RTS(XON-RS)



•CTS-RTS(CS-RS)



2.4 Handshaking

To use an RS-232 interface for transferring data between this instrument and a computer, it is necessary to use certain procedures by mutual agreement to ensure the proper transfer of data. These procedures are called “handshaking.” Various handshaking systems are available depending on the computer to be used; the same handshaking system must be used for both the computer and this instrument.

This instrument allows you to choose any handshaking mode from the following four modes.

Handshake format Descriptions→○

Handshake Method		Data Sending Control (control method when sending data to a computer)			Data Receiving Control (control method when receiving data from a computer)		
		Software Handshake	Hardware Handshake		Software Handshake	Hardware Handshake	
	The menu of this instrument	Sending stops when X-off is received, and sending is resumed when X-on is received.	Sending stops when CB(CTS) is False, and sending is resumed when CB is True.	No handshake	X-off is sent when received data buffer becomes 3/4-full, and X-on is sent when the received data buffer is only 1/4-full.	CA (RTS) is set to False when received data buffer is only 3/4-full, and is set to True when received data buffer is only 1/4-full.	No handshake
OFF-OFF	NO-NO			○			○
XON-XON	XON-XON	○			○		
XON-RS	XON-RTS	○				○	
CS-RS	CTS-RTS		○			○	

1 OFF-OFF

- **Transmission data control**

  There is no handshake status between the instrument and host computer. The X-OFF and X-ON signal from the host computer is processed as data, and the CS signal is ignored.
- **Reception data control**

  There is no handshake status between the recorder and host computer. When the recorder reception buffer becomes full, the excess data is discarded. RS = True (fixed).

2 XON-XON

- **Transmission data control**

  A software handshake status is established between the instrument and host computer. The instrument will stop a data transmission when an X-OFF signal is received from the host computer, and will resume transmission when the next X-ON signal is received. A CS signal from the host computer is ignored.
- **Reception data control**

  A software handshake status is established between the instrument and host computer. When the intstruments reception buffer vacancy reaches 64bytes, the X-OFF signal will be sent to the host computer. When the reception buffer vacancy reaches 192 bytes, the X-ON signal will be sent. RS = True (fixed).

3 XON-RS

- **Transmission data control**

  A software handshake status is established between the instrument and host computer. The instrument will stop a data transmission when an X-OFF signal is received from the host computer, and will resume transmission when the next X-ON signal is received. A CS signal from the host computer is ignored.
- **Reception data control**

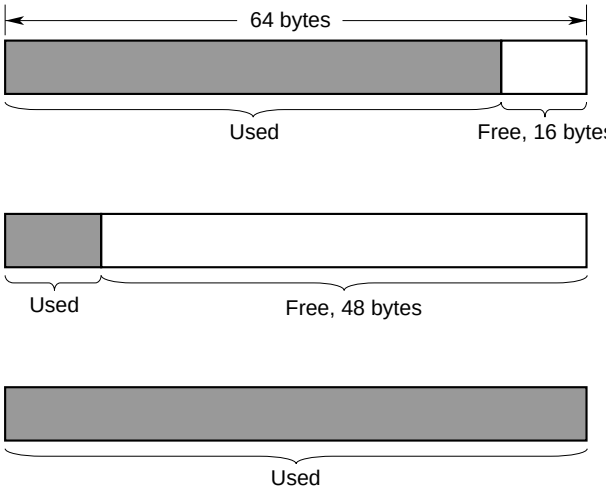
  A hardware handshake status is established between the instrument and host computer. When the intstruments reception buffer vacancy reaches 64bytes, an “RS=False” status will be established. When the reception buffer vacancy reaches 192 bytes, an “RS=True” status will be established.

4 CS-RS

- **Transmission data control**
  A software handshake status is established between the instrument and host computer. The instrument will stop a data transmission if a “CS = False” status is established, and will resume the transmission when a “CS = True” status is established. The X-OFF and X-ON signals from the host computer are processed as data.
- **Reception data control**
  A hardware handshake status is established between the instrument and host computer. When the intstruments reception buffer vacancy reaches 64bytes, an “RS=False” status will be established. When the reception buffer vacancy reaches 192 bytes, an “RS=True” status will be established.

Precautions Regarding Data Receiving Control

When handshaking is used to control the reception of data, data may still be sent from the computer even if the free space in the receive buffer drops below 64 bytes. In this case, after the receive buffer becomes full, the excess data will be lost, whether handshaking is in effect or not. Data storage to the buffer will begin again when there is free space in the buffer.



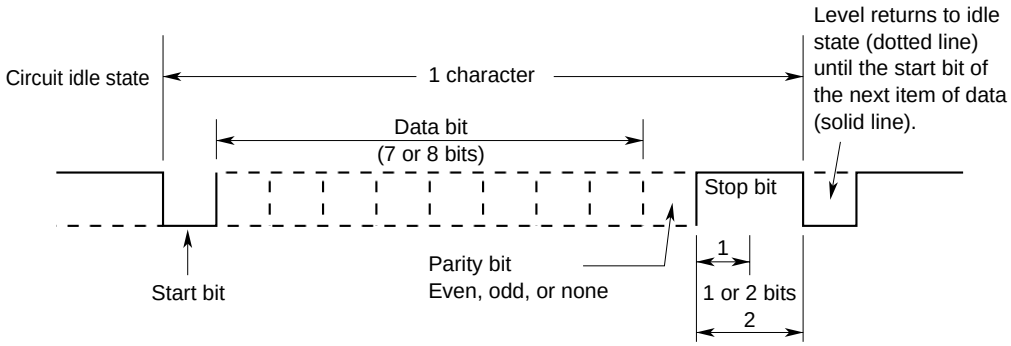
Data Receiving Control using Handshaking

Note

It is necessary to create a host computer program which prevents the buffers of both the intrument and the computer from becoming full.

2.5 Matching the Data Format

The RS-232 interface of this instrument performs communications using start-stop synchronization. In start-stop synchronization, one character is transmitted at a time. Each character consists of a start bit, data bits, a parity bit, and a stop bit. Refer to the figure below.



2.6 Setting up this Instrument

Before You Begin

When using the controller to set the items which can be set locally using the keys on the instrument, or when outputting the setup information or the waveform data to the controller, set the following items.

Baud rate

Select from the following choices.

1200, 2400, 4800, 9600, 19200

Data format

Select the combination of the data length and the stop bit from the following choices.

8-NO-1, 7-EVEN-1, 7-ODD-1, 7-NO-2

Handshaking method

Select the transmit data control and the receive data control from the following choices.

NO-NO, XON-XON, XON-RTS, CTS-RTS

Terminator

Select from the following choices. The terminator used when sending the data from this instrument is selected on the menu. Use either "LF" or "CR+LF" for the terminator in receiving the data.

CR, LF, CR+LF

Turning the display of the menu and setting parameters ON/OFF

This function allows you to select whether or not to display the menu and setting parameters on the right side of screen when the setting parameters of the instrument are being altered by communication interface. If this setting is set to ON, the new setting parameters will be updated on the screen. If this setting is set to OFF, and a command is received, the menu and setting parameters will be deleted from the screen. If this function is set to ON, however, the execution speed of commands is reduced.

Operating Procedure

Displaying the RS-232 menu

1. Press the ***MISC*** key.
2. Press the "**GP-IB/RS232**" soft key.
3. Press the "**Comm Device**" soft key to select "**RS232**."

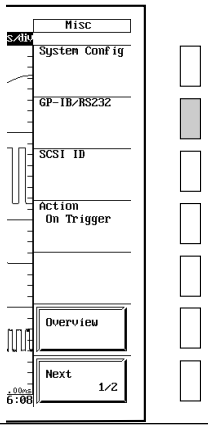
Selecting the baud rate, the data format and etc.

4. Press the "**BaudRate**" (baud rate), "**Format**" (data format), "**Rx-Tx**" (handshaking method), and the "**Terminator**" (terminator) soft keys individually, and turn the jog shuttle to set each item.

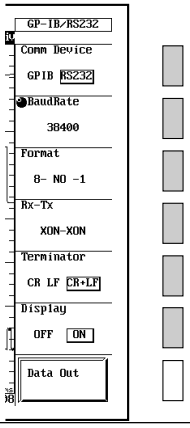
Menu display ON/OFF

5. Press the "**display**" soft key to select either "**ON**" or "**OFF**."

Step 1, 2



Step 3, 4, 5



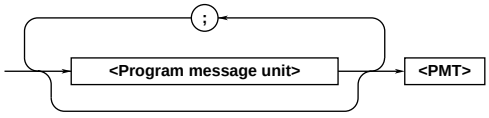
3 Before Programming

3.1 Messages

Blocks of message data are transferred between the controller and this instrument during communications. Messages sent from the controller to this instrument are called program messages, and messages sent back from this instrument to the controller are called response messages. If a program message contains a query command, i.e. a command which requests a response, this instrument returns a response message. A single response message is always returned in reply to a program message.

Program Messages

The format of a program message is shown below.



<Program message unit>

A program message consists of zero or more program message units; each unit corresponds to one command. This instrument executes commands one by one according to the order in which they are received.
Program message units are delimited by a ";."
For a description of the format of the program message unit, refer to the explanation given further below.

Example `:ACQuire:MODE NORMal;COUNt 1<PMT>`
Unit Unit

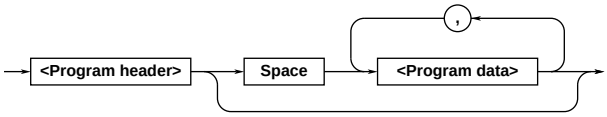
<PMT>

PMT is a terminator used to terminate each program message. The following three types of terminator are available.

`NL` (New Line) : Same as LF (Line Feed). ASCII code "0AH" is used.
`^END` : END message defined in IEEE488.1. (EOI signal) (The data byte sent with an END message will be the final item of the program message unit.)
`NL^END` : NL with an END message attached (NL is not included in the program message unit.)

Program message unit format

The format of a program message unit is shown below.



<Program header>

A program header is used to indicate the command type. For details, refer to page 3-3.

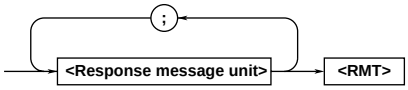
<Program data>

If certain conditions are required for the execution of a command, program data must be added. Program data must be separated from the header by a space (ASCII code "20H"). If multiple items of program data are included, they must be separated by a "," (comma). For details, refer to page 3-5.

Example `:ACQuire:MODE NORMal<PMT>`
Header Data

Response Messages

The format of a response message is shown below.



<Response message units>

A response message consists of one or more response message units: each response message unit corresponds to one response.
Response message units are delimited by a ";."
For the response message format, refer to the next page.

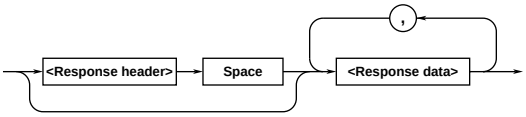
Example `:ACQUIRE:MODE NORMAL;COUNT 1<RMT>`
Unit Unit

<RMT>

RMT is the terminator used for every response message. Only one type of response message is available; `NL^END`.

Response message unit format

The format of a program message unit is shown below.



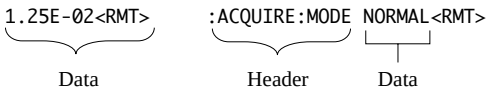
<Response header>

A response header sometimes precedes the response data. Response data must be separated from the header by a space. For details, refer to page 3-4.

<Response data>

Response data is used to define a response. If multiple items of response data are used, they must be separated by a “,” (comma). For details, refer to page 3-5.

Example



If a program message contains more than one query, responses are made in the same order as the queries. Normally, each query returns only one response message unit, but there are some queries which return more than one response message unit. The first response message unit always responds to the first query, but it is not always true that the ‘n’th unit always responds to the ‘n’th query. Therefore, if you want to make sure that a response is made to each query, the program message must be divided up into individual messages.

Points to Note concerning Message Transmission

- It is always possible to send a program message if the previous message which was sent did not contain any queries.
- If the previous message contained a query, it is not possible to send another program message until a response message has been received. An error will occur if a program message is sent before a response message has been received in its entirety. A response message which has not been received will be discarded.
- If an attempt is made by the controller to receive a response message, even if there it no response message, an error will occur. An error will also occur if the controller makes an attempt to receive a response message before transmission of a program message has been completed.
- If a program message of more than one unit is sent and some of the units are incomplete, this instrument receives program message units which the instrument thinks complete and attempts to execute them. However, these attempts may not always be successful and a response may not always be returned, even if the program message contains queries.

Deadlock

This instrument has a buffer memory in which both program and response messages of 1024 bytes or more can be stored. (The number of bytes available will vary depending on the operating state of the instrument.) If both buffer memories become full at the same time, this instrument becomes inoperative. This state is called deadlock. In this case, operation can be resumed by discarding the response message.

No dead lock will occur, if the size of the program message including the PMT is kept below 1024 bytes. Furthermore, no deadlock will occur if the program message does not contain a query.

3.2 Commands

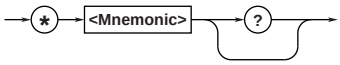
There are three types of command (program header) which can be sent from the controller to this instrument. They differ in the format of their program headers.

They are

- Common command header
- Compound header
- Simple header

Common Command Header

Commands defined in IEEE 488.2-1987 are called common commands. The header format of a common command is shown below. An asterisk (*) must always be attached to the beginning of a command.

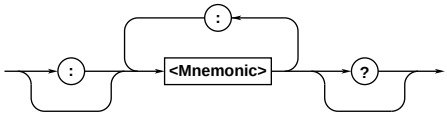


An example of a common command

```
*CLS
```

Compound Header

Commands designed to be used only with this instrument are classified and arranged in a hierarchy according to their function. The format of a compound header is illustrated below. A colon (:) must be used when specifying a lower-level header.

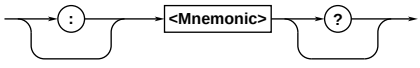


An example of a compound header

```
:ACQuire:MODE
```

Simple Header

These commands (headers) are functionally independent of each other and are not arranged hierarchically. The format of a simple header is shown below.



An example of a simple header

```
:STARt
```

Note

A mnemonic is a character string made up of alphanumeric characters.

When Concatenating Commands

Command Group

A command group is a group of commands which have the same compound header. A command group may contain sub-groups.

Example Commands relating to acquisition settings

```
:ACQuire:AVERage:COUNt          :ACQuire:MODE
:ACQuire:AVERage:EWEight        :ACQuire:RLENgth
:ACQuire:BAVerage:COUNt         :ACQuire:RTIMe
:ACQuire:COUNt                  :ACQuire:RTOut
:ACQuire:ENVelope:COUNt         :ACQuire:SLENgth
:ACQuire:SEQuential:COUNt
```

When Concatenating Commands of the Same Group

This instrument stores the hierarchical level of the command which is currently being executed, and performs analysis on the assumption that the next command to be sent will also belong to the same level. Therefore, it is possible to omit the header if the commands belong to the same group.

Example `:ACQuire:MODE NORMal;COUNt 1<PMT>`

When Concatenating Commands of Different Groups

A colon (:) must be included before the header of a command, if the command does not belong to the same group as the preceding command.

Example

```
:ACQuire:MODE NORMal;:DISPlay:FORMat SINGle<PMT>
```

When Concatenating Simple Headers

When you type in a simple header after another command, you must include a colon (:) before the simple header.

Example `:ACQuire:MODE NORMal;:STARt<PMT>`

When Concatenating Common Commands

Common commands defined in IEEE 488.2-1987 are independent of hierarchical level. Thus, it is not necessary to add a colon (:) before a common command.

Example `:ACQuire:MODE NORMal;*CLS;COUNt 1<PMT>`

When Separating Commands with <PMT>

If a terminator is used to separate two commands, each command is a separate message. Therefore, the common header must be typed in for each command even when commands of the same command group are being concatenated.

Example

```
:ACQuire:MODE NORMal<PMT>:ACQuire:COUNt 1<PMT>
```

Upper-level Query

An upper-level query is a compound header to which a question mark is appended. Execution of an upper-level query allows all a group's settings to be output at once. Some query groups comprising more than three hierarchical levels can output all their lower level settings.

Example

```
:CHANnel1?<PMT>→:CHANNEL1:DISPLAY ON;
LABEL ”CH1    ”;VOLTAGE:COUPLING DC;
POSITION 0.00;PROBE 10;VDIV 50.0E+00;
BWIDTH FULL;INVERT 0;OFFSET 0.0E+00;LSCALE:
MODE 0
```

In reply to a query, a response can be returned as a program message to this instrument. Transmitting a response can restore the settings made when the query was executed. However, some upper-level queries will not return set-up data which is not currently in use. Note that not all a group's information will necessarily be sent out as a response.

Header Interpretation Rules

This instrument interprets the header received according to the following rules.

- Mnemonics are not case sensitive.
  Example
  “CURSor” can also be written as “cursor” or “Cursor.”
- The lower-case part of a header can be omitted.
  Example
  “CURSor” can also be written as “CURSO” or “CURS.”
- If the header ends with a question mark, the command is a query. It is not possible to omit the question mark.
  Example
  “CURSor?” cannot be abbreviated to anything shorter than “CURS?.”
- If the “x” at the end of a mnemonic is omitted, it is assumed to be “1.”
  Example
  If “CHANnel<x>” is written as “CHAN,” this represents “CHANnel1.”
- Any part of a command enclosed by [ ] can be omitted.
  Example
  “TRIGger[:SIMPLle]:LEVel” can be written as “TRIG:LEV.”
- However, a part enclosed by [ ] cannot be omitted if it is located at the end of an upper-level query.
  Example
  “TRIGger?” and “TRIGger:SIMPle?” belong to different upper-level query levels.

3.3 Response

On receiving a query from the controller, this instrument returns a response message to the controller. A response message is sent in one of the following two forms.

- Response consisting of a header and data
  If the query can be used as a program message without any change, a command header is attached to the query, which is then returned.
  Example
  `:ACQUire:MODE?<PMT>→:ACQUire:MODE NORMAL<RMT>`
- Response consisting of data only
  If the query cannot be used as a program message unless changes are made to it (i.e. it is a query-only command), no header is attached and only the data is returned. Some query-only commands can be returned after a header is attached to them.
  Example

```
:MEASure:CHANnel1:PTOPeak:VALue?<PMT>→
10.0E+00<RMT>
```

When returning a response without a header

It is possible to remove the header from a response consisting of a header and data. The “COMMunicate:HEADer” command is used to do this.

Abbreviated form

Normally, the lower-case part is removed from a response header before the response is returned to the controller. Naturally, the full form of the header can also be used. For this, the “COMMunicate:VERBose” command is used. The part enclosed by [ ] is also omitted in the abbreviated form.

3.4 Data

Data

A data section comes after the header. A space must be included between the header and the data. The data contains conditions and values. Data is classified as below.

Data	Description
<Decimal>	Value expressed as a decimal number (Example: CH2's probe attenuation →CHANnel2:PROBe 100)
<Voltage><Time><Frequency>	Physical value (Example: Time axis range→TIMebase:TDIV 1US)
<Register>	Register value expressed as either binary, octal, decimal or hexadecimal (Example: Extended event register value →STATus:EESE #HFE)
<Character data>	Specified character string (mnemonic). Can be selected from { } (Example: CH2 input coupling →CHANnel2:COUPling {AC|DC|GND})
<Boolean>	Indicates ON/OFF. Set to ON, OFF or value (Example: CH2 display ON →CHANnel2:DISPlay ON)
<Character string data>	Arbitrary character string (Example: Comment on screen-data output →HCOPy:COMMent "ABCDEF")
<Filename>	Gives the name of a file. (Example: Name of file to be saved →FILE:SAVE:WAVeform:NAME "CASE1")
<Block data>	Arbitrary 8-bit data (Example: Response to acquired waveform data →#800000010ABCDEFGHIJ)

<Decimal>

<Decimal> indicates a value expressed as a decimal number, as shown in the table below. Decimal values are given in the NR form specified in ANSI X3. 42-1975.

Symbol	Description	Example
<NR1>	Integer	125 −1 +1000
<NR2>	Fixed point number	125.0 −.90 +001.
<NR3>	Floating point number	125.0E+0 −9E−1 +.1E4
<NRf>	Any of the forms <NR1> to <NR3> is allowed.	

Decimal values which are sent from the controller to this instrument can be sent in any of the forms to <NR3>. In this case, <NRf> appears.
For response messages which are returned from this instrument to the controller, the form (<NR1> to <NR3> to be used) is determined by the query. The same form is used, irrespective of whether the value is large or small.
In the case of <NR3>, the "+" after the "E" can be omitted, but the "−" cannot.
If a value outside the setting range is entered, the value will be normalized so that it is just inside the range.
If the value has more than the significant number of digits, the value will be rounded.

<Voltage>, <Time>, <Frequency>

<Voltage>, <Time>, and <Frequency> indicate decimal values which have physical significance. <Multiplier> or <Unit> can be attached to <NRf>. They can be entered in any of the following forms.

Form	Example
<NRf><Multiplier><Unit>	5MV
<NRf><Unit>	5E-3V
<NRf><Multiplier>	5M
<NRf>	5E-3

<Multiplier>

Multipliers which can be used are shown below.

Symbol	Word	Description
EX	Exa	10^{18}
PE	Peta	10^{15}
T	Tera	10^{12}
G	Giga	10^{9}
MA	Mega	10^{6}
K	Kilo	10^{3}
M	Mili	10^{-3}
U	Micro	10^{-6}
N	Nano	10^{-9}
P	Pico	10^{-12}
F	Femto	10^{-15}
A	Atto	10^{-18}

<Unit>

Units which can be used are shown below.

Symbol	Word	Description
V	Volt	Voltage
S	Second	Time
HZ	Hertz	Frequency
MHZ	Megahertz	Frequency

<Multiplier> and <Unit> are not case sensitive.

"U" is used to indicate "μ."

"MA" is used for Mega (M) to distinguish it from Mili, except for in the case of Megahertz, which is expressed as "MHZ." Hence, it is not permissible to use "M" (Mili) for Hertz.

If both <Multiplier> and <Unit> are omitted, the default unit will be used.

Response messages are always expressed in <NR3> form. Neither <Multiplier> nor <Unit> is used, therefore the default unit is used.

<Register>

<Register> indicates an integer, and can be expressed in hexadecimal, octal, or binary as well as as a decimal number. <Register> is used when each bit of a value has a particular meaning. <Register> is expressed in one of the following forms.

Form	Example
<NRf>	1
#H<Hexadecimal value made up of the digits 0 to 9, and A to F>	#H0F
#Q<Octal value made up of the digits 0 to 7>	#Q777
#B<Binary value made up of the digits 0 and 1>	#B001100

<Register> is not case sensitive.
Response messages are always expressed as <NR1>.

<Character Data>

<Character data> is a specified string of character data (a mnemonic). It is mainly used to indicate options, and is chosen from the character strings given in { }. For interpretation rules, refer to "Header Interpretation Rules" on page 3-4.

Form	Example
{AC|DC|GND}	AC

As with a header, the "COMMunicate:VERBose" command can be used to return a response message in its full form. Alternatively, the abbreviated form can be used.
The "COMMunicate:HEADer" command does not affect <character data>.

<Boolean>

<Boolean> is data which indicates ON or OFF, and is expressed in one of the following forms.

Form	Example
{ON|OFF|<NRf>}	ON OFF 1 0

When <Boolean> is expressed in <NRf> form, OFF is selected if the rounded integer value is "0" and ON is selected if the rounded integer is "Not 0."
A response message is always "1" if the value is ON and "0" if it is OFF.

<Character String Data>

<Character string data> is not a specified character string like <Character data>. It is an arbitrary character string. A character string must be enclosed in single quotation marks (') or double quotation marks (").

Form	Example
<Character string data>	'ABC' "IEEE488.2-1987"

Response messages are always enclosed in double quotation marks.

If a character string contains a double quotation mark ("), the double quotation mark will be replaced by two concatenated double quotation marks (""). This rule also applies to a single quotation mark within a character string.

<Character string data> is an arbitrary character string, therefore this instrument assumes that the remaining program message units are part of the character string if no single (') or double quotation mark (") is encountered. As a result, no error will be detected if a quotation mark is omitted.

<Filename>

Gives the name of a file. The format is as follows.

Form	Example
{<NRf>|<Character data>|<Character string>}	1 CASE "CASE"

If you input an <NRf> value, the system converts the value (after rounding to the nearest integer) to the corresponding 8-character ASCII string. (If you set the value to 1, the name becomes "00000001".) Note that negative values are not allowed.
If you enter a <character data> or <character string> argument that is longer than eight characters, only the first eight characters are used.
Response messages always return filenames as <character string> arguments.

<Block data>

<Block data> is arbitrary 8-bit data. <Block data> is only used for response messages. Response messages are expressed in the following form.

Form	Example
#N<N-digit decimal value><Data byte string>	#800000010ABCDEFGHIJ

#N

Indicates that the data is <Block data>. "N" is an ASCII character string number (digits) which indicates the number of data bytes that follow.

<N-digits decimal value>

Indicates the number of bytes of data. (Example: 00000010=10 bytes)

<Data byte string>

The actual data. (Example: ABCDEFGHIJ)

Data is comprised of 8-bit values (0 to 255). This means that the ASCII code "0AH," which stands for "NL," can also be a code used for data. Hence, care must be taken when programming the controller.

3.5 Synchronization with the Controller

Overlap Commands and Sequential Commands

There are two kinds of command; overlap commands and sequential commands. Execution of an overlap command may start before execution of the previously sent command is completed.

The "CHANnel1:VOLTage:VDIV" command, for example, is a sequential command. Assume that you set a new V/div value and immediately request return of the new value, as follows:

```
:CHANnel1:VOLTage:VDIV 5V;VIDV?<PMT>
```

In this case, the oscilloscope always returns the newest setting ("5V"). This is because it always completes processing of the current sequential command (in this case, "VDIV 5V") before moving on to the next command ("VDIV?").

In contrast, assume that you begin a file load and then immediately query the V/div value:

```
:FILE:LOAD:SETup:EXECute;:CHANnel1:VOLTage:VDIV?
```

Because "FILE:LOAD:SETup:EXECute" is an overlapped command, the oscilloscope will advance to the "CHANNel1:VOLTage:VDIV?" command before it finishes the load. The returned V/div value will not show the newest setting, but will rather show the setting in use before the setup was changed.

Obviously, use of overlapped commands may in some cases produce inappropriate results. Where necessary, you can avoid such problems as described below.

Synchronization with an Overlap Command

Using the *WAI command

The *WAI command causes the commands which follow it to wait until an overlap command has been executed.

Example

```
:COMMunicate:OPSE #0040;:FILE:LOAD:SETup:
EXECute;*WAI;:CHANnel1:VOLTage:VDIV?<PMT>
```

The "COMMunicate:OPSE" command is used to designate which commands are to be subject to the *WAI command. In the above example, only auto set-up is designated.

Since a *WAI command is executed just before "CHANnel1:VOLTage:VDIV?," "CHANnel1:VOLTage:VDIV?" will not be executed until auto set-up has been completed.

Using the COMMunicate:OVERlap command

The "COMMunicate:OVERlap" command is used to enable or disable overlap operation.

Example

```
:COMMunicate:OVERlap #HFFBF;:FILE:LOAD:SETup:
EXECute;:CHANnel1:VOLTage:VDIV:VALue?<PMT>
```

The "COMMunicate:OVERlap #HFFBF" command disables overlapped operation of the medium access command, while enabling all other overlap-type operations. The oscilloscope will therefore handle "FILE:LOAD:SETup:EXECute" as sequential command, ensuring that the "CHANnel1:VOLTage:VIDV?" command (in the above example) will not execute until file loading is completed.

Using the *OPC command

The *OPC command causes the OPC bit (bit 0) of the standard event register (page 5-3) to be set to "1" when an overlap operation has been completed.

Example

```
:COMMunicate:OPSE #H0040;*ESE 1;*ESR?;
*SRE 32;:FILE:LOAD:SETup:EXECute;*OPC<PMT>
```

(Response to *ESR? is decoded.)

(Service request is awaited.)

```
CHANnel1:VOLTage:VDIV:VALue?<PMT>
```

The "COMMunicate:OPSE" command is used to designate which commands are to be subject to the *OPC command. In the above example, only medium access commands are designated.

*ESE 1 and *SRE 32 stipulate that a service request is generated only when the OPC bit is set to "1."

*ESR? is used to clear the standard event register.

In the above example, "CHANnel1:VOLTage:VDIV?" will not be executed until a service request is generated.

Using the *OPC? query

The *OPC? query generates a response when an overlap operation has been completed.

Example

```
:COMMunicate:OPSE #H0040;:FILE:LOAD:SETup:
EXECute;*OPC?<PMT>
```

(Response to *OPC? is decoded.)

```
:CHANnel1:VOLTage:VDIV?<PMT>
```

The "COMMunicate:OPSE" command is used to designate which commands are to be subject to the *OPC? command. In the above example, only medium access commands are designated.

Since *OPC? does not generate a response until an overlap operation is completed, file loading will have been completed when a response to *OPC? is read.

Note

Most commands are sequential commands. Commands used in Chapter 4 are sequential commands unless otherwise specified.

Synchronization with Non-Overlap Commands

Synchronization is sometimes required for reasons other than communications-related reasons, such as the activation of a trigger, even if a sequential command is used.
As an example, the following message is properly used to query waveform data obtained by a "single start" operation:
`SSTart;WAVeform:SEND?<PMT>`
But sending this message (executing this command) before a single-start reading has been registered may result in a command error.
In this case, synchronization with the time at which acquisition is completed must be accomplished, as shown next.

Using STATus:CONDition? query

A "`STATus:CONDition?`" query is used to make an query about the contents of the condition register (page 5-4). It is possible to judge whether acquisition is in progress or not by reading bit 0 of the condition register. Bit 0 is "`1`" if acquisition is in progress, and "`0`" if acquisition is stopped.

Example
```
:SSTart<PMT>
:STATus:CONDition?<PMT>
```
(Returns to the previous status if bit 0 is found to be "`1`" when the response is decoded.)
```
:WAVeform:SEND?<PMT>
```

A "`WAVeform:SEND?`" query will not be executed until bit 0 of the condition register has been set to "`0`."

Using the extended event register

Changes in the condition register are reflected in the extended event register (page 5-4).

Example
```
:STATus:FILTer1 FALL;:STATus:EESE 1;EESR?;
*SRE 8;:SSTart<PMT>
```
(Response to "`STATus:EESR?`" is decoded.)
(Service request is awaited.)
```
:WAVeform:SEND?<PMT>
```

The "`STATus:FILTer1 FALL`" command sets the transition filter such that Bit 0 (FILTer1) of the Extended Event Register sets to 1 when Bit 0 of the Condition Register changes from 1 to 0.
"`STATus:EESE 1`" is a command used only to reflect the status of bit 0 of the extended event register in the status byte.
"`STATus:EESR?`" is used to clear the extended event register.
The `*SRE` command is used to generate a service request caused solely by the extended event register.
"`WAVeform:SEND?`" will not be executed until a service request is generated.

Using the COMMunicate:WAIT command

The "`COMMunicate:WAIT`" command halts communications until a specific event is generated.

Example
```
:STATus:FILTer1 FALL;:STATus:EESR?;:SSTart<PMT>
```
(Response to "`STATus:EESR?`" is decoded.)
```
:COMMunicate:WAIT 1;:WAVeform:SEND?<PMT>
```

For a description of "`STATus:FILTer1 FALL`" and "`STATus:EESR?`," refer to "Using the extended event register" on this page.
"`COMMunicate:WAIT 1`" means that communications is halted until bit 0 of the extended event register is set to "`1`."

4 Commands

4.1 Command List

Command	Description	Page
:CHANnel<x>:STRain:BALance:EXECute	Execute the balancing when the strain module is installed.	4-21
:CHANnel<x>:STRain:BALance:CHANnel<x>	Sets/queries the channel to be balanced.	4-21
:CHANnel<x>:STRain:BWIDth	Sets/queries the filter when the strain module is installed.	4-21
:CHANnel<x>:STRain:EXCitation	Sets/queries the Excitation when the strain module is installed.	4-21
:CHANnel<x>:STRain:GFACtor	Sets/queries the Gauge Factor when the strain module is installed.	4-21
:CHANnel<x>:STRain:LSCale?	Queries all settings relating to the linear scaling when the strain module is installed.	4-22
:CHANnel<x>:STRain:LSCale:AVALue	Set/queries the scaling coefficient A when the strain module is installed.	4-22
:CHANnel<x>:STRain:LSCale:BVALue	Sets/queries the offset B when the strain module is installed.	4-22
:CHANnel<x>:STRain:LSCale:MODE	Sets/queries the linear scaling mode when the strain module is installed.	4-22
:CHANnel<x>:STRain:LSCale:{P1X|P1Y|P2X|P2Y}	Sets/queries the P1:X|P1:Y|P2:X|P2:Y values of the linear scaling when the strain module is installed.	4-22
:CHANnel<x>:STRain:LSCale:UNIT	Sets/queries the unit that is added to the results of the linear scaling when the strain module is installed.	4-22
:CHANnel<x>:STRain:RANGe	Sets/queries the measurement range when the strain module is installed.	4-22
:CHANnel<x>:STRain:SCALe	Sets/queries the upper and lower limits of the display when the strain module is installed.	4-23
:CHANnel<x>:TEMPerature?	Queries all settings, when the temperature module is installed.	4-23
:CHANnel<x>:TEMPerature:BWIDth	Sets/queries the filter, when the temperature module is installed.	4-23
:CHANnel<x>:TEMPerature:RJC	Sets/queries the RJC, when the temperature module is installed.	4-23
:CHANnel<x>:TEMPerature:SCALe	Sets/queries the upper and lower limits on the screen, when the temperature module is installed.	4-23
:CHANnel<x>:TEMPerature:TYPE	Sets/queries the type of thermocouples to use,when the temperature module is installed.	4-23
:CHANnel<x>:TEMPerature:UNIT	Sets/queries the units for the upper and lower limits, when the temperature module is installed.	4-23
:CHANnel<x>:VOLTage?	Sets/queries all settings, when the voltage module is installed.	4-23
:CHANnel<x>[:VOLTage]:BWIDth	Sets/queries the filter, when the voltage module is installed.	4-24
:CHANnel<x>[:VOLTage]:COUPling	Sets/queries the input coupling, when the voltage module is installed.	4-24
:CHANnel<x>[:VOLTage]:INVert	Sets/queries how the waveform is to be displayed, inverted (ON) or not inverted (OFF), when the voltage module is installed.	4-24
:CHANnel<x>[:VOLTage]:LSCale?	Queries all linear scaling settings, when the voltage module is installed.	4-24
:CHANnel<x>[:VOLTage]:LSCale:AVALue	Sets/queries the scaling coefficient A, when the voltage module is installed.	4-24
:CHANnel<x>[:VOLTage]:LSCale:BVALue	Sets/queries the offset value B, when the voltage module is installed.	4-24
:CHANnel<x>[:VOLTage]:LSCale:MODE	Sets/queries the ON/OFF of linear scaling, when the voltage module is installed.	4-24
:CHANnel<x>[:VOLTage]:LSCale:UNIT	Sets/queries the unit which is appended to the linear scaling result, when the voltage module is installed.	4-24
:CHANnel<x>[:VOLTage]:OFFSet	Sets/queries the offset voltage, when the voltage module is installed.	4-25
:CHANnel<x>[:VOLTage]:POSition	Sets/queries the vertical position, when the voltage module is installed.	4-25
:CHANnel<x>[:VOLTage]:PROBe	Sets/queries the probe attenuation, when the voltage module is installed.	4-25
:CHANnel<x>[:VOLTage]:VDIV	Sets/queries the V/div setting, when the voltage module is installed.	4-25
:CHANnel<x>[:VOLTage]:ZOOM	Sets/queries the zooming ratio in the vertical direction, when the voltage module is installed.	4-25
CLEar Group		
:CLEar	Clears trace.	4-26
COMMunicate Group		
:COMMunicate?	Queries all communications settings.	4-27
:COMMunicate:HEADer	Selects whether response messages include headers.	4-27
:COMMunicate:LOCKout	Sets/releases the local lockout.	4-27
:COMMunicate:OPSE	Designates/queries the overlapped commands affected by *OPC, *OPC?, and *WAI commands.	4-27
:COMMunicate:OPSR?	Queries the Operation Pending Status Register.	4-27
:COMMunicate:OVERlap	Sets/queries overlap enable/disable for overlapped commands.	4-27
:COMMunicate:REMote	Sets remote/local.	4-27
:COMMunicate:STATus?	Queries circuit status.	4-27
:COMMunicate:VERBose	Selects/queries whether full spellings are used in response messages.	4-27
:COMMunicate:WAIT	Instructs oscilloscope to wait for specified extended event.	4-28
:COMMunicate:WAIT?	Instructs oscilloscope to return response upon occurrence of specified extended event.	4-28

Command	Description	Page
CURSor Group		
:CURSor?	Queries all cursor measurement settings.	4-31
:CURSor:HORizontal?	Queries all H cursor settings.	4-31
:CURSor:HORizontal:DY?	Queries Y distance between H cursors.	4-31
:CURSor:HORizontal:POSition<x>	Sets/queries H cursor position.	4-31
:CURSor:HORizontal:TRACe	HSets/queries H cursor's target waveform.	4-31
:CURSor:HORizontal:Y<x>?	Queries H cursor's Y-axis value.	4-31
:CURSor:LINKage	Sets/queries cursor linkage.	4-31
:CURSor:MARKer?	Queries all marker settings.	4-31
:CURSor:MARKer:DX?	Queries X distance between markers.	4-31
:CURSor:MARKer:DY?	Queries Y distance between markers.	4-31
:CURSor:MARKer:JUMP	Jumps marker to zoomed waveform, or queries the jump status.	4-31
:CURSor:MARKer:PERDt?(1 PER Delta T)	Queries the (horizontal) 1/ΔT differential between markers.	4-31
:CURSor:MARKer:POSition<x>	Sets/queries marker position.	4-32
:CURSor:MARKer:TRACe<x>	Sets/queries marker's target waveform.	4-32
:CURSor:MARKer:Y<x>?	Queries marker's Y-axis value.	4-32
:CURSor:TYPE	Sets/queries cursor type.	4-32
:CURSor:USERdefine?	Queries all settings relating to the user defined cursor.	4-32
:CURSor:USERdefine:DX?	Queries the X-axis value between the user defined cursors (relative value with respect to the reference cursor).	4-32
:CURSor:USERdefine:POSition<x>	Sets/queries the position of the user defined cursor.	4-32
:CURSor:USERdefine:REFerence<x>	Sets/queries the position of the user defined reference cursor.	4-32
:CURSor:USERdefine:RVALue	Sets/queries the reference width of the user defined cursor.	4-32
:CURSor:USERdefine:TRACe	Sets/queries the waveform to measure with the user defined cursor.	4-32
:CURSor:USERdefine:UNIT	Sets/queries the unit that is added to the cursor value of the user defined cursor.	4-32
:CURSor:USERdefine:X<x>?	Queries the X-axis value (relative value from the reference cursor) of the user defined cursor position.	4-33
:CURSor:VERTical?	Queries all V cursor settings.	4-33
:CURSor:VERTical:DX?	Queries X distance between V cursors.	4-33
:CURSor:VERTical:PERDt?	Queries the 1/ΔT differential between V cursors.	4-33
:CURSor:VERTical:POSition<x>	Sets/queries V cursor position.	4-33
:CURSor:VERTical:TRACe	Sets/queries V cursor's target waveform.	4-33
:CURSor:VERTical:X<x>?	Queries V cursor's X-axis value.	4-33
DISPlay Group		
:DISPlay?	Queries all display settings.	4-37
:DISPlay:ACCumulate?	Queries all waveform accumulation display settings.	4-37
:DISPlay:ACCumulate:COLor	Sets/queries color grade width.	4-37
:DISPlay:ACCumulate:MODE	Sets/queries the accumulate mode.	4-37
:DISPlay:ACCumulate:PERSistence	Sets/queries the accumulate count.	4-37
:DISPlay:CLEar	Clears trace.	4-37
:DISPlay:COLor?	Queries all display color settings.	4-37
:DISPlay:COLor:GRAPh?	Queries all color settings for graphs.	4-37
:DISPlay:COLor:GRAPh:{BACKground|CURSor|GRATicule|SNAP|CHANnel<x>}	Sets/queries color for cursors, waveforms, etc.	4-37
:DISPlay:COLor:GRAPh:MODE	Sets/queries the color mode for graph items.	4-37
:DISPlay:COLor:TEXT?	Queries all color settings for text items	4-38
:DISPlay:COLor:TEXT:{BACKground|BOX|LETTer|SELected|SUB}	Sets/queries color for menus, boxes, keys, or letters.	4-38
:DISPlay:COLor:TEXT:MODE	Sets/queries color for text items.	4-38
:DISPlay:DATE	Sets date/time display ON or OFF, or queries the setting.	4-38
:DISPlay:EWINdow(Extra WINDOW)	Sets extra window ON/OFF, or queries the setting.	4-38
:DISPlay:FORMat	Sets/queries display format.	4-38
:DISPlay:GRATicule	Sets/queries graticule type.	4-38
:DISPlay:INTerpolate	Sets/queries display interpolation method.	4-38
:DISPlay:MAPPing	Sets/queries mapping mode.	4-38
:DISPlay:SMAPping?	Queries all settings relating to the assignment of the waveforms to the split window.	4-38
:DISPlay:SMAPping:{CHANnel<x>|MATH<x>}	Sets/queries the assignment of the waveforms to the split window.	4-39
:DISPlay:SNAP	Takes a snapshot.	4-39
:DISPlay:SSAVer(Screen SAVER)	Sets/queries screen saver ON/OFF.	4-39

Command	Description	Page
:DISPlay:STYLe	Sets/queries display compression.	4-39
:DISPlay:SVALue(Scale VALUE)	Sets/queries scale display.	4-39
:DISPlay:TLABel(Trace LABEL)	Selects/queries whether waveform labels are displayed.	4-39
:DISPlay:TRANslucent	Sets/queries the ON/OFF of the translucent mode of the pop-up menu.	4-39
:DISPlay:TRIGger	Sets/queries trigger mark ON/OFF.	4-39
FILE Group		
:FILE?	Queries all settings for the specified medium.	4-43
:FILE:COPY?	Queries all file copy settings.	4-43
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}	Executes copy.	4-43
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}?	Queries copy settings.	4-43
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:ABORt	Aborts copy.	4-43
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:COMMent	Sets/queries comment on copy destination file.	4-43
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:DDIRectory(Destination DIRECTORY)	Sets/queries destination directory for copy.	4-43
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:DMEDia(Destination MEDIA)	Sets/queries destination medium.	4-43
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:DNAMe	Sets/queries destination filename.	4-44
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:EXECute	Executes copy.	4-44
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:SNAMe	Sets/queries source filename.	4-44
:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}	Deletes data file.	4-44
:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:EXECute	Deletes data file.	4-44
:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:NAME	Sets/queries filename of file for deletion.	4-44
:FILE:DIRectory?	Queries file settings in specified directory.	4-44
:FILE:DIRectory:COMMent	Sets comment ON/OFF, or queries current setting.	4-44
:FILE:DIRectory:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}?	Requests name(s) of file(s) in directory.	4-45
:FILE:FORMat?	Queries medium format settings.	4-45
:FILE:FORMat:EXECute	Formats a medium.	4-45
:FILE:FORMat:MODE	Sets/queries the format mode.	4-45
:FILE:FORMat:TYPE	Sets/queries format type for floppy disk format.	4-45
:FILE:FREE?	Queries amount of free space on the medium.	4-45
:FILE:LOAD:{SETup|WAVeform}	Loads data from file.	4-45
:FILE:LOAD[:{SETup|WAVeform}]:ABORt	Aborts data load.	4-45
:FILE:LOAD:{SETup|WAVeform}:EXECute	Loads data from file.	4-45
:FILE:LOAD:{SETup|WAVeform}:NAME	Sets/queries name of file to be loaded.	4-46
:FILE:MISC?	Queries directory and medium settings.	4-46
:FILE[:MISC]:ADIRectory(Add DIRECTORY)	Adds one directory.	4-46
:FILE[:MISC]:CDIRectory	Sets/queries directory No. of directory to be moved.	4-46
:FILE:MISC:DIRectory?	Queries current directory No. and maximum directory No.	4-46
:FILE[:MISC]:MEDia	Sets/queries medium for file access.	4-46
:FILE:PROTect?	Queries all file protection settings.	4-46
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}	Sets file protection ON/OFF.	4-46
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}?	Queries file protection settings.	4-46
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:EXECute	Sets file protection ON/OFF.	4-46
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:NAME	Sets/queries file to be protected/unprotected.	4-47

Command	Description	Page
:FILE:PROTect[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:STATe		
	Sets/queries file protection status.	4-47
:FILE:SAVE?	Queries all file save settings.	4-47
:FILE:SAVE[:AGForm]:AGLength	Sets/queries data length for AG format file save.	4-47
:FILE:SAVE[:AGForm]:AGSTart	Sets/queries start location for AG format file save.	4-47
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}		
	Executes file save.	4-47
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}?		
	Queries all file save settings.	4-47
:FILE:SAVE[:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}]:ABORt		
	Aborts file save.	4-47
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}:ANAMing		
	Enables/disables/queries automatic naming of save files.	4-47
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}:COMMent		
	Sets/queries file save comment.	4-48
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}:EXECute		
	Executes file save.	4-48
:FILE:SAVE[:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}]:NAME		
	Sets/queries name of save file.	4-48
:FILE:SAVE:{ASCii|FLOat|WAVeform}:RANGe		
	Sets/queries the range in saving the waveform data.	4-48
:FILE:SAVE[:{AGForm|ASCii|FLOat}]:TRACe		
	Selects/queries waveform to be saved.	4-48
:FILE:TYPE?	Queries current medium type.	4-48
HCOPy Group		
:HCOPy? (Hard COPY)	Queries all screen-image data output settings.	4-51
:HCOPy:ABORt	Aborts data output and paper feed.	4-51
:HCOPy:{BMP|TIFF}?	Queries all BMP-format (TIFF) settings.	4-51
:HCOPy:{BMP|TIFF}:COMPression	Sets/queries use of BMP (TIFF) compression.	4-51
:HCOPy:{BMP|TIFF}:TONE	Sets/queries BMP (TIFF) tonality.	4-51
:HCOPy:CENTronics?	Queries all settings relating to the output to the external printer.	4-51
:HCOPy:CENTronics:FORMat	Sets/queries the command type to output to the external printer.	4-51
:HCOPy:CENTronics:LONG	Sets/queries the magnification ratio of the external printer output.	4-51
:HCOPy:CENTronics:TONE	Sets/queries the half tone setting for the external printer output.	4-51
:HCOPy:COMMent	Sets/queries comment for top of the screen.	4-51
:HCOPy:DIRection	Sets/queries data output destination.	4-52
:HCOPy:EXECute	Executes data output.	4-52
:HCOPy:FORMat	Sets/queries format for screen imade data output.	4-52
:HCOPy:HPGL?	Queries all settings related to output to HP-GL type plotter.	4-52
:HCOPy:HPGL:CONNect	Selects/deselects line connection of graph points; or queries the setting.	4-52
:HCOPy:HPGL:FORMat	Sets/queries the plot size.	4-52
:HCOPy:HPGL:HADJust(Horizontal ADJUST)	Sets/queries horizontal adjustment of drawing position.	4-52
:HCOPy:HPGL:PAPer	Sets/queries paper size.	4-52
:HCOPy:HPGL:PEN?	Queries all pen assignment settings.	4-52
:HCOPy:HPGL:PEN:MODE	Sets/queries pen assignment mode.	4-52
:HCOPy:HPGL:PEN:NUMBer	Sets/queries number of pens for automatic assignment.	4-52
:HCOPy:HPGL:PEN:{CHANnel<x>|GRATicule|MATH<x>|TEXT}		
	Sets/queries pen setting for manual pen assignment.	4-53
:HCOPy:HPGL:SP0	Sets/queries use of appended SP0.	4-53
:HCOPy:HPGL:SPEed	Sets/queries pen speed.	4-53
:HCOPy:HPGL:VADJust(Vertical ADJUST)	Sets/queries vertical adjustment of drawing position.	4-53
:HCOPy:PRINter?	Queries all internal printer settings.	4-53
:HCOPy:PRINter:FEED	Feeds paper in internal printer.	4-53
:HCOPy:PRINter:LONG	Sets/queries long-copy operation.	4-53
:HCOPy:PRINter:SPLit	Sets/queries split print ON/OFF.	4-53
:HCOPy:SAVE?	Queries all file output settings.	4-53
:HCOPy:SAVE:ANAMing	Enables/disables/queries automatic naming of output files.	4-53
:HCOPy:SAVE:NAME	Sets/queries name for new file.	4-53

Command	Description	Page
HISTory Group		
:HISTory?	Queries all history settings.	4-54
:HISTory:DISPlay	Sets/queries data display method.	4-54
:HISTory:MAP	Sets/queries map number.	4-54
:HISTory:RECord	Sets/queries target record.	4-54
:HISTory:RECord? MINimum	Queries lowest available record under current settings.	4-54
:HISTory:TIME?	Queries trigger time of the specified record number.	4-54
IMAGe Group		
:IMAGe?	Sets/queries all settings relating to the screen image data output.	4-55
:IMAGe:FORMat	Sets/queries the output format of the screen image data.	4-55
:IMAGe:SEND?	Queries the screen image data.	4-55
:IMAGe:TONE	Sets/queries the color tone of the BMP (TIFF) format of the screen image data.	4-55
INITialize Group		
:INITialize[:EXECute]	Executes initialization.	4-55
:INITialize:UNDO	Cancels (undoes) previous initialization.	4-55
LSTart Group		
:LSTart(Log START)	Executes log start.	4-55
:LSTart?	Executes the log start and waits for the completion.	4-55
MATH Group		
:MATH<x>?	Queries all math (computation) settings.	4-61
:MATH<x>:BASic?	Queries all BASIC mode settings.	4-61
:MATH[1]:BASic:DEFine	Sets/queries BASIC mode's "Math1" expression.	4-61
:MATH2:BASic:DEFine	Sets/queries BASIC mode's "Math2" expression.	4-61
:MATH<x>:BASic:FFT?	Queries all FFT settings.	4-61
:MATH<x>:BASic:FFT:POINt	Sets/queries the number of points for the FFT.	4-61
:MATH<x>:BASic:FFT:WINDow	Sets/queries the window function of the FFT.	4-61
:MATH<x>:BASic:THReshold	Sets/queries BASIC mode's binarizing threshold.	4-61
:MATH<x>:BINary?	Queries all BINARY mode settings.	4-62
:MATH<x>:BINary:CHANnel<x>	Sets/queries BINARY mode's binarizing threshold of specified channel.	4-62
:MATH2:BINary:DA	Sets/queries BINARY mode's D/A conversion method.	4-62
:MATH[1]:BINary:MODE	Sets/queries BINARY mode's "Math1" computation mode.	4-62
:MATH2:BINary:MODE	Sets/queries BINARY mode's "Math2" computation mode.	4-62
:MATH<x>:EXECute	Executes math (computation) operation.	4-62
:MATH<x>:FFT?	Queries all FFT computation settings.	4-62
:MATH<x>:FFT:POINt	Sets/queries number of points for FFT computation.	4-62
:MATH<x>:FFT:WINDow	Sets/queries FFT window function.	4-62
:MATH<x>:MODE	Sets/queries math (computation) mode.	4-62
:MATH<x>:MREFerence(Math REFERENCE)	Sets/queries computation range.	4-62
:MATH<x>:PHASe?	Queries all PHASE mode settings.	4-63
:MATH[1]:PHASe:CHANnel<x>	Sets/queries phase shift for CH2 or CH5.	4-63
:MATH2:PHASe:CHANnel<x>	Sets/queries phase shift for CH3 or CH7.	4-63
:MATH[1]:PHASe:DEFine	Sets/queries PHASE mode's "Math1" expression.	4-63
:MATH2:PHASe:DEFine	Sets/queries PHASE mode's "Math2" expression.	4-63
:MATH<x>:SCALe?	Queries all scaling settings.	4-63
:MATH<x>:SCALe:MODE	Sets/queries the scaling method.	4-63
:MATH<x>:SCALe:VALue	Sets/queries the upper and lower limits for manual scaling.	4-63
:MATH<x>:UNIT	Sets/queries dimensional unit appended to computation result.	4-63
:MATH<x>:USERdefine?	Queries all settings regarding the user defined computation.	4-63
:MATH<x>:USERdefine:AVERage?	Queries all settings regarding the average of the user defined computation.	4-64
:MATH<x>:USERdefine:AVERage:CCOunt	Sets/queries the number of cycles (cycle count) to cycle average.	4-64
:MATH<x>:USERdefine:AVERage:COUNt	Sets/queries the number of acquisitions for linear averaging.	4-64
:MATH<x>:USERdefine:AVERage:EWEight	Sets/queries the attenuation constant for exponential averaging.	4-64
:MATH<x>:USERdefine:AVERage:MODE	Sets/queries the averaging mode.	4-64
:MATH<x>:USERdefine:AVERage:TYPE	Sets/queries the domain to average.	4-64
:MATH<x>:USERdefine:CONStant<x>	Sets/queries the constant for the user defined computation.	4-64
:MATH<x>:USERdefine:DEFine	Sets/queries the equation for the user defined computation.	4-64
:MATH<x>:USERdefine:FFT?	Queries all FFT settings.	4-64
:MATH<x>:USERdefine:FFT:POINt	Sets/queries the number of points for the FFT.	4-64

Command	Description	Page
:MATH<x>:USERdefine:FFT:WINDow	Sets/queries the FFT window function.	4-65
:MATH<x>:USERdefine:FILTer<x>?	Queries all filter settings.	4-65
:MATH<x>:USERdefine:FILTer<x>:BAND	Sets/Queries the bandwidth of the filter.	4-65
:MATH<x>:USERdefine:FILTer<x>:CUToff<x>	Sets/Queries the cutoff frequency.	4-65
:MATH<x>:USERdefine:FILTer<x>:TYPE	Sets/queries the filter type.	4-65
:MATH<x>:USERdefine:MODE	Sets/queries the enable/disable condition of the user defined computation.	4-65
:MATH<x>:USERdefine:THReshold?	Queries all threshold level settings for binary computation and pulse width computation.	4-65
:MATH<x>:USERdefine:THReshold:LEVel	Sets/queries the threshold level setting for binary computation and pulse width computation.	4-65
:MATH<x>:USERdefine:THReshold:TRACe	Sets/queries the trace to set the threshold.	4-65
MEASure Group		
:MEASure?	Queries all settings of automatic measurement of waveform parameter.	4-67
:MEASure:{CHANnel<x>|MATH<x>}?	Queries all parameter ON/OFF settings for specified waveform.	4-67
:MEASure:{CHANnel<x>|MATH<x>}:CLEar	Clears waveform's parameters.	4-67
:MEASure:{CHANnel<x>|MATH<x>}:<Parameter>[:STATe]	Sets/queries parameter ON/OFF for specified waveform.	4-67
:MEASure:{CHANnel<x>|MATH<x>}:<Parameter>:VALue?	Queries automatic measurement result (parameter value).	4-67
:MEASure:{CHANnel<x>|MATH<x>}:DPRoximal(Distal PROXIMAL)	Sets/queries waveform's distal, medial, and proximal points.	4-67
:MEASure:{CHANnel<x>|MATH<x>}:HISTogram?	Queries all histrogram display settings.	4-67
:MEASure:{CHANnel<x>|MATH<x>}:HISTogram[:STATe]	Sets/queries histrogram display ON/OFF.	4-68
:MEASure:CLEar	Clears measurement of all waveform parameters.	4-68
:MEASure:DELay?	Queries all delay settings.	4-68
:MEASure:DELay:RANGe	Sets/queries stating point for delayed measurement.	4-68
:MEASure:DELay:REFerence	Sets/queries delay reference point.	4-68
:MEASure:DELay:SLOPe	Sets/queries the slope of the delay reference waveform.	4-68
:MEASure:DPRoximal(Distal PROXIMAL)	Sets/queries distal, medial, and proximal values.	4-68
:MEASure:HREFerence(Horizontal REFERENCE)	Sets/queries measurement range.	4-68
:MEASure:MODE	Enables/disables/queries automatic measurement.	4-68
:MEASure:MREFerence(Magnitude REFERENCE)	Sets or queries HIGH/LOW point reference.	4-68
:MEASure:WAIT?	Waits for the completion of the automatic measurement	4-68
SNAP Group		
:SNAP	Executes a snapshot.	4-69
SSTart Group		
:SSTart(Single Start)	Execute single start	4-69
:SSTart?	Execute single start and wait for the completion with timeout.	4-69
STARt Group		
:STARt	Starts waveform acquisition.	4-69
STATus Group		
:STATus?	Queries all communication status settings.	4-70
:STATus:CONDition?	Queries the Condition Register.	4-70
:STATus:EESE(Extended Event Status Enable register)	Sets/queries the Extended Event Enable Register.	4-70
:STATus:EESR?(Extended Event Status Register)	Queries or clears the Extended Event Register.	4-70
:STATus:ERRor?	Queries error code and content.	4-70
:STATus:FILTer<x>	Sets/queries the transition filter.	4-70
:STATus:QENable	Selects/queries whether messages other than errors are stored in the error queue.	4-71
:STATus:QMESsage	Selects/queries whether message content is appended to responses to "STATus:ERRor?"	4-71
:STATus:SPOLl?(Serial Poll)	Executes the serial polling.	4-71

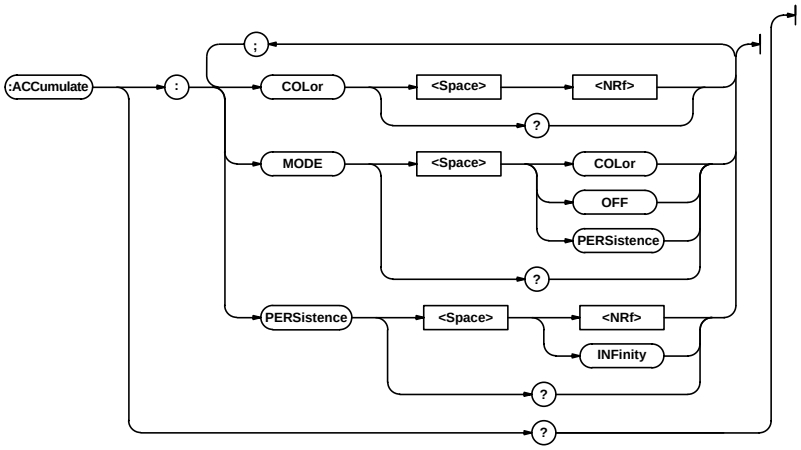
Command	Description	Page
STOP Group		
:STOP	Stops waveform acquisition.	4-71
SYSTem Group		
:SYSTem?	Queries all system settings.	4-73
:SYSTem:BATTery?	Queries status of internal lithium battery.	4-73
:SYSTem:BLIGht	Turns the screen backlight ON/OFF.	4-73
:SYSTem:CDISplay	Sets/queries the ON/OFF of the display of the setting values.	4-73
:SYSTem:CLICk	Sets/queries clock sound ON/OFF.	4-73
:SYSTem:DATE	Sets/queries date.	4-73
:SYSTem:HDMotor	Sets/queries the ON/OFF of the internal hard disk motor	4-73
:SYSTem:LANGuage	Sets/queries language used for messages.	4-73
:SYSTem:LBRightness	Sets/queries the brightness of the screen.	4-73
:SYSTem:LSCRounding	Sets/queries the rounding of the linear scaling values ON/OFF.	4-73
:SYSTem:PACTion	Sets/queries whether or not to enable the action on trigger mode setting at power up.	4-74
:SYSTem:PSTart	Sets/queries whether or not to start waveform acquisition at power up.	4-74
:SYSTem:SCSI?	Queries all SCSI ID settings.	4-74
:SYSTem:SCSI:EXTernalid	Sets/queries external device SCSI ID	4-74
:SYSTem:SCSI:INITialize	Initializes SCSI.	4-74
:SYSTem:SCSI:INTernalid	Sets/queries the SCSI-ID of the internal HD.	4-74
:SYSTem:SCSI:OWNid	Sets/queries own SCSI ID.	4-74
:SYSTem:SCSI:TERMinator	Sets/queries the SCSI terminator ON/OFF switch.	4-74
:SYSTem:TIME	Sets/queries time.	4-74
:SYSTem:VIDeo	Sets/queries the ON/OFF of the video output.	4-74
TIMebase Group		
:TIMebase?	Queries all the time base settings.	4-75
:TIMebase:FREQuency	Sets/queries the sampling rate.	4-75
:TIMebase:SRATe(Sample RATE)	Sets/queries the sampling rate.	4-75
:TIMebase:TDIV	Sets/queries the T/div setting.	4-75
TRIGger Group		
:TRIGger?	Queries all trigger settings.	4-79
:TRIGger:ABN?(A→B(n))	Queries all A→B(n) trigger settings.	4-79
:TRIGger:ABN:COUNt	Sets/queries "condition B=TRUE" count.	4-79
:TRIGger:ACTion?	Queries all action on trigger settings.	4-79
:TRIGger:ACTion:BUZZer	Sets/queries ON/OFF of the buzzer at the time of trigger action.	4-79
:TRIGger:ACTion:HCOPy	Sets/queries whether screen image data is output at the time of trigger action.	4-79
:TRIGger:ACTion:MODE	Sets/queriese ON/OFF of action on trigger function.	4-79
:TRIGger:ACTion:SAVE	Sets/queries whether waveform data is saved to the medium at the time of trigger action.	4-80
:TRIGger:ACTion:SEQuence	Sets/queries whether trigger action is executed continuously.	4-80
:TRIGger:ADB?(A Delay B)	Queries all "A delay B" trigger settings.	4-80
:TRIGger:ADB:DELay	Sets/queries delay time for condition B.	4-80
:TRIGger:ATRigger?	Queries all condition A settings.	4-80
:TRIGger:ATRigger:CHANnel<x>	Sets/queries condition A status on specified channel.	4-80
:TRIGger:ATRigger:CONDition	Sets/queries condition A TRUE criterion.	4-80
:TRIGger:BGTime?(B Greater TIME)	Queries all "B>Time" trigger settings.	4-80
:TRIGger:BGTime:TIME	Sets/queries "B>Time" trigger pulse width.	4-80
:TRIGger:BLTime?(B Less TIME)	Queries all "B<Time" trigger settings.	4-80
:TRIGger:BLTime:TIME	Sets/queries "B<Time" trigger pulse width.	4-81
:TRIGger:BTOut?(B Time OUT)	Queries all "B Time Out" trigger settings.	4-81
:TRIGger:BTOut:TIME	Sets/queries "B Time Out" trigger pulse width.	4-81
:TRIGger:BTRigger?	Queries all condition B settings.	4-81
:TRIGger:BTRigger:CHANnel<x>	Sets/queries condition B status on specified channel.	4-81
:TRIGger:BTRigger:CONDition	Sets/queries condition B TRUE criterion.	4-81
:TRIGger:CHANnel<x>?	Queries all trigger conditions on specified channel.	4-81
:TRIGger:CHANnel<x>:HYSTeresis	Sets/queries trigger hysteresis of specified channel.	4-81
:TRIGger:CHANnel<x>:LEVel	Sets/queries trigger level for specified channel.	4-81
:TRIGger:CHANnel<x>:PODA	Sets/queries the bit pattern of Pod A	4-82
:TRIGger:CHANnel<x>:PODB	Sets/queries the bit pattern of Pod B,	4-82

Command	Description	Page
:TRIGger:DELay	Sets/queries trigger delay.	4-82
:TRIGger:DREFerence(Delay REFERENCE)	Sets/queries trigger position.	4-82
:TRIGger:EOA?(Edge On A)	Queries all "Edge on A" trigger settings.	4-82
:TRIGger:EOA:CHANnel<x>	Sets/queries "Edge on A" trigger edge setting.	4-82
:TRIGger:EOR?(Edge OR)	Queries all "OR" trigger settings.	4-82
:TRIGger:EOR:CHANnel<x>	Sets/queries "OR" trigger edge setting.	4-83
:TRIGger:HOLDoff	Sets/queries holdoff time.	4-83
:TRIGger:MODE	Sets/queries the trigger mode.	4-83
:TRIGger:SIMPle?	Queries all "simple trigger" settings.	4-83
:TRIGger[:SIMPle]:HYSTeresis	Sets/queries trigger hysteresis of the channel specified by "TRIGger[:SIMPle]:SOURce"	4-83
:TRIGger[:SIMPle]:LEVel	Sets/queries simple trigger level.	4-83
:TRIGger[:SIMPle]:LOGic	Sets/queries source bit of logic input module	4-83
:TRIGger[:SIMPle]:SLOPe	Sets/queries simple trigger slope.	4-83
:TRIGger[:SIMPle]:SOURce	Sets/queries simple trigger source.	4-83
:TRIGger:TIMer?	Queries all settings relating to the timer trigger.	4-84
:TRIGger:TIMer:DATE	Sets/queries the trigger date of the timer trigger.	4-84
:TRIGger:TIMer:INTerval	Sets/queries the trigger time interval of the timer trigger.	4-84
:TRIGger:TIMer:TIME	Sets/queries the trigger time of the timer trigger.	4-84
:TRIGger:TYPE	Selects/queries the trigger type.	4-84
:TRIGger:WINDow?	Queries all window trigger settings	4-84
:TRIGger:WINDow:CENTer	Sets/queries window trigger center.	4-84
:TRIGger:WINDow:CONDition	Sets/queries window trigger condition.	4-84
:TRIGger:WINDow:HYSTeresis	Sets/queries trigger hysteresis	4-84
:TRIGger:WINDow:SOURce	Sets/querieswindow trigger source.	4-84
:TRIGger:WINDow:WIDTh	Sets/queries window trigger width.	4-84
WAVeform Group		
:WAVeform?	Queries all waveform data.	4-86
:WAVeform:BITS?	Queries bit length of specified waveform.	4-86
:WAVeform:BYTeorder	Sets/queries byte order for word-data transmission.	4-86
:WAVeform:END	Selects/queries final data point for specified waveform.	4-86
:WAVeform:FORMat	Sets/queries data transmission format.	4-86
:WAVeform:LENGth?	Queries waveform's data length (total number of data points).	4-86
:WAVeform:MODule?	Queries module of specified waveform.	4-86
:WAVeform:OFFSet?	Queries offset value used in converting specified waveform data.	4-86
:WAVeform:RANGe?	Queries range value used in converting specified waveform data.	4-86
:WAVeform:RECord	Sets/queries target record number.	4-86
:WAVeform:SEND?	Requests return of specified waveform (data string).	4-87
:WAVeform:SIGN?	Queries the presence of the sign of specified waveform data.	4-87
:WAVeform:SRATe?(Sample RATE)	Queries sampling rate of specified record.	4-87
:WAVeform:STARt	Sets/queries first data point for specified waveform.	4-87
:WAVeform:TRACe	Sets/queries target waveform.	4-87
:WAVeform:TRIGger?	Queries trigger position within specified record.	4-88
:WAVeform:TYPE?	Queries acquisition mode for specified waveform.	4-88
XY Group		
:XY?	Queries all X/Y display settings.	4-89
:XY:DREFerence(Division REFERENCE)	Sets/queries range for T/Y waveform on X/Y display.	4-89
:XY:MODE	Sets/queries display mode.	4-89
:XY:XTRace	Sets/queries X-axis channel for X/Y display.	4-89
ZOOM Group		
:ZOOM?	Queries all waveform zoom settings.	4-91
:ZOOM:ALLOcation?	Queries all settings for zoomed waveform.	4-91
:ZOOM:ALLOcation:{CHANnel<x>|MATH<x>}	Sets/queries zoomed waveform.	4-91
:ZOOM:FITMeasure	Move the range of the automatic measurement of waveform parameters.	4-91
:ZOOM:FORMat	Sets/queries zoom display format.	4-91
:ZOOM:LINKage	Enables/disables/queries zoom box linkage.	4-91

Command	Description	Page
:ZOOM:MAG<x>	Sets/queries zoom ratio.	4-91
:ZOOM:MODE	Sets/queries zoom display mode.	4-91
:ZOOM:POSition<x>	Sets/queries zoom box position.	4-91
:ZOOM:SCRoll?	Queries all settings relating to the auto scroll of the waveform.	4-91
:ZOOM:SCRoll:ASCRoll	Executes auto scroll of the waveform.	4-91
:ZOOM:SCRoll:DIRection	Sets/queries the direction to auto scroll the waveform.	4-91
:ZOOM:SCRoll:POSition	Sets/queries display position of waveform data.	4-91
:ZOOM:SCRoll:PSCRoll	Execute the page scroll of the waveform.	4-92
:ZOOM:SCRoll:SPEed	Sets/queries auto scroll speed of the waveform.	4-92
:ZOOM:SCRoll:STOP	Stop auto scroll of the waveform.	4-92
:ZOOM:TDIV<x>	Sets/queries zoomed waveform's T/div value.	4-92
Common Command Group		
*CAL?(CALibrate)	Executes calibration and returns the result.	4-93
*CLS(CLear Status)	Clears the Standard Event Register, Extended Event Register, and Error Queue.	4-93
*ESE(standard Event Status Enable register)	Sets/queries the Standard Event Enable register.	4-93
*ESR?(standard Event Status Register)	Queries and then clears the Standard Event Register.	4-94
*IDN?(IDeNtify)	Queries the instrument model.	4-94
*LRN?(LeaRN)	Queries current group settings.	4-94
*OPC(OPeration Complete)	Sets OPC event upon completion of specified overlapped operation.	4-94
*OPC?(OPeration Complete)	Returns response upon completion of specified overlapped commands.	4-94
*OPT?(OPTion)	Queries the installed options.	4-94
*PSC(Power-on Status Clear)	Enables/disables/queries power-on clearing of all registers.	4-94
*RST(ReSeT)	Executes a reset.	4-95
*SRE(Service Request Enable register)	Sets/queries the Service Request Enable Register.	4-95
*STB?(STatus Byte)	Queries the Status-Byte Register.	4-95
*WAI(WAIt)	Places subsequent commands on hold until completion of specified overlapped operation.	4-95

4.2 ACCumulate Group

The commands in the ACCumulate group are used to make settings and queries about the accumulation of waveforms. This allows you to make the same settings that you can make using the DISPLAY key on the front panel.



:ACCumulate?

Function Queries settings relating to accumulation of waveforms.
Syntax `:ACCumulate?`
Example `:ACCUMULATE?→:ACCUMULATE:MODE PERSISTENCE;PERSISTENCE 16`
Description The same query can be made using "`DISPlay:ACCumulate?`."

:ACCumulate:COLor

Function Sets the color grading width/queries about the current setting. "`ACCumulate:MODE COLor`" must be selected, otherwise this command is meaningless.
Syntax `:ACCumulate:COLor {<NRf>}`
`:ACCumulate:COLor?`
`<NRf>=2 to 32 (in steps of 2^n)`
Example `:ACCUMULATE:COLOR 16`
`:ACCUMULATE:COLOR?→:ACCUMULATE:COLOR 16`
Description The same setting and query can be made using "`DISPlay:ACCumulate:COLor`."

:ACCumulate:MODE

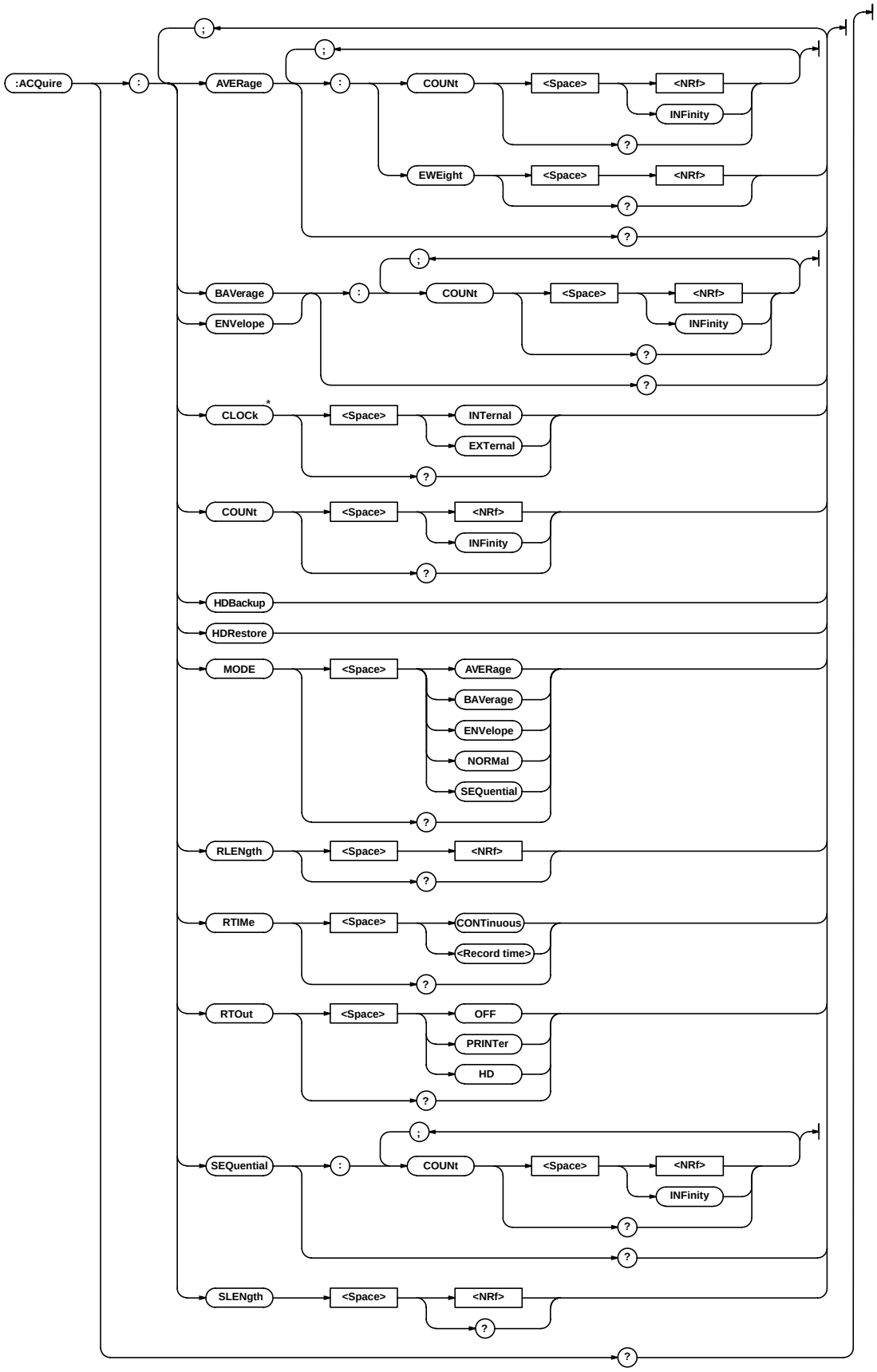
Function Sets/queries the accumulation mode.
Syntax `:ACCumulate:MODE {COLor|OFF|PERSistence}`
`:ACCumulate:MODE?`
Example `:ACCUMULATE:MODE PERSISTENCE`
`:ACCUMULATE:MODE?→:ACCUMULATE:MODE PERSISTENCE`
Description The same setting and query can be made using "`DISPlay:ACCumulate:MODE`."

:ACCumulate:PERSistence

Function Sets/queries the accumulation count.
Syntax `:ACCumulate:PERSistence {<NRf>|INFinity}`
`:ACCumulate:PERSistence?`
`<NRf>=2 to 128 (in steps of 2^n)`
Example `:ACCUMULATE:PERSISTENCE 16`
`:ACCUMULATE:PERSISTENCE?→:ACCUMULATE:PERSISTENCE 16`
Description The same setting and query can be made using "`DISPlay:ACCumulate:PERSistence`."

4.3 ACQuire Group

The commands in the ACQuire group are used for making settings and queries about acquisition. This allows you to make the same settings that you can make using the ACQ key on the front panel.



*** You can set and query only when using the external clock option.**

:ACQuire?

Function Quiries all the acquisition settings.
Syntax :ACQuire?
Example :ACQUIRE?→:ACQUIRE:RTOUT OFF;
MODE NORMAL;RLENGTH 1000;COUNT INFINITY

:ACQuire:AVERage?

Function Quiries all the averaging and acquisition count settings.
Syntax :ACQuire:AVERage?
Example :ACQUIRE:AVERAGE?→:ACQUIRE:AVERAGE:
COUNT INFINITY:EWEIGHT 16

:ACQuire:AVERage:COUNt

Function Sets/queries the acquisition count in averaging mode.
Syntax :ACQuire:AVERage:COUNt {<NRf>|INFinity}
:ACQuire:AVERage:COUNt?
<NRf>=2 to 65536 (in steps of 2^n)
Example :ACQUIRE:AVERAGE:COUNT INFINITY
:ACQUIRE:AVERAGE:COUNT?→:ACQUIRE:
AVERAGE:COUNT INFINITY

:ACQuire:AVERage:EWEight (Exponent WEIGHT)

Function Sets/queries the average weight for infinite averaging mode.
Syntax :ACQuire:AVERage:EWEight {<NRf>}
:ACQuire:AVERage:EWEight?
<NRf>=2 to 256 (in steps of 2^n)
Example :ACQUIRE:AVERAGE:EWEIGHT 16
:ACQUIRE:AVERAGE:EWEIGHT?→:ACQUIRE:
AVERAGE:EWEIGHT 16

:ACQuire:{BAVerage|ENVelope}?

Function Queries all box-average or envelope settings.
Syntax :ACQuire:{BAVerage|ENVelope}?
Example :ACQUIRE:BAVERAGE:COUNT INFINITY

:ACQuire:{BAVerage|ENVelope}:COUNt

Function Sets/queries the waveform acquisition count for box average or envelope mode.
Syntax :ACQuire:{BAVerage|ENVelope}:
COUNt {<NRf>|INFinity}
:ACQuire:{BAVerage|ENVelope}:COUNt?
<NRf>=2 to 65535
Example :ACQUIRE:BAVERAGE:COUNT INFINITY
:ACQUIRE:BAVERAGE:COUNT?→:ACQUIRE:
BAVERAGE:COUNT INFINITY

:ACQuire:CLOCk

Function Sets/queries the time base.
Syntax :ACQuire:{INTernal|EXTernal}
:ACQuire:CLOCk?
Example :ACQUIRE:CLOCK INTERNAL
:ACQUIRE:CLOCK?→:ACQUIRE:CLOCK INTERNAL
Description You can set and query only when using the external clock option.

:ACQuire:COUNt

Function Sets/queries the waveform acquisition count for "normal" mode.
Syntax :ACQuire:COUNt {<NRf>|INFinity}
:ACQuire:COUNt?
<NRf>=2 to 65535
Example :ACQUIRE:COUNT INFINITY
:ACQUIRE:COUNT?→:ACQUIRE:COUNT INFINITY

:ACQuire:MODE

Function Sets/queries the acquisition mode.
Syntax :ACQuire:MODE{AVERage|BAVerage|
ENVelope|NORMal|SEQuential}
:ACQuire:MODE?
Example :ACQUIRE:MODE NORMAL
:ACQUIRE:MODE?→:ACQUIRE:MODE NORMAL

:ACQuire:HDBackup

Function Backs up the waveform data that is realtime recorded to the internal hard disk.
Syntax :ACQuire:HDBackup
Example :ACQUIRE:HDBACKUP

:ACQuire:HDRestore

Function Loads the waveform data that is backed up to the internal hard disk.
Syntax :ACQuire:HDRestore
Example :ACQUIRE:HDRESTORE

:ACQuire:RLENgth

Function Sets/queries the record length.
Syntax :ACQuire:RLENgth {<NRf>}
:ACQuire:RLENgth?
<NRf>=1000 to 64000000
(The step refers to User's Manusl
IM701830-01E)
Example :ACQUIRE:RLENGTH 1000
:ACQUIRE:RLENGTH?→:ACQUIRE:RLENGTH 1000

:ACQuire:RTIMe

Function Sets/queries the record time for the realtime print or for the realtime record to the internal hard disk of the datas of the acquired waveform.
Syntax :ACQuire:RTIMe {OFF|SEC10|SEC20|SEC30|
SEC40|SEC50|MIN1|MIN2|MIN3|MIN4|MIN5|MIN6|
MIN7|MIN8|MIN9|MIN10|MIN20|MIN30|MIN40|
MIN50|HOUR1|HOUR2|HOUR3|HOUR4|HOUR5|HOUR6|
HOUR12|DAY1|DAY2|DAY3|DAY4|DAY5|DAY6|DAY7|
DAY8|DAY9|DAY10}
:ACQuire:RTIMe?
Example :ACQUIRE:RTIME SEC10
:ACQUIRE:RTIME?→:ACQUIRE:RTIME SEC10
Description Realtime print is executed with the STARt command.

:ACQuire:RTOut

Function Sets/queries whether to realtime print or to realtime record to the internal hard disk, the datas of the acquired waveform.

Syntax `:ACQuire:RTOut {HD|PRINter|OFF}`
`:ACQuire:RTOut?`

Example `:ACQUIRE:RTOUT PRINTER`
`:ACQUIRE:RTOUT?→:ACQUIRE:RTOUT PRINTER`

Description The internal hard disk is an option. An error will occur, if "HD" is selected when there is no hard disk.

:ACQuire:SEQuential?

Function Queries all sequential store mode settings.

Syntax `:ACQuire:SEQuential?`

Example `:ACQUIRE:SEQUENTIAL?→:ACQUIRE:SEQUENTIAL:COUNT INFINITY`

:ACQuire:SEQuential:COUNt

Function Sets/queries the acquisition count in sequential store mode.

Syntax `:ACQuire:SEQuential:COUNt {<NRf>|INFinity}`
`:ACQuire:SEQuential:COUNt?`
`<NRf>=2 to 1000`

Example `:ACQUIRE:SEQUENTIAL:COUNT INFINITY`
`:ACQUIRE:SEQUENTIAL:COUNT?→:ACQUIRE:SEQUENTIAL:COUNT INFINITY`

Description Maximum count value varies according to machine model and interleave ON/OFF setting. Refer to the User's Manual IM701830-01E for details.

:ACQuire:SLENgth (Store Length)

Function Sets/queries the record length for realtime recording the datas of the acquired waveform to the internal hard disk.

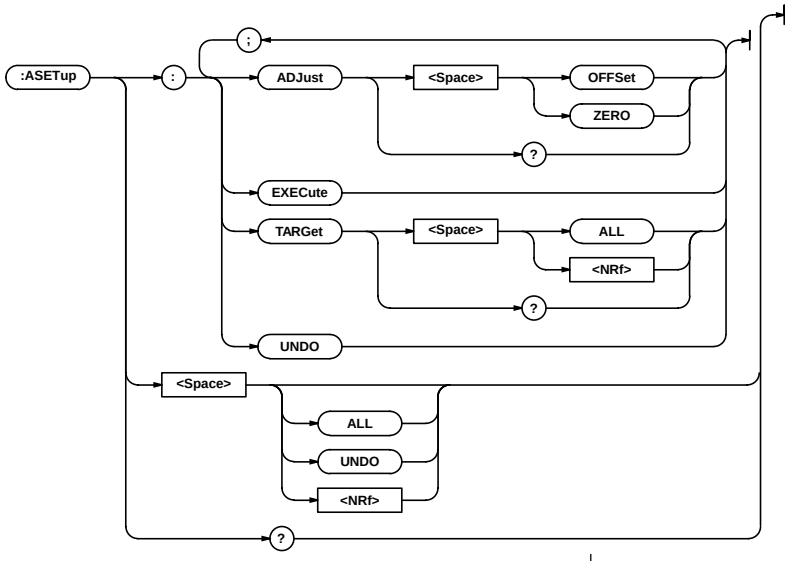
Syntax `:ACQuire:SLENgth {<NRf>}`
`:ACQuire:SLENgth?`
`<NRf>=1000000 to 256000000`

Example `:ACQUIRE:SLENGTH 1000000`
`:ACQUIRE:SLENGTH?→:ACQUIRE:SLENGTH 1000000`

Description An error will occur, if there is no hard disk.

4.4 ASETup Group

The commands in the ASETup group are used for setting and querying auto-setup. This allows you to make the same settings and queries that you can make using the AUTO-SETUP key on the front panel.



:ASETup (Auto SETUP)

Function Executes or cancels auto-setup.

Syntax `:ASETup [{ALL|UNDO|<NRf>}]`
`<NRf>=1 to 8`

Example `:ASETUP (Executes auto-setup using the current settings.)`
`:ASETUP ALL (Executes auto-setup for all channels.)`
`:ASETUP UNDO (Cancels auto-setup.)`
`:ASETUP 1 (Executes auto-setup for CH1.)`

Description
- The same function can be performed using "`ASETup:EXECute`" or "`ASETup:UNDO`."
- An error will occur, if a channel which does not have a module installed is specified.

:ASETup?

Function Queries all auto-setup settings.

Syntax `:ASETup?`

Example `:ASETUP?→:ASETUP:TARGET ALL;ADJUST ZERO`

:ASETup:ADJust

Function Sets/queries the center position for use after completion of auto-setup.

Syntax `:ASETup:ADJust {OFFSet|ZERO}`
`:ASETup:ADJust?`

Example `:ASETUP:ADJUST ZERO`
`:ASETUP:ADJUST?→:ASETUP:ADJUST ZERO`

:ASETup:EXECute

Function	Executes auto-setup. This command is an overlap command.
Syntax	:ASETup:EXECute
Example	:ASETUP:EXECUTE
Description	The same function can be performed using "ASETup."

:ASETup:TARGet

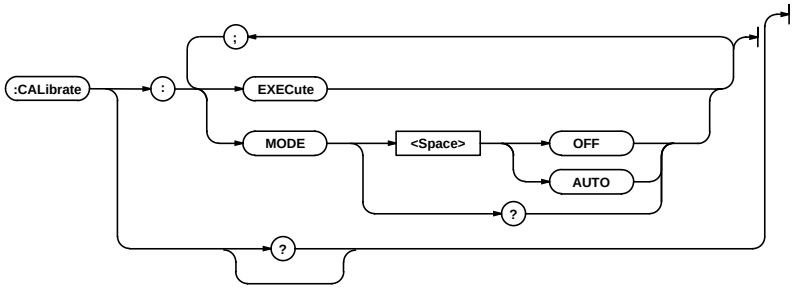
Function	Sets/queries the target channel for auto-setup.
Syntax	:ASETup:TARGet {ALL|<NRf>} :ASETup:TARGet? <NRf>=1 to 16
Example	:ASETUP:TARGET ALL :ASETUP:TARGET?→:ASETUP:TARGET ALL
Description	An error will occur, if a channel which does not have a module installed is specified.

:ASETup:UNDO

Function	Cancels auto set-up settings.
Syntax	:ASETup:UNDO
Example	:ASETUP:UNDO
Description	The same function can be performed using "ASETup UNDO."

4.5 CALibrate Group

The commands in the CALibrate group are used for setting and querying the calibration. This allows you to make the same settings that you can make using the SHIFT+SETUP key on the front panel.



:CALibrate?

Function	Queries all calibration settings.
Syntax	:CALibrate?
Example	:CALIBRATE?→:CALIBRATE:MODE AUTO

:CALibrate[:EXECute]

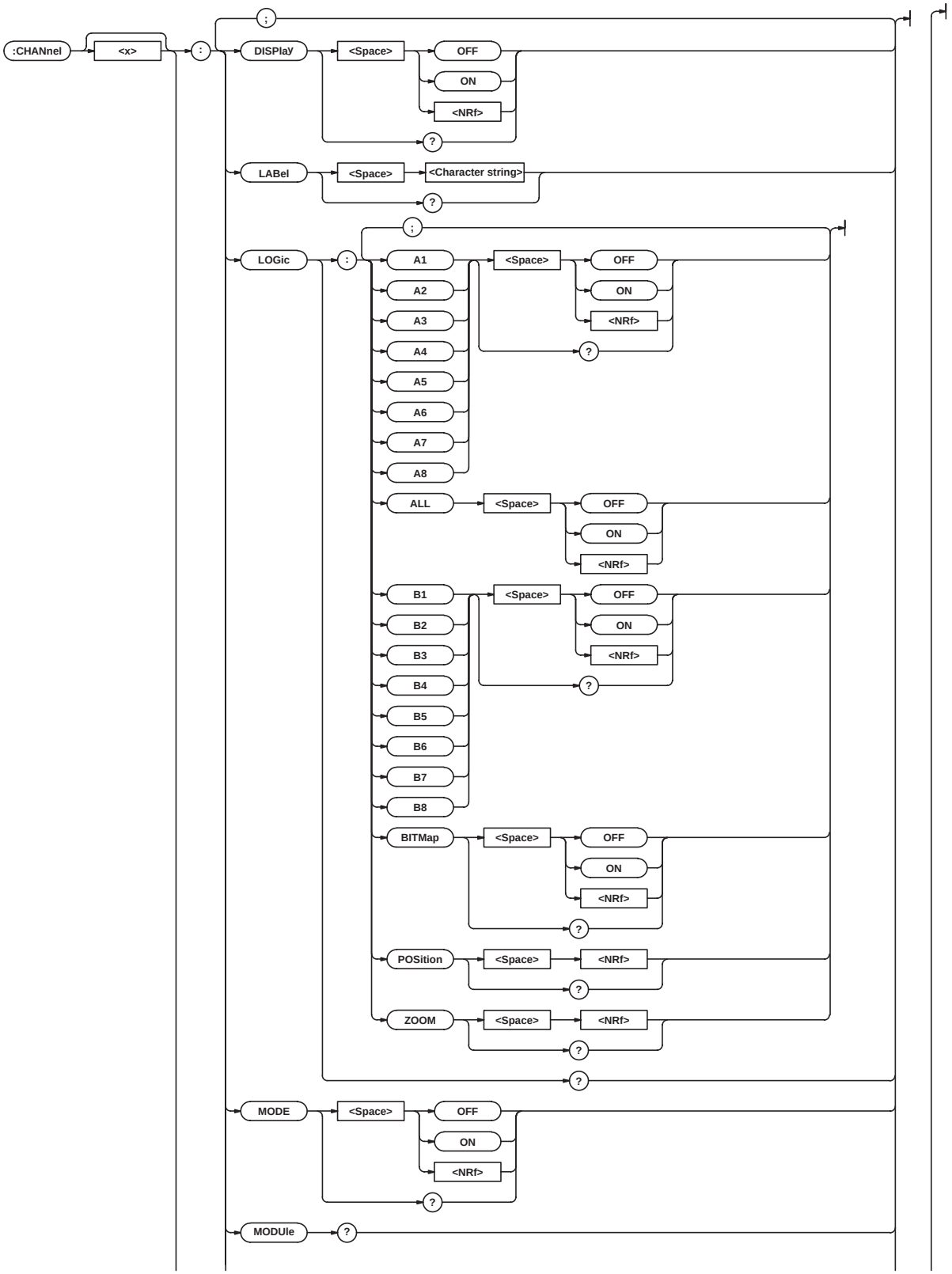
Function	Executes calibration. This is an overlap command.
Syntax	:CALibrate[:EXECute]
Example	:CALIBRATE:EXECUTE

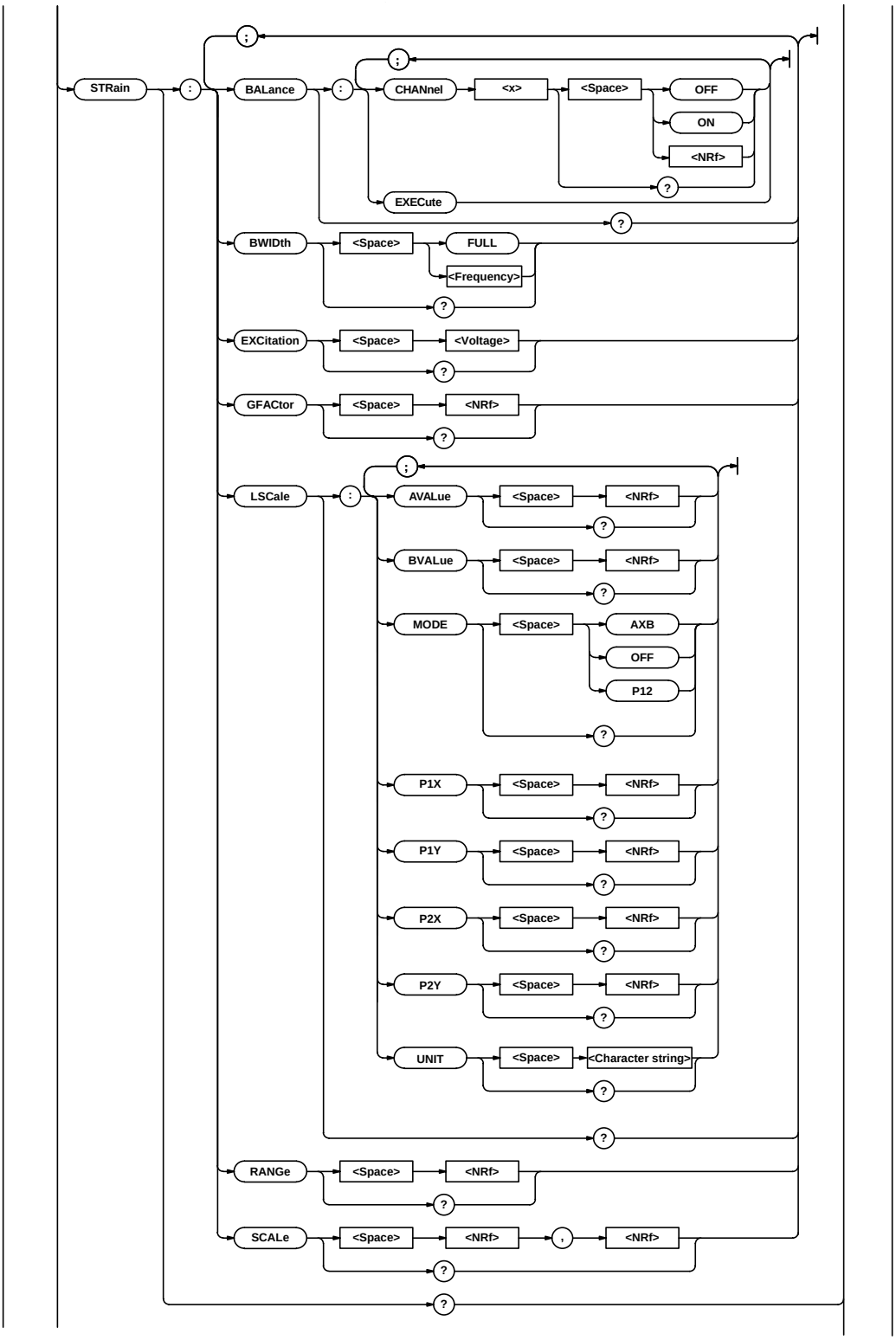
:CALibrate:MODE

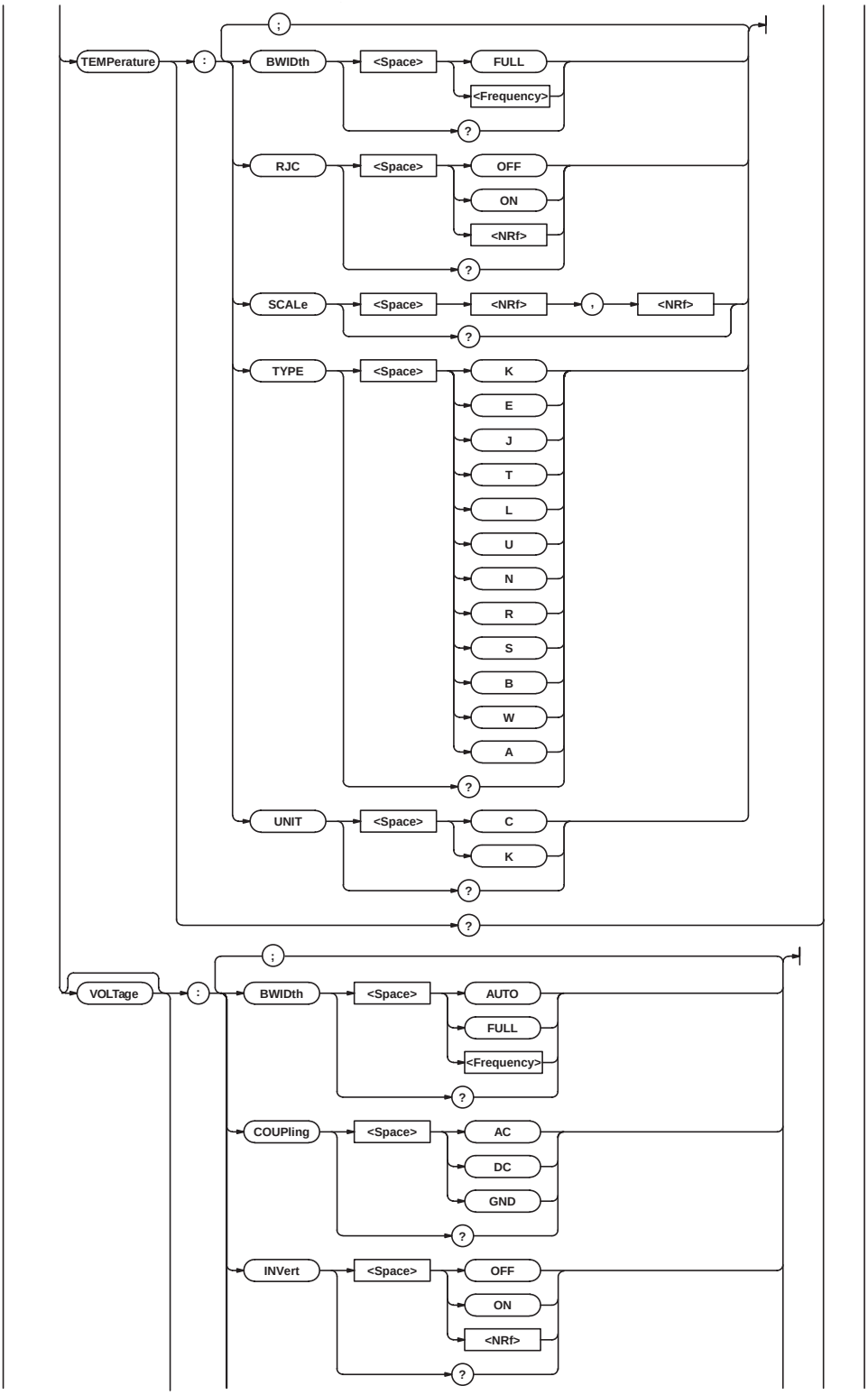
Function	Sets/queries the ON/OFF of the auto calibration.
Syntax	:CALibrate:MODE {AUTO|OFF} :CALibrate?
Example	:CALIBRATE:MODE AUTO :CALIBRATE?→:CALIBRATE:MODE AUTO

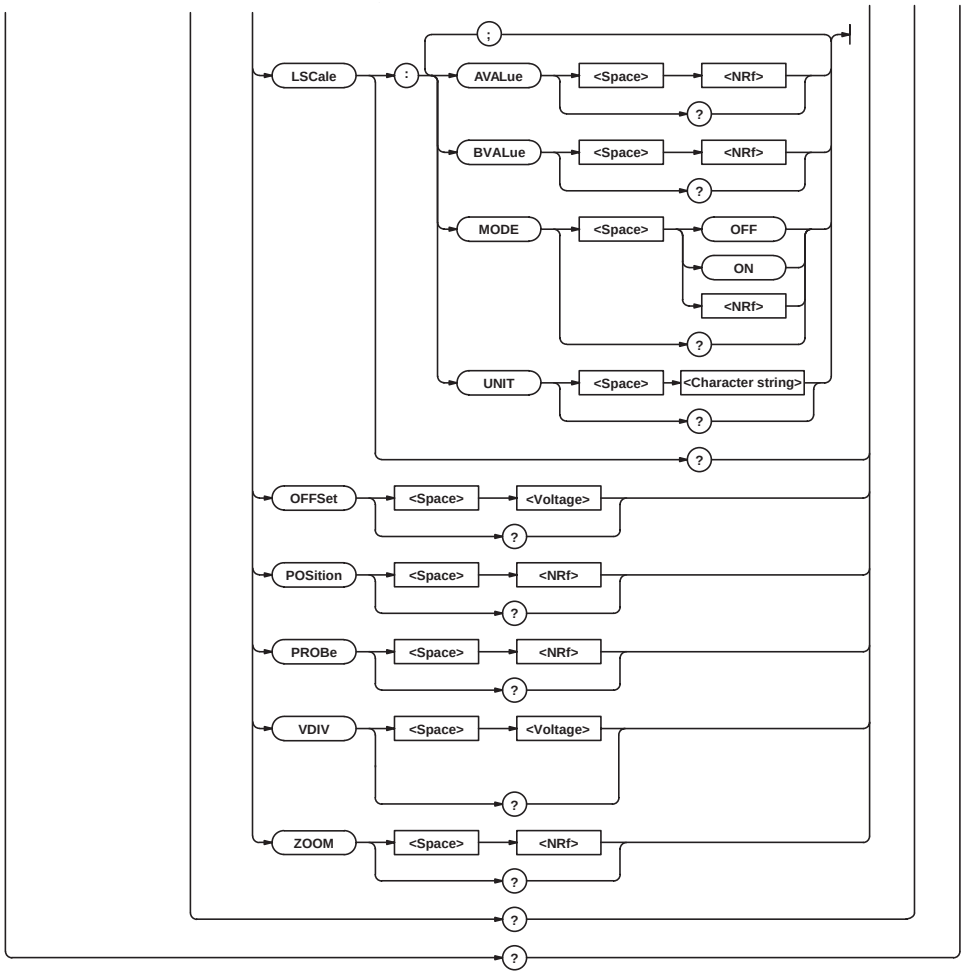
4.6 CHANnel Group

The commands in the CHANnel group are used to make settings and queries about the vertical axis of the specified channel. This allows you to make the same settings that you can make using the VERTICAL keys (CH key, V/DIV key).




STRain
:
;
BALance
:
;
CHANnel
<x>
<Space>
OFF
ON
<NRf>
?
EXECute
?
BWIDth
<Space>
FULL
<Frequency>
?
EXCitation
<Space>
<Voltage>
?
GFACtor
<Space>
<NRf>
?
LSCale
:
;
AVALue
<Space>
<NRf>
?
BVALue
<Space>
<NRf>
?
MODE
<Space>
AXB
OFF
P12
?
P1X
<Space>
<NRf>
?
P1Y
<Space>
<NRf>
?
P2X
<Space>
<NRf>
?
P2Y
<Space>
<NRf>
?
UNIT
<Space>
<Character string>
?
?
RANGe
<Space>
<NRf>
?
SCALe
<Space>
<NRf>
,
<NRf>
?
?


TEMPerature
:
;
BWIDth
<Space>
FULL
<Frequency>
?
RJC
<Space>
OFF
ON
<NRf>
?
SCALe
<Space>
<NRf>
,
<NRf>
?
TYPE
<Space>
K
E
J
T
L
U
N
R
S
B
W
A
?
UNIT
<Space>
C
K
?
?
VOLTage
:
;
BWIDth
<Space>
AUTO
FULL
<Frequency>
?
COUPling
<Space>
AC
DC
GND
?
INVert
<Space>
OFF
ON
<NRf>
?



:CHANnel<x>?

Function Queries all vertical axis settings for the specified channel.

Syntax
```
:CHANnel<x>?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:CHANNEL?→:CHANNEL1:DISPLAY ON;
LABEL "CH1    ";VOLTAGE:COUPLING DC;
POSITION 0.00;PROBE 10;VDIV 50.0E+00;
BWIDTH FULL;INVERT 0;OFFSET 0.0E+00;
LSCALE:MODE 0
```

Description An error will occur, if there is no module installed at the channel (slot) or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:DISPlay

Function Sets/queries display ON/OFF for the specified channel.

Syntax
```
:CHANnel<x>:DISPlay {<Boolean>}
:CHANnel<x>:DISPlay?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:CHANNEL1:DISPLAY ON
:CHANNEL1:DISPLAY?→:CHANNEL1:DISPLAY 1
```

Description
- The same function can be performed using "CHANnel<x>:MODE."
- An error will occur, if there is no module installed at the channel (slot) or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:LABel

Function Sets/queries the channel's waveform label.

Syntax
```
:CHANnel<x>:LABel {<Character string>}
:CHANnel<x>:LABel?
<NRf>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
<Character string>=up to 8 characters
```

Example
```
:CHANNEL1:LABEL "TRACE1"
:CHANNEL1:LABEL?→:CHANNEL1:
LABEL "TRACE1"
```

Description
- Only characters and symbols on the displayed keyboard are available for use in the label.
- ASCII codes for "Ω" and "μ" are 1EH and 1FH, respectively.
- An error will occur, if there is no module installed at the channel (slot) or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:LOGic?

Function Queries all setting values when the logic input module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:LOGic?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:CHANNEL1:LOGIC?→:CHANNEL1:LOGIC:A1 1;
A2 1;A3 1;A4 1;A5 1;A6 1;A7 1;A8 1;B1 1;
B2 1;B3 1;B4 1;B5 1;B6 1;B7 1;B8 1;
BITMAP 1;POSITION 0.00;ZOOM 1.00
```

Description An error will occur, if the logic input module is not installed or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:LOGic:{A1|A2|A3|A4|A5|A6|A7|A8|B1|B2|B3|B4|B5|B6|B7|B8}

Function Sets/queries the ON/OFF condition of each bit, when the logic input module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:LOGic:{A1|A2|A3|A4|A5|A6|A7|
A8|B1|B2|B3|B4|B5|B6|B7|B8} {<Boolean>}
:CHANnel<x>:LOGic:{A1|A2|A3|A4|A5|A6|A7|
A8|B1|B2|B3|B4|B5|B6|B7|B8}?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:CHANNEL1:LOGIC:A1 ON
:CHANNEL1:LOGIC:A1?→:CHANNEL1:LOGIC:A1 1
```

Description An error will occur, if the logic input module is not installed or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:LOGic:ALL

Function Sets/queries the ON/OFF condition of all of the bits, when the logic input module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:LOGic:ALL {<Boolean>}
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:CHANNEL1:LOGIC:ALL ON
```

Description An error will occur, if the logic input module is not installed or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:LOGic:BITMap

Function Sets/queries whether to position the bit that is turned ON in the space for the bit that is turned OFF, when the logic input module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:LOGic:BITMap {<Boolean>}
:CHANnel<x>:LOGic:BITMap?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:CHANNEL1:LOGIC:BITMAP ON
:CHANNEL1:LOGIC:BITMAP?→:CHANNEL1:LOGIC:
BITMAP 1
```

Description An error will occur, if the logic input module is not installed or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:LOGic:POSition

Function Sets/queries the vertical position, when the logic input module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:LOGic:POSition {<NRf>}
:CHANnel<x>:LOGic:POSition?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
<NRf>=-4.00 to 4.00(div, in steps of
      0.01div)
```

Example
```
:CHANNEL1:LOGIC:POSITION 1.00
:CHANNEL1:LOGIC:POSITION?→:CHANNEL1:
LOGIC:POSITION 1.00
```

Description An error will occur, if the logic input module is not installed or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:LOGic:ZOOM

Function Sets/queries the zoom ratio in the vertical direction, when the logic input module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:LOGic:ZOOM {<NRf>}
:CHANnel<x>:LOGic:ZOOM?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
<NRf>=0.1,0.2,0.5,0.75,1,1.5,2,5,10,20,
      50,100
```

Example
```
:CHANNEL1:LOGIC:ZOOM 10
:CHANNEL1:LOGIC:ZOOM?→:CHANNEL1:LOGIC:
ZOOM? 10.00
```

Description An error will occur, if the logic input module is not installed or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:MODE

Function Sets/queries display ON/OFF for the specified channel.

Syntax
```
:CHANnel<x>:MODE {<Boolean>}
:CHANnel<x>:MODE?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:CHANNEL1:MODE ON
:CHANNEL1:MODE?→:CHANNEL1:MODE 1
```

Description
- The same function can be performed using "CHANnel<x>:DISPlay."
- An error will occur, if there is no module installed at the channel (slot) or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:CHANnel<x>:MODUle?

Function Queries the modules installed at each channel (slot).

Syntax
```
:CHANnel<x>:MODUle?
<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:CHANNEL1:MODULE?→:CHANNEL1:
MODULE M701856
```

Description The values returned by each module are as follows.

NOMODULE	No module
M701855	High-Speed Isolation Module
M701856	High-Speed Module
M701852	High-Resolution, High-Voltage Isolation Module
M701853	High-Resolution, Isolation Module
M701860	Temperature Module
M701870	Logic Input Module
M701880	Strain Module

:CHANnel<x>:STRain?

Function Queries all settings when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain?
<x>=1 to 16
```

Example
```
:CHANNEL1:STRAIN?→:CHANNEL1:STRAIN:
RANGE 20000;SCALE 11000,-2000;
EXCITATION 2.00E+00;GFACTOR 2.00;
BWIDTH FULL;BALANCE:CHANNEL1 0;
CHANNEL2 0;CHANNEL3 0;CHANNEL4 0;
CHANNEL5 0;CHANNEL6 0;CHANNEL7 0;
CHANNEL8 0;:CHANNEL1:STRAIN:LSCALE:
MODE OFF
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:BALance?

Function Queries all settings relating to the balancing when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:BALance?
<x>=1 to 16
```

Example
```
:CHANNEL1:STRAIN:BALANCE?→:CHANNEL1:
STRAIN:BALANCE:CHANNEL1 0;CHANNEL2 0;
CHANNEL3 0;CHANNEL4 0;CHANNEL5 0;
CHANNEL6 0;CHANNEL7 0;CHANNEL8 0
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:BALance:EXECute

Function Executes the balancing when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:BALance:EXECute
<x>=1 to 16
```

Example
```
:CHANNEL1:STRAIN:BALANCE:EXECUTE
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:BALance:CHANnel<x>

Function Sets/queries the channel to balance when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:BALance:CHANnel<x>
{<Boolean>}
:CHANnel<x>:STRain:BALance:CHANnel<x>?
<x>=1 to 16
```

Example
```
:CHANNEL1:STRAIN:BALANCE:CHANNEL1 ON
:CHANNEL1:STRAIN:BALANCE:CHANNEL1?
→:CHANNEL1:STRAIN:BALANCE:CHANNEL1 1
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:BWIDth

Function Sets/queries the filter when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:BWIDth
{FULL|<Frequency>}
:CHANnel<x>:STRain:BWIDth?
<x>=1 to 16
<Frequency>=1kHz, 100Hz, 10Hz
```

Example
```
:CHANNEL1:STRAIN:BWIDTH FULL
:CHANNEL1:STRAIN:BWIDTH?→:CHANNEL1:
STRAIN:BWIDTH FULL
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:EXCitation

Function Sets/queries the excitation when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:EXCitation {<Voltage>}
:CHANnel<x>:STRain:EXCitation?
<x>=1 to 16
<Voltage>=2V, 5V
```

Example
```
:CHANNEL1:STRAIN:EXCITATION 2V
:CHANNEL1:STRAIN:EXCITATION?→:CHANNEL1:
STRAIN:EXCITATION 2.00E+00
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:GFACtor

Function Sets/queries the gauge factor when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:GFACtor {<NRf>}
:CHANnel<x>:STRain:GFACtor?
<x>=1 to 16
<NRf>=1.90 to 2.20
```

Example
```
:CHANNEL1:STRAIN:GFACTOR 2.00
:CHANNEL1:STRAIN:GFACTOR?→:CHANNEL1:
STRAIN:GFACTOR 2.00
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:LSCale?

Function Queries all settings relating to the linear scaling when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale?
<x>=1 to 16
```

Example
```
:CHANNEL1:STRAIN:LSCALE?→:CHANNEL1:
STRAIN:LSCALE:MODE 1;AVALUE 1.0000E+00;
BVALUE 0.0000E+00;UNIT "A   "
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:LSCale:AVALue

Function Sets/queries the scaling coefficient A of the linear scaling when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:AVALue {<NRf>}
:CHANnel<x>:STRain:LSCale:AVALue?
<x>=1 to 16
<NRf>=-1E+30 to 1E+30
```

Example
```
:CHANNEL1:STRAIN:LSCALE:AVALUE 10
:CHANNEL1:STRAIN:LSCALE:AVALUE?
→:CHANNEL1:STRAIN:LSCALE:
AVALUE 10.000E+00
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:LSCale:BVALue

Function Sets/queries the offset value B of the linear scaling when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:BVALue {<NRf>}
:CHANnel<x>:STRain:LSCale:BVALue?
<x>=1 to 16
<NRf>=-1E+30 to 1E+30
```

Example
```
:CHANNEL1:STRAIN:LSCALE:BVALUE 10
:CHANNEL1:STRAIN:LSCALE:BVALUE?
→:CHANNEL1:STRAIN:LSCALE:
BVALUE 10.000E+00
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:LSCale:MODE

Function Sets/queries the linear scaling mode when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:
MODE {AXB|OFF|P12}
:CHANnel<x>:STRain:LSCale:MODE?
<x>=1 to 16
```

Example
```
:CHANNEL1:STRAIN:LSCALE:MODE AXB
:CHANNEL1:STRAIN:LSCALE:MODE?→:CHANNEL1:
STRAIN:LSCALE:MODE AXB
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:LSCale:{P1X|P1Y|P2X|P2Y}

Function Sets/queries the P1:X|P1:Y|P2:X|P2:Y values of the linear scaling when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:
{P1X|P1Y|P2X|P2Y} {<NRf>}
:CHANnel<x>:STRain:LSCale:
{P1X|P1Y|P2X|P2Y}?
<x>=1 to 16
<NRf>=-1E+30 to 1E+30
```

Example
```
:CHANNEL1:STRAIN:LSCALE:P1X 10
:CHANNEL1:STRAIN:LSCALE:P1X?→:CHANNEL1:
STRAIN:LSCALE:P1X 10.000E+00
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:LSCale:UNIT

Function Sets/queries the unit to be added to the linear scaling results when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:UNIT
{<Character string>}
:CHANnel<x>:STRain:LSCale:UNIT?
<x>=1 to 16
<Character string>=4 characters or less
```

Example
```
:CHANNEL1:STRAIN:LSCALE:UNIT "X"
:CHANNEL1:STRAIN:LSCALE:UNIT?→:CHANNEL1:
STRAIN:LSCALE:UNIT "X   "
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:RANGe

Function Sets/queries the measurement range when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:RANGe {<NRf>}
:CHANnel<x>:STRain:RANGe?
<x>=1 to 16
<NRf>=1000, 2000, 5000, 10000, 20000
      (×10^-6 strain)
```

Example
```
:CHANNEL1:STRAIN:RANGE 20000
:CHANNEL1:STRAIN:RANGE?→:CHANNEL1:
STRAIN:RANGE 20000
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:STRain:SCALe

Function Sets/queries the upper and lower limits of the display when the strain module is installed into the channel (slot).

Syntax
```
:CHANnel<x>:STRain:SCALe {<NRf>,<NRf>}
:CHANnel<x>:STRain:SCALe?
<x>=1 to 16
<NRf>=-30000 to 30000 (×10^-6 strain)
```

Example
```
:CHANNEL1:STRAIN:SCALE 11000,-2000
:CHANNEL1:STRAIN:SCALE?→:CHANNEL1:
STRAIN:SCALE 11000,-2000
```

Description An error will occur, if the strain module is not installed.

:CHANnel<x>:TEMPerature?

Function Queries all setting values, when the temperature module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature?
<x>=1 to 16
```

Example
```
:CHANNEL1:TEMPERATURE?→:CHANNEL1:
TEMPERATURE:TYPE K;UNIT C;
SCALE 1200.0E+00,-200.0E+00;BWIDTH FULL;
RJC 1
```

Description An error will occur, if the temperature module is not installed.

:CHANnel<x>:TEMPerature:BWIDth

Function Sets/queries the filter, when the temperature module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:BWIDth {FULL|
<Frequency>}
:CHANnel<x>:TEMPerature:BWIDth?
<x>=1 to 16
<Frequency>=2, 8(Hz)
```

Example
```
:CHANNEL1:TEMPERATURE:BWIDTH 2HZ
:CHANNEL1:TEMPERATURE:BWIDTH?→:CHANNEL1:
TEMPERATURE:BWIDTH 2E+00
```

Description An error will occur, if the temperature module is not installed.

:CHANnel<x>:TEMPerature:RJC

Function Sets/queries the RJC, when the temperature module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:RJC {<Boolean>}
:CHANnel<x>:TEMPerature:RJC?
<x>=1 to 16
```

Example
```
:CHANNEL1:TEMPERATURE:RJC ON
:CHANNEL1:TEMPERATURE:RJC?→:CHANNEL1:
TEMPERATURE:RJC 1
```

Description An error will occur, if the temperature module is not installed.

:CHANnel<x>:TEMPerature:SCALe

Function Sets/queries the upper and lower limits on the screen, when the temperature module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:
SCALe {<NRf>,<NRf>}
:CHANnel<x>:TEMPerature:SCALe?
<x>=1 to 16
<NRf>=-3000 to 3000 (if in °C, in 0.1°C
      steps)
```

Example
```
:CHANNEL1:TEMPERATURE:SCALE 1200.0,-200.0
:CHANNEL1:TEMPERATURE:SCALE?→:CHANNEL1:
TEMPERATURE:SCALE 1200.0E+00,-200.0E+00
```

Description
- The unit for the setting value is the unit set with ":CHANnel<x>:TEMPerature:UNIT."
- An error will occur, if the temperature module is not installed.

:CHANnel<x>:TEMPerature:TYPE

Function Sets/queries the type of thermocouples to use, when the temperature module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:TYPE {K|E|J|T|L|
U|N|R|S|B|W|A}
:CHANnel<x>:TEMPerature:TYPE?
<x>=1 to 16
```

Example
```
:CHANNEL1:TEMPERATURE:TYPE K
:CHANNEL1:TEMPERATURE:TYPE?→:CHANNEL1:
TEMPERATURE:TYPE K
```

Description An error will occur, if the temperature module is not installed.

:CHANnel<x>:TEMPerature:UNIT

Function Sets/queries the units for the upper and lower limits, when the temperature module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:UNIT {C|K}
:CHANnel<x>:TEMPerature:UNIT?
<x>=1 to 16
```

Example
```
:CHANNEL1:TEMPERATURE:UNIT C
:CHANNEL1:TEMPERATURE:UNIT?→:CHANNEL1:
TEMPERATURE:UNIT C
```

Description An error will occur, if the temperature module is not installed.

:CHANnel<x>:VOLTage?

Function Sets/queries all setting values, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>:VOLTage?
<x>=1 to 16
```

Example
```
:CHANNEL1:VOLTAGE?→:CHANNEL1:VOLTAGE:
COUPLING DC;POSITION 0.00;PROBE 10;
VDIV 50.0E+00;BWIDTH FULL;INVERT 0;
OFFSET 0.0E+00;LSCALE:MODE 0
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:BWIDth

Function Sets/queries the filter, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:BWIDth {AUTO|FULL|
<Frequency>}
:CHANnel<x>[:VOLTage]:BWIDth?
<x>=1 to 16
<Frequency>
=5MHz,500kHz(M701851)
 5MHz,500kHz,50kHz,5kHz,500Hz(M701850)
 4kHz,400Hz,40Hz(M701852,M701853)
```

Example
```
:CHANNEL1:VOLTAGE:BWIDTH AUTO
:CHANNEL1:VOLTAGE:BWIDTH?→:CHANNEL1:
VOLTAGE:BWIDTH AUTO
```

Description
- An error will occur, if the voltage module is not installed.
- "AUTO" is not available on M701855/M701856.

:CHANnel<x>[:VOLTage]:COUPling

Function Sets/queries the input coupling, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:COUPling {AC|DC|
GND}
:CHANnel<x>[:VOLTage]:COUPling?
<x>=1 to 16
```

Example
```
:CHANNEL1:VOLTAGE:COUPlING DC
:CHANNEL1:VOLTAGE:COUPlING?→:CHANNEL1:
VOLTAGE:COUPlING DC
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:INVert

Function Sets/queries how the waveform is to be displayed, inverted (ON) or not inverted (OFF), when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:INVert {<Boolean>}
:CHANnel<x>[:VOLTage]:INVert?
<x>=1 to 16
```

Example
```
:CHANNEL1:VOLTAGE:INVERT ON
:CHANNEL1:VOLTAGE:INVERT?→:CHANNEL1:
VOLTAGE:INVERT 1
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale?

Function Queries all linear scaling settings, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale?
<x>=1 to 16
```

Example
```
:CHANNEL1:VOLTAGE:LSCALE?→:CHANNEL1:
VOLTAGE:LSCALE:MODE 1;AVALUE 1.0000E+00;
BVALUE 0.0000E+00;UNIT "A   "
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:AVALue

Function Sets/queries the constant (coefficient) A of linear scaling when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
AVALue {<NRf>}
:CHANnel<x>[:VOLTage]:LSCale:AVALue?
<x>=1 to 16
<NRf>=-1E+30 to 1E+30
```

Example
```
:CHANNEL1:VOLTAGE:LSCALE:AVALUE 10
:CHANNEL1:VOLTAGE:LSCALE:AVALUE?
→:CHANNEL1:VOLTAGE:LSCALE:
AVALUE 10.000E+00
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:BVALue

Function Sets/queries the offset value B of linear scaling when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
BVALue {<NRf>}
:CHANnel<x>[:VOLTage]:LSCale:BVALue?
<x>=1 to 16
<NRf>=-1E+30 to 1E+30
```

Example
```
:CHANNEL1:VOLTAGE:LSCALE:BVALUE 10
:CHANNEL1:VOLTAGE:LSCALE:BVALUE?
→:CHANNEL1:VOLTAGE:LSCALE:
BVALUE 10.000E+00
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:MODE

Function Sets/queries the ON/OFF condition of linear scaling, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
MODE {<Boolean>}
:CHANnel<x>[:VOLTage]:LSCale:MODE?
<x>=1 to 16
```

Example
```
:CHANNEL1:VOLTAGE:LSCALE:MODE ON
:CHANNEL1:VOLTAGE:LSCALE:MODE?
→:CHANNEL1:VOLTAGE:LSCALE:MODE 1
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:UNIT

Function Sets/queries the dimensional unit which is appended to the linear scaling result, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
UNIT {<Character string>}
:CHANnel<x>[:VOLTage]:LSCale:UNIT?
<x>=1 to 16
<Character string>=up to 4 characters
```

Example
```
:CHANNEL1:VOLTAGE:LSCALE:UNIT "RPM"
:CHANNEL1:VOLTAGE:LSCALE:UNIT?
→:CHANNEL1:VOLTAGE:LSCALE:UNIT "RPM"
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:OFFSet

Function Sets/queries the offset voltage, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:OFFSet {<Voltage>}
:CHANnel<x>[:VOLTage]:OFFSet?
<x>=1 to 16
<Voltage>=Refer to User's Manual
          IM701830-01E.
```

Example
```
:CHANNEL1:VOLTAGE:OFFSET 5V
:CHANNEL1:VOLTAGE:OFFSET?→:CHANNEL1:
VOLTAGE:OFFSET 5.000E+00
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:POSition

Function Sets/queries the vertical position, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:POSition {<NRf>}
:CHANnel<x>[:VOLTage]:POSition?
<x>=1 to 16
<NRf>=-4.00 to 4.00div (in 0.01div steps)
```

Example
```
:CHANNEL1:VOLTAGE:POSITION 1.00
:CHANNEL1:VOLTAGE:POSITION?→:CHANNEL1:V
OLTAGE:POSITION 1.00
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:PROBe

Function Sets/queries the probe attenuation, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:PROBe {<NRf>}
:CHANnel<x>[:VOLTage]:PROBe?
<x>=1 to 16
<NRf>=1,10,100,1000
```

Example
```
:CHANNEL1:VOLTAGE:PROBE 10
:CHANNEL1:VOLTAGE:PROBE?→:CHANNEL1:
VOLTAGE:PROBE 10
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:VDIV

Function Sets/queries the V/div setting, when the voltage module is installed at the channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:VDIV {<Voltage>}
:CHANnel<x>[:VOLTage]:VDIV?
<x>=1 to 16
<Voltage>=5mV to 20V
(M701855 and M701856, probe attenuation
1:1)
50mV to 200V
(M701852, probe attenuation 1:1)
5mV to 20V
(M701853, probe attenuation 1:1)
```

Example
```
:CHANNEL1:VOLTAGE:VDIV 5V
:CHANNEL1:VOLTAGE:VDIV?→:CHANNEL1:
VOLTAGE:VDIV 5.000E+00
```

Description An error will occur, if the voltage module is not installed.

:CHANnel<x>[:VOLTage]:ZOOM

Function Sets/queries the zoom ratio in the vertical direction, when the voltage module is installed at the channel (slot).

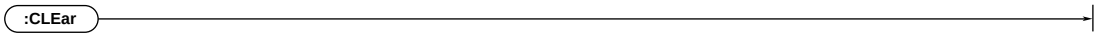
Syntax
```
:CHANnel<x>[:VOLTage]:ZOOM {<NRf>}
:CHANnel<x>[:VOLTage]:ZOOM?
<x>=1 to 16
<NRf>=0.1,0.2,0.5,0.75,1,1.5,2,5,10,20,
      50,100
```

Example
```
:CHANNEL1:VOLTAGE:ZOOM 10
:CHANNEL1:VOLTAGE:ZOOM?→:CHANNEL1:
VOLTAGE:ZOOM 10
```

Description An error will occur, if the voltage module is not installed.

4.7 CLEar Group

The CLEar command is used to clear the trace. This allows you to perform the same operation that you can perform using the CLEAR key on the front panel.

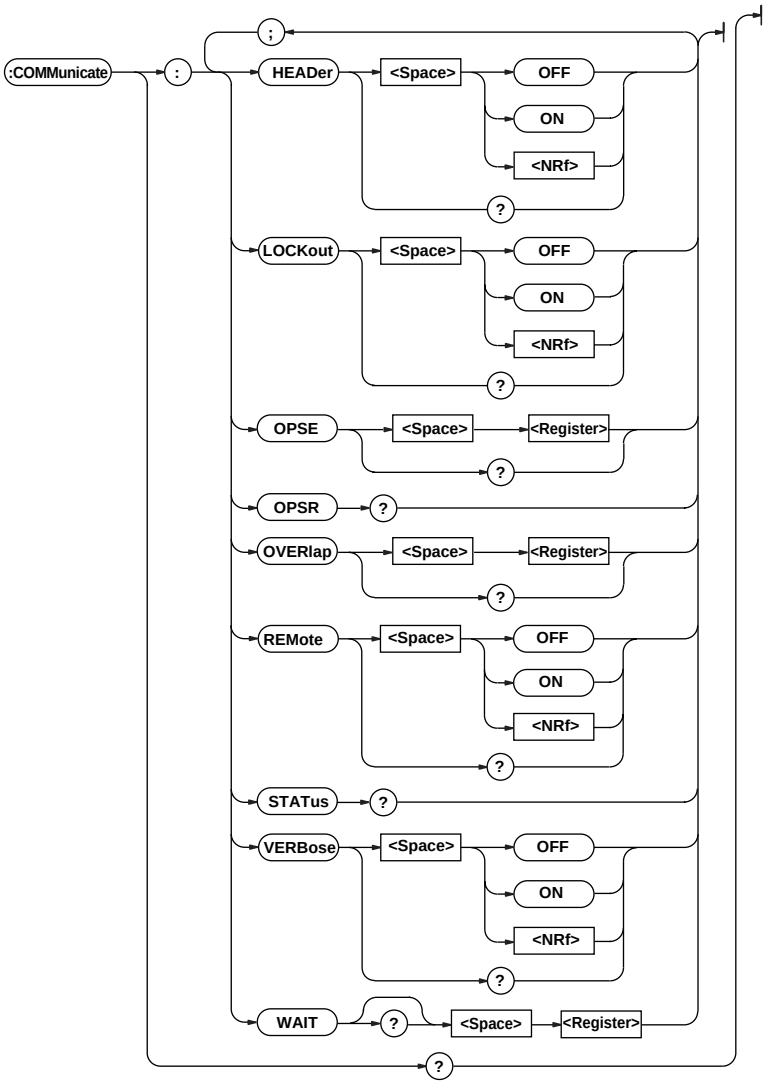


:CLEar

Function	Clears trace.
Syntax	`:CLEar`
Example	`:CLEAR`
Description	• The same function can be performed using "`DISPlay:CLEar`."
	• Use "`SNAP`" or "`DISPlay:SNAP`" for a snap shot.

4.8 COMMunicate Group

The commands in the COMMunicate group are used to make settings and queries about communications. There is no front panel key with this function.



:COMMunicate?

Function Queries all communication settings.
Syntax :COMMunicate?
Example :COMMUNICATE?→:COMMUNICATE:HEADER 1;OPSE 96;OVERLAP 96;VERBOSE 1

:COMMunicate:HEADer

Function Determines whether a header is to be added (for example: CHANNEL1:VOLTAGE:PROBE 10) or not (example: 10) when sending a response to a query; or queries the current setting.
Syntax :COMMunicate:HEADer {<Boolean>}
:COMMunicate:HEADer?
Example :COMMUNICATE:HEADER ON
:COMMUNICATE:HEADER?→:COMMUNICATE:HEADER 1

:COMMunicate:LOCKout

Function Sets/releases the local lockout.
Syntax :COMMunicate:LOCKout {<Boolean>}
:COMMunicate:LOCKout?
Example :COMMUNICATE:LOCKOUT ON
:COMMUNICATE:LOCKOUT?→:COMMUNICATE:LOCKOUT 1
Description This is an exclusive command for the RS-232 interface.

:COMMunicate:OPSE

(Operation Pending Status Enable register)

Function Sets the overlap command to be used with *OPC, *OPC?, and *WAI, or queries the current setting.
Syntax :COMMunicate:OPSE <Register>
:COMMunicate:OPSE?
<Register>=0 to 65535, refer to the figure on the next page.
Example :COMMUNICATE:OPSE 65535
:COMMUNICATE:OPSE?→:COMMUNICATE:OPSE 96
Description In the above example, all bits are masked to "1" so that all overlap commands can be used by this command. However, bits that are fixed as "0" cannot be set to "1," so only bits 5 and 6 are actually set to "1" and appears as "1" when a query is made.

:COMMunicate:OPSR?

(Operation Pending Status Register)

Function Inquires about the value in the operation pending status register.
Syntax :COMMunicate:OPSR?
Example :COMMUNICATE:OPSR?→0
Description For a description of the operation pending status register, refer to the figure on the next page.

:COMMunicate:OVERlap

Function Selects/queries the commands enabled for overlap.
Syntax :COMMunicate:OVERlap <Register>
:COMMunicate:OVERlap?
<Register>=0 to 65535, refer to the figure on the next page.
Example :COMMUNICATE:OVERLAP 65535
:COMMUNICATE:OVERLAP?→:COMMUNICATE:OVERLAP 96
Description
- In the above example, all bits are set to "1" so that all overlap commands can be used by this command. However, bits that are fixed as "0" cannot be set to "1," so only bits 5 and 6 are actually set to "1" and appears as "1" when a query is made.
- COMMunicate: For the synchronization method using "OVERlap," refer to page 3-7.
- In the above example, bits 5 and 6 are set to "1" so that all overlap commands can be used by this command. (Refer to the figure on the next page.)

:COMMunicate:REMote

Function Sets remote/local. It is in remote mode when it is set to ON.
Syntax :COMMunicate:REMote {<Boolean>}
:COMMunicate:REMote?
Example :COMMUNICATE:REMOTE ON
:COMMUNICATE:REMOTE?→:COMMUNICATE:REMOTE 1
Description This is an exclusive command for the RS-232 interface.

:COMMunicate:STATus?

Function Queries the circuit status.
Syntax :COMMunicate:STATus?
Example :COMMUNICATE:STATUS?→:COMMUNICATE:STATUS 0
Description Status-bit meanings are as follows.

Bit	GP-IB	RS-232
0	Unrecoverable transmission error	Parity error
1	Always 0	Framing error
2	Always 0	Break character detected
3 to	Always 0	Always 0

Status bit sets when cause occurs, and clears when read.

:COMMunicate:VERBose

Function Determines whether a response to a query is to be returned in full form (for example: CHANNEL1:VOLTAGE:PROBE 10) or in abbreviated form (for example: CHAN:PROB 10), or queries the current setting.
Syntax :COMMunicate:VERBose {<Boolean>}
:COMMunicate:VERBose?
Example :COMMUNICATE:VERBOSE ON
:COMMUNICATE:VERBOSE?→:COMMUNICATE:VERBOSE 1

:COMMunicate:WAIT

Function Waits until one of the specified extended events occurs.

Syntax `:COMMunicate:WAIT <Register>`
`<Register>=0 to 65535 (Extended event register; refer to page 5-4.)`

Example `:COMMUNICATE:WAIT 65535`

Description COMMunicate: For a description of the synchronizing method using "WAIT," refer to page 3-8.

:COMMunicate:WAIT?

Function Generates a response when one of the specified extended events occurs.

Syntax `:COMMunicate:WAIT? <Register>`
`<Register>=0 to 65535 (Extended event register; refer to page 5-4.)`

Example `:COMMUNICATE:WAIT? 65535→1`

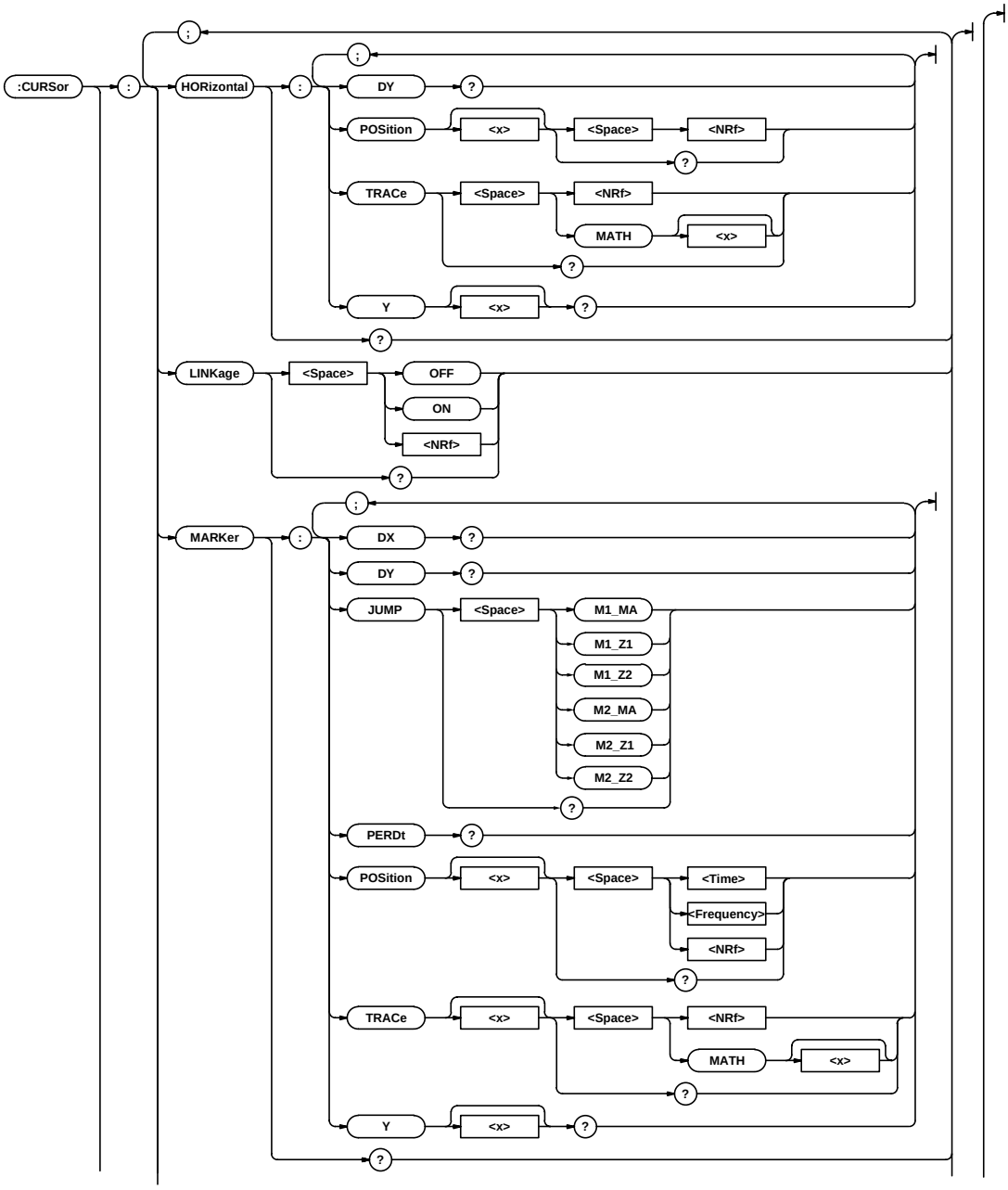
Operation pending status register/overlap enable register

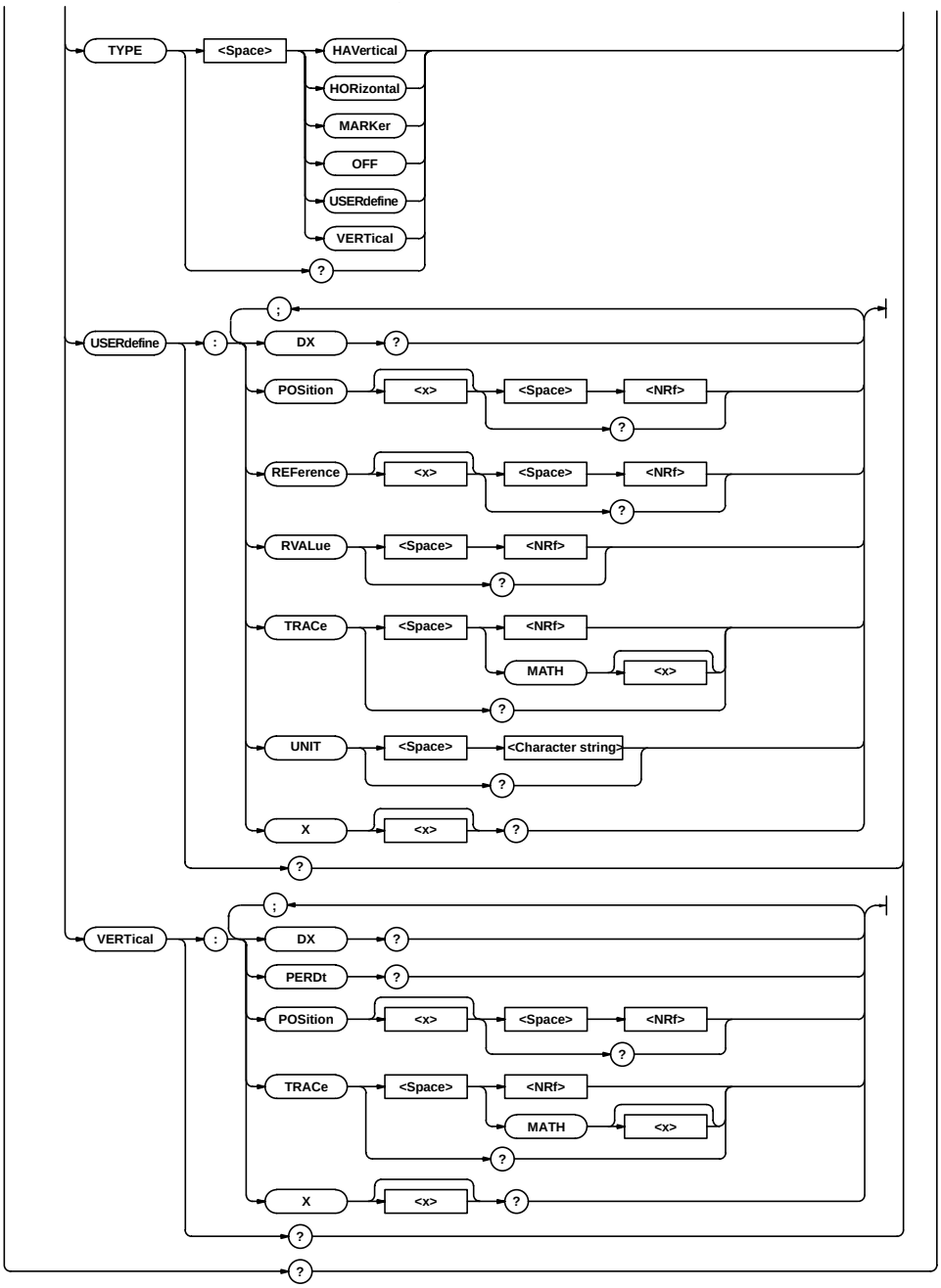
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	ø	0	0	ACS	PRN	0	ø	0	0	0

When bit 5 (PRN) = 1:
Printer operation not completed
When bit 6 (ACS) = 1:
Medium not accessed

4.9 CURSor Group

The commands in the CURSor group are used to make cursor measurement settings and queries. This allows you to make the same settings that you can make using the CURSOR key on the front panel.




TYPE
<Space>
HAVertical
HORizontal
MARKer
OFF
USERdefine
VERTical
?
;
USERdefine
:
DX
?
POSition
<x>
<Space>
<NRf>
?
REFerence
<x>
<Space>
<NRf>
?
RVALue
<Space>
<NRf>
?
TRACe
<Space>
<NRf>
MATH
<x>
?
UNIT
<Space>
<Character string>
?
X
<x>
?
?
;
VERTical
:
DX
?
PERDt
?
POSition
<x>
<Space>
<NRf>
?
TRACe
<Space>
<NRf>
MATH
<x>
?
X
<x>
?
?
?

:CURSor?

Function Queries all cursor measurement settings.
Syntax :CURSor?
Example :CURSOR?→:CURSOR:TYPE HORIZONTAL;
LINKAGE 0;HORIZONTAL:TRACE 1;
POSITION1 3.000E+00;POSITION2 -3.000E+00

:CURSor:HORizontal?

Function Queries all H cursor settings.
Syntax :CURSor:HORizontal?
Example :CURSOR:HORIZONTAL?→:CURSOR:HORIZONTAL:
TRACE 1;POSITION1 3.000E+00;
POSITION2 -3.000E+00

:CURSor:HORizontal:DY?

Function Queries the Y-axis distance between the H cursors.
Syntax :CURSor:HORizontal:DY?
Example :CURSOR:HORIZONTAL:DY?→4.50E-01
Description If linear scaling is ON, this message queries the scaling value.

:CURSor:HORizontal:POSition<x>

Function Sets/queries the H cursor position(Y-axis).
Syntax :CURSor:HORizontal:POSition<x> {<NRf>}
:CURSor:HORizontal:POSition<x>?
<x>=1, 2
<NRf>=-4 to 4div (in 1/48 V/div steps)
Example :CURSOR:HORIZONTAL:POSITION1 2
:CURSOR:HORIZONTAL:POSITION1?→:CURSOR:
HORIZONTAL:POSITION1 2.00E+00
Description <NRf> value is effective to second decimal place.

:CURSor:HORizontal:TRACe

Function Sets the waveform for which the H cursors are used, or queries the current setting.
Syntax :CURSor:HORizontal:TRACe {<NRf>|
MATH<x>}
:CURSor:HORizontal:TRACe?
<NRf>=1 to 18 (However, 17 and 18 correspond to LOGIC1 and LOGIC2 of the optional extended logic input.)
<x>=1, 2
Example :CURSOR:HORIZONTAL:TRACE 1
:CURSOR:HORIZONTAL:TRACE?→:CURSOR:
HORIZONTAL:TRACE 1

:CURSor:HORizontal:Y<x>?

Function Queries the Y-axis value of the H cursor.
Syntax :CURSor:HORizontal:Y<x>?
<x>=1, 2
Example :CURSOR:HORIZONTAL:Y1?→:CURSOR:
HORIZONTAL:Y1 -4.50E-03
Description If linear scaling is ON, this message queries the scaling value.

:CURSor:LINKage

Function Sets/queries cursor linkage ON/OFF.
Syntax :CURSor:LINKage {<Boolean>}
:CURSor:LINKage?
Example :CURSOR:LINKAGE OFF
:CURSOR:LINKAGE?→:CURSOR:LINKAGE 0

:CURSor:MARKer?

Function Queries all marker settings.
Syntax :CURSor:MARKer?
Example :CURSOR:MARKER?→:CURSOR:MARKER:
TRACE1 1;TRACE2 1;POSITION1 -8.00E-03;
POSITION2 8.00E-03

:CURSor:MARKer:DX?

Function Queries the X-axis distance between the markers.
Syntax :CURSor:MARKer:DX?
Example :CURSOR:MARKER:DX?→:CURSOR:MARKER:
DX 2.50E-06
Description • If time domain, the command queries the time between markers.
• If frequency domain, the command queries the frequency between markers.

:CURSor:MARKer:DY?

Function Queries the Y-axis distance between the markers.
Syntax :CURSor:MARKer:DY?
Example :CURSOR:MARKER:DY?→:CURSOR:MARKER:
DY -4.5000E-03
Description • If "CURSor:TYPe" is not "MARKer," the command returns "NAN" ("Not A Number").
• If linear scaling is ON, this message queries the scaling value.

:CURSor:MARKer:JUMP

Function Jumps marker to zoomed waveform, or queries the jump status.
Syntax :CURSor:MARKer:JUMP {M1_MA|M1_Z1|M1_Z2|
M2_MA|M2_Z1|M2_Z2}
Example :CURSOR:MARKER:JUMP M1_Z1
:CURSOR:MARKER:JUMP?→:CURSOR:MARKER:
JUMP M1_Z1
Description • M1 and M2 represent marker cursors 1 and 2, respectively.
• "M_Z" jumps to the center position of the zoomed waveform.
• "M_MA" jumps to the center position of the main waveform.

:CURSor:MARKer:PERDt? (1 PER Delta T)

Function Queries the 1/ΔT value between the markers.
Syntax :CURSor:MARKer:PERDt?
Example :CURSOR:MARKER:PERDT?→:CURSOR:MARKER:
PERDT 2.50E+06
Description This query is meaningless if horizontal axis is not a time domain.

:CURSor:MARKer:POSition<x>

Function Sets/queries the X-axis value of the marker.
Syntax :CURSor:MARKer:POSition<x> {<Time>|<Frequency>|<NRf>}
:CURSor:MARKer:POSition<x>?
<x>=1, 2
<Time>, <Frequency>, and <NRf> value cannot exceed 10 screen divisions.
Example :CURSOR:MARKER:POSITION1 2.5US
:CURSOR:MARKER:POSITION1?→:CURSOR:MARKER:POSITION1 2.50E-06
Description If time domain, the command sets/queries the time. If frequency domain, the command sets/queries the frequency.

:CURSor:MARKer:TRACe<x>

Function Sets the waveform for which the markers are to be used, or queries the current setting.
Syntax :CURSor:MARKer:TRACe<x> {<NRf>|MATH<x>}
:CURSor:MARKer:TRACe<x>?
<NRf>=1 to 18 (However, 17 and 18 correspond to LOGIC1 and LOGIC2 of the optional extended logic input.)
<x>=1, 2
Example :CURSOR:MARKER:TRACE1 1
:CURSOR:MARKER:TRACE1?→:CURSOR:MARKER:TRACE1 1

:CURSor:MARKer:Y<x>?

Function Queries the Y-axis value of the marker .
Syntax :CURSor:MARKer:Y<x>?
<x>=1, 2
Example :CURSOR:MARKER:Y1?→:CURSOR:MARKER:Y1 -4.5000E-03
Description
- If "CURSor:TYPe" is not "MARKer," the command returns "NAN" ("Not A Number").
- If linear scaling is ON, this message queries the scaling value.

:CURSor:TYPE

Function Sets/queries the cursor type.
Syntax :CURSor:TYPE {HAVertical|HORizontal|MARKer|OFF|VERTical}
:CURSor:TYPE?
Example :CURSOR:TYPE HORIZONTAL
:CURSor:TYPE?→:CURSOR:TYPE HORIZONTAL

:CURSor:USERdefine?

Function Queries all settings relating to the user defined cursor.
Syntax :CURSor:USERdefine?
Example :CURSOR:USERDEFINE?→:CURSOR:USERDEFINE:TRACE 1;POSITION1 -4.00E+00;POSITION2 4.00+00;REFERENCE1 -2.00+00;REFERENCE2 2.00+00;RVALUE 3.600E+02;UNIT "deg"

:CURSor:USERdefine:DX?

Function Queries the X-axis value between the user defined cursors (relative value with respect to the reference cursor).
Syntax :CURSor:USERdefine:DX?
Example :CURSOR:USERDEFINE:DX?→1.800E+02

:CURSor:USERdefine:POSition<x>

Function Sets/queries the position of the user defined cursor.
Syntax :CURSor:USERdefine:POSition<x> {<NRf>}
:CURSor:USERdefine:POSition<x>?
<NRf>=-5 to 5div(10div/display record length steps)
<x>=1, 2
Example :CURSOR:USERDEFINE:POSITION1 -4
:CURSOR:USERDEFINE:POSITION1?→:CURSOR:USERDEFINE:POSITION1 -4.00E+00

:CURSor:USERdefine:REFerence<x>

Function Sets/queries the position of the user defined reference cursor.
Syntax :CURSor:USERdefine:REFerence<x> {<NRf>}
:CURSor:USERdefine:REFerence<x>?
<NRf>=-5 to 5div(10div/display record length steps)
<x>=1, 2
Example :CURSOR:USERDEFINE:REFERENCE1 -2
:CURSOR:USERDEFINE:REFERENCE1?→:CURSOR:USERDEFINE:REFERENCE1 -2.00E+00

:CURSor:USERdefine:RVALue

Function Sets/queries the reference width of the user defined cursor.
Syntax :CURSor:USERdefine:RVALue {<NRf>}
:CURSor:USERdefine:RVALue?
Example :CURSOR:USERDEFINE:RVALUE 180
:CURSOR:USERDEFINE:RVALUE?→:CURSOR:USERDEFINE:RVALUE 1.800E+02

:CURSor:USERdefine:TRACe

Function Sets/queries the waveform to measure with the user defined cursor.
Syntax :CURSor:USERdefine:TRACe {<NRf>|MATH<x>}
:CURSor:USERdefine:TRACe?
<NRf>=1 to 18 (However, 17 and 18 correspond to LOGIC1 and LOGIC2 of the optional extended logic input.)
<x>=1, 2
Example :CURSOR:USERDEFINE:TRACE 1
:CURSOR:USERDEFINE:TRACE→:CURSOR:USERDEFINE:TRACE 1

:CURSor:USERdefine:UNIT

Function Sets/queries the unit that is added to the cursor value of the user defined cursor.
Syntax :CURSor:USERdefine:UNIT {<Character string>}
:CURSor:USERdefine:UNIT?
<Character string>=3 characters or less
Example :CURSOR:USERDEFINE:UNIT "deg"
:CURSOR:USERDEFINE:UNIT?→:CURSOR:USERDEFINE:UNIT "deg"

:CURSor:USERdefine:X<x>?

Function Queries the X-axis value (relative value from the reference cursor) of the user defined cursor position.
Syntax `:CURSor:USERdefine:X<x>?`
`<x>=1, 2`
Example `:CURSOR:USERDEFINE:X1?→9.000E+01`

:CURSor:VERTical?

Function Queries all V cursor settings.
Syntax `:CURSor:VERTical?`
Example `:CURSOR:VERTICAL→:CURSOR:VERTICAL:TRACE 1;POSITION1 -4.00E+00;POSITION2 4.00E+00`

:CURSor:VERTical:DX?

Function Queries the X-axis distance between the V cursors .
Syntax `:CURSor:VERTical:DX?`
Example `:CURSOR:VERTICAL:DX?→:CURSOR:VERTICAL:DX 2.50E-06`
Description
- Queries the time when in the time domain and the frequency when in the frequency domain.
- If the X-Y mode is set to "X-Y" display, the command queries the physical value of the vertical axis of the X trace. If linear scaling is ON, the command queries the linear scaling value.
- When the trace is that of a logic waveform or simultaneous binary computation of all channels, "NAN"(Not A Number) is returned.

:CURSor:VERTical:PERDt?

Function Queries the 1/ΔT value between the V cursors.
Syntax `:CURSor:VERTical:PERDt?`
Example `:CURSOR:VERTICAL:PERDT?→:CURSOR:VERTICAL:PERDT 2.50E+06`
Description When the trace is that of a frequency domain, logic waveform or simultaneous binary computation of all channels, "NAN"(Not A Number) is returned.

:CURSor:VERTical:POSition<x>

Function Sets/queries the position of the V cursor.
Syntax `:CURSor:VERTical:POSition<x> {<NRf>}`
`:CURSor:VERTical:POSition<x>?`
`<NRf>=-5 to 5div (in steps of [10div/displayed record length])`
`<x>=1, 2`
Example `:CURSOR:VERTICAL:POSITION1 2`
`:CURSOR:VERTICAL:POSITION1?→:CURSOR:VERTICAL:POSITION1 2.00E+00`

:CURSor:VERTical:TRACe

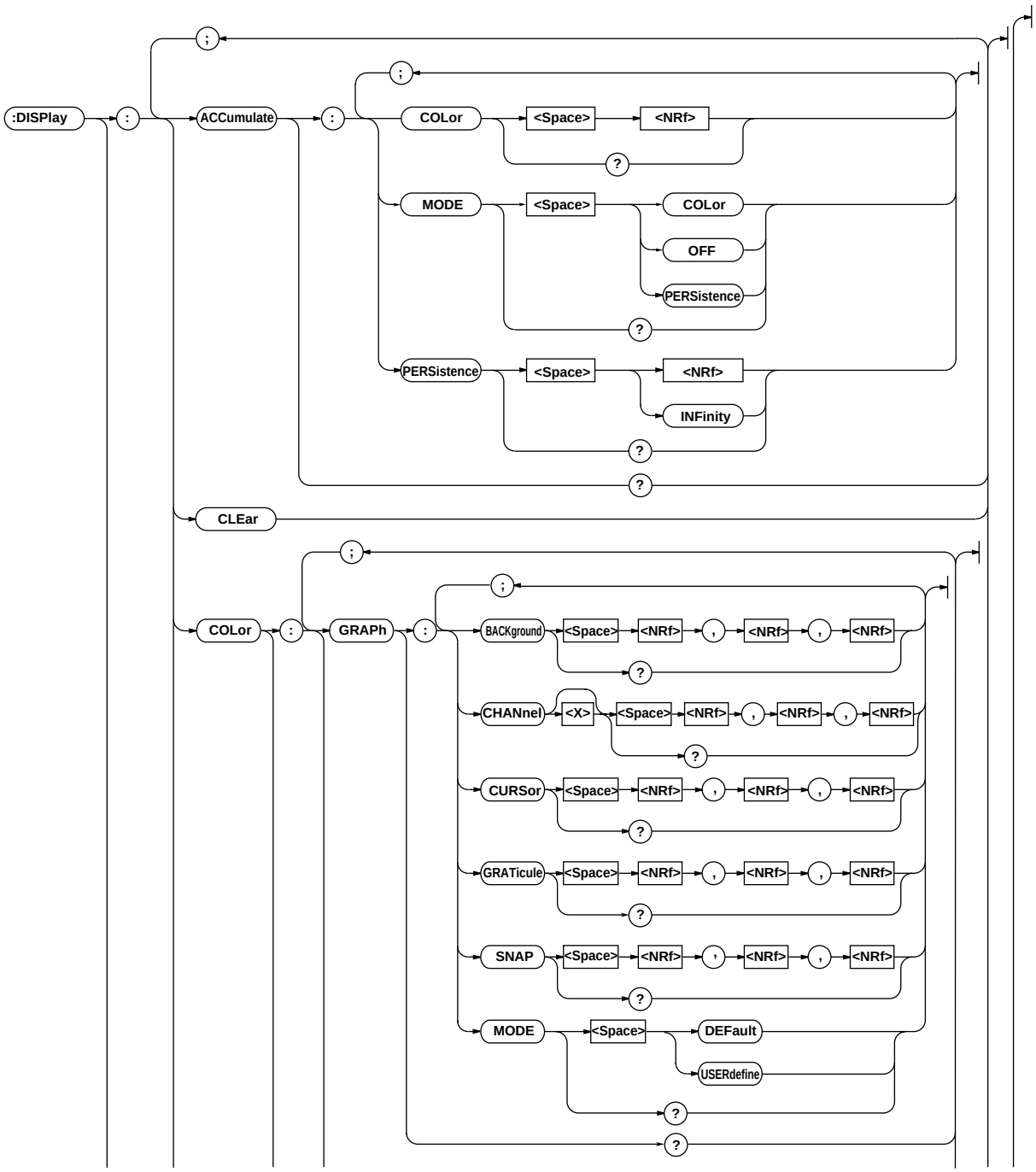
Function Sets the waveform for which the V cursor is to be used, or queries the current setting.
Syntax `:CURSor:VERTical:TRACe {<NRf>|MATH<x>}`
`:CURSor:VERTical:TRACe?`
`<NRf>=1 to 18 (However, 17 and 18 correspond to LOGIC1 and LOGIC2 of the optional extended logic input.)`
`<x>=1, 2`
Example `:CURSOR:VERTICAL:TRACE 1`
`:CURSOR:VERTICAL:TRACE?→:CURSOR:VERTICAL:TRACE 1`

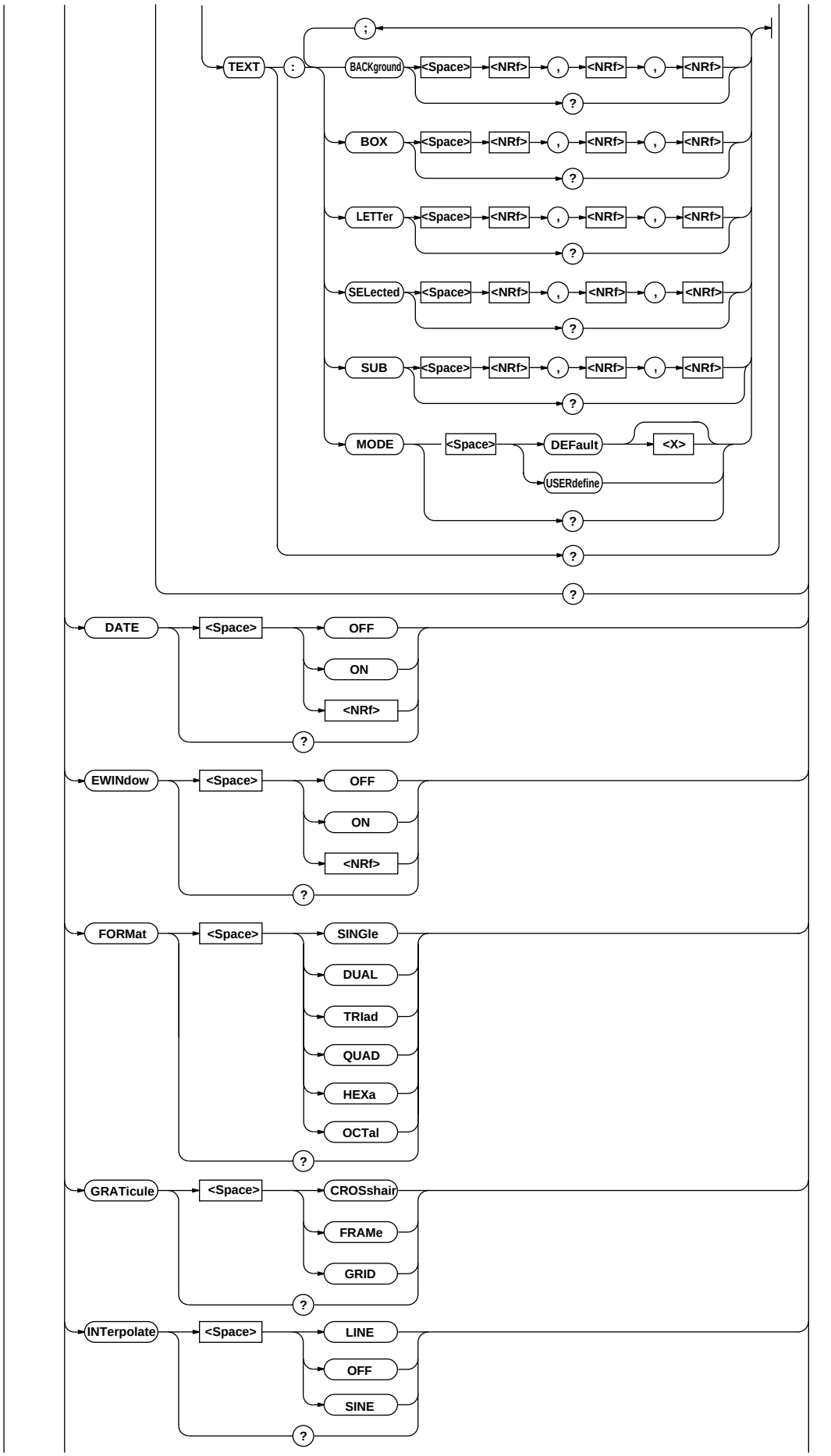
:CURSor:VERTical:X<x>?

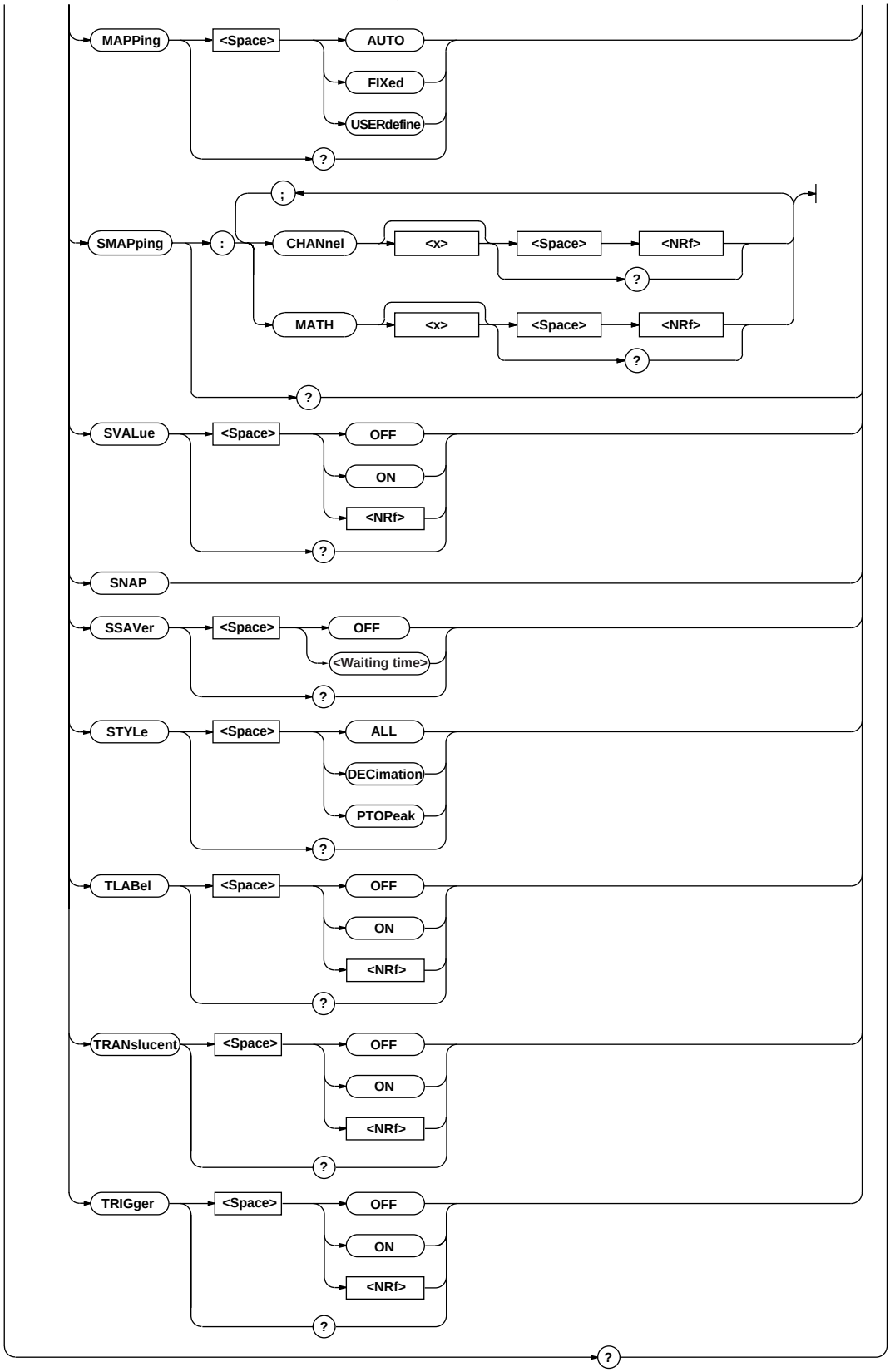
Function Queries the X-axis value of the V cursor.
Syntax `:CURSor:VERTical:X<x>?`
`<x>=1, 2`
Example `:CURSOR:VERTICAL:X1?→:CURSOR:VERTICAL:X1 -2.50E-06`
Description
- Queries the time when in the time domain and the frequency when in the frequency domain.
- If the X-Y mode is set to "X-Y" display, the command queries the physical value of the vertical axis of the X trace. If linear scaling is ON, the command queries the linear scaling value.
- When the trace is that of a logic waveform (optional) and simultaneous binary computation of all channels, the query is made on a decimal value with bit8 to bit1 (CH1) of PadB being the MSB side and bit8 to bit1 (CH16) of PadA being the LSB side.

4.10 DISPlay Group

The commands in the DISPlay group are used to set or query the display parameters. This allows you to make the same settings that can be made using the DISPLAY, CLEAR, SNAP SHOT, and MISC keys etc.




;
TEXT
:
BACKground
<Space>
<NRf>
,
?
BOX
LETTer
SELected
SUB
MODE
DEFault
<X>
USERdefine
DATE
OFF
ON
EWINdow
FORMat
SINGle
DUAL
TRIad
QUAD
HEXa
OCTal
GRATicule
CROSshair
FRAMe
GRID
INTerpolate
LINE
SINE


MAPPing
<Space>
AUTO
FIXed
USERdefine
?
SMAPping
:
;
CHANnel
<x>
<Space>
<NRf>
?
MATH
<x>
<Space>
<NRf>
?
?
SVALue
<Space>
OFF
ON
<NRf>
?
SNAP
SSAVer
<Space>
OFF
<Waiting time>
?
STYLe
<Space>
ALL
DECimation
PTOPeak
?
TLABel
<Space>
OFF
ON
<NRf>
?
TRANslucent
<Space>
OFF
ON
<NRf>
?
TRIGger
<Space>
OFF
ON
<NRf>
?
?

:DISPlay?

Function Queries all display settings.
Syntax :DISPlay?
Example :DISPLAY?→:DISPLAY:FORMAT QUAD;
EWINDOW 0;GRATICULE GRID;
INTERPOLATE SINE;MAPPING AUTO;
STYLE PTOPEAK;SVALUE 0;TLABEL 1;
ACCUMULATE:MODE OFF;:DISPLAY:DATE 1;
SSAVER OFF;TRIGGER 1;TRANSLUCENT OFF;
COLOR:GRAPH:MODE DEFAULT;:DISPLAY:COLOR:
TEXT:MODE DEFAULT1

:DISPlay:ACCumulate?

Function Queries all accumulation settings.
Syntax :DISPlay:ACCumulate?
Example :DISPLAY:ACCUMULATE?→:DISPLAY:
ACCUMULATE:MODE PERSISTENCE;
PERSISTENCE 16
Description The same query can be made using "ACCumulate?."

:DISPlay:ACCumulate:COLor

Function Sets/queries the color grading width.
Syntax :DISPlay:ACCumulate:COLor {<NRf>}
:DISPlay:ACCumulate:COLor?
<NRf>=2 to 32 (in steps of 2^n)
Example :DISPLAY:ACCUMULATE:COLOR 16
:DISPLAY:ACCUMULATE:COLOR?→:DISPLAY:
ACCUMULATE:COLOR 16
Description The same query can be made using
"ACCumulate:COLor."

:DISPlay:ACCumulate:MODE

Function Selects/queries the accumulation mode.
Syntax :DISPlay:ACCumulate:MODE {COLor|OFF|
PERSistence}
:DISPlay:ACCumulate:MODE?
Example :DISPLAY:ACCUMULATE:MODE PERSISTENCE
:DISPLAY:ACCUMULATE:MODE?→:DISPLAY:
ACCUMULATE:MODE PERSISTENCE
Description The same setting/query can be made using
"ACCumulate:MODE."

:DISPlay:ACCumulate:PERSistence

Function Sets/queries the accumulation count.
Syntax :DISPlay:ACCumulate:PERSistence {<NRf>|
INFinity}
:DISPlay:ACCumulate:PERSistence?
<NRf>=2 to 128 (in steps of 2^n)
Example :DISPLAY:ACCUMULATE:PERSISTENCE 16
:DISPLAY:ACCUMULATE:PERSISTENCE?
→:DISPLAY:ACCUMULATE:PERSISTENCE 16
Description The same setting/query can be made using
"ACCumulate:PERSistence."

:DISPlay:CLEar

Function Clears the trace.
Syntax :DISPlay:CLEar
Example :DISPLAY:CLEAR
Description • The same function can be performed using "CLEar."
• Use "SNAP" or "DISPlay:SNAP" for a snap shot.

:DISPlay:COLor?

Function Queries all screen color settings.
Syntax :DISPlay:COLor?
Example :DISPLAY:COLOR?→:DISPLAY:COLOR:GRAPH:
MODE USERDEFINE;BACKGROUND 0,0,0;
CURSOR 7,7,7;GRATICULE 4,4,4;SNAP 7,7,7;
CHANNEL1 7,7,0;CHANNEL2 0,7,0;
CHANNEL3 7,0,7;CHANNEL4 0,7,7;
CHANNEL5 7,0,0;CHANNEL6 7,4,0;
CHANNEL7 0,4,7;CHANNEL8 5,5,5:DISPLAY:
COLOR:TEXT:MODE USERDEFINE;
BACKGROUND 2,2,6;BOX 0,0,7;LETTER 7,7,7;
SELECTED 0,4,7;SUB 4,4,4

:DISPlay:COLor:GRAPh?

Function Queries all settings relating to the color of graphic items.
Syntax :DISPlay:COLor:GRAPh?
Example :DISPLAY:COLOR:GRAPH?→:DISPLAY:COLOR:
GRAPH:MODE USERDEFINE;BACKGROUND 0,0,0;
CURSOR 7,7,7;GRATICULE 4,4,4;SNAP 7,7,7;
CHANNEL1 7,7,0;CHANNEL2 0,7,0;
CHANNEL3 7,0,7;CHANNEL4 0,7,7;
CHANNEL5 7,0,0;CHANNEL6 7,4,0;
CHANNEL7 0,4,7;CHANNEL8 5,5,5

:DISPlay:COLor:GRAPh:{BACKground|CURSor|GRATicule|SNAP|CHANnel<x>}

Function Sets/queries the color for the background, cursor, graticule (scale), snapshot waveform, or channel waveform.
Syntax :DISPlay:COLor:GRAPh:{BACKground|CURSor|
GRATicule|SNAP|CHANnel<x>} {<NRf>,<NRf>,
<NRf>}
:DISPlay:COLor:GRAPh:{BACKground|CURSor|
GRATicule|SNAP|CHANnel<x>}?
<x>=1 to 8
<NRf>=0 to 7 (for R, G and B
respectively)
Example (Examples of color settings are given
below.)
:DISPLAY:COLOR:GRAPH:BACKGROUND 0,0,0
:DISPLAY:COLOR:GRAPH:BACKGROUND?
→:DISPLAY:COLOR:GRAPH:BACKGROUND 0,0,0

:DISPlay:COLor:GRAPh:MODE

Function Sets/queries the color mode for graphic items.
Syntax :DISPlay:COLor:GRAPh:MODE {DEFault|
USERdefine}
:DISPlay:COLor:GRAPh:MODE?
Example :DISPLAY:COLOR:GRAPH:MODE DEFAULT
:DISPLAY:COLOR:GRAPH:MODE?→:DISPLAY:
COLOR:GRAPH:MODE DEFAULT

:DISPlay:COLor:TEXT?

Function Queries all text color settings.
Syntax :DISPlay:COLor:TEXT?
Example :DISPLAY:COLOR:TEXT?→:DISPLAY:COLOR:TEXT:MODE USERDEFINE;BACKGROUND 2,2,6;BOX 0,0,7;LETTER 7,7,7;SELECTED 0,4,7;SUB 4,4,4

:DISPlay:COLor:TEXT:{BACKground|BOX|LETTer|SELected|SUB}

Function Sets/queries the color for the menu background (Menu Back), selected menu (Select Box), characters (Menu Fore), selected keys (Selected Key), or pop-up menu (Submenu).
Syntax :DISPlay:COLor:TEXT:{BACKground|BOX|LETTer|SELected|SUB} {<NRf>,<NRf>,<NRf>}
:DISPlay:COLor:TEXT:{BACKground|BOX|LETTer|SELected|SUB}?
<NRf>=0 to 7 (for R, G, and B respectively)
Example (Examples of menu background color setting are given below.)
:DISPLAY:COLOR:TEXT:BACKGROUND 2,2,6
:DISPLAY:COLOR:TEXT:BACKGROUND?
→:DISPLAY:COLOR:TEXT:BACKGROUND 2,2,6

:DISPlay:COLor:TEXT:MODE

Function Sets/queries the color mode for text items.
Syntax :DISPlay:COLor:TEXT:MODE {DEFault<x>|USERdefine}
:DISPlay:COLor:TEXT:MODE?
<x>=1 to 3
Example :DISPLAY:COLOR:TEXT:MODE DEFAULT1
:DISPLAY:COLOR:TEXT:MODE?→:DISPLAY:COLOR:TEXT:MODE DEFAULT1

:DISPlay:DATE

Function Selects whether to display the date and time, or queries the current setting.
Syntax :DISPlay:DATE {<Boolean>}
:DISPlay:DATE?
Example :DISPLAY:DATE ON
:DISPLAY:DATE?→:DISPLAY:DATE 1
Description The lower two digits of the year are displayed. Years 2000 to 2079 are represented by 00 to 79, and years 1980 to 1999 are represented by 80 to 99.

:DISPlay:EWINdow(Extra WINDOW)

Function Selects whether to display the extra window, or queries the current setting.
Syntax :DISPlay:EWINdow {<Boolean>}
:DISPlay:EWINdow?
Example :DISPLAY:EWINDOW OFF
:DISPLAY:EWINDOW?→:DISPLAY:EWINDOW 0

:DISPlay:FORMat

Function Sets/queries the display format.
Syntax :DISPlay:FORMat {SINGle|DUAL|TRIad|QUAD|HEXa|OCTal}
:DISPlay:FORMat?
Example :DISPLAY:FORMAT QUAD
:DISPLAY:FORMAT?→:DISPLAY:FORMAT QUAD

:DISPlay:GRATicule

Function Sets/queries the graticule.
Syntax :DISPlay:GRATicule {CROSshair|FRAMe|GRID}
:DISPlay:GRATicule?
Example :DISPLAY:GRATICULE GRID
:DISPLAY:GRATICULE?→:DISPLAY:GRATICULE GRID

:DISPlay:INTerpolate

Function Selects/queries the interpolation method.
Syntax :DISPlay:INTerpolate {LINE|OFF|SINE}
:DISPlay:INTerpolate?
Example :DISPLAY:INTERPOLATE LINE
:DISPLAY:INTERPOLATE?→:DISPLAY:INTERPOLATE LINE

:DISPlay:MAPPing

Function Sets/queries the mapping mode.
Syntax :DISPlay:MAPPing {AUTO|FIXed|USERdefine}
:DISPlay:MAPPing?
Example :DISPLAY:MAPPING AUTO
:DISPLAY:MAPPING?→:DISPLAY:MAPPING AUTO

:DISPlay:SMAPping?

Function Queries all settings relating to the assignment of the waveforms to the split window.
Syntax :DISPlay:SMAPping?
Example :DISPLAY?→:DISPLAY:SMAPPING:CHANNEL1 0;CHANNEL2 1;CHANNEL5 0;CHANNEL6 1;CHANNEL7 2;CHANNEL8 3;CHANNEL10 1;CHANNEL11 2;CHANNEL12 3;CHANNEL13 0;CHANNEL14 1;CHANNEL15 2;CHANNEL16 3;CHANNEL17 0;CHANNEL18 1;MATH1 0;MATH2 1;MATH3 2;MATH4 3;MATH5 0;MATH6 1;MATH7 2;MATH8 3

:DISPlay:SMAPping:{CHANnel<x>|MATH<x>}

Function Sets/queries the assignment of the waveforms to the split window.

Syntax
```
:DISPlay:SMAPping:{CHANnel<x>|
MATH<x>} {<NRf>}
:DISPlay:SMAPping:{CHANnel<x>|MATH<x>}?
```

```
CHANnel<x>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
<x> of MATH<x>=1 to 8
<NRf>=0 to 7
```

Example
```
(Below is an example of CH1.)
:DISPLAY:SMAPPING:CHANNEL1 0
:DISPLAY:SMAPPING:CHANNEL1?→DISPLAY:
SMAPPING:CHANNEL1 0
```

Description This setting is valid only when "`:MATH:MAPPing`" is set to "`USERdefine`."

:DISPlay:SNAP

Function Performs a snapshot.

Syntax
```
:DISPlay:SNAP
```

Example
```
:DISPLAY:SNAP
```

Description
- The same function can be performed using "`SNAP`."
- "`DISPlay:CLEar`" or "`CLEar`" must be used when you perform clear a trace.

:DISPlay:SSAVer (Screen SAVER)

Function Sets/queries screen-saver ON/OFF.

Syntax
```
:DISPlay:SSAVer {OFF|MIN10|MIN30|
HOUR1|HOUR2|HOUR5}
:DISPlay:SSAVer?
```

Example
```
:DISPLAY:SSAVER HOUR1
:DISPLAY:SSAVER?→:DISPLAY:SSAVER HOUR1
```

:DISPlay:STYLe

Function Sets/queries display compression.

Syntax
```
:DISPlay:STYLe {ALL|DECimation|PTOPeak}
:DISPlay:STYLe?
```

Example
```
:DISPLAY:STYLE ALL
:DISPLAY:STYLE?→:DISPLAY:STYLE ALL
```

:DISPlay:SVALue (Scale VALUE)

Function Selects/queries whether scaling values are displayed.

Syntax
```
:DISPlay:SVALue {<Boolean>}
:DISPlay:SVALue?
```

Example
```
:DISPLAY:SVALUE OFF
:DISPLAY:SVALUE?→:DISPLAY:SVALUE 0
```

:DISPlay:TLABel (Trace LABEL)

Function Selects/queries whether waveform labels are displayed.

Syntax
```
:DISPlay:TLABel {<Boolean>}
:DISPlay:TLABel?
```

Example
```
:DISPLAY:TLABEL ON
:DISPLAY:TLABEL?→:DISPLAY:TLABEL 1
```

Description You can set a user defined waveform label using the "`:CHANnel<x>:LABel`" command.

:DISPlay:TRANslucent

Function Sets/queries the ON/OFF condition of the translucent mode of the pop-up menu.

Syntax
```
:DISPlay:TRANslucent {<Boolean>}
:DISPlay:TRANslucent?
```

Example
```
:DISPLAY:TRANSLUCEENT ON
:DISPLAY:TRANSLUCEENT?→:DISPLAY:
TRANSLUCEENT 1
```

:DISPlay:TRIGger

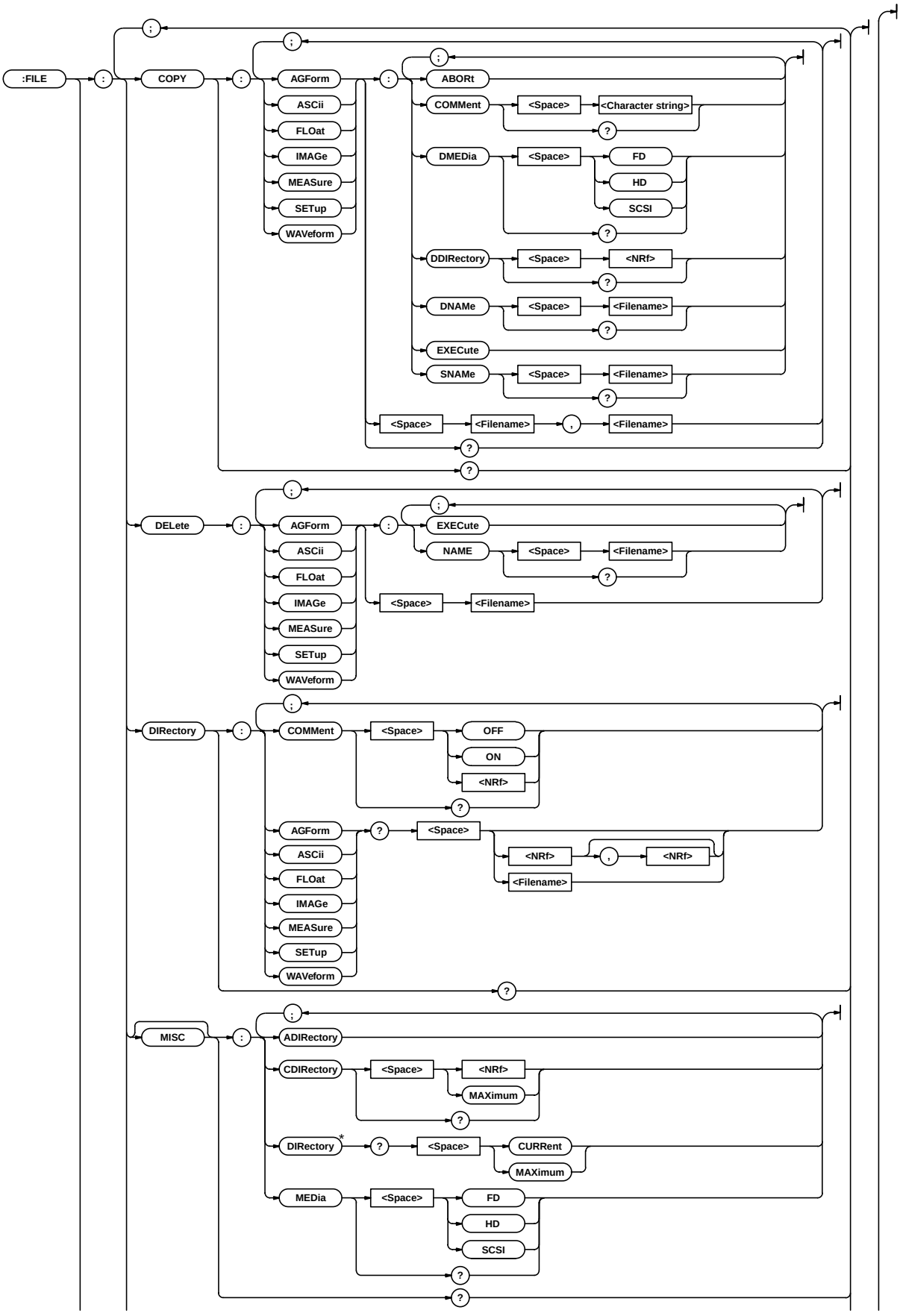
Function Sets/queries trigger mark ON/OFF.

Syntax
```
:DISPlay:TRIGger {<Boolean>}
:DISPlay:TRIGger?
```

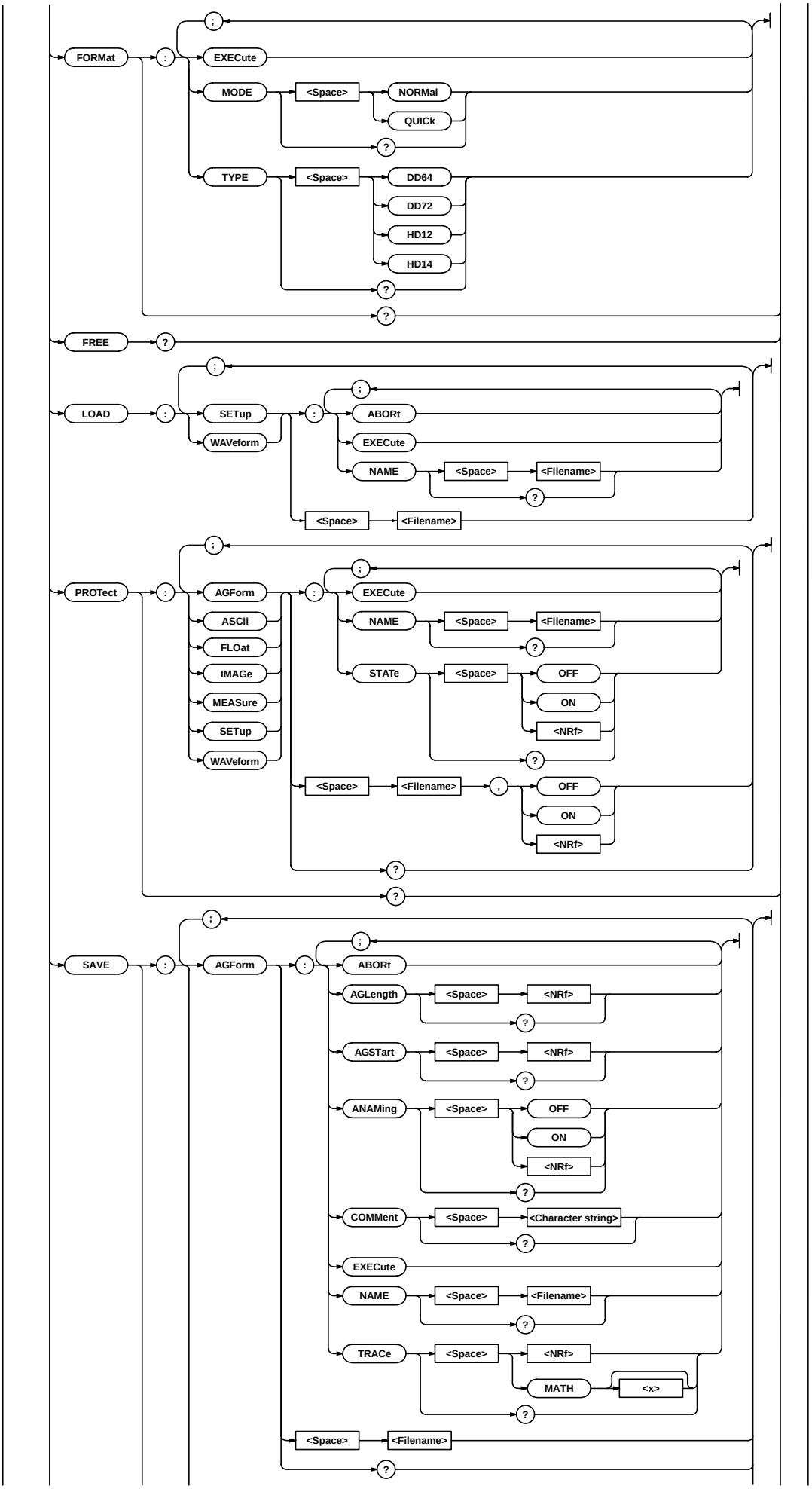
Example
```
:DISPLAY:TRIGGER ON
:DISPLAY:TRIGGER?→:DISPLAY:TRIGGER 1
```

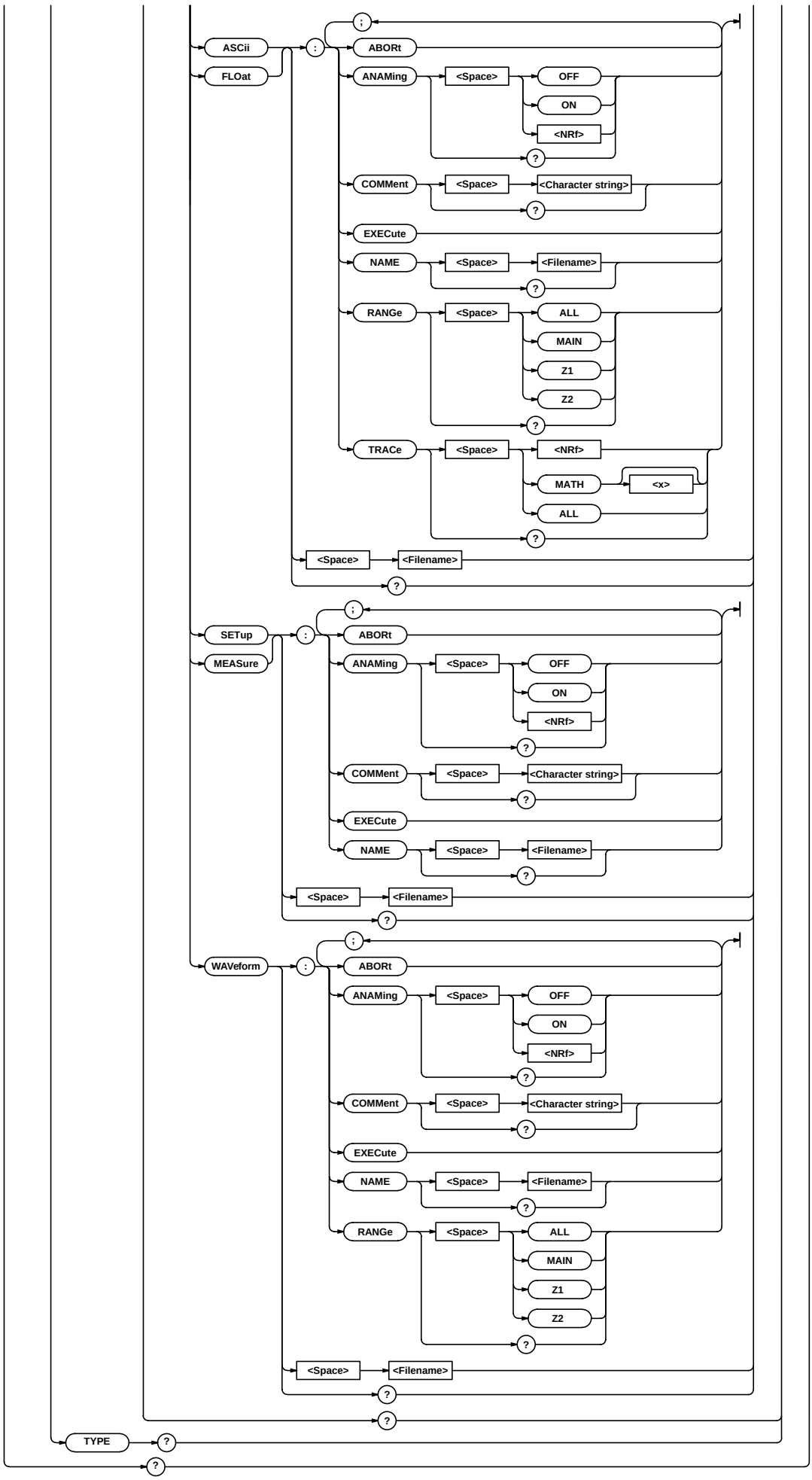
4.11 FILE Group

These commands control operation of floppy disks, HD, and external SCSI devices. This allows you to make the same settings and queries that can be made using the FILE key.



* "MISC" cannot be omitted when using "DIRectory".


FORMat
:
;
EXECute
MODE
<Space>
NORMal
QUICk
?
TYPE
<Space>
DD64
DD72
HD12
HD14
?
?
FREE
?
LOAD
:
;
SETup
WAVeform
:
;
ABORt
EXECute
NAME
<Space>
<Filename>
?
<Space>
<Filename>
PROTect
:
;
AGForm
ASCii
FLOat
IMAGe
MEASure
SETup
WAVeform
:
;
EXECute
NAME
<Space>
<Filename>
?
STATe
<Space>
OFF
ON
<NRf>
?
<Space>
<Filename>
,
OFF
ON
<NRf>
?
?
SAVE
:
;
AGForm
:
;
ABORt
AGLength
<Space>
<NRf>
?
AGSTart
<Space>
<NRf>
?
ANAMing
<Space>
OFF
ON
<NRf>
?
COMMent
<Space>
<Character string>
?
EXECute
NAME
<Space>
<Filename>
?
TRACe
<Space>
<NRf>
MATH
<x>
?
<Space>
<Filename>
?


ASCii
FLOat
:
;
ABORt
ANAMing
<Space>
OFF
ON
<NRf>
?
COMMent
<Space>
<Character string>
?
EXECute
NAME
<Space>
<Filename>
?
RANGe
<Space>
ALL
MAIN
Z1
Z2
?
TRACe
<Space>
<NRf>
MATH
<x>
ALL
?
<Space>
<Filename>
?
SETup
MEASure
:
;
ABORt
ANAMing
<Space>
OFF
ON
<NRf>
?
COMMent
<Space>
<Character string>
?
EXECute
NAME
<Space>
<Filename>
?
<Space>
<Filename>
?
WAVeform
:
;
ABORt
ANAMing
<Space>
OFF
ON
<NRf>
?
COMMent
<Space>
<Character string>
?
EXECute
NAME
<Space>
<Filename>
?
RANGe
<Space>
ALL
MAIN
Z1
Z2
?
<Space>
<Filename>
?
?
TYPE
?
?

:FILE?

Function Queries the settings for the specified medium.

Syntax
```
:FILE?
```

Example
```
:FILE?→:FILE:COPY:DMEDIA FD;:FILE:
DIRECTORY:COMMENT 0;:FILE:FORMAT:
TYPE HD14;:FILE:MISC:MEDIA FD;:FILE:
PROTECT:STATE 0;:FILE:SAVE:AGLENGTH 1000;
AGSTART -5.000E+00;ANAMING 0;
COMMENT "                    ";NAME "";
TRACE 1
```

:FILE:COPY?

Function Queries all settings relating to file copying.

Syntax
```
:FILE:COPY?
```

Example
```
:FILE:COPY?→:FILE:COPY:
COMMENT "THIS IS TEST.       ";
DNAME "CASE1";DMEDIA FD
```

:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}

Function Makes a copy of a file. This is an overlap command.

Syntax
```
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform} <Filename>,
<Filename>
```

Example
```
(An example of copying waveform data is
given below.)
:FILE:COPY:WAVEFORM "CASE1","COPYED1"
```

Description
- The first <Filename> is the name of the source file and the second <Filename> is the name of the destination file.
- The above example and the following message have the same result, but differ in that they do not belong to the same hierarchical level and that the remaining operations are not performed in case of an error.
```
:FILE:COPY:WAVEFORM:SNAME "CASE1";
DNAME "COPYED1";EXECUTE
```

:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}?

Function Queries all settings relating to file copying.

Syntax
```
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}?
```

Example
```
(An example of copying waveform data is
given below.)
:FILE:COPY:WAVEFORM?→:FILE:COPY:
WAVEFORM:COMMENT "THIS IS TEST.       ";
DMEDIA FD;DNAME "CASE1"
```

:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:ABORt

Function Aborts file copying.

Syntax
```
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:ABORt
```

Example
```
(An example of aborting copying of
waveform data is given below.)
:FILE:COPY:WAVEFORM:ABORT
```

:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:COMMent

Function Sets a comment for the destination file, or queries the current setting.

Syntax
```
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:
COMMent <Character string>
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:COMMent?
<Character string>=Within 20 characters
```

Example
```
(Example for waveform data is given
below.)
:FILE:COPY:WAVEFORM:
COMMENT "THIS IS TEST.       "
:FILE:COPY:WAVEFORM:COMMENT?→:FILE:COPY:
WAVEFORM:COMMENT "THIS IS TEST.       "
```

Description
- The comment set for copying is also used when a save operation is performed.
- It is not possible to change or query the comment used in an existing file.
- Only characters and symbols available on the keyboard displayed on the screen can be used. "Ω" will be replaced by "1EH" and "μ" by "1FH" (their ASCII codes).

:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:DDIRectory

(Destination DIRECTORY)

Function Sets/queries the destination directory for copy.

Syntax
```
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:
DDIRectory {<NRf>}
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:DDIRectory?
<NRf>=1 to 65535
```

Example
```
:FILE:COPY:WAVEFORM:DDIRECTORY 1
:FILE:COPY:WAVEFORM:DDIRECTORY?→:FILE:
COPY:WAVEFORM:DDIRECTORY 1
```

Description This command is meaningless if the target medium is a floppy disk.

:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:DMEDia

(Destination MEDIA)

Function Sets/queries the destination medium.

Syntax
```
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:DMEDia {FD|MO|
SCSI}
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:DMEDia?
```

Example
```
:FILE:COPY:WAVEFORM:DMEDIA FD
:FILE:COPY:WAVEFORM:DMEDIA?→:FILE:COPY:
WAVEFORM:DMEDIA FD
```

Description HD is available as the option.

:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}]:DNAMe

(Distination NAME)

Function Sets/queries the name of the destination file.

Syntax
```
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:DNAMe <Filename>
:FILE:COPY[:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}]:DNAMe?
```

Example
```
(An example for waveform data is given
below.)
:FILE:COPY:WAVEFORM:DNAME "COPYED1"
:FILE:COPY:WAVEFORM:DNAME?→:FILE:COPY:
WAVEFORM:DNAME "COPYED1"
```

Description
- The same file name is used exactly as set, irrespective of the data type.
- The name set is also used when a save operation is performed.

:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:EXECute

Function Copies a file. This is an overlap command.

Syntax
```
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:EXECute
```

Example
```
(An example for waveform data is given
below.)
:FILE:COPY:WAVEFORM:EXECUTE
```

:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:SNAMe

(Source NAME)

Function Sets/queries the name of the source file.

Syntax
```
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:SNAMe <Filename>
:FILE:COPY:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:SNAMe?
```

Example
```
(An example for waveform data is given
below.)
:FILE:COPY:WAVEFORM:SNAME "CASE1"
:FILE:COPY:WAVEFORM:SNAME?→"CASE1"
```

Description
- The file name set is also used when loading, protection setting and protection canceling operations are performed.
- If there is no file of the specified name, an error will occur when the file name is set and "" will be returned when a query is made.
- If the disk is replaced with a new one, the first file on the new disk will be copied.

:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}

Function Deletes a file. This is an overlap command.

Syntax
```
:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform} <Filename>
```

Example
```
(An example for waveform data is given
below.)
:FILE:DELETE:WAVEFORM "CASE1"
```

Description The above example and the following message perform similar operations, but differ in that they do not belong to the same hierarchical level and that the remaining operations are not performed in case of an error.

```
:FILE:DELETE:WAVEFORM:NAME "CASE1";EXECUTE
```

:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:EXECute

Function Deletes a file. This is an overlap command.

Syntax
```
:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:EXECute
```

Example
```
(An example for waveform data is given
below.)
:FILE:DELETE:WAVEFORM:EXECUTE
```

:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:NAME

Function Sets/queries the name of the file to be deleted.

Syntax
```
:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:NAME <Filename>
:FILE:DELete:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:NAME?
```

Example
```
(An example for waveform data is given
below.)
:FILE:DELETE:WAVEFORM:NAME "CASE1"
:FILE:DELETE:WAVEFORM:NAME?→"CASE1"
```

Description
- The file name set is also used when loading, protection setting and protection canceling operations are performed.
- If there is no file with the specified name, an error will occur when the file name is set and "" will be returned when a query is made.
- If the disk is replaced with a new one, the first file on the new disk will be deleted.

:FILE:DIRectory?

Function Queries all settings relating to querying of files in a directory.

Syntax
```
:FILE:DIRectory?
```

Example
```
:FILE:DIRECTORY?→:FILE:DIRECTORY:
COMMENT 0
```

:FILE:DIRectory:COMMent

Function Selects/queries whether a comment is added when making query about files in the specified directory.

Syntax
```
:FILE:DIRectory:COMMent {<Boolean>}
:FILE:DIRectory:COMMent?
```

Example
```
:FILE:DIRECTORY:COMMENT OFF
:FILE:DIRECTORY:COMMENT?→:FILE:
DIRECTORY:COMMENT 0
```

:FILE:DIRectory:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}?

Function Queries the files in the specified directory.

Syntax
```
:FILE:DIRectory:{AGForm|ASCii|FLOat|
IMAGe|MEASure|SETup|WAVeform}? [{<NRf>
[,<NRf>]|<Filename>}]
<NRf>=1 to the maximum number of files
```

Example
```
(An example for waveform data is given
below.)
:FILE:DIRECTORY:WAVEFORM?→"CASE1",
"CASE2","CASE3"
```

Description
- If all parameters are omitted, the names of all the files in the directory will be returned.
- When making a query in <NRf> format, designate the start file using the first <NRf> parameter and the number of files using the second <NRf> parametr. If the second <NRf> parameter is omitted, a query is made about all files from the specified first file up to the last file.
- When making a query using <Filename>, a query is made about files whose file name begins with the specified character string.
- If no file with the specified name exists, "" will be returned.

:FILE:FORMat?

Function Queries all settings relating to the medium formatting.

Syntax `:FILE:FORMat?`

Example `:FILE:FORMAT?→:FILE:FORMAT:TYPE HD12`

:FILE:FORMat:EXECute

Function Formats a medium. This is an overlap command.

Syntax `:FILE:FORMat:EXECute`

Example `:FILE:FORMAT:EXECUTE`

Description This command operates only on the medium selected by the "`FILE:MISC:MEDia`" command.

:FILE:FORMat:MODE

Function Sets/queries the format mode.

Syntax
```
:FILE:FORMat:MODE {NORMal|QUICk}
:FILE:FORMat:MODE?
```

Example
```
:FILE:FORMAT:MODE NORMAL
:FILE:FORMAT:MODE?→:FILE:FORMAT:
MODE NORMAL
```

Description
- Description "`NORMal`" is for executing the physical and logical format, and "`QUICk`" is for executing only the logical format.
- "`QUICk`" is effective only for "`:FILE:MISC:MEDia SCSI`." For other medium, the mode is fixed to "`NORMal`."

:FILE:FORMat:TYPE

Function Selects/queries the floppy disk format to be used.

Syntax
```
:FILE:FORMat:TYPE {DD64|DD72|HD12|HD14}
:FILE:FORMat:TYPE?
```

Example
```
:FILE:FORMAT:TYPE HD12
:FILE:FORMAT:TYPE?→:FILE:FORMAT:
TYPE HD12
```

Description
- This command is effective only for floppy disks.
- You can use the "`FILE:TYPE?`" query to find the floppy disk's current format type.

:FILE:FREE?

Function Queries amount of free space on the medium.

Syntax `:FILE:FREE?`

Example `:FILE:FREE?→655360`

Description
- The number of bytes will be returned in reply to this query.
- If there is no medium in the drive, "`0`" will be returned.
- This command operates only on the medium selected by the "`FILE:MISC:MEDia`" command.

:FILE:LOAD:{SETup|WAVeform}

Function Loads data. This is an overlap command.

Syntax `:FILE:LOAD:{SETup|WAVeform} <Filename>`

Example
```
(An example for waveform data is given
below.)
:FILE:LOAD:WAVEFORM "CASE1"
```

Description The above example and the following message perform similar operations, but differ in that they do not belong to the same hierarchical level and that the remaining operations are not performed in case of an error.

```
:FILE:LOAD:WAVEFORM:NAME "CASE1";EXECUTE
```

:FILE:LOAD[:{SETup|WAVeform}]:ABORt

Function Aborts loading of data.

Syntax `:FILE:LOAD[:{SETup|WAVeform}]:ABORt`

Example
```
(An example for waveform data is given
below.)
:FILE:LOAD:WAVEFORM:ABORT
```

:FILE:LOAD:{SETup|WAVeform}:EXECute

Function Loads data. This is an overlap command.

Syntax `:FILE:LOAD:{SETup|WAVeform}:EXECute`

Example
```
(An example for waveform data is given
below.)
:FILE:LOAD:WAVEFORM:EXECUTE
```

:FILE:LOAD:{SETup|WAVeform}:NAME

Function Sets/queries the name of the file to be loaded.

Syntax
```
:FILE:LOAD:{SETup|WAVeform}:
NAME <Filename>
:FILE:LOAD:{SETup|WAVeform}:NAME?
```

Example
```
(An example for waveform data is given
below.)
:FILE:LOAD:WAVEFORM:NAME "CASE1"
:FILE:LOAD:WAVEFORM:NAME?→"CASE1"
```

Description
- The file set is also used when loading, protection setting and protection canceling operations are performed.
- If there is no file with the specified name, an error will occur when the file name is set and "" will be returned when a query is made.
- If the disk is replaced with a new one, the first file on the new disk will be loaded.

:FILE:MISC?

Function Queries the directory and medium settings.

Syntax
```
:FILE:MISC?
```

Example
```
:FILE:MISC?→:FILE:MISC:MED FD
```

:FILE[:MISC]:ADIRectory (Add DIRECTORY)

Function Adds one directory.

Syntax
```
:FILE:MISC:ADIRectory {<NRf>}
:FILE:MISC:ADIRectory?
<NRf>=1 to (maximum number for targeted
      medium)
```

Example
```
:FILE:MISC:ADIRECTORY 10
:FILE:MISC:ADIRECTORY?→:FILE:MISC:
ADIRECTORY 10
```

Description
- The total number of root directories that can be added varies depending on the medium.
- This command is irrelevant, when the medium is a floppy disk.

:FILE[:MISC]:CDIRectory

Function Changes to the selected directory. Queries the number of directories to be moved.

Syntax
```
:FILE:MISC:CDIRectory {<NRf>|MAXimum}
:FILE:MISC:CDIRectory?
<NRf>=1 to (maximum number for targeted
      medium)
```

Example
```
:FILE:MISC:CDIRECTORY 5
:FILE:MISC:CDIRECTORY?→:FILE:MISC:
CDIRECTORY 5
```

Description This command/query is irrelevant, when the medium is a floppy disk.

:FILE:MISC:DIRectory?

Function Queries the current directory No. and maximum directory No.

Syntax
```
:FILE:MISC:DIRectory? {CURRent|MAXimum}
```

Example
```
:FILE:MISC:DIRECTORY?
CURRENT→:FILE:MISC:DIRECTORY 4
```

Description Query returns a "0" if the medium is a floppy disk.

:FILE[:MISC]:MEDia

Function Sets/queries the medium used for file access.

Syntax
```
:FILE:MISC:MEDia {FD|HD|SCSI}
:FILE:MISC:MEDia?
```

Example
```
:FILE:MISC:MEDIA FD
:FILE:MISC:MEDIA?→:FILE:MISC:MEDIA FD
```

Description HD is available as the option.

:FILE:PROTect?

Function Queries all file protection settings.

Syntax
```
:FILE:PROTect?
```

Example
```
:FILE:PROTECT?→:FILE:PROTECT:STATE 0
```

:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}

Function Protects a file or cancels a file's protection. This is an overlap command.

Syntax
```
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform} <Filename>,
{<Boolean>}
```

Example
```
(An example for waveform data is given
below.)
:FILE:PROTECT:WAVEFORM "CASE1",ON
```

Description Enter ON (any value other than "0") to turn protection on, or OFF ("0") to cancel protection.
The above example and the following message perform similar operations, but differ in that they do not belong to the same hierarchical level and that the remaining operations are not performed in case of an error.
```
:FILE:PROTECT:WAVEFORM:NAME "CASE1";
STATE 1;EXECUTE
```

:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}?

Function Queries all file pretection settings.

Syntax
```
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}?
```

Example
```
(An example for waveform data is given
below.)
:FILE:PROTECT:WAVEFORM:STATE 1
```

:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:EXECute

Function Protects a file or cancels a file's protection. This is an overlap command.

Syntax
```
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:EXECute
```

Example
```
(An example for waveform data is given
below.)
:FILE:PROTECT:WAVEFORM:EXECUTE
```

:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEAS ure|SETup|WAVeform}:NAME

Function Sets/queries the name of the file to be protected or unprotected.

Syntax
```
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:NAME <Filename>
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:NAME?
```

Example
```
(An example for waveform data is given
below.)
:FILE:PROTECT:WAVEFORM:NAME "CASE1"
:FILE:PROTECT:WAVEFORM:NAME?→"CASE1"
```

Description
- The file name set is also used when deletion and loading operations are performed.
- If there is no file with the specified name, an error will occur when the file name is set and "" will be returned when a query is made.
- If the disk is replaced with a new one, the first file on the new disk will be protected or will have its protection cancelled.

:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|MEASure|SETup|WAVeform}:STATe

Function Sets the protection state of a file ON (protected) or OFF (unprotected), or queries the current setting.

Syntax
```
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:STATe {<Boolean>}
:FILE:PROTect:{AGForm|ASCii|FLOat|IMAGe|
MEASure|SETup|WAVeform}:STATe?
```

Example
```
(An example for waveform data is given
below.)
:FILE:PROTECT:WAVEFORM:STATE ON
:FILE:PROTECT:WAVEFORM:STATE?→:FILE:
PROTECT:WAVEFORM:STATE 1
```

:FILE:SAVE?

Function Queries all file-save settings.

Syntax
```
:FILE:SAVE?
```

Example
```
:FILE:SAVE?→:FILE:SAVE:ANAMING
0;AGLENGTH 1000;AGSTART -5.00E+00;
COMMENT "THIS IS TEST.       ";
NAME "CASE1";TRACE 1
```

:FILE:SAVE[:AGForm]:AGLength

Function Sets/queries the data length for AG-format file save.

Syntax
```
:FILE:SAVE:AGForm:AGLength {<NRf>}
:FILE:SAVE:AGForm:AGLength?
<NRf>=1000, 2000, 4000, 8000
```

Example
```
:FILE:SAVE:AGFORM:AGLENGTH 1000
:FILE:SAVE:AGFORM:AGLENGTH?→:FILE:SAVE:
AGFORM:AGLENGTH 1000
```

:FILE:SAVE[:AGForm]:AGSTart

Function Sets/queries the start location for AG-format file save.

Syntax
```
:FILE:SAVE:AGForm:AGSTart {<NRf>}
:FILE:SAVE:AGForm:AGSTart?
<NRf>=-5 to 5div (in steps of [10div/
      displayed record length])
```

Example
```
:FILE:SAVE:AGFORM:AGSTART 1
:FILE:SAVE:AGFORM:AGSTART?→:FILE:SAVE:
AGFORM:AGSTART 1.00E+00
```

:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}

Function Saves a file. This is an overlap command.

Syntax
```
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform} <Filename>
```

Example
```
(An example for waveform data is given
below.)
:FILE:SAVE:WAVEFORM "CASE1"
```

Description The above example and the following message perform similar operations, but differ in that they do not belong to the same hierarchical level and that the remaining operations are not performed in case of an error.
```
:FILE:SAVE:WAVEFORM:NAME "CASE1";EXECUTE
```

:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}?

Function Queries all file saving settings.

Syntax
```
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}?
```

Example
```
(An example for waveform data is given
below.)
:FILE:SAVE:WAVEFORM?→:FILE:SAVE:
WAVEFORM:COMMENT "THIS IS TEST.       ";
NAME "CASE1"
```

:FILE:SAVE[:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}]:ABORt

Function Aborts saving of data.

Syntax
```
:FILE:SAVE[:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}]:ABORt
```

Example
```
(An example for waveform data is given
below.)
:FILE:SAVE:WAVEFORM:ABORT
```

:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}:ANAMing

Function Sets/queries whether or not to automatically name the file to be saved (ON/OFF).

Syntax
```
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}:ANAMing {<Boolean>}
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}:ANAMing?
```

Example
```
FILE:SAVE:WAVEFORM:ANAMING ON
:FILE:SAVE:WAVEFORM:ANAMING?→:FILE:SAVE:
WAVEFORM:ANAMING 1
```

:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}:COMMent

Function Sets/queries a comment for saved data.

Syntax
```
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}:COMMent <Character string>
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}:COMMent?
<Character string>=Within 20 characters
```

Example
```
(An example for waveform data is given
below.)
:FILE:SAVE:WAVEFORM:COMMENT "THIS IS
TEST.       "
:FILE:SAVE:WAVEFORM:COMMENT?→:FILE:SAVE:
WAVEFORM:COMMENT "THIS IS TEST.       "
```

Description
- The comment set for saving is also used when a copy operation is performed.
- It is not possible to change or query the comment used in an existing file.
- Only characters and symbols available on the keyboard displayed on the screen can be used. "Ω" will be replaced by "1EH" and "μ" by "1FH" (their ASCII codes).

:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}:EXECute

Function Saves data. This is an overlap command.

Syntax
```
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}:EXECute
```

Example
```
(An example for waveform data is given
below.)
:FILE:SAVE:WAVEFORM:EXECUTE
```

:FILE:SAVE[:{AGForm|ASCii|FLOat|MEASure|SETup|WAVeform}]:NAME

Function Sets/queries the name of the file to be saved.

Syntax
```
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}:NAME <Filename>
:FILE:SAVE:{AGForm|ASCii|FLOat|MEASure|
SETup|WAVeform}:NAME?
```

Example
```
(An example for waveform data is given
below.)
:FILE:SAVE:WAVEFORM:NAME "CASE1"
:FILE:SAVE:WAVEFORM:NAME?→:FILE:SAVE:
WAVEFORM:NAME "CASE1"
```

Description The file name set is also used when a copy operation is performed.

:FILE:SAVE:{ASCii|FLOat|WAVeform}:RANGe

Function Sets/queries the range of the waveform data to save.

Syntax
```
:FILE:SAVE:{ASCii|FLOat|WAVeform}:
RANGe {ALL|MAIN|Z1|Z2}
:FILE:SAVE:{ASCii|FLOat|WAVeform}:RANGe?
```

Example
```
:FILE:SAVE:WAVEFORM:RANGE Z1
:FILE:SAVE:WAVEFORM:RANGE?→:FILE:SAVE:
WAVEFORM:RANGE Z1
```

:FILE:SAVE[:{AGForm|ASCii|FLOat}]:TRACe

Function Sets/queries the waveform for data (except for SETup, MEASure and WAVeform) to be saved.

Syntax
```
:FILE:SAVE[:{AGForm|ASCii|FLOat}]:
TRACe {<NRf>|MATH[<x>]|ALL}
:FILE:SAVE[:{AGForm|ASCii|FLOat}]:TRACe?
<NRf>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
<x>=1, 2
```

Example
```
FILE:SAVE:ASCII:TRACE 1
FILE:SAVE:ASCII:TRACE?→:FILE:SAVE:
ASCII:TRACE 1
```

Description
- The same waveform is saved, irrespective of the selected data type.
- "ALL" can be specified only for ":FILE:SAVE:ASCii:TRACe."

:FILE:TYPE?

Function Queries the current medium type.

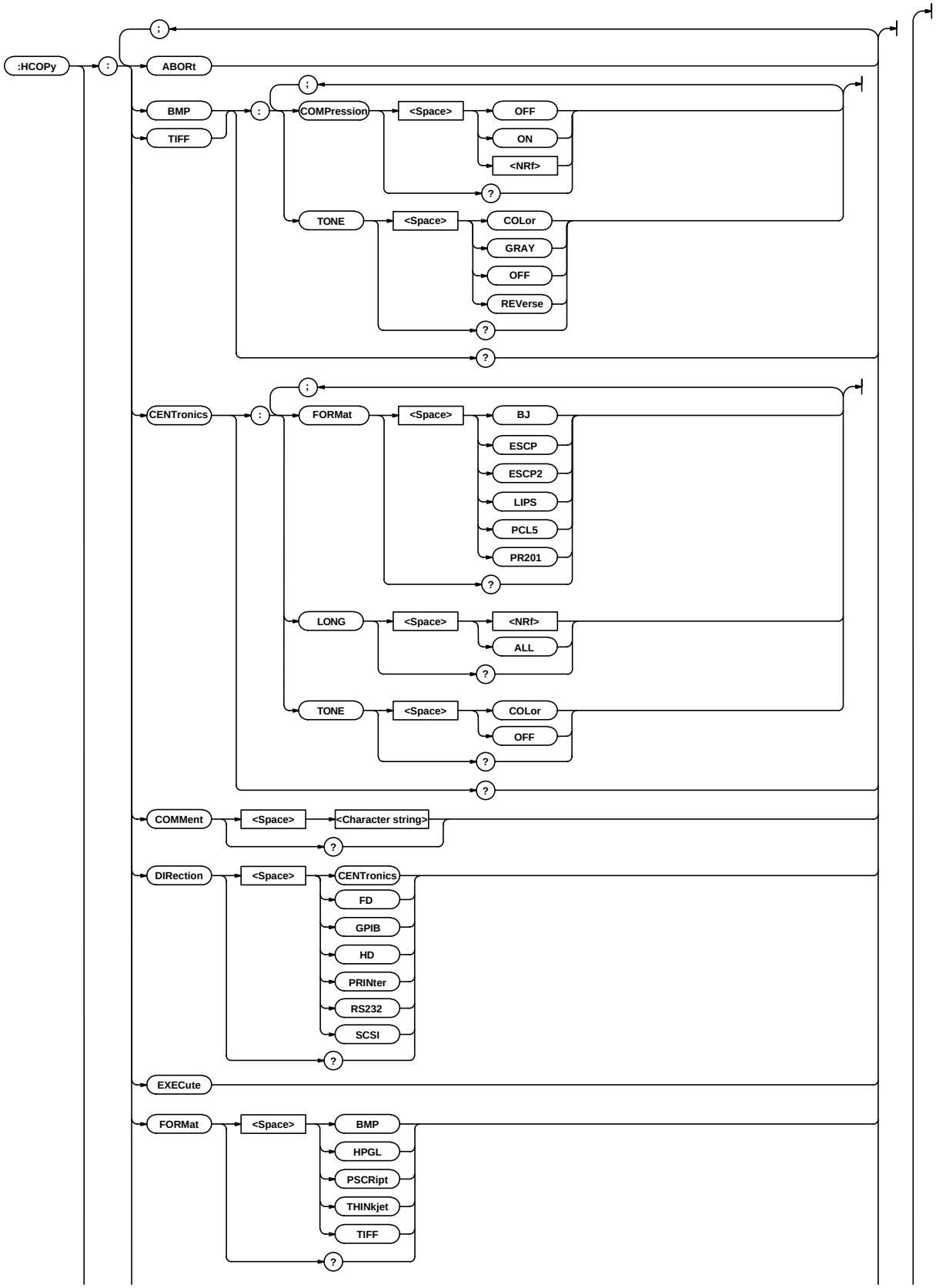
Syntax
```
:FILE:TYPE?
```

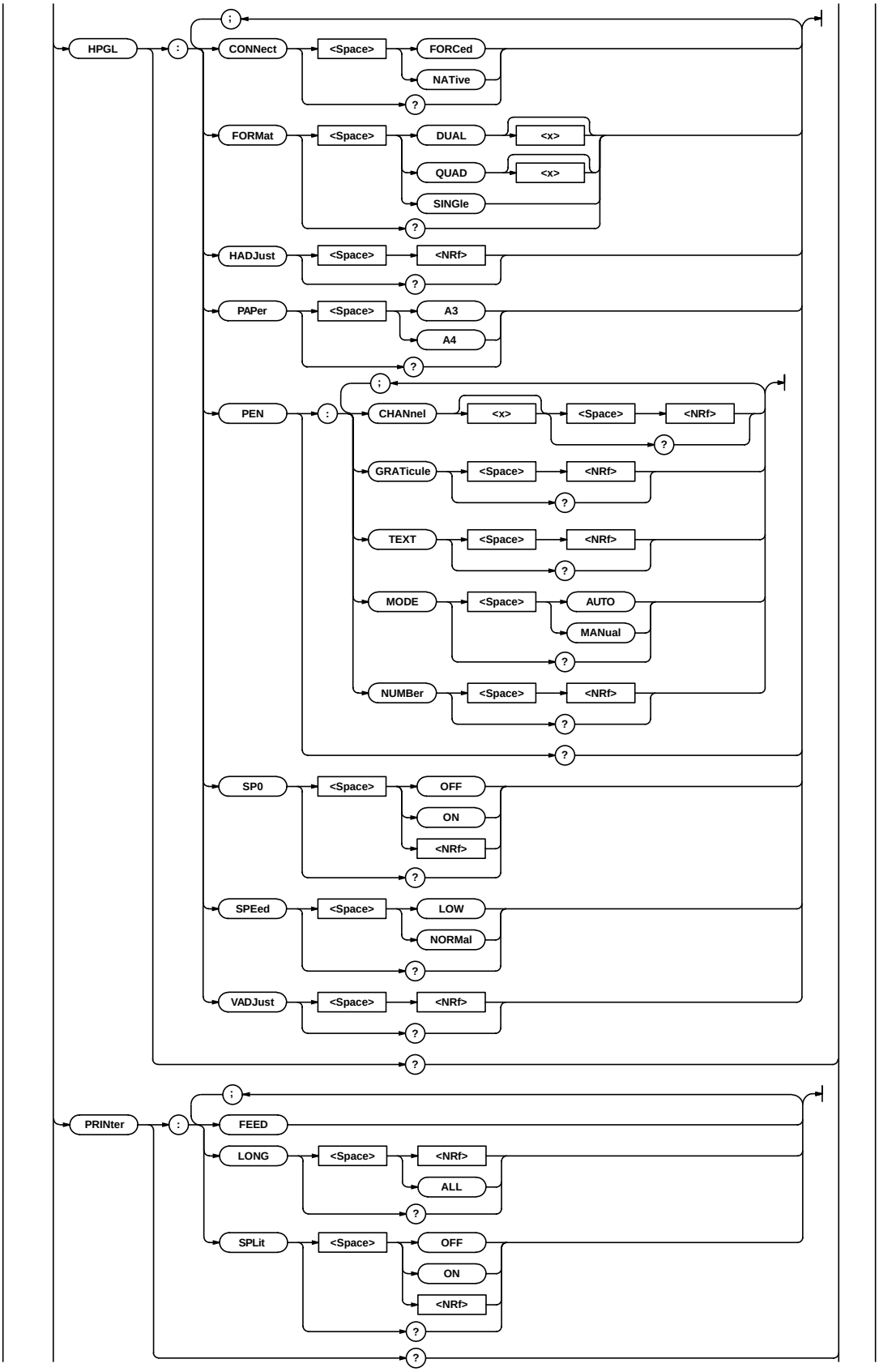
Example
```
:FILE:TYPE?→:FILE:TYPE "2HD(1.44MB)"
```

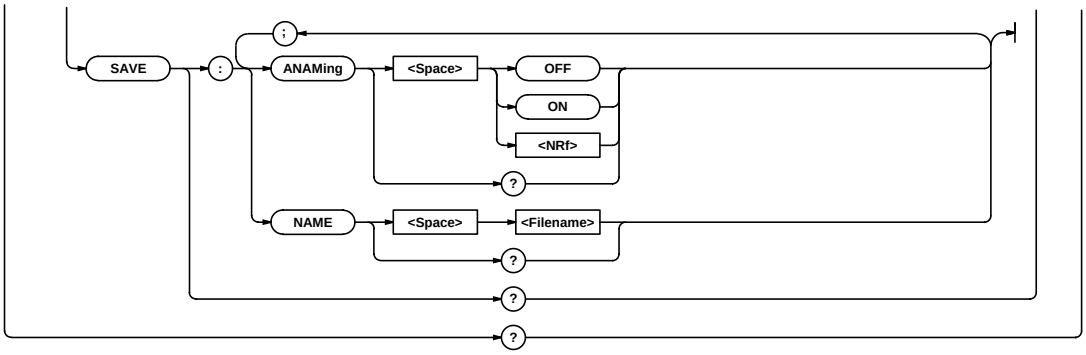
Description
- If medium is a floppy disk, this command returns the format type.
- You can use the "FILE:FORMat:TYPE?" query to find the format type the oscilloscope uses when formatting.

4.12 HCOPy Group

The commands in the HCOPy group are used to make settings and queries about screen image data output, for example, output to the optional built-in printer. You can make the same settings that you can make by pressing the SHIFT key, then pressing the COPY key.




HPGL
:
;
CONNect
<Space>
FORCed
NATive
?
FORMat
<Space>
DUAL
<x>
QUAD
<x>
SINGle
?
HADJust
<Space>
<NRf>
?
PAPer
<Space>
A3
A4
?
PEN
:
;
CHANnel
<x>
<Space>
<NRf>
?
GRATicule
<Space>
<NRf>
?
TEXT
<Space>
<NRf>
?
MODE
<Space>
AUTO
MANual
?
NUMBer
<Space>
<NRf>
?
?
SP0
<Space>
OFF
ON
<NRf>
?
SPEed
<Space>
LOW
NORMal
?
VADJust
<Space>
<NRf>
?
?
PRINter
:
;
FEED
LONG
<Space>
<NRf>
ALL
?
SPLit
<Space>
OFF
ON
<NRf>
?
?



:HCOPy? (Hard COPY)

Function Queries all settings relating to output of screen image data.
Syntax :HCOPy?
Example :HCOPY?→:HCOPY:DIRECTION PRINTER;
COMMENT "THIS IS TEST. ";
PRINTER:LONG 1;SPLIT 0

:HCOPy:ABORt

Function Aborts data output and paper feed. This is an overlap command.
Syntax :HCOPy:ABORt
Example :HCOPY:ABORT

:HCOPy:{BMP|TIFF}?

Function Queries all the settings relating to BMP (TIFF) format.
Syntax :HCOPy:{BMP|TIFF}?
Example (An example for BMP is given below.)
:HCOPY:BMP?→:HCOPY:BMP:TONE OFF

:HCOPy:{BMP|TIFF}:COMPression

Function Sets/queries use of screen image data compression relating to BMP (or TIFF) format.
Syntax :HCOPy:{BMP|TIFF}:COMPression {<Boolean>}
:HCOPy:{BMP|TIFF}:COMPression?
Example (An example for BMP is given below.)
:HCOPy:BMP:COMPression ON
:HCOPy:BMP:COMPression?→:HCOPY:BMP:COMPRESSION 1
Description This setting cannot be used when "HCOPY:BMP:TONE OFF" has been set.

:HCOPy:{BMP|TIFF}:TONE

Function Sets/queries the color tone for BMP (TIFF) format.
Syntax :HCOPy:{BMP|TIFF}:TONE {COLor|GRAY|OFF}
:HCOPy:{BMP|TIFF}:TONE?
Example (An example for BMP is given below.)
:HCOPY:BMP:TONE COLOR
:HCOPY:BMP:TONE?→:HCOPY:BMP:TONE COLOR

:HCOPy:CENTronics?

Function Queries all setting values relating to the output to the external printer.
Syntax :HCOPy:CENTronics?
Example :HCOPY:CENTRONICS?→:HCOPY:CENTRONICS:FORMAT ESCP;LONG 1;TONE OFF

:HCOPy:CENTronics:FORMat

Function Sets/queries the command type to output to the external printer.
Syntax :HCOPy:CENTronics:FORMat {BJ|ESCP|ESCP2|LIPS|PCL5|PR201}
:HCOPy:CENTronics:FORMat?
Example :HCOPY:CENTRONICS:FORMAT ESCP
:HCOPY:CENTRONICS:FORMAT?→:HCOPY:CENTRONICS:FORMAT ESCP

:HCOPy:CENTronics:LONG

Function Sets/queries the magnification ratio (long copy) of the external printer output.
Syntax :HCOPy:CENTronics:LONG {ALL|<NRf>}
:HCOPy:CENTronics:LONG?
<NRf>=1,2,5,10,20,50
Example :HCOPY:CENTRONICS:LONG 1
:HCOPY:CENTRONICS:LONG?→:HCOPY:CENTRONICS:LONG 1
Description Depending on the T/div setting, selecting 2 results in a scaling factor of 2.5. Similarly, 5 becomes 4; 20 becomes 25; and 50 becomes 40.

:HCOPy:CENTronics:TONE

Function Sets/queries the half tone setting for the external printer.
Syntax :HCOPy:CENTronics:TONE {COLor|OFF}
:HCOPy:CENTronics:TONE?
Example :HCOPY:CENTRONICS:TONE COLOR
:HCOPY:CENTRONICS:TONE?→:HCOPY:CENTRONICS:TONE COLOR

:HCOPy:COMMent

Function Sets/queries the comment to be displayed at the top of the screen.
Syntax :HCOPy:COMMent <Character string>
:HCOPy:COMMent?
<Character string>=Within 20 characters
Example :HCOPY:COMMENT "THIS IS TEST."
:HCOPY:COMMENT?→:HCOPY:COMMENT "THIS IS TEST."

:HCOPy:DIRection

Function Sets/queries the data output destination.
Syntax :HCOPy:DIRection {FD|HD|PRINter|RS232|SCSI}
:HCOPy:DIRection?
Example :HCOPY:DIRECTION FD
:HCOPY:DIRECTION?→:HCOPY:DIRECTION FD
Description HD is available as an option.

:HCOPy:EXECute

Function Executes data output. This is an overlap command.
Syntax :HCOPy:EXECute
Example :HCOPY:EXECute
Description An error will occur, if the data output destination is GPIB or RS232.

:HCOPy:FORMat

Function Sets/queries the format for screen image data output.
Syntax :HCOPy:FORMat {BMP|HPGL|PSCRipt|THINkjet|TIFF}
:HCOPy:FORMat?
Example :HCOPY:FORMAT BMP
:HCOPY:FORMAT?→:HCOPY:FORMAT BMP
Description This command is meaningless if "HCOPy:DIRection" is "PRINter" or "CENTronics."

:HCOPy:HPGL?

Function Queries all settings relating to output to a HP-GL plotter.
Syntax :HCOPy:HPGL?
Example :HCOPY:HPGL?→:HCOPY:HPGL:CONNECT FORCED;FORMAT SINGLE;HADJUST 0;PAPER A4;SP0 1;SPEED NORMAL;VADJUST 0;PEN:MODE AUTO;NUMBER 4

:HCOPy:HPGL:CONNect

Function Selects whether , or queries to connect the measured points in a graph the current setting.
Syntax :HCOPy:HPGL:CONNect {FORCed|NATive}
:HCOPy:HPGL:CONNect?
Example :HCOPY:HPGL:CONNECT FORCED
:HCOPY:HPGL:CONNECT?→:HCOPY:HPGL:CONNECT FORCED

:HCOPY:HPGL:FORMat

Function Sets/queries the plot size.
Syntax :HCOPy:HPGL:FORMat {SINGle|DUAL<x>|QUAD<x>}
:HCOPy:HPGL:FORMat?
<x> (DUAL)=1(1/2) to 2(2/2)
<x> (QUAD)=1(1/4) to 4(4/4)
Example :HCOPY:HPGL:FORMAT SINGLE
:HCOPY:HPGL:FORMAT?→:HCOPY:HPGL:FORMAT SINGLE

:HCOPy:HPGL:HADJust (Horizontal ADJUST)

Function Sets/queries horizontal adjustment of the plotting position (in mm).
Syntax :HCOPy:HPGL:HADJust {<NRf>}
:HCOPy:HPGL:HADJust?
<NRf>=-99 to 99
Example :HCOPY:HPGL:HADJUST 0
:HCOPY:HPGL:HADJUST?→:HCOPY:HPGL:HADJUST 0

:HCOPy:HPGL:PAPer

Function Sets/queries the paper size.
Syntax :HCOPy:HPGL:PAPer {A3|A4}
:HCOPy:HPGL:PAPer?
Example :HCOPY:HPGL:PAPER A4
:HCOPY:HPGL:PAPER?→:HCOPY:HPGL:PAPER A4
Description This command is not used to load paper into the plotter or to inquire about the size of the paper loaded in the plotter. It is used to set or inquire about the paper size, used when the instrument uses a HP-GL command.

:HCOPy:HPGL:PEN?

Function Queries all pen assignment settings.
Syntax :HCOPy:HPGL:PEN?
Example :HCOPY:HPGL:PEN?→:HCOPY:HPGL:PEN:MODE AUTO ;NUMBER 4

:HCOPy:HPGL:PEN:MODE

Function Sets/queries the pen assignment method.
Syntax :HCOPy:HPGL:PEN:MODE {AUTO|MANual}
:HCOPy:HPGL:PEN:MODE?
Example :HCOPY:HPGL:PEN:MODE AUTO
:HCOPY:HPGL:PEN:MODE?→:HCOPY:HPGL:PEN:MODE AUTO

:HCOPy:HPGL:PEN:NUMBer

Function Sets/queries the number of pens referred to under automatic pen assignment.
Syntax :HCOPy:HPGL:PEN:NUMBer {<NRf>}
:HCOPy:HPGL:PEN:NUMBer?
<NRf>=1 to 12
Example :HCOPY:HPGL:PEN:NUMBER 4
:HCOPY:HPGL:PEN:NUMBER?→:HCOPY:HPGL:PEN:NUMBER 4
Description This command is not used to query the actual number of plotter pens available. It is used to set or query the maximum number of pens available for automatic pen assignment.

:HCOPy:HPGL:PEN:{CHANnel<x>|GRATicule|MATH<x>|TEXT}

Function Sets/queries the pen for the specified item when pen assignment is made manually.

Syntax
```
:HCOPy:HPGL:PEN:{CHANnel<x>|GRATicule|TEXT} {<NRf>}
:HCOPy:HPGL:PEN:{CHANnel<x>|GRATicule|TEXT}?
<x> (CHANnel)=1 to 8
<x> (MATH)=1, 2
<NRf>=1 to 12
```

Example
```
:HCOPY:HPGL:PEN:CHANNEL1 1
:HCOPY:HPGL:PEN:CHANNEL1?→:HCOPY:HPGL:PEN:CHANNEL1 1
```

:HCOPy:HPGL:SP0

Function Selects/queries whether SP0 is appended.

Syntax
```
:HCOPy:HPGL:SP0 {<Boolean>}
:HCOPy:HPGL:SP0?
```

Example
```
:HCOPY:HPGL:SP0 ON
:HCOPY:HPGL:SP0→:HCOPY:HPGL:SP0 1
```

:HCOPy:HPGL:SPEed

Function Sets/queries the pen speed.

Syntax
```
:HCOPy:HPGL:SPEed {LOW|NORMal}
:HCOPy:HPGL:SPEed?
```

Example
```
:HCOPY:HPGL:SPEED NORMAL
:HCOPY:HPGL:SPEED?→:HCOPY:HPGL:SPEED NORMAL
```

:HCOPy:HPGL:VADJust (Vertical ADJUST)

Function Sets/queries the vertical adjustment of the plotting position (in mm).

Syntax
```
:HCOPy:HPGL:VADJust {<NRf>}
:HCOPy:HPGL:VADJust?
<NRf>=-99 to 99
```

Example
```
:HCOPY:HPGL:VADJUST 0
:HCOPY:HPGL:VADJUST?→:HCOPY:HPGL:VADJUST 0
```

:HCOPy:PRINter?

Function Queries all settings relating to the built-in printer.

Syntax
```
:HCOPy:PRINter?
```

Example
```
:HCOPY:PRINTER?→:HCOPY:PRINTER:LONG 1;SPLIT 0
```

:HCOPy:PRINter:FEED

Function Feeds a roll chart into the built-in printer. This is an overlap command.

Syntax
```
:HCOPy:PRINter:FEED
```

Example
```
:HCOPY:PRINTER:FEED
```

:HCOPy:PRINter:LONG

Function Sets/queries the output magnification ratio (long copy) for the built-in printer.

Syntax
```
:HCOPy:PRINter:LONG {ALL|<NRf>}
:HCOPy:PRINter:LONG?
<NRf>=1, 2, 5, 10, 20, 50
```

Example
```
:HCOPY:PRINTER:LONG 1
:HCOPY:PRINTER:LONG?→:HCOPY:PRINTER:LONG 1
```

Description If the scaling factor is 1, the screen display is output as is. Depending on the T/div setting, selecting 2 results in a scaling factor of 2.5. Similarly, 5 becomes 4; 20 becomes 25; and 50 becomes 40.

:HCOPy:PRINTer:SPLit

Function Sets/queries the ON/OFF condition of the split print to the internal printer.

Syntax
```
:HCOPy:PRINter:SPLit {<Bodean>}
:HCOPy:PRINter:SPLit?
```

Example
```
:HCOPY:PRINTER:SPLIT ON
:HCOPY:PRINTER:SPLIT?→:HCOPY:PRINTER:SPLIT 1
```

:HCOPy:SAVE?

Function Queries all file saving settings.

Syntax
```
:HCOPy:SAVE?
```

Example
```
:HCOPY:SAVE?→:HCOPY:SAVE:ANAMING 0;NAME "CASE1"
```

:HCOPy:SAVE:ANAMing

Function Enables/disables/queries automatic naming of output files.

Syntax
```
:HCOPy:ANAMing {<Boolean>}
:HCOPy:SAVE:ANAMing?
```

Example
```
:HCOPY:ANAMING ON
:HCOPY:SAVE:ANAMING?→:HCOPY:SAVE:ANAMING 1
```

:HCOPy:SAVE:NAME

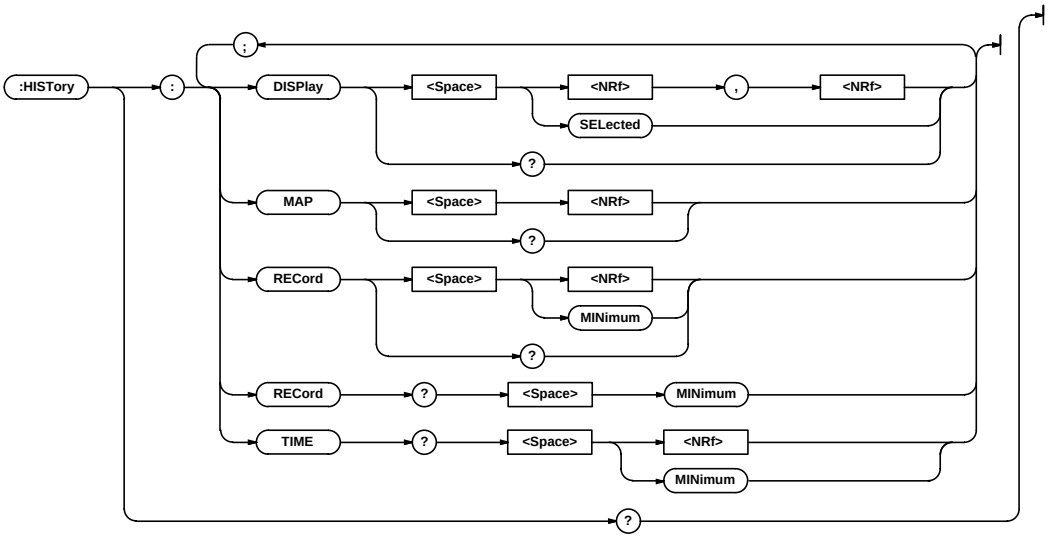
Function Sets/queries the name of the file to be created.

Syntax
```
:HCOPy:SAVE:NAME <Filename>
:HCOPy:SAVE:NAME?
```

Example
```
:HCOPY:SAVE:NAME "DISP_1"
:HCOPY:SAVE:NAME?→:HCOPY:SAVE:NAME "DISP_1"
```

4.13 HISTory Group

The commands in the HISTory group are used to make settings and queries about recalling of data from the history memory. You can make the same settings that you can make using the HISTORY key on the front panel.



:HISTory?

Function Queries all settings relating to the history memory function.

Syntax `:HISTory?`

Example `:HISTORY?→:HISTORY:DISPLAY SELECTED;RECORD 0;MAP 1`

:HISTory:DISPlay

Function Selects whether to display one record or accumulated records, or queries the current setting.

Syntax `:HISTory:DISPlay {<NRf>,<NRf>|SELected}`
`:HISTory:DISPlay?`
`<NRf>=0 to -999`

Example `:HISTORY:DISPLAY SELECTED`
`:HISTORY:DISPLAY?→:HISTORY:DISPLAY SELECTED`

Description Maximum number of displayable records varies according to acquisition settings and machine model. For details, refer to the User's Manual IM701830-01E.

:HISTory:MAP

Function Sets/queries the map number.

Syntax `:HISTory:MAP {<NRf>}`
`:HISTory:MAP?`
`<NRf>=1 to 10`

Example `:HISTORY:MAP 1`
`:HISTORY:MAP?→:HISTORY:MAP 1`

:HISTory:RECord

Function Sets/queries the target record.

Syntax `:HISTory:RECord {<NRf>|MINimum}`
`:HISTory:RECord?`
`<NRf>=0 to -999`

Example `:HISTORY:RECORD 0`
`:HISTORY:RECORD?→:HISTORY:RECORD 0`

Description
- The oldest record is selected when "MINimum" is selected.
- Available record setting varies according to machine model, extension memory, and interleave mode. For details, refer to the User's Manual IM701830-01E.

:HISTory:RECord? MINimum

Function Queries the lowest record No.

Syntax `:HISTory:RECord? MINimum`

Example `:HISTORY:RECORD? MINIMUM→-81`

Description The lowest record varies according to machine model, record length, and interleave mode. For details, refer to the User's Manual IM701830-01E.

:HISTory:TIME?

Function Queries the trigger time of the specified record number.

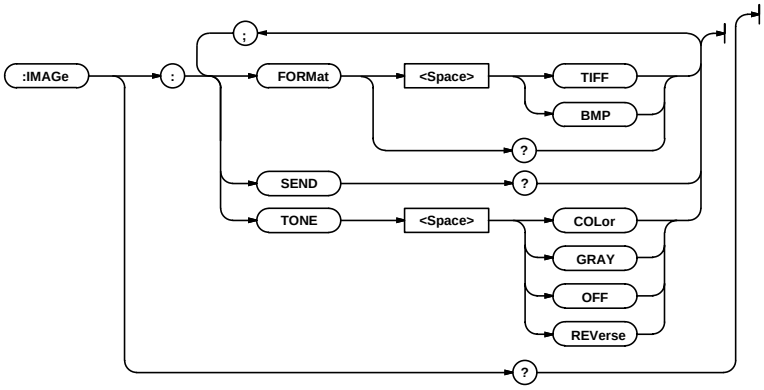
Syntax `:HISTory:TIME? {<NRf>|MINimum}`
`<NRf>=0 to -999`

Example `:HISTORY:TIME? -1→"-001 12:34:56"`

Description
- The oldest record is selected when "MINimum" is selected.
- The record that can be specified varies depending on the record length or the extension memory setting. For details, refer to the User's Manual IM701830-01E.

4.14 IMAGe Group

Use this group to set or query the screen image data output settings. There are no corresponding front-panel keys for these operations.



:IMAGe?

Function	Queries all screen image data output settings.
Syntax	IMAGe?
Example	IMAGE?→:IMAGE:FORMAT TIFF

:IMAGe:FORMat

Function	Sets/queries the screen image data output format.
Syntax	:IMAGe:FORMat {TIFF|BMP} :IMAGe:FORMat?
Example	:IMAGE:FORMAT TIFF :IMAGE:FORMAT?→:IMAGE:FORMAT TIFF

:IMAGe:SEND?

Function	Queries screen image data.
Syntax	:IMAGe:SEND?
Example	:IMAGE:SEND?→#6 (number of bytes[6-digit value]) (Data byte string) (Block data)
Description	Number of bytes in <block data> is {(2+6+Number of data+1(delimiter)}. For information about block data, refer to page 3-6.

:IMAGe:TONE

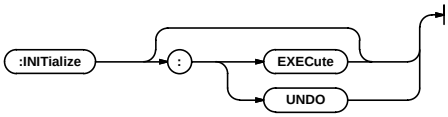
Function	Sets/queries the color tone of the BMP (TIFF) format of the screen image data to output.
Syntax	:IMAGe:TONE {COLor|GRAY|OFF|REVerse} :IMAGe:TONE?
Example	:IMAGE:TONE COLOR :IMAGE:TONE?→:IMAGE:TONE COLOR

4.15 INITialize Group

The commands in the INITialize group are related to initi

of the SETUP key on the front panel.

You can make the same settings using the Initialize menu



:INITialize[:EXECute]

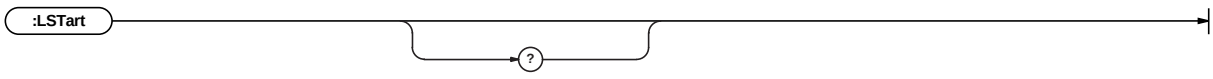
Function	Executes initializing.
Syntax	:INITialize:EXECute
Example	:INITIALIZE:EXECUTE

:INITialize:UNDO

Function	Nullifies initializing.
Syntax	:INITialize:UNDO
Example	:INITIALIZE:UNDO

4.16 LSTart Group

LStart group is the group to execute the log start. This is the same as executing the Log Start in the ACQ menu from the front panel.



:LSTart (Log START)

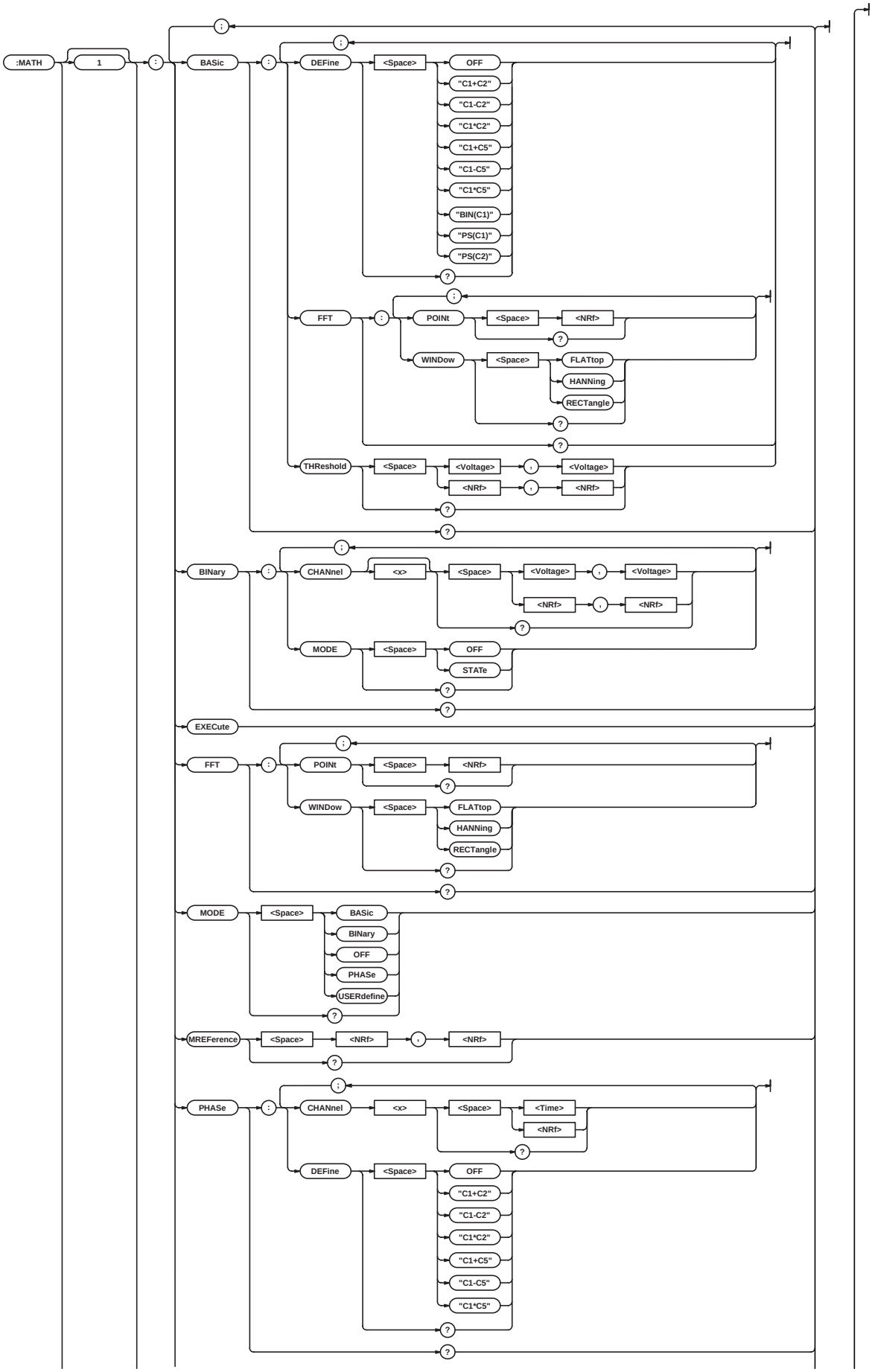
Function	Executes the log start.
Syntax	:LSTart
Example	:LSATART

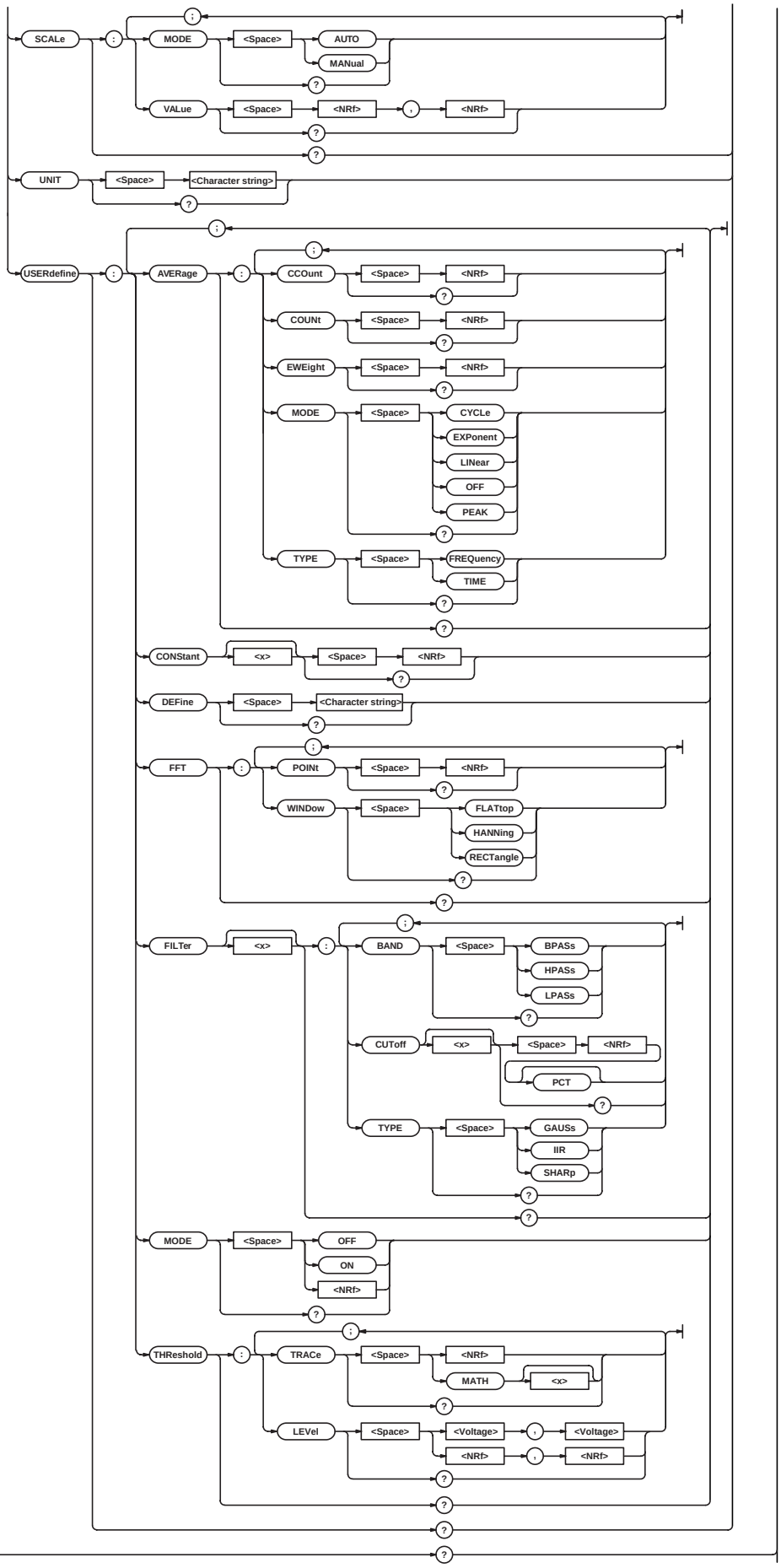
:LSTart?

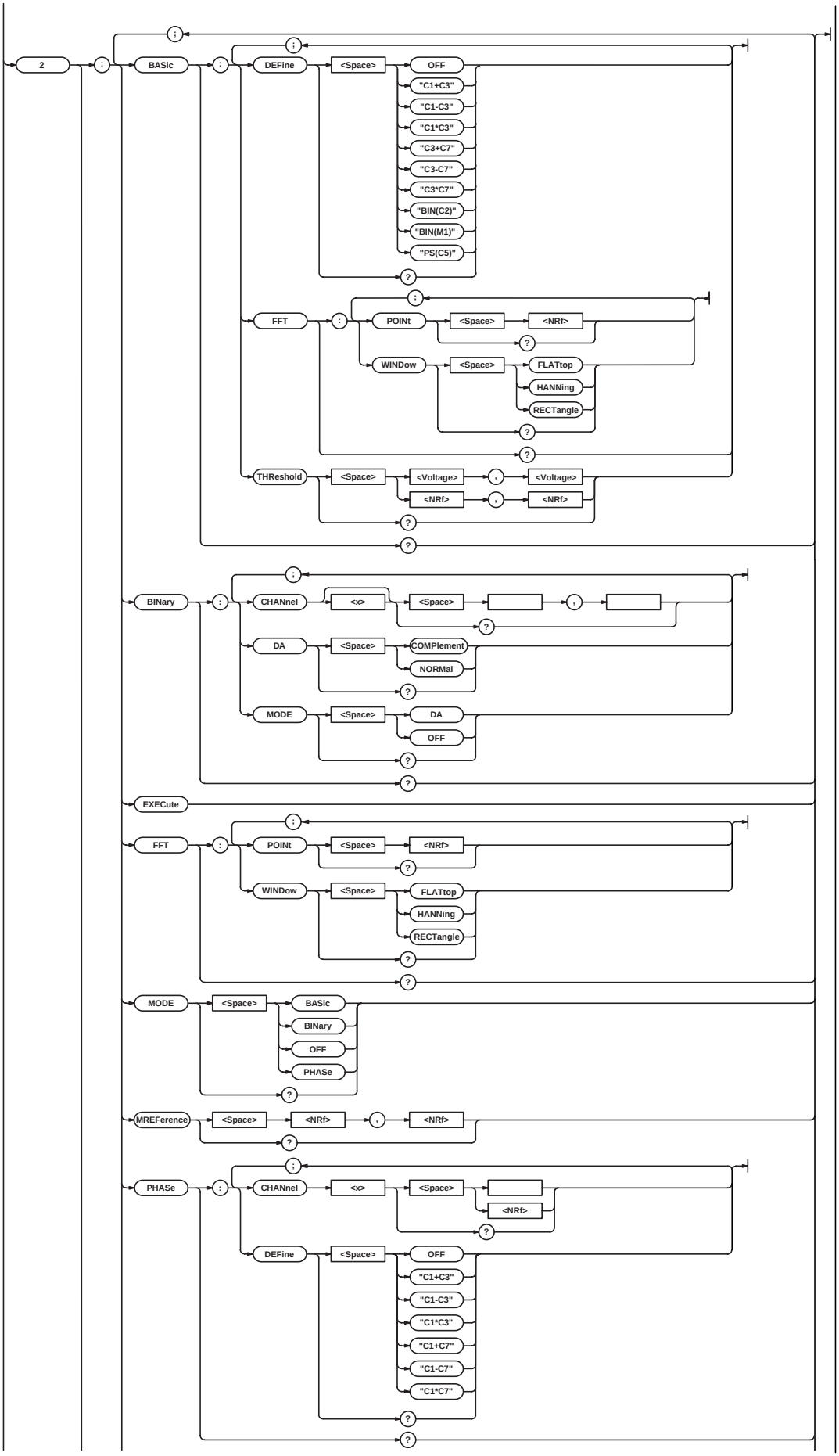
Function	Executes the log start and waits for the completion.
Syntax	:LSTart?
Example	:LSTART?→0
Description	0 is always returned when it is complete.

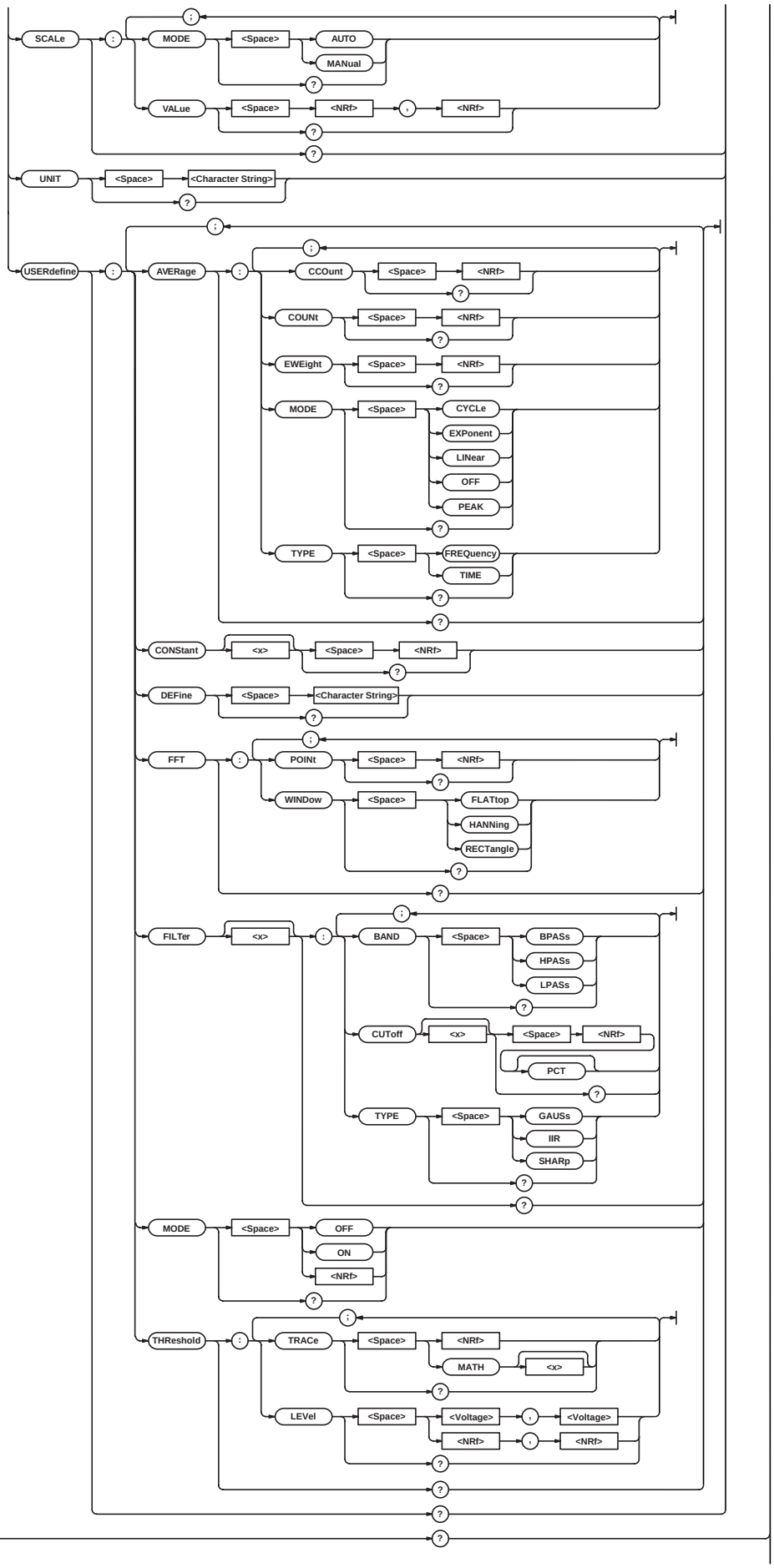
4.17 MATH Group

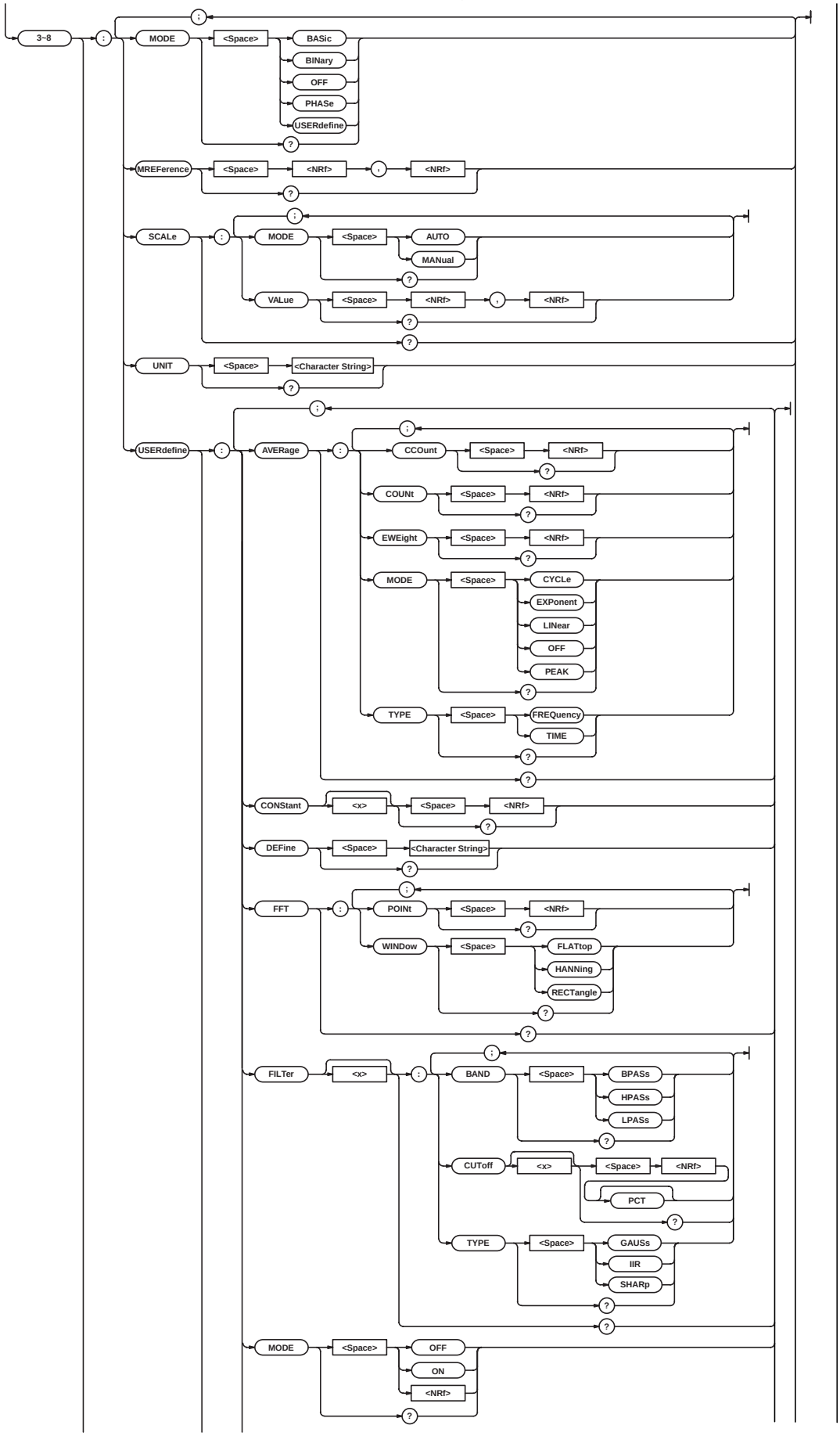
The commands in the MATH group are used to make settings and queries about computation. You can make the same settings that you can make using the MATH key on the front panel.

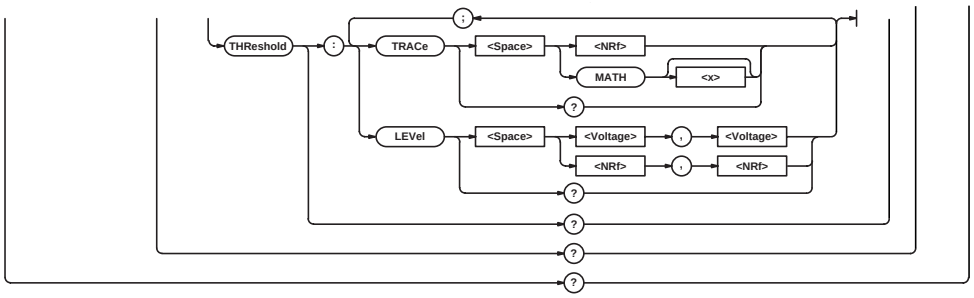



SCALe
MODE
<Space>
AUTO
MANual
VALue
<NRf>
UNIT
<Character string>
USERdefine
AVERage
CCOunt
COUNt
EWEight
CYCLe
EXPonent
LINear
OFF
PEAK
TYPE
FREQuency
TIME
CONStant
<x>
DEFine
FFT
POINt
WINDow
FLATtop
HANNing
RECTangle
FILTer
BAND
BPASs
HPASs
LPASs
CUToff
PCT
GAUSs
IIR
SHARp
THReshold
TRACe
LEVel
<Voltage>
ON


2
BASic
DEFine
<Space>
OFF
"C1+C3"
"C1-C3"
"C1*C3"
"C3+C7"
"C3-C7"
"C3*C7"
"BIN(C2)"
"BIN(M1)"
"PS(C5)"
FFT
POINt
<Space>
<NRf>
WINDow
<Space>
FLATtop
HANNing
RECTangle
THReshold
<Space>
<Voltage>
,
<Voltage>
<NRf>
,
<NRf>
BINary
CHANnel
<x>
<Space>
,
DA
<Space>
COMPlement
NORMal
MODE
<Space>
DA
OFF
EXECute
FFT
POINt
<Space>
<NRf>
WINDow
<Space>
FLATtop
HANNing
RECTangle
MODE
<Space>
BASic
BINary
OFF
PHASe
MREFerence
<Space>
<NRf>
,
<NRf>
PHASe
CHANnel
<x>
<Space>
<NRf>
DEFine
<Space>
OFF
"C1+C3"
"C1-C3"
"C1*C3"
"C1+C7"
"C1-C7"
"C1*C7"


SCALe
MODE
<Space>
AUTO
MANual
VALue
<Space>
<NRf>
,
<NRf>
UNIT
<Space>
<Character String>
USERdefine
AVERage
CCOunt
<Space>
<NRf>
COUNt
<Space>
<NRf>
EWEight
<Space>
<NRf>
MODE
<Space>
CYCLe
EXPonent
LINear
OFF
PEAK
TYPE
<Space>
FREQuency
TIME
CONStant
<x>
<Space>
<NRf>
DEFine
<Space>
<Character String>
FFT
POINt
<Space>
<NRf>
WINDow
<Space>
FLATtop
HANNing
RECTangle
FILTer
<x>
BAND
<Space>
BPASs
HPASs
LPASs
CUToff
<x>
<Space>
<NRf>
PCT
TYPE
<Space>
GAUSs
IIR
SHARp
MODE
<Space>
OFF
ON
<NRf>
THReshold
TRACe
<Space>
<NRf>
MATH
<x>
LEVel
<Space>
<Voltage>
,
<Voltage>
<NRf>
,
<NRf>


3–8
:
;
MODE
<Space>
BASic
BINary
OFF
PHASe
USERdefine
?
MREFerence
<Space>
<NRf>
,
<NRf>
?
SCALe
:
;
MODE
<Space>
AUTO
MANual
?
VALue
<Space>
<NRf>
,
<NRf>
?
?
UNIT
<Space>
<Character String>
?
USERdefine
:
;
AVERage
:
;
CCOunt
<Space>
<NRf>
?
COUNt
<Space>
<NRf>
?
EWEight
<Space>
<NRf>
?
MODE
<Space>
CYCLe
EXPonent
LINear
OFF
PEAK
?
TYPE
<Space>
FREQuency
TIME
?
?
CONStant
<x>
<Space>
<NRf>
?
DEFine
<Space>
<Character String>
?
FFT
:
;
POINt
<Space>
<NRf>
?
WINDow
<Space>
FLATtop
HANNing
RECTangle
?
?
FILTer
<x>
:
;
BAND
<Space>
BPASs
HPASs
LPASs
?
CUToff
<x>
<Space>
<NRf>
PCT
?
TYPE
<Space>
GAUSs
IIR
SHARp
?
?
MODE
<Space>
OFF
ON
<NRf>
?



:MATH<x>?

Function Queries all settings relating to computation.
Syntax :MATH<x>?
<x>=1, 2
Example :MATH1?→:MATH1:MODE BASIC;BASIC:
DEFINE "C1+C2"

:MATH<x>:BASic?

Function Queries all settings relating to basic mode (addition and subtraction between channels, binary computation for the specified channel).
Syntax :MATH<x>:BASic?
<x>=1, 2
Example :MATH1:BASIC?→:MATH1:BASIC:
DEFINE "C1+C2"

:MATH[1]:BASic:DEFine

Function Sets/queries the equation for Math1 in basic mode.
Syntax :MATH[1]:BASic:DEFine {OFF|"C1+C2"|
"C1-C2"|"C1*C2"|"C1+C5"|"C1-C5"|"C1*C5"|
"BIN(C1)"|"PS(C1)"|"PS(C2)"}
:MATH[1]:BASic:DEFine?
Example :MATH1:BASIC:DEFINE "C1+C2"
:MATH1:BASIC:DEFINE?→:MATH1:BASIC:DEFINE
"C1+C2"

:MATH2:BASic:DEFine

Function Sets/queries the equation for Math2 in basic mode.
Syntax :MATH2:BASic:DEFine {OFF|"C1+C3"|"C1-C3"|
"C1*C3"|"C3+C7"|"C3-C7"|"C3*C7"|
"BIN(C2)"|"BIN(M1)"|"PS(C5)"}
:MATH2:BASic:DEFine?
Example :MATH2:BASIC:DEFINE "BIN(C2)"
:MATH2:BASIC:DEFINE?→:MATH2:BASIC:
DEFINE "BIN(C2)"

:MATH<x>:BASic:FFT?

Function Queries all FFT settings.
Syntax :MATH<x>:BASic:FFT?
<x>=1, 2
Example :MATH1:BASIC:FFT?→:MATH1:BASIC:FFT:
POINT 1000;WINDOW RECTANGLE

:MATH<x>:BASic:FFT:POINt

Function Sets/queries the number of points for the FFT.
Syntax :MATH<x>:BASic:FFT:POINt {<NRf>}
:MATH<x>:BASic:FFT:POINt?
<x>=1, 2
<NRf>=1000, 2000, 10000
Example :MATH1:BASIC:FFT:POINT 1000
:MATH1:BASIC:FFT:POINT?→:MATH1:FFT:
POINT 1000

:MATH<x>:BASic:FFT:WINDow

Function Sets/queries the window function of the FFT.
Syntax :MATH<x>:BASic:FFT:WINDow {FLATtop|
HANNing|RECTangle}
:MATH<x>:BASic:FFT:WINDow?
<x>=1, 2
Example :MATH1:BASIC:FFT:WINDOW HANNING
:MATH1:BASIC:FFT:WINDOW?→:MATH1:
FFT:WINDOW HANNING

:MATH<x>:BASic:THReshold

Function Sets/queries the threshold level for binary computation in basic mode. Both MATH<x>:MODE BASic and MATH<x>:BASic:DEFine "BIN(□)" must be selected, otherwise the command cannot be executed.
Syntax :MATH<x>:BASic:THReshold {<Voltage>,
<Voltage>|<NRf>,<NRf>}
:MATH<x>:BASic:THReshold?
<x>=1, 2
<Voltage>=Within 8 divisions
<NRf>=-1E+30 to 1E+30 (for BIN(M1))
=-3000 to 3000 (for °C with
tempareture module)
=-20000 to 20000 (×10^{-6} strain, for
strain module)
Example :MATH1:BASIC:THRESHOLD 5V,-5V
:MATH1:BASIC:THRESHOLD?→:MATH1:BASIC:
THRESHOLD 5.0E+00,-5.0E+00
Description If the specified channel is a temperature/strain module, or ":MATH2:BASic:DEFine "BIN(M1)"," requires/returns <NRf> arguments.

:MATH<x>:BINary?

Function Queries all settings relating to binary mode (simultaneous binary computation for all channels and D/A conversion of the results). "MATH<x>:MODE BINary" must be selected, otherwise this command is meaningless.

Syntax
```
:MATH<x>:BINary?
<x>=1, 2
```

Example
```
:MATH1:BINARY?→:MATH1:BINARY:
MODE STATE;CHANNEL1 0.000E+03,0.000E+03;
CHANNEL2 0.000E+03,0.000E+03;
CHANNEL3 0.000E+03,0.000E+03;
CHANNEL4 0.000E+03,0.000E+03
```

:MATH<x>:BINary:CHANnel<x>

Function Sets/queries the threshold level for the specified channel for simultaneous binary computation in binary mode.

Syntax
```
:MATH<x>:BINary:CHANnel<x> {<Voltage>,
<Voltage>|<NRf>,<NRf>}
:MATH<x>:BINary:CHANnel<x>?
<x>(MATH)=1, 2
<x>(CHANnel)=1 to 8
<Voltage>=Within 8 divisions(in steps of
          0.1div)
<NRf>=-3000 to 3000 (for °C with
      tempareture module)
     =-20000 to 20000 (×10^-6 strain, for
      strain module)
```

Example
```
:MATH1:BINARY:CHANNEL1 0V,0V
:MATH1:BINARY:CHANNEL1?→:MATH1:BINARY:
CHANNEL1 0.000E+03,0.000E+03
```

Description When the specified channel is a temperature/strain module, set and query are done with <NRf>.

:MATH2:BINary:DA

Function Sets/queries the D/A conversion method for binary mode.

Syntax
```
:MATH2:BINary:DA {COMPlement|NORMal}
:MATH2:BINary:DA?
```

Example
```
:MATH2:BINARY:DA NORMAL
:MATH2:BINARY:DA?→:MATH2:BINARY:
DA NORMAL
```

:MATH[1]:BINary:MODE

Function Sets/queries the computation mode for Math1 for binary mode.

Syntax
```
:MATH[1]:BINary:MODE {OFF|STATe}
:MATH[1]:BINary:MODE?
```

Example
```
:MATH1:BINARY:MODE STATE
:MATH1:BINARY:MODE?→:MATH1:BINARY:
MODE STATE
```

:MATH2:BINary:MODE

Function Sets/queries the computation mode for Math2 for binary mode.

Syntax
```
:MATH2:BINary:MODE {DA|OFF}
:MATH2:BINary:MODE?
```

Example
```
:MATH2:BINARY:MODE DA
:MATH2:BINARY:MODE?→:MATH2:BINARY:
MODE DA
```

:MATH<x>:EXECute

Function Executes a math operation.

Syntax
```
:MATH<x>:EXECute
<x>=1 to 8
```

Example
```
:MATH1:EXECUTE
```

:MATH<x>:FFT?

Function Queries all FFT computation settings.

Syntax
```
:MATH<x>:FFT?
<x>=1, 2
```

Example
```
:MATH1:FFT?→:MATH1:FFT:POINT 1000;
WINDOW RECTANGLE
```

:MATH<x>:FFT:POINt

Function Sets/queries the number of points for FFT computation.

Syntax
```
:MATH<x>:FFT:POINt {<NRf>}
:MATH<x>:FFT:POINt?
<x>=1, 2
<NRf>=1000, 2000, 10000
```

Example
```
:MATH1:FFT:POINT 1000
:MATH1:FFT:POINT?→:MATH1:FFT:POINT 1000
```

:MATH<x>:FFT:WINDow

Function Sets/queries FFT window function.

Syntax
```
:MATH<x>:FFT:WINDow {FLATtop|HANNing|
RECTangle}
:MATH<x>:FFT:WINDow?
<x>=1, 2
```

Example
```
:MATH1:FFT:WINDOW HANNING
:MATH1:FFT:WINDOW?→:MATH1:FFT:
WINDOW HANNING
```

:MATH<x>:MODE

Function Sets/queries the computation mode.

Syntax
```
:MATH<x>:MODE {BASic|BINary|OFF|PHASe|
USERdefine}
:MATH<x>:MODE?
<x>=1 to 8
```

Example
```
:MATH1:MODE BASIC
:MATH1:MODE?→:MATH1:MODE BASIC
```

Description User defined computation is an option.

:MATH<x>:MREFerence(Math REFERENCE)

Function Sets/queries the computation range.

Syntax
```
:MATH<x>:MREFerence {<NRf>,<NRf>}
:MATH<x>:MREFerence?
 <x>=1 to 8
<NRf>=-5 to 5div(in steps of [10div/
displayed record length])
```

Example
```
:MATH1:MREFERENCE -4,4
:MATH1:MREFERENCE?→:MATH1:
MREFERENCE -4.00E+00,4.00E+00
```

:MATH<x>:PHASe?

Function Queries all settings relating to phase mode (addition and subtraction between channels during phase shifting).
Syntax :MATH<x>:PHASe?
<x>=1, 2
Example :MATH1:PHASE?→:MATH1:PHASE:
DEFINE "C1+C2";CHANNEL2 0.00E-06

:MATH[1]:PHASe:CHANnel<x>

Function Sets/queries the shift for CH2/CH5 in phase mode.
Syntax :MATH[1]:PHASe:CHANnel<x> {<Time>}
:MATH[1]:PHASe:CHANnel<x>?
<x>=2,5
<Time>=Time range equivalent to -1000/SR to 1000/SR (SR=Sample rate)
Example :MATH1:PHASE:CHANNEL2 0S
:MATH1:PHASE:CHANNEL2?→:MATH1:PHASE:
CHANNEL2 0.00E-06

:MATH2:PHASe:CHANnel<x>

Function Sets/queries the shift for CH3/CH7 in phase mode.
Syntax :MATH2:PHASe:CHANnel<x> {<Time>}
:MATH2:PHASe:CHANnel<x>?
<x>=3, 7
<Time>=Time range equivalent to -1000/SR to 1000/SR (SR=Sample rate)
Example :MATH2:PHASE:CHANNEL3 0S
:MATH2:PHASE:CHANNEL3?→:MATH2:PHASE:
CHANNEL3 0.00E-06

:MATH[1]:PHASe:DEFine

Function Sets/queries the equation for Math1 in phase mode.
Syntax :MATH[1]:PHASe:DEFine {OFF|"C1+C2"|
"C1-C2"|"C1*C2"|"C1+C5"|"C1-C5"|"C1*C5"}
:MATH[1]:PHASe:DEFine?
Example :MATH1:PHASE:DEFINE "C1+C2"
:MATH1:PHASE:DEFINE?→:MATH1:PHASE:
DEFINE "C1+C2"

:MATH2:PHASe:DEFine

Function Sets/queries the equation for Math2 in phase mode.
Syntax :MATH2:PHASe:DEFine {OFF|"C1+C3"|"C1-C3"|
"C1*C3"|"C1+C7"|"C1-C7"|"C1*C7"}
:MATH2:PHASe:DEFine?
Example :MATH2:PHASE:DEFINE "C1+C3"
:MATH2:PHASE:DEFINE?→:MATH2:PHASE:
DEFINE "C1+C3"

:MATH<x>:SCALe?

Function Queries all scaling setting values.
Syntax :MATH<x>:SCALe?
<x>=1 to 8
Example :MATH:SCALE?→:MATH1:SCALE:MODE AUTO;
VALUE 1.0000E+00,-1.0000E+00

:MATH<x>:SCALe:MODE

Function Sets/queries the scaling method.
Syntax :MATH<x>:SCALe:MODE {AUTO|MANual}
:MATH<x>:SCALe:MODE?
<x>=1 to 8
Example :MATH1:SCALE:MODE AUTO
:MATH1:SCALE:MODE?→:MATH1:SCALE:
MODE AUTO

:MATH<x>:SCALe:VALue

Function Sets/queries the upper and lower limits for the manual scaling.
Syntax :MATH<x>:SCALe:VALue {<NRf>,<NRf>}
:MATH<x>:SCALe:VALue?
<x>=1 to 8
<NRf>=-1.0000E+30 to 1.0000E+30
Example :MATH1:SCALE:VALUE 1,-1
:MATH1:SCALE:VALUE?→:MATH1:SCALE:
VALUE 1.0000E+00,-1.0000E+00

:MATH<x>:UNIT

Function Sets/queries the dimensional unit appended to computation result.
Syntax :MATH<x>:UNIT {<Character string>}
:MATH<x>:UNIT?
<x>=1 to 8
<Character string>=Within 4 characters
Example :MATH1:UNIT "RPM"
:MATH1:UNIT?→:MATH1:UNIT "RPM"
Description The unit is reflected in the scale value. The unit setting has no effect on the calculation result.

:MATH<x>:USERdefine?

Function Queries all settings regarding the user defined computation.
Syntax :MATH<x>:USERdefine?
<x>=1 to 8
Example :MATH1:USERDEFINE?→:MATH1:USERDEFINE:
MODE 1;CONSTANT1 1.0000E+00;
CONSTANT2 1.0000E+00;
CONSTANT3 1.0000E+00;
CONSTANT4 1.0000E+00;
CONSTANT5 1.0000E+00;
CONSTANT6 1.0000E+00;
CONSTANT7 1.0000E+00;
CONSTANT8 1.0000E+00;
DEFINE "C1 ";
AVERAGE:MODE OFF;:MATH1:USERDEFINE:
FILTER1:CUTOFF1 10.0E+00;TYPE GAUSS;
BANDLPASS;:MATH1:USERDEFINE:FILTER2:
CUTOFF1 10.0E+00;TYPE GAUSS;BAND LPASS;:
MATH1:USERDEFINE:FFT:POINT 1000;
WINDOW RECTANGLE;:MATH1:USERDEFINE:
THRESHOLD:TRACE 1;
LEVEL 0.005E+03,-0.005E+03
Description User defined computation is an option.

:MATH<x>:USERdefine:AVERage?

Function Queries all settings regarding the average of the user defined computation.

Syntax
```
:MATH<x>:USERdefine:AVERage?
<x>=1 to 8
```

Example
```
:MATH1:USERDEFINE:AVERAGE?→:MATH1:
USERDEFINE:AVERAGE:MODE LINEAR;TYPE TIME;
CCOUNT 10;COUNT 16;EWEIGHT 16
```

:MATH<x>:USERdefine:AVERage:CCOunt

Function Sets/queries the number of cycles (cycle count) to cycle average.

Syntax
```
:MATH<x>:USERdefine:AVERage:
CCOunt {<NRf>}
:MATH<x>:USERdefine:AVERage:CCOunt?
<x>=1 to 8
<NRf>=10 to 1000
```

Example
```
:MATH1:USERDEFINE:AVERAGE:CCOUNT 100
:MATH1:USERDEFINE:AVERAGE:CCOUNT?
→:MATH1:USERDEFINE:AVERAGE:CCOUNT 100
```

:MATH<x>:USERdefine:AVERage:COUNt

Function Sets/queries the number of waveform acquisitions for linear averaging.

Syntax
```
:MATH<x>:USERdefine:AVERage:COUNt {<NRf>}
:MATH<x>:USERdefine:AVERage:COUNt?
<x>=1 to 8
<NRf>=2 to 128(in steps of 2^n)
```

Example
```
:MATH1:USERDEFINE:AVERAGE:COUNT 16
:MATH1:USERDEFINE:AVERAGE:COUNT?→:MATH1:
USERDEFINE:AVERAGE:COUNT 16
```

:MATH<x>:USERdefine:AVERage:EWEight

Function Sets/queries the attenuation constant for exponential averaging.

Syntax
```
:MATH<x>:USERdefine:AVERage:
EWEight {<NRf>}
:MATH<x>:USERdefine:AVERage:EWEight?
<x>=1 to 8
<NRf>=2 to 256(in steps of 2^n)
```

Example
```
:MATH1:USERDEFINE:AVERAGE:EWEIGHT 16
:MATH1:USERDEFINE:AVERAGE:EWEIGHT?
→:MATH1:USERDEFINE:AVERAGE:EWEIGHT 16
```

:MATH<x>:USERdefine:AVERage:MODE

Function Sets/queries the averaging mode.

Syntax
```
:MATH<x>:USERdefine:AVERage:
MODE {CYCLe|EXPonent|LINear|OFF|PEAK}
:MATH<x>:USERdefine:AVERage:MODE?
<x>=1 to 8
```

Example
```
:MATH1:USERDEFINE:AVERAGE:MODE LINEAR
:MATH1:USERDEFINE:AVERAGE:MODE?→:MATH1:
USERDEFINE:AVERAGE:MODE LINEAR
```

:MATH<x>:USERdefine:AVERage:TYPE

Function Sets/queries the domain to average.

Syntax
```
:MATH<x>:USERdefine:AVERage:
TYPE {FREQuency|TIME}
:MATH<x>:USERdefine:AVERage:TYPE?
<x>=1 to 8
```

Example
```
:MATH1:USERDEFINE:AVERAGE:TYPE TIME
:MATH1:USERDEFINE:AVERAGE:TYPE?→:MATH1:
USERDEFINE:AVERAGE:TYPE TIME
```

:MATH<x>:USERdefine:CONStant<x>

Function Sets/queries the constant for the user defined computation.

Syntax
```
:MATH<x>:USERdefine:CONStant<x> {<NRf>}
:MATH<x>:USERdefine:CONStant<x>?
<x>(MATH)=1 to 8
<x>(CONStant)=1 to 8
<NRf>=-1.0000E+30 to +1.0000E+30
```

Example
```
:MATH1:USERDEFINE:CONSTANT1 1
:MATH1:USERDEFINE:CONSTANT1?→:MATH1:
USERDEFINE:CONSTANT1 1.0000E+00
```

:MATH<x>:USERdefine:DEFine

Function Sets/queries the equation for the user defined computation.

Syntax
```
:MATH<x>:USERdefine:
DEFine {<Character string>}
:MATH<x>:USERdefine:DEFine?
<x>=1 to 8
<Character string>=within 50 characters
```

Example
```
:MATH1:USERDEFINE:DEFINE "C1-C2"
:MATH1:USERDEFINE:DEFINE?→:MATH1:
USERDEFIN E:DEFINE "C1-C2                "
```

Description Characters or symbols other than the ones on the keyboard displayed on the screen can not be used.

:MATH<x>:USERdefine:FFT?

Function Queries all FFT settings.

Syntax
```
:MATH<x>:USERdefine:FFT?
<x>=1 to 8
```

Example
```
:MATH1:USERDEFINE:FFT?→:MATH1:
USERDEFINE:FFT:POINT 1000;
WINDOW RECTANGLE
```

:MATH<x>:USERdefine:FFT:POINt

Function Sets/queries the number of points for the FFT.

Syntax
```
:MATH<x>:USERdefine:FFT:POINt {<NRf>}
:MATH<x>:USERdefine:FFT:POINt?
<x>=1 to 8
<NRf>=1000, 2000, 10000
```

Example
```
:MATH1:USERDEFINE:FFT:POINT 1000
:MATH1:USERDEFINE:FFT:POINT?→:MATH1:
USERDEFINE:FFT:POINT 1000
```

:MATH<x>:USERdefine:FFT:WINDow

Function Sets/queries the FFT window function.
Syntax :MATH<x>:USERdefine:FFT:WINDow {FLATtop|HANNing|RECTangle}
:MATH<x>:USERdefine:FFT:WINDow?
<x>=1 to 8
Example :MATH1:USERDEFINE:FFT:WINDOW RECTANGLE
:MATH1:USERDEFINE:FFT:WINDOW?→:MATH1:USERDEFINE:FFT:WINDOW RECTANGLE

:MATH<x>:USERdefine:FILTer<x>?

Function Queries all filter setting values.
Syntax :MATH<x>:USERdefine:FILTer<x>?
<x>(MATH)=1 to 8
<x>(FILTer)=1, 2
Example :MATH1:USERDEFINE:FILTER1?→:MATH1:USERDEFINE:FILTER1:CUTOFF1 10.0E+00;TYPE GAUSS;BAND LPASS

:MATH<x>:USERdefine:FILTer<x>:BAND

Function Sets/Queries the bandwidth of the filter.
Syntax :MATH<x>:USERdefine:FILTer<x>:BAND {BPASs|HPASs|LPASs}
:MATH<x>:USERdefine:FILTer<x>:BAND?
<x>(MATH)=1 to 8
<x>(FILTer)=1, 2
Example :MATH1:USERDEFINE:FILTER1:BAND LPASS
:MATH1:USERDEFINE:FILTER1:BAND?→:MATH1:USERDEFINE:FILTER1:BAND LPASS
Description When the BAND is ":MATH<x>:USERdefine:FILTer<x>:TYPE GAUSs", only LPAS can be set.

:MATH<x>:USERdefine:FILTer<x>:CUToff<x>

Function Sets/Queries the cutoff frequency.
Syntax :MATH<x>:USERdefine:FILTer<x>:CUToff<x> {<NRf>[PCT]}
:MATH<x>:USERdefine:FILTer<x>:CUToff<x>?
<x>(MATH)=1 to 8
<x>(FILTer)=1, 2
<x>(CUToff)=1, 2
<NRf>=2 to 30%(in steps of 0.2%)
Example :MATH1:USERDEFINE:FILTER1:CUTOFF1 10PCT
:MATH1:USERDEFINE:FILTER1:CUTOFF1?
→:MATH1:USERDEFINE:FILTER1:CUTOFF1 10.0E+00
Description CUToff2 can be set only when the MODE is ":MATH<x>:USERdefine:FILTer<x>:BAND BPASs."

:MATH<x>:USERdefine:FILTer<x>:TYPE

Function Sets/queries the filter type.
Syntax :MATH<x>:USERdefine:FILTer<x>:TYPE {GAUSs|IIR|SHARp}
:MATH<x>:USERdefine:FILTer<x>:TYPE?
<x>=1 to 8
Example :MATH1:USERDEFINE:FILTER1:TYPE GAUSS
:MATH1:USERDEFINE:FILTER1:TYPE?→:MATH1:USERDEFINE:FILTER1:TYPE GAUSS

:MATH<x>:USERdefine:MODE

Function Sets/queries the enable/disable condition of the user defined computation.
Syntax :MATH<x>:USERdefine:MODE {<Boolean>}
:MATH<x>:USERdefine:MODE?
<x>=1 to 8
Example :MATH1:USERDEFINE:MODE ON
:MATH1:USERDEFINE:MODE?→:MATH1:USERDEFINE:MODE 1

:MATH<x>:USERdefine:THReshold?

Function Queries all threshold level settings for binary computation/pulse width computation.
Syntax :MATH<x>:USERdefine:THReshold?
<x>=1 to 8
Example :MATH1:USERDEFINE:THRESHOLD?→MATH1:USERDEFINE:THRESHOLD:TRACE 1;LEVEL 0.005E+03,-0.005E+03

:MATH<x>:USERdefine:THReshold:LEVel

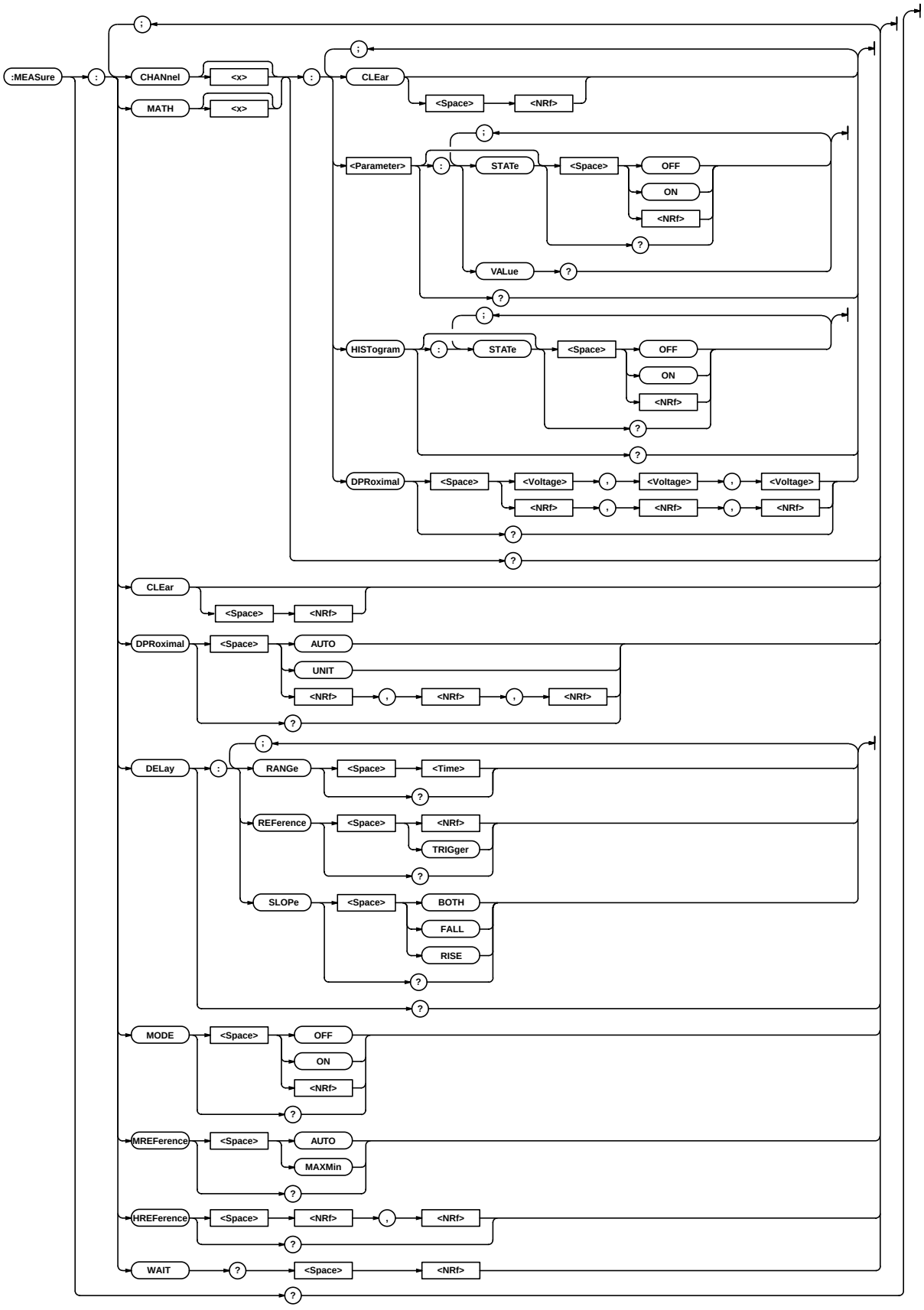
Function Sets/queries the threshold level setting for binary computation/pulse width computation.
Syntax :MATH<x>:USERdefine:THReshold:LEVel {<Voltage>,<Voltage>|<NRf>,<NRf>}
:MATH<x>:USERdefine:THReshold:LEVel?
<x>=1 to 8
<Voltage>=Within 8 divisions(in steps of 0.1div)
<NRf>=-1.0000E+30 to +1.0000E+30
Example :MATH1:USERDEFINE:THRESHOLD:LEVEL 5V,-5V
:MATH1:USERDEFINE:THRESHOLD:LEVEL?
→:MATH1:USERDEFINE:THRESHOLD:LEVEL 0.005E+03,-0.005E+03
Description If the specified channel is a temperature/strain module, or when setting the threshold level of a computed waveform, it is set in <NRf> format.

:MATH<x>:USERdefine:THReshold:TRACe

Function Sets/queries the trace to set the threshold.
Syntax :MATH<x>:USERdefine:THReshold:TRACe {<NRf>|MATH<x>}
:MATH<x>:USERdefine:THReshold:TRACe?
<x>(MATH<x>, Header)=1 to 8
<x>(MATH<x>, Data)=1 to 7
<NRf>=1 to 8
Example :MATH1:USERDEFINE:THRESHOLD:TRACE 1
:MATH1:USERDEFINE:THRESHOLD:TRACE?
→:MATH1:USERDEFINE:THRESHOLD:TRACE 1

4.18 MEASure Group

The commands in the MEASure group are used to make settings and queries about automatic measurement of waveform parameters. You can make the same settings which you can make using the MEASURE key on the front panel.



:MEASure?

Function Queries all settings relating to automatic measurement of waveform parameters.

Syntax :MEASure?

Example :MEASURE?→:MEASURE:MODE 1;CLEAR 1;
DPROXIMAL AUTO;CHANNEL1:CLEAR 1;
FREQUENCY:STATE 1;:MEASURE:CHANNEL1:
MAXIMUM:STATE 1;:MEASURE:CHANNEL1:
MINIMUM:STATE 1;:MEASURE:CHANNEL1:
NOVERSHOOT:STATE 1;:MEASURE:CHANNEL1:
PERIOD:STATE 1;:MEASURE:CHANNEL1:
POVERSHOOT:STATE 1;:MEASURE:CHANNEL1:
TY1INTEG:STATE 1;:MEASURE:CHANNEL1:
TY2INTEG:STATE 1;:MEASURE:DELAY:
REFERENCE TRIGGER;:MEASURE:
HREFERENCE -5.00E+00,5.00E+00;
MREFERENCE AUTO

:MEASure:{CHANnel<x>|MATH<x>}?

Function Queries all specified parameter ON/OFF settings.

Syntax :MEASure:{CHANnel<x>|MATH<x>}?
<x>(CHANnel)=1 to 16
<x>(MATH)=1 to 8

Example :MEASURE:CHANNEL1?→:MEASURE:CHANNEL1:
CLEAR 1;FREQUENCY:STATE 1;:MEASURE:
CHANNEL1:MAXIMUM:STATE 1;:MEASURE:
CHANNEL1:MINIMUM:STATE 1;:MEASURE:
CHANNEL1:NOVERSHOOT:STATE 1;:MEASURE:
CHANNEL1:PERIOD:STATE 1;:MEASURE:
CHANNEL1:POVERSHOOT:STATE 1;:MEASURE:
CHANNEL1:TY1INTEG:STATE 1;:MEASURE:
CHANNEL1:TY2INTEG:STATE 1

:MEASure:{CHANnel<x>|MATH<x>}:CLEar

Function Clears waveform's parameters.

Syntax :MEASure:{CHANnel<x>|MATH<x>}:
CLEar[<NRf>]
<x>(CHANnel)=1 to 16
<x>(MATH)=1 to 8

Example :MEASURE:CHANNEL1:CLEAR

Description The <NRf> argument is meaningless and can be omitted.

:MEASure:{CHANnel<x>|MATH<x>}:<Parameter>[:STATe]

Function Turns the specified waveform parameter ON or OFF, or queries the current setting.

Syntax :MEASure:{CHANnel<x>|MATH<x>}:
<Parameter>[:STATe] {<Boolean>}
:MEASure:{CHANnel<x>|MATH<x>}:
<Parameter>[:STATe]?
<x>(CHANnel)=1 to 16
<x>(MATH)=1 to 8
<Parameter>
={AMPLitude|AVERage|FALL|FDELay|
FREQuency|HIGH|LOW|MAXimum|MINimum|
NDUTycycle|NOVershoot|NWIDth|PDUTycycle|
PERiod|POVershoot|PTOPeak|PWIDth|RDELay|RISE|
RMS|SDEViation|TY1Integ|TY2Integ|
XY1Integ|XY2Integ}

Example :MEASURE:CHANNEL1:AVERAGE:STATE ON
:MEASURE:CHANNEL1:AVERAGE:STATE?
→:MEASURE:CHANNEL1:AVERAGE:STATE 1

:MEASure:{CHANnel<x>|MATH<x>}:<Parameter>:VALue?

Function Queries automatically measured values for the specified waveform parameter.

Syntax :MEASure:CHANnel<x>:<Parameter>:VALue?
<x> (CHANnel)=1 to 16
<x> (MATH)=1 to 8
<Parameter>
={AMPLitude|AVERage|FALL|FDELay|
FREQuency|HIGH|LOW|MAXimum|MINimum|
NDUTycycle|NOVershoot|NWIDth|PDUTycycle|
PERiod|POVershoot|PTOPeak|PWIDth|RDELay|
RISE|RMS|SDEViation|TY1Integ|TY2Integ|
XY1Integ|XY2Integ}

Example :MEASURE:CHANNEL1:AVERAGE:
VALUE?→4.950E+00

Description "NAN (Not A Number)" will be returned if the measurement cannot be performed.

:MEASure:{CHANnel<x>|MATH<x>}:DPRoximal (Distal PROXIMAL)

Function Sets/queries the waveform's distal, medial, and proximal points.

Syntax :MEASure:{CHANnel<x>|MATH<x>}:
DPRoximal {(<Voltage>,<Voltage>,
<Voltage>)|(<NRf>,<NRf>,<NRf>)}
:MEASure:{CHANnel<x>|MATH<x>}:DPRoximal?
<x>(CHANnel)=1 to 16
<x>(MATH)=1 to 8
<Voltage>=Within 8 divisions
<NRf>=-1E+30 to 1E+30 (for MATH<x>)
<NRf>=-3000 to 3000 (when set to °C on the temperature module)
=-20000 to 20000 (×10^{-6} strain, for strain module)

Example :MEASURE:CHANNEL1:DPROXIMAL -6V,0V,6V
:MEASURE:CHANNEL1:DPROXIMAL?
→:MEASURE:CHANNEL1:
DPROXIMAL -6.0E+00,0.0E+00,6.0E+00

Description
- If the specified channel is a temperature/strain module, or "MEASure:MATH<x>:DPRoximal," requires/returns <NRf> arguments.
- To incorporate the setting into automatic waveform parameter measurement, you must use the ":MEASure:DPRoximal UNIT" command.

:MEASure:{CHANnel<x>|MATH<x>}:HISTogram?

Function Queries all settings relating to histogram display of the specified waveform.

Syntax :MEASure:{CHANnel<x>|MATH<x>}:HISTogram?
<x> (CHANnel)=1 to 16
<x> (MATH)=1 to 8

Example :MEASURE:CHANNEL1:HISTOGRAM?→:MEASURE:
CHANNEL1:HISTOGRAM:STATE 1

:MEASure:{CHANnel<x>|MATH<x>}: HISTogram[:STATe]

Function Sets histogram display of the specified waveform ON or OFF, or queries the current setting.
Syntax :MEASure:{CHANnel<x>|MATH<x>}:HISTogram[:STATe] {<Boolean>}
:MEASure:{CHANnel<x>|MATH<x>}:HISTogram[:STATe]?
<x> (CHANnel)=1 to 16
<x> (MATH)=1 to 8
Example :MEASURE:CHANNEL1:HISTOGRAM:STATE ON
:MEASURE:CHANNEL1:HISTOGRAM:STATE?
→:MEASURE:CHANNEL1:HISTOGRAM:STATE 1

:MEASure:CLEar

Function Clears measurement of all waveform parameters.
Syntax :MEASure:CLEar[<NRf>]
Example :MEASURE:CLEAR
Description <NRf> is a dummy, a meaningless data. It can be omitted.

:MEASure:DELay?

Function Queries all the delay settings.
Syntax :MEASure:DELay?
Example :MEASURE:DELAY?→:MEASURE:DELAY:REFERENCE TRIGGER

:MEASure:DELay:RANGe

Function Sets/queries the starting point for delay measurement.
Syntax :MEASure:DELay:RANGe {<Time>}
:MEASure:DELay:RANGe?
<Time>=Within 10 div
Example :MEASURE:DELAY:RANGE -2us
:MEASURE:DELAY:RANGE?→:MEASURE:DELAY:RANGE -2.00E-06
Description This setting is invalid if the reference is a trigger (MEASure:DELay:REFerence TRIGger).

:MEASure:DELay:REFerence

Function Sets the delay reference/inquires about the current setting.
Syntax :MEASure:DELay:REFerence {<NRf>|TRIGger}
:MEASure:DELay:REFerence?
<NRf>=1 to 16
Example :MEASURE:DELAY:REFERENCE TRIGGER
:MEASURE:DELAY:REFERENCE?→:MEASURE:DELAY:REFERENCE TRIGGER

:MEASure:DELay:SLOPe

Function Sets/queries the slope of the delay reference waveform.
Syntax :MEASure:DELay:SLOPe {BOTH|FALL|RISE}
:MEASure:DELay:SLOPe?
Example :MEASURE:DELAY:SLOPE BOTH
:MEASURE:DELAY:SLOPE?→:MEASURE:DELAY:SLOPE BOTH
Description This setting/query is invalid if the reference is a trigger (MEASure:DELay:REFerence TRIGger).

:MEASure:DPRoximal (Distal PROXIMAL)

Function Sets/queries the distal, mesial, and proximal points.
Syntax :MEASure:DPRoximal{AUTO|UNIT|<NRf>,<NRf>,<NRf>}
:MEASure:DPRoximal?
<NRf>=0 to 100(%)
Example :MEASURE:DPROXIMAL AUTO
:MEASURE:DPROXIMAL?→:MEASURE:DPROXIMAL AUTO

:MEASure:HREFerence (Horizontal REFERENCE)

Function Sets/queries the measurement range.
Syntax :MEASure:HREFerence {<NRf>,<NRf>}
:MEASure:HREFerence?
<NRf>=-5 to 5div(in steps of 10div/display-record-length)
Example :MEASURE:HREFERENCE -4,4
:MEASURE:HREFERENCE?→:MEASURE:HREFERENCE -4.00E+00,4.00E+00

:MEASure:MODE

Function Sets automatic measurement of waveform parameters ON or OFF, or queries the current setting.
Syntax :MEASure:MODE {<Boolean>}
:MEASure:MODE?
Example :MEASURE:MODE ON
:MEASURE:MODE?→:MEASURE:MODE 1

:MEASure:MREFerence (Magnitude REFERENCE)

Function Sets/queries the High and Low points.
Syntax :MEASure:MREFerence {AUTO|MAXMin}
:MEASure:MREFerence?
Example :MEASURE:MREFERENCE AUTO
:MEASURE:MREFERENCE?→:MEASURE:MREFERENCE AUTO

:MEASure:WAIT?

Function Wait for the completion of the automatic measurement of waveform parameters with timeout.
Syntax :MEASure:WAIT? {<NRf>}
<NRf>=1 to 36000(Timeout period in units of 100 ms)
Example :MEASURE:WAIT?→0
Description
- "0" is returned if the automatic measurement of waveform parameters completes before the timeout. "1" is returned if it is not complete or automatic measurement is not being made.
- "0" is returned at the time the automatic measurement of waveform parameters completes, even if the timeout period is set to a large value.

4.19 SNAP Group

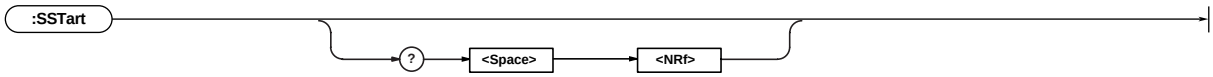
The SNAP command is used to execute a snapshot. The same function can be performed using the SNAPSHOT key on the front panel.

:SNAP

Function	Executes a snapshot.
Syntax	:SNAP
Example	:SNAP
Description	• The same function can be performed using "DISPlay:SNAP."
	• Use "CLEar" or "DISPlay:CLEar" to perform a clear trace.

4.20 SSTart Group

The SSTart command is used to execute a single start. You can make the same setting using the Single Start menu of the ACQ key on the front panel.



:SSTart (Single STart)

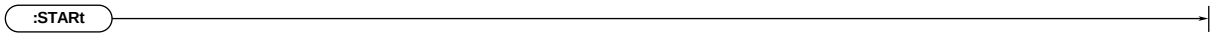
Function	Executes a single start.
Syntax	:SSTart
Example	:SSTART

:SSTart?

Function	Execute single start and wait for the completion with timeout.
Syntax	:SSTart? {<NRf>} <NRf>=1 to 2592000 (in units of 100 ms)
Example	:SSTART?→0
Description	• "0" is returned if it completes before the timeout. "1" is returned if it is not complete.
	• "0" is returned at the time the single start completes, even if the timeout period is set to a large value.

4.21 STARt Group

The STARt command is used to start acquisition. The same function can be performed by pressing the START/STOP key on the front panel.

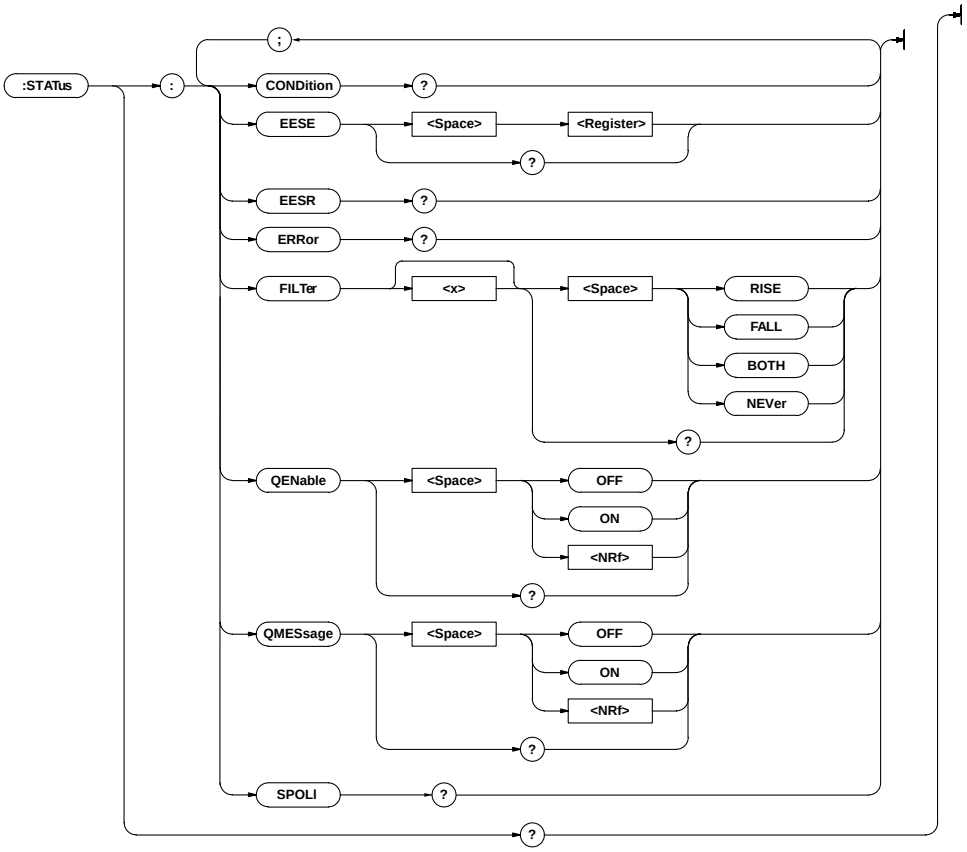


:STARt

Function	Starts acquisition.
Syntax	:STARt
Example	:START
Description	To stop acquisition, use "STOP."

4.22 STATus Group

The commands in the STATus group are used to make settings and queries about the status report. There is no front panel key for this function. For details of the status report, refer to Chapter 5.



:STATus?

Function Inquires about all the settings relating to the communications status function.

Syntax `:STATus?`

Example
```
:STATUS?→:STATUS:EESE 0;FILTER1 NEVER;
FILTER2 NEVER;FILTER3 NEVER;
FILTER4 NEVER;FILTER5 NEVER;
FILTER6 NEVER;FILTER7 NEVER;
FILTER8 NEVER;FILTER9 NEVER;
FILTER10 NEVER;FILTER11 NEVER;
FILTER12 NEVER;FILTER13 NEVER;
FILTER14 NEVER;FILTER15 NEVER;
FILTER16 NEVER;QENABLE 0;QMESSAGE 1
```

:STATus:CONDition?

Function Queries the contents of the condition register, and clears the register.

Syntax `:STATus:CONDition?`

Example `:STATUS:CONDITION?→16`

Description For a description of the synchronizion method using "`STATus:CONDition`," refer to page 3-8.

:STATus:EESE (Extended Event Status Enable register)

Function Sets/queries the extended event enable register.

Syntax
```
:STATus:EESE <Register>
:STATus:EESE?
<Register>=0 to 65535
```

Example
```
:STATUS:EESE #B00000000
:STATUS:EESE?→:STATUS:EESE 0
```

:STATus:EESR? (Extended Event Status Register)

Function Queries the content of the extended event register, and clears the register.

Syntax `:STATus:EESR?`

Example `:STATUS:EESR?→0`

:STATus:ERRor?

Function Queries the code and message (at the beginning of the error queue) of the error which has occurred.

Syntax `:STATus:ERRor?`

Example `:STATUS:ERROR?→901",Backup failure"`

Description
- "`0`," or ""`No error`"" is returned if no error is occurring.
- Message contents cannot be returned in Japanese.
- It is possible to select whether or not to add the message contents using "`STATus:QMESsage`."

:STATus:FILTer<x>

Function Sets/queries the transit filter.

Syntax
```
:STATus:FILTer<x> {RISE|FALL|BOTH|NEVer}
:STATus:FILTer<x>?
<x>=1 to 16
```

Example
```
:STATUS:FILTER2 RISE
:STATUS:FILTER2?→:STATUS:FILTER2 RISE
```

Description Determines which direction Condition Register bits must change in order to set an event. If "`RISE`," event is set when bit changes from 0 to 1.

:STATus:QENable

Function Selects/queries whether messages other than errors are stored in the error queue.
Syntax `:STATus:QENable {<Boolean>}`
`:STATus:QENable?`
Example `:STATUS:QENABLE ON`
`:STATUS:QENABLE?→:STATUS:QENABLE 1`

:STATus:QMESsage

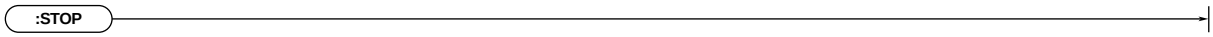
Function Selects/queries whether message content is appended to responses to "`STATus:ERRor?`".
Syntax `:STATus:QMESsage {<Boolean>}`
`:STATus:QMESsage?`
Example `:STATUS:QMESSAGE ON`
`:STATUS:QMESSAGE?→:STATUS:QMESSAGE 1`

:STATus:SPOLl? (Serial Poll)

Function Executes the serial polling.
Syntax `:STATus:SPOLl?`
Example `:STATUS:SPOLL?→:STATUS:SPOLL 0`
Description This is an exclusive command for the RS-232 interface.

4.23 STOP Group

The STOP command is used to stop acquisition. The same function can be performed using the START/STOP key on the front panel.

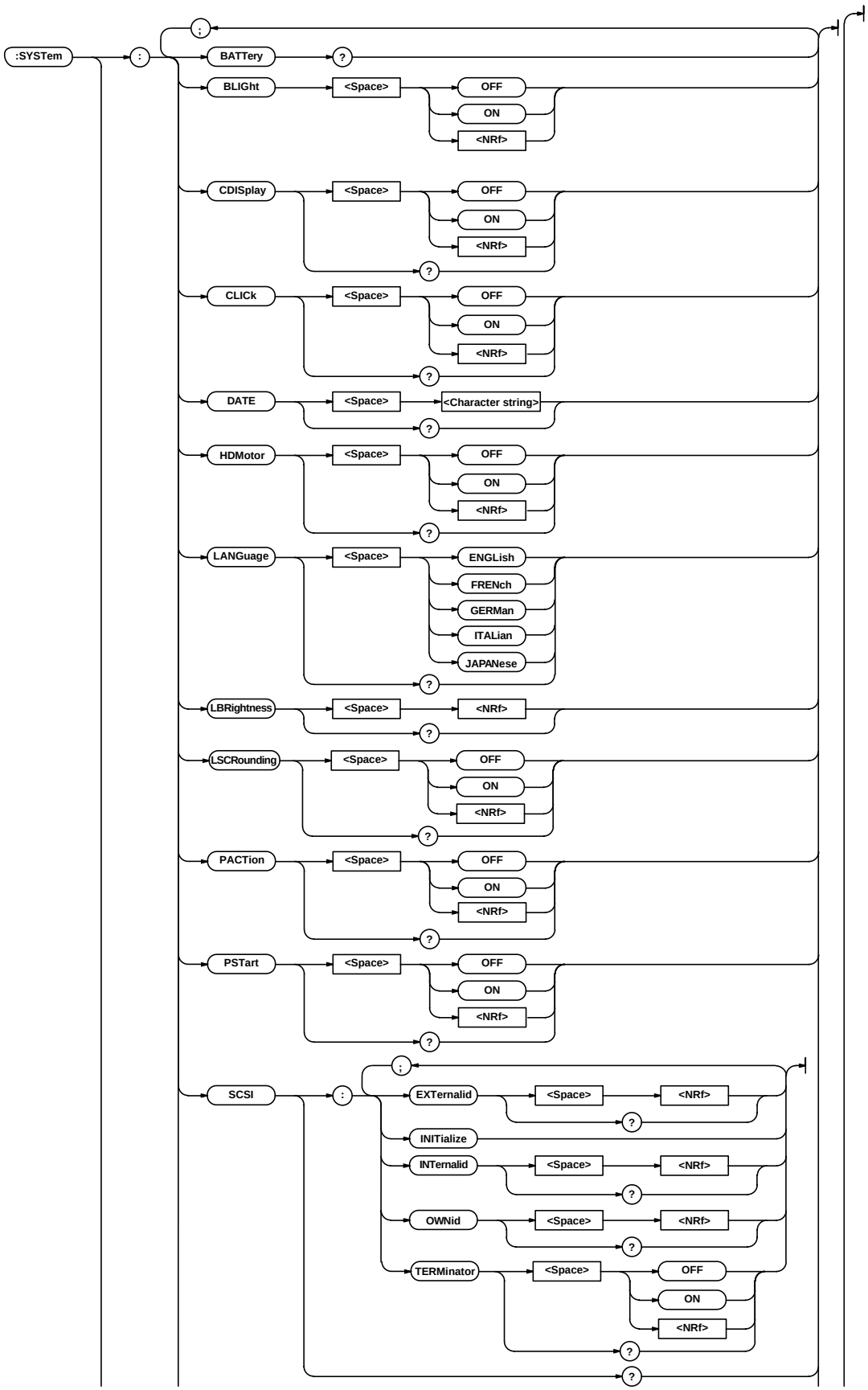


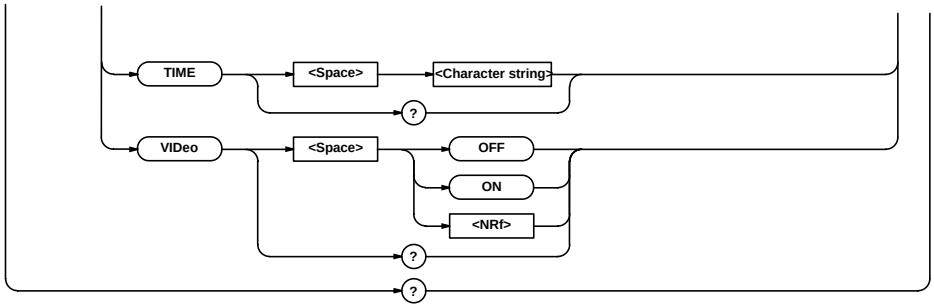
:STOP

Function Stops acquisition.
Syntax `:STOP`
Example `:STOP`
Description To start acquisition, use "`STARt`."

4.24 SYSTem Group

The commands in the SYSTem group are used to make settings and queries about the system. The same settings can be made using the System Config menu obtained when the MISC key on the front panel is pressed.





:SYSTem?

Function Queries all system settings.
Syntax :SYSTem?
Example :SYSTEM:CLICK 1;LANGUAGE JAPANESE;
LBRIGHTNESS 7;SCSI:OWNID 6;EXTERNALID 5;
INTERNALID 4;TERMINATOR 1;:SYSTEM:
CDISPLAY 1;HDMOTOR 1;LSCROUNDING 1;
PACTION 0;PSTART 0;VIDEO 0

:SYSTem:BATTery?

Function Queries the condition of the internal lithium battery.
Syntax :SYSTem:BATTery?
Example :SYSTEM:BATTERY?→1
Description "1" is returned if the battery is functioning, and "0" is returned if the battery has run out.

:SYSTem:BLIGht

Function Turns the screen backlight ON/OFF.
Syntax :SYSTem:BLIGht {<Boolean>}
Example :SYSTEM:BLIGHT OFF

:SYSTem:CDISplay

Function Sets/queries the ON/OFF condition of the display of the setting values when executing a communication command.
Syntax :SYSTem:CDISplay {<Boolean>}
:SYSTem:CDISplay?
Example :SYSTEM:CDISPLAY ON
:SYSTEM:CDISPLAY?→:SYSTEM:CDISPLAY 1
Description If this setting is set to ON, the new setting values are displayed on the screen. If it is set to OFF, the setting value display disappears from the screen at the time of the command execution, but the execution time is slightly shortened as compared to when it is set to ON.

:SYSTem:CLICk

Function Sets the click sound ON or OFF, or queries the current setting.
Syntax :SYSTem:CLICk {<Boolean>}
:SYSTem:CLICk?
Example :SYSTEM:CLICK ON
:SYSTEM:CLICK?→:SYSTEM:CLICK 1

:SYSTem:DATE

Function Sets/queries the date.
Syntax :SYSTem:DATE <Character string>
:SYSTem:DATE?
<Character string>
=YY/MM/DD,refer to User's Manual
IM701830-01E
Example :SYSTEM:DATE "96/07/26"
:SYSTEM:DATE?→"96/07/26"
Description The "year" is represented with the lower two digits. Years 2000 to 2079 are represented by 00 to 79, and years 1980 to 1999 are represented by 80 to 99.

:SYSTem:HDMotor

Function Sets/queries the ON/OFF of the internal hard disk motor.
Syntax :SYSTem:HDMotor {<Boolean>}
:SYSTem:HDMotor?
Example :SYSTEM:HDMOTOR ON
:SYSTEM:HDMOTOR?→:SYSTEM:HDMOTOR 1
Description An error will occur, if there is no internal hard disk.

:SYSTem:LANGuage

Function Sets/queries the message language.
Syntax :SYSTem:LANGuage {ENGLish|FRENch|GERMan|
ITALian|JAPANese}
:SYSTem:LANGuage?
Example :SYSTEM:LANGUAGE JAPANESE
:SYSTEM:LANGUAGE?→:SYSTEM:
LANGUAGE JAPANESE

:SYSTem:LBRightness

Function Sets/queries the brightness of the screen.
Syntax :SYSTem:LBRightness{<NRf>}
:SYSTEM:LBRightness?
<NRf>=0 to 7
Example :SYSTEM:LBRIGHTNESS 4
:SYSTEM:LBRIGHTNESS?→:SYSTEM:
LBRIGHTNESS 4

:SYSTem:LSCRounding

Function Sets/queries the rounding of the upper and lower limits of the linear scaling values ON/OFF.
Syntax :SYSTem:LSCRounding {<Boolean>}
:SYSTem:LSCRounding?
Example :SYSTEM:LSCROUNDING ON
:SYSTEM:LSCROUNDING?→:SYSTEM:
LSCROUNDING 1

:SYSTem:PACTion

Function Sets/queries whether to make the action on trigger mode valid or set the mode to OFF when the power turns ON.

Syntax
```
:SYSTem:PACTion {<Boolean>}
:SYSTem:PACTion?
```

Example
```
:SYSTEM:PACTION ON
:SYSTEM:PACTION?→:SYSTEM:PACTION 1
```

:SYSTem:PSTart

Function Sets/queries whether or not to start waveform acquisition when the power turns ON.

Syntax
```
:SYSTem:PSTart {<Boolean>}
:SYSTem:PSTart?
```

Example
```
:SYSTEM:PSTART ON
:SYSTEM:PSTART?→:SYSTEM:PSTART 1
```

:SYSTem:SCSI?

Function Queries all SCSI-ID settings.

Syntax
```
:SYSTem:SCSI?
```

Example
```
:SYSTEM:SCSI?→:SYSTEM:SCSI:OWNID 7;
EXTERNALID 0;INTERNALID 4:TERMINATOR 1
```

:SYSTem:SCSI:EXTernalid

Function Sets/queries external device SCSI-ID.

Syntax
```
:SYSTem:SCSI:EXTernalid {<NRf>}
:SYSTem:SCSI:EXTernalid?
<NRf>=0 to 7
```

Example
```
:SYSTEM:SCSI:EXTERNALID 1
:SYSTEM:SCSI:EXTERNALID?→:SYSTEM:SCSI:
EXTERNALID 1
```

:SYSTem:SCSI:INITialize

Function Initializes the SCSI.

Syntax
```
:SYSTem:SCSI:INITialize
```

Example
```
:SYSTEM:SCSI:INITIALIZE
```

Description This command should always be issued following change of the oscilloscope's SCSI-ID (by the "`:SYSTem:SCSI:OWNid`" command).

:SYSTem:SCSI:INTernalid

Function Sets/queries the SCSI-ID of the internal hard disk.

Syntax
```
:SYSTem:SCSI:INTernalid {<NRf>}
:SYSTem:SCSI:INTernalid?
<NRf>=0 to 7
```

Example
```
:SYSTEM:SCSI:INTERNALID 4
:SYSTEM:SCSI:INTERNALID?→:SYSTEM:SCSI:
INTERNALID 4
```

:SYSTem:SCSI:OWNid

Function Sets/queries own SCSI-ID.

Syntax
```
:SYSTem:SCSI:OWNid {<NRf>}
:SYSTem:SCSI:OWNid?
<NRf>=0 to 7
```

Example
```
:SYSTEM:SCSI:OWNID 7
:SYSTEM:SCSI:OWNID?→:SYSTEM:SCSI:OWNID 7
```

:SYSTem:SCSI:TERMinator

Function SCSI Sets/queries the SCSI terminator ON/OFF switch.

Syntax
```
:SYSTem:SCSI:TERMinator {<Boolean>}
:SYSTem:SCSI:TERMinator?
```

Example
```
:SYSTEM:SCSI:TERMINATOR ON
:SYSTEM:SCSI:TERMINATOR?→:SYSTEM:SCSI:
TERMINATOR 1
```

:SYSTem:TIME

Function Sets/queries the time.

Syntax
```
:SYSTem:TIME <Character string>
:SYSTem:TIME?
<Character string>
=HH:MM:SS,refer to User's ManualIM701830-
 01E
```

Example
```
:SYSTEM:TIME "14:30:00"
:SYSTEM:TIME?→"14:30:00"
```

:SYSTem:VIDeo

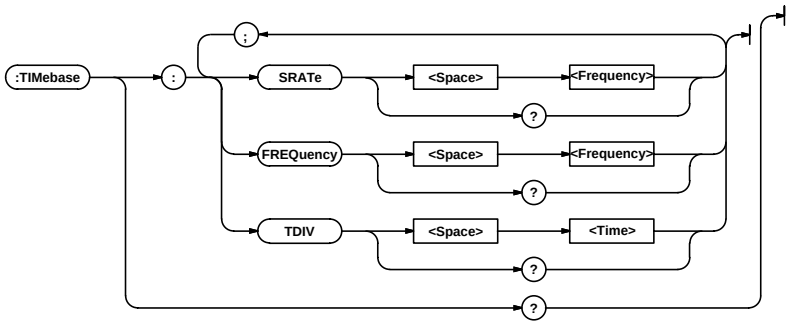
Function Sets/queries the ON/OFF condition of the video output.

Syntax
```
:SYSTem:VIDeo {<Boolean>}
:SYSTem:VIDeo?
```

Example
```
:SYSTEM:VIDEO ON
:SYSTEM:VIDEO?→:SYSTEM:VIDEO 1
```

4.25 TIMebase Group

The commands in the TIMebase group are used to make settings and queries about the time base. These settings and inquiries can also be made using the TIME/DIV knob on the front panel.



:TIMebase?

Function Queries all the time base settings.
Syntax `:TIMebase?`
Example `:TIMEBASE?→:TIMEBASE:SRATE 0.1E+08;`
`TDIV 0.0001E-03`

:TIMebase:FREQuency

Function Sets/queries the sampling rate.
Syntax `:TIMebase:FREQuency {<Frequency>}`
`:TIMebase:FREQuency?`
`<Frequency>=1Hz to 10MHz`
Example `:TIMEBASE:FREQUENCY 10MHZ`
`:TIMEBASE:FREQUENCY?→:TIMEBASE:`
`FREQUENCY 0.1E+08`
Description
- T/div varies according to the sampling rate set.
- The same setting and query can be made using "`TIMebase:SRATe`."

:TIMebase:SRATe(Sample RATE)

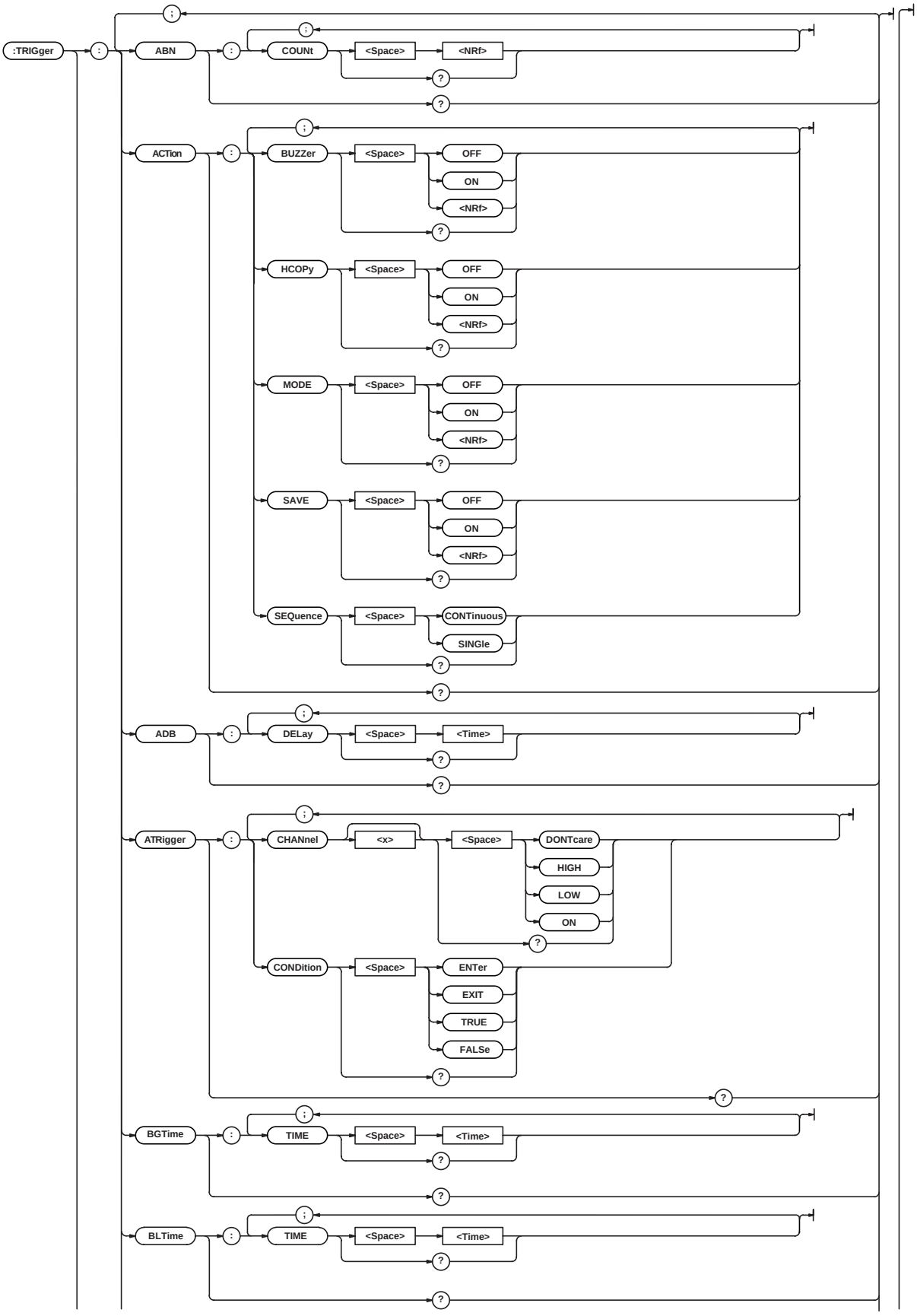
Function Sets/queries the sampling rate.
Syntax `:TIMebase:SRATe {<Frequency>}`
`:TIMebase:SRATe?`
`<Frequency>=1Hz to 10MHz`
Example `:TIMEBASE:SRATE 10MHZ`
`:TIMEBASE:SRATE?→:TIMEBASE:SRATE 0.1E+08`
Description
- T/div varies according to the set sampling rate.
- The same setting and query can be made using "`TIMebase:FREQuency`." In this case, "`TIMebase:SRATe`" will be the target for "`TIMebase?`."

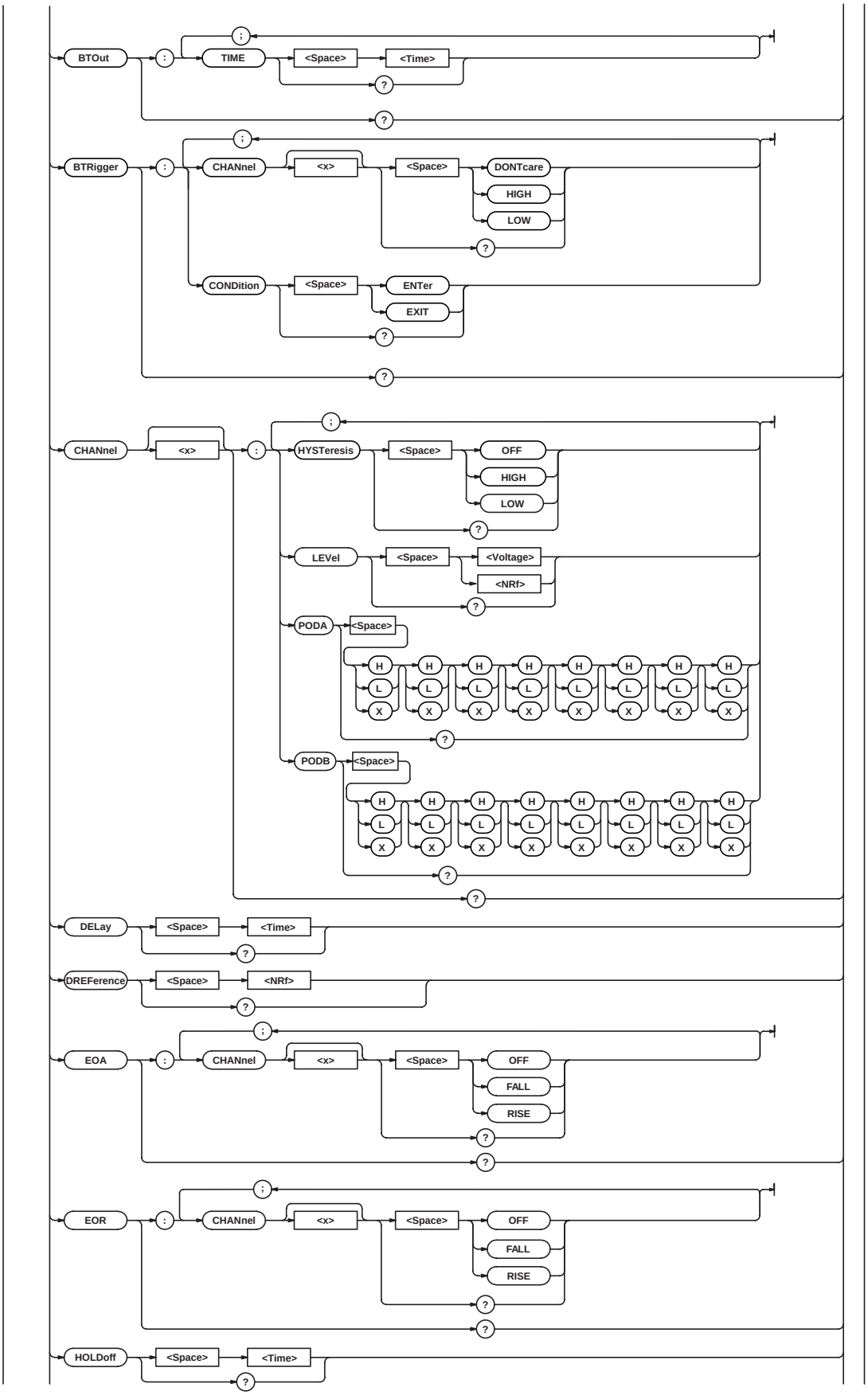
:TIMebase:TDIV

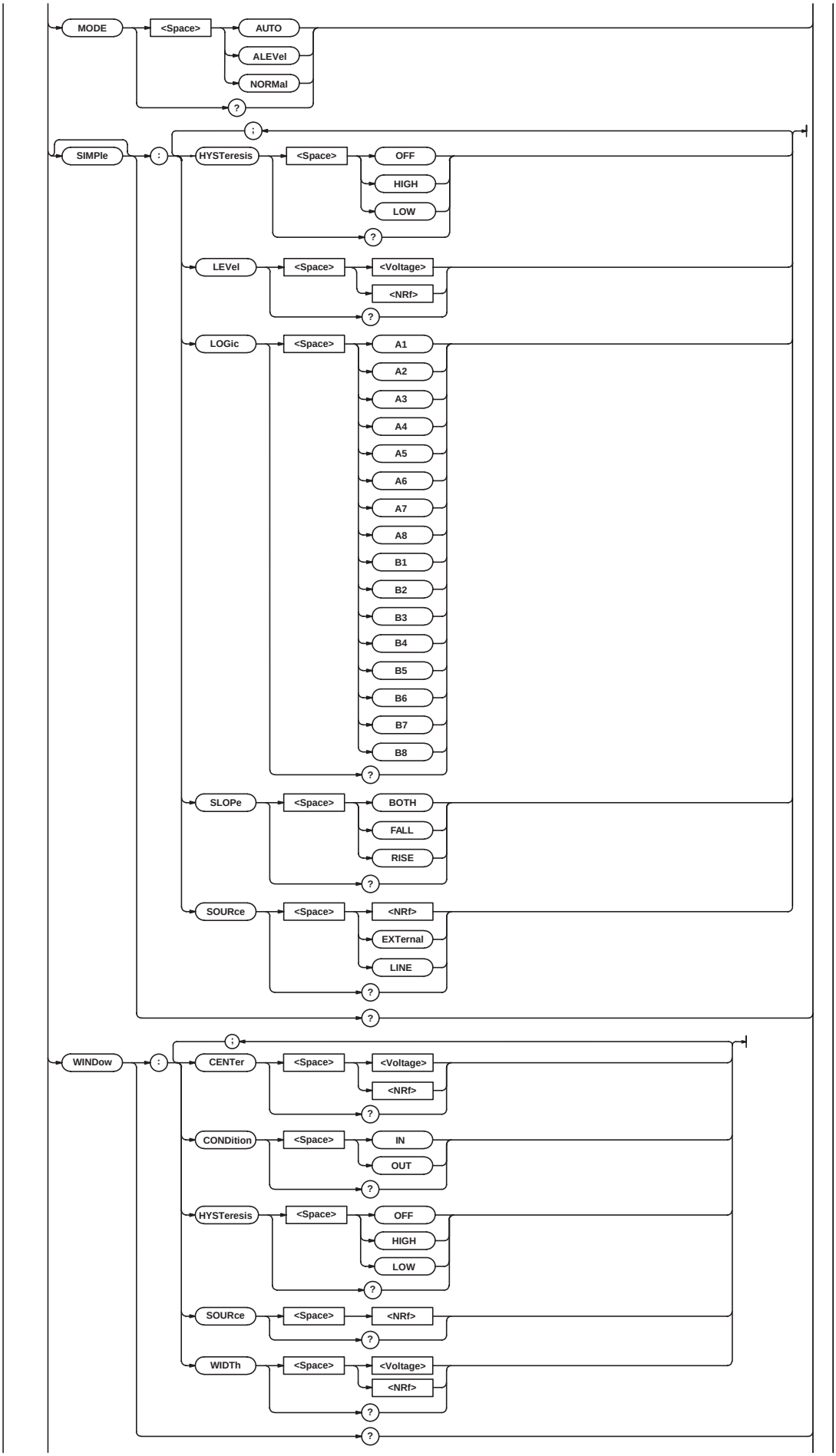
Function Sets/queries the T/div setting.
Syntax `:TIMebase:TDIV {<Time>}`
`:TIMebase:TDIV?`
`<Time>=500ns to 100ks`
Example `:TIMEBASE:TDIV 1US`
`:TIMEBASE:TDIV?→:TIMEBASE:TDIV 0.001E-03`

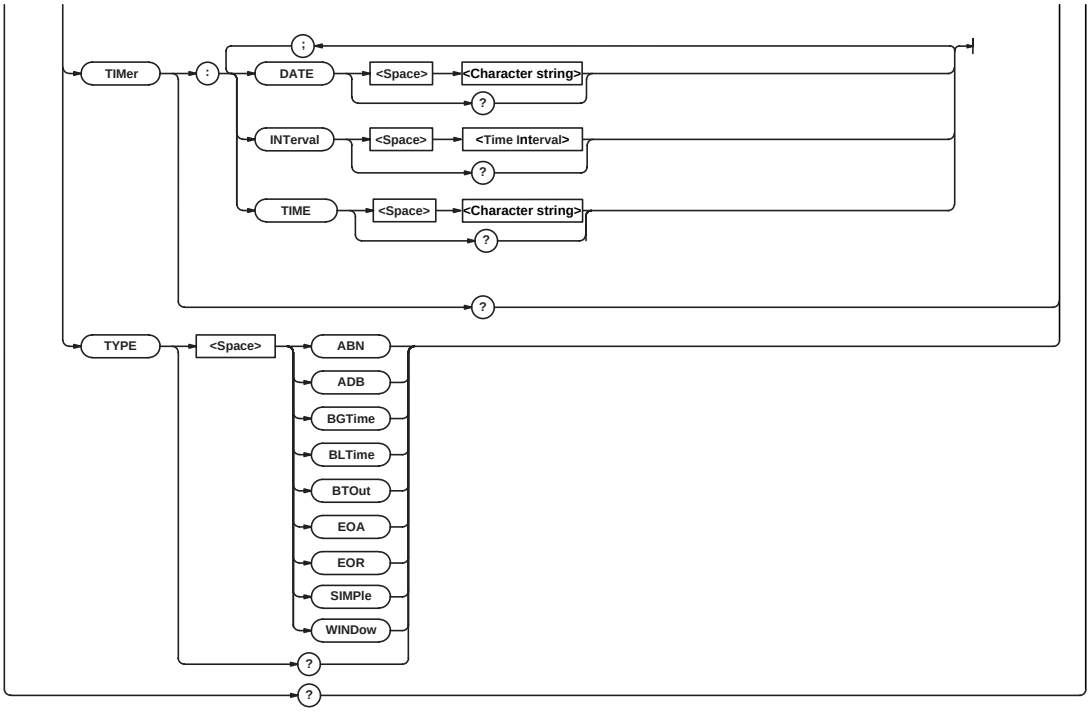
4.26 TRIGger Group

The commands in the TRIGger group are used to make settings and queries about the trigger. The same settings can be made using the TRIGGER group keys (the SIMPLE and ENHANCED keys, and LEVEL nob, and POSITION/DELAY keys).




BTOut
TIME
<Space>
<Time>
BTRigger
CHANnel
<x>
DONTcare
HIGH
LOW
CONDition
ENTer
EXIT
CHANnel
<x>
HYSTeresis
OFF
HIGH
LOW
LEVel
<Voltage>
<NRf>
PODA
PODB
H
L
X
DELay
DREFerence
EOA
FALL
RISE
EOR
HOLDoff


MODE
<Space>
AUTO
ALEVel
NORMal
?
:
SIMPle
:
HYSTeresis
<Space>
OFF
HIGH
LOW
?
LEVel
<Space>
<Voltage>
<NRf>
?
LOGic
<Space>
A1
A2
A3
A4
A5
A6
A7
A8
B1
B2
B3
B4
B5
B6
B7
B8
?
SLOPe
<Space>
BOTH
FALL
RISE
?
SOURce
<Space>
<NRf>
EXTernal
LINE
?
?
;
WINDow
:
CENTer
<Space>
<Voltage>
<NRf>
?
CONDition
<Space>
IN
OUT
?
HYSTeresis
<Space>
OFF
HIGH
LOW
?
SOURce
<Space>
<NRf>
?
WIDTh
<Space>
<Voltage>
<NRf>
?
?



:TRIGger?

Function Queries all trigger settings.
Syntax `:TRIGger?`
Example
```
:TRIGGER?→:TRIGGER:TYPE SIMPLE;
MODE AUTO;DELAY 0.000E-06;DREFERENCE 50;
HOLDOFF 0.05E-06;SIMPLE:SOURCE 1;
SLOPE RISE;LEVEL 0.000E+03;HYSTERESIS 0;:
TRIGGER:ACTION:MODE 0
```

:TRIGger:ABN?(A→B (n))

Function Queries all A→B (n) trigger settings.
Syntax `:TRIGger:ABN?`
Example `:TRIGGER:ABN?→:TRIGGER:ABN:COUNT 1`

:TRIGger:ABN:COUNt

Function Sets/queries the number of times the pattern for the A→B (n) trigger should become true.
Syntax
```
:TRIGger:ABN:COUNt {<NRf>}
:TRIGger:ABN:COUNt?
```
`<NRf>=1 to 255`
Example
```
:TRIGGER:ABN:COUNT 1
:TRIGGER:ABN:COUNT?→:TRIGGER:ABN:COUNT 1
```

:TRIGger:ACTion?

Function Queries all settings relating to the action on trigger.
Syntax `:TRIGger:ACTion?`
Example
```
:TRIGGER:ACTION?→:TRIGGER:ACTION:MODE 1;
SEQUENCE SINGLE;HCOPY 0;SAVE 0;BUZZER 0
```

:TRIGger:ACTion:BUZZer

Function Sets/queries the ON/OFF condition of the buzzer at the time of the trigger action.
Syntax
```
:TRIGger:ACTion:BUZZer {<Boolean>}
:TRIGger:ACTion:BUZZer?
```
Example
```
:TRIGGER:ACTION:BUZZER ON
:TRIGGER:ACTION:BUZZER?→:TRIGGER:ACTION:
BUZZER 1
```

:TRIGger:ACTion:HCOPy

Function Sets/queries whether the screen image data is output at the time of the trigger action.
Syntax
```
:TRIGger:ACTion:HCOPy {<Boolean>}
:TRIGger:ACTion:HCOPy?
```
Example
```
:TRIGGER:ACTION:HCOPY ON
:TRIGGER:ACTION:HCOPY?→:TRIGGER:ACTION:
HCOPY 1
```
Description Set the output destination of the screen image data and other settings in the HCOPy group.

:TRIGger:ACTion:MODE

Function Sets/queries the ON/OFF condition of the action on trigger function.
Syntax
```
:TRIGger:ACTion:MODE {<Boolean>}
:TRIGger:ACTion:MODE?
```
Example
```
:TRIGGER:ACTION:MODE ON
:TRIGGER:ACTION:MODE?→:TRIGGER:ACTION:
MODE 1
```

:TRIGger:ACTion:SAVE

Function Sets/queries whether the waveform data is saved to the medium at the time of the trigger action.
Syntax `:TRIGger:ACTion:SAVE {<Boolean>}`
`:TRIGger:ACTion:SAVE?`
Example `:TRIGGER:ACTION:SAVE ON`
`:TRIGGER:ACTION:SAVE?→:TRIGGER:ACTION:SAVE 1`
Description Set the medium and other settings in the FILE group.

:TRIGger:ACTion:SEQuence

Function Sets/queries whether the trigger action is executed continuously.
Syntax `:TRIGger:ACTion:SEQuence {CONTinuous|SINGle}`
`:TRIGger:ACTion:SEQuence?`
Example `:TRIGGER:ACTION:SEQUENCE CONTINUOUS`
`:TRIGGER:ACTION:SEQUENCE?→:TRIGGER:ACTION:SEQUENCE CONTINUOUS`

:TRIGger:ADB? (A Delay B)

Function Queries all A Delay B trigger settings.
Syntax `:TRIGger:ADB?`
Example `:TRIGGER:ADB?→:TRIGGER:ADB:DELAY 0.005E-6`

:TRIGger:ADB:DELay

Function Sets/queries the delay time for pattern B for an A Delay B trigger.
Syntax `:TRIGger:ADB:DELay {<Time>}`
`:TRIGger:ADB:DELay?`
`<Time>=0 to 1s (in steps of 500ns)`
Example `:TRIGGER:ADB:DELAY 500NS`
`:TRIGGER:ADB:DELAY?→:TRIGGER:ADB:DELAY 0.5E-06`

:TRIGger:ATRigger?

Function Queries all the pattern A settings.
Syntax `:TRIGger:ATRigger?`
Example `:TRIGGER:ATRIGGER?→:TRIGGER:ATRIGGER:CONDITION ENTER;CHANNEL1 HIGH;CHANNEL2 DONTCARE;CHANNEL3 DONTCARE;CHANNEL4 DONTCARE;CHANNEL5 DONTCARE;CHANNEL6 DONTCARE;CHANNEL7 DONTCARE;CHANNEL8 DONTCARE;CHANNEL9 DONTCARE;CHANNEL10 DONTCARE;CHANNEL11 DONTCARE;CHANNEL12 DONTCARE;CHANNEL13 DONTCARE;CHANNEL14 DONTCARE;CHANNEL15 DONTCARE;CHANNEL16 DONTCARE`

:TRIGger:ATRigger:CHANnel<x>

Function Sets/queries the channel condition for pattern A.
Syntax `:TRIGger:ATRigger:CHANnel<x> {DONTcare|HIGH|LOW|ON}`
`:TRIGger:ATRigger:CHANnel<x>?`
`<x>=1 to 16`
Example `(An example for CH1 is given below.)`
`:TRIGGER:ATRIGGER:CHANNEL1 HIGH`
`:TRIGGER:ATRIGGER:CHANNEL1?→:TRIGGER:ATRIGGER:CHANNEL1 HIGH`
Description If the logic input module is installed at the channel (slot), select from {DONTcare|ON}. If other modules are installed, select from {DONTcare|HIGH|LOW}. If there is no modules installed, an error occurs.

:TRIGger:ATRigger:CONDition

Function Sets/queries the conditions under which pattern A should become true.
Syntax `:TRIGger:ATRigger:CONDition {ENTer|EXIT}`
`(Trigger type:"ABN", "ADB")`
`:TRIGger:ATRigger:CONDition {TRUE|FALSe}`
`(Trigger type:"EOA")`
`:TRIGger:ATRigger:CONDition?`
Example `:TRIGGER:ATRIGGER:CONDITION ENTER`
`:TRIGGER:ATRIGGER:CONDITION?→:TRIGGER:ATRIGGER:CONDITION ENTER`
Description Select {ENTer|EXIT} if the trigger type is "ABN" or "ADB," {TRUE|FALSe} if the trigger type is "EOA."

:TRIGger:BGTime? (B Greater TIME)

Function Queries all the B>Time trigger settings.
Syntax `:TRIGger:BGTime?`
Example `:TRIGGER:BGTIME?→:TRIGGER:BGTIME:TIME 0.01E-06`

:TRIGger:BGTime:TIME

Function Sets/queries the pulse width for the B>Time trigger.
Syntax `:TRIGger:BGTime:TIME {<Time>}`
`:TRIGger:BGTime:TIME?`
`<Time>=200ns to 1s (in steps of 100ns)`
Example `:TRIGGER:BGTIME:TIME 200NS`
`:TRIGGER:BGTIME:TIME?→:TRIGGER:BGTIME:TIME 0.2E-06`

:TRIGger:BLTime? (B Less TIME)

Function Queries all B<Time trigger settings.
Syntax `:TRIGger:BLTime?`
Example `:TRIGGER:BLTIME?→:TRIGGER:BLTIME:TIME 0.1E-06`

:TRIGger:BLTime:TIME

Function Sets/queries the pulse width for the B<Time trigger.
Syntax :TRIGger:BLTime:TIME {<Time>}
:TRIGger:BLTime:TIME?
<Time>=200ns to 1s (in steps of 100ns)
Example :TRIGGER:BLTIME:TIME 200NS
:TRIGGER:BLTIME:TIME?→:TRIGGER:BLTIME:
TIME 0.2E-06

:TRIGger:BTOut?(B Time OUT)

Function Queries all B Time Out trigger settings.
Syntax :TRIGger:BTOut?
Example :TRIGGER:BTOUT?→:TRIGGER:BTOUT:
TIME 0.1E-06

:TRIGger:BTOut:TIME

Function Sets/queries the pulse width for the B Time Out trigger.
Syntax :TRIGger:BTOut:TIME {<Time>}
:TRIGger:BTOut:TIME?
<Time>=200ns to 1s (in steps of 100ns)
Example :TRIGGER:BTOUT:TIME 200NS
:TRIGGER:BTOUT:TIME?→:TRIGGER:BTOUT:
TIME 0.2E-06

:TRIGger:BTRigger?

Function Queries all pattern B settings.
Syntax :TRIGger:BTRigger?
Example :TRIGGER:BTRIGGER?→:TRIGGER:BTRIGGER:
CONDITION ENTER;CHANNEL1 HIGH;
CHANNEL2 DONTCARE;CHANNEL3 DONTCARE;
CHANNEL4 DONTCARE;CHANNEL5 DONTCARE;
CHANNEL6 DONTCARE;CHANNEL7 DONTCARE;
CHANNEL8 DONTCARE;CHANNEL9 DONTCARE;
CHANNEL10 DONTCARE;CHANNEL11 DONTCARE;
CHANNEL12 DONTCARE;CHANNEL13 DONTCARE;
CHANNEL14 DONTCARE;CHANNEL15 DONTCARE;
CHANNEL16 DONTCARE

:TRIGger:BTRigger:CHANnel<x>

Function Sets/queries the channel condition for pattern B.
Syntax :TRIGger:BTRigger:
CHANnel<x> {DONTcare|HIGH|LOW}
:TRIGger:BTRigger:CHANnel<x>?
<x>=1 to 8
Example (An example for CH1 is given below.)
:TRIGGER:BTRIGGER:CHANNEL1 HIGH
:TRIGGER:BTRIGGER:CHANNEL1?→:TRIGGER:
BTRIGGER:CHANNEL1 HIGH
Description If the modules except the logic input module are installed, select from {DONTcare|HIGH|LOW}. An error will occur, if there is no module installed or if there is a logic input module is installed at the channel (slot).

:TRIGger:BTRigger:CONDition

Function Sets/queries the conditions under which pattern B becomes true.
Syntax :TRIGger:BTRigger:CONDition {ENTer|EXIT}
:TRIGger:BTRigger:CONDition?
Example :TRIGGER:BTRIGGER:CONDITION ENTER
:TRIGGER:BTRIGGER:CONDITION?→:TRIGGER:
BTRIGGER:CONDITION ENTER

:TRIGger:CHANnel<x>?

Function Queries the trigger conditions for the specified channel.
Syntax :TRIGger:CHANnel<x>?
<x>=1 to 8
Example :TRIGGER:CHANNEL1?→:TRIGGER:CHANNEL1:
LEVEL 0.000E+03;HYSTERESIS OFF
Description An error will occur, if there is no module installed at the channel (slot).

:TRIGger:CHANnel<x>:HYSTeresis

Function Sets/queries the trigger hysteresis of each channel.
Syntax :TRIGger:CHANnel<x>:HYSTeresis {OFF|HIGH|
LOW}
:TRIGger:CHANnel<x>:HYSTeresis?
<x>=1 to 16
Example :TRIGGER:CHANNEL1:HYSTERESIS HIGH
:TRIGGER:CHANNEL1:HYSTERESIS?→:TRIGGER:
CHANNEL1:HYSTERESIS HIGH
Description An error will occur, if the voltage, the temperature, or the strain module is not installed.

:TRIGger:CHANnel<x>:LEVel

Function Sets/queries the trigger level of each channel.
Syntax :TRIGger:CHANnel<x>:LEVel {<Voltage>|
<NRf>}
:TRIGger:CHANnel<x>:LEVel?
<x>=1 to 16
<Voltage>=Within the 8 div on the screen
(for voltage modules)
<NRf>=-3000 to 3000 (when set to °C on
the temperature module)
=-2000 to 2000 (×10 $^{-6}$ strain, for
strain module)
Example :TRIGGER:CHANNEL1:LEVEL 0V
:TRIGGER:CHANNEL1:LEVEL?→:TRIGGER:
CHANNEL1:LEVEL 0.000E+03
Description An error will occur, if the voltage, the temperature, or the strain module is not installed.

:TRIGger:CHANnel<x>:PODA

Function Sets/queries the bit pattern of Pod A, when the logic input module is installed at the channel (slot).

Syntax
```
:TRIGger:CHANnel<x>:PODA {(H|L|X)(H|L|X)
(H|L|X)(H|L|X)(H|L|X)(H|L|X)(H|L|X)(H|L|X)}
(From the left of the data, bit 8 to bit 1)
:TRIGger:CHANnel<x>:PODA?
<x>=1 to 16
H=HIGH, L=LOW, X=Don't care
```

Example
```
(Below is an example when setting the bits
8, 5, and 2 of Pod A of channel 1 to High;
bits 7, 4, and 1 to Low; and bits 6 and 3
to Don't care.)
:TRIGGER:CHANNEL1:PODA HLXHLXHL
:TRIGGER:CHANNEL1:PODA?→:TRIGGER:
CHANNEL1:PODA HLXHLXHL
```

Description
- Displays the bit status of bit 8 to 1 of Pod A from the left of the data. Each bit can be set to H(HIGH), L(LOW), or X(Don't care).
- An error will occur if all 8 bits are not set.
- An error will occur, if the logic module is not installed at the channel (slot).

:TRIGger:CHANnel<x>:PODB

Function Sets/queries the bit pattern of Pod B, when the logic input module is installed at the channel (slot).

Syntax
```
:TRIGger:CHANnel<x>:PODB {(H|L|X)(H|L|X)
(H|L|X)(H|L|X)(H|L|X)(H|L|X)(H|L|X)(H|L|X)}
(From the left of the data, bit 8 to bit 1)
:TRIGger:CHANnel<x>:PODB?
<x>=1 to 16
H=HIGH, L=LOW, X=Don't care
```

Example
```
(Below is an example when setting the
bits8, 5, and 2 of Pod B of channel 1 to
High; bits 7, 4, and 1 to Low; and bits 6
and 3 to Don't care.)
:TRIGGER:CHANNEL1:PODB HLXHLXHL
:TRIGGER:CHANNEL1:PODB?→:TRIGGER:
CHANNEL1:PODB HLXHLXHL
```

Description
- Displays the bit status of bit 8 to 1 of Pod B from the left of the data. Each bit can be set to H(HIGH), L(LOW), or X(Don't care).
- An error will occur if all 8 bits are not set.
- An error will occur, if the logic module is not installed at the channel (slot).

:TRIGger:DELay

Function Sets/queries the trigger delay (time between trigger point and trigger position).

Syntax
```
:TRIGger:DELay {<Time>}
:TRIGger:DELay?
<Time>=0 to 1s (in steps of 500ns)
```

Example
```
:TRIGGER:DELLAY 2US
:TRIGGER:DELAY?→:TRIGGER:DELAY 2.00E-06
```

Description If timebase is external clock, the value is fixed at 0. On this oscilloscope, the delay is the time difference between the trigger point and the trigger position. You can set/query the trigger position using the ":TRIGger:DREFerence" command.

:TRIGger:DREFerence(Delay REFERENCE)

Function Sets/queries the trigger position.

Syntax
```
:TRIGger:DREFerence {<NRf>}
:TRIGger:DREFerence?
<NRf>=0 to 100(%)
```

Example
```
:TRIGGER:DREFERENCE 10
:TRIGGER:DREFERENCE?→:TRIGGER:
DREFERENCE 10
```

:TRIGger:EOA?(Edge On A)

Function Queries all Edge on A trigger settings.

Syntax
```
:TRIGger:EOA?
```

Example
```
:TRIGGER:EOA?→:TRIGGER:EOA:
CHANNEL1 RISE;CHANNEL2 OFF;CHANNEL3 OFF;
CHANNEL4 OFF;CHANNEL5 OFF;CHANNEL6 OFF;
CHANNEL7 OFF;CHANNEL8 OFF;CHANNEL9 OFF;
CHANNEL10 OFF;CHANNEL11 OFF;
CHANNEL12 OFF;CHANNEL13 OFF;
CHANNEL14 OFF;CHANNEL15 OFF;CHANNEL16 OFF
```

:TRIGger:EOA:CHANnel<x>

Function Sets/queries the edge of each channel of the Edge on A trigger.

Syntax
```
:TRIGger:EOA:CHANnel<x> {OFF|FALL|RISE}
:TRIGger:EOA:CHANnel<x>?
<x>=1 to 16
```

Example
```
(Below is an example on CH1.)
:TRIGGER:EOA:CHANNEL1 RISE
:TRIGGER:EOA:CHANNEL1?→:TRIGGER:EOA:
CHANNEL1 RISE
```

Description
- Cannot set the channels which is being used in condition A.
- An error will occur, if there is no module installed or if there is a logic input module is installed at the channel (slot).

:TRIGger:EOR?(Edge OR)

Function Queries all OR trigger settings.

Syntax
```
:TRIGger:EOR?
```

Example
```
:TRIGGER:EOR→:TRIGGER:EOR:CHANNEL1 RISE;
CHANNEL2 OFF;CHANNEL3 OFF;CHANNEL4 OFF;
CHANNEL5 OFF;CHANNEL6 OFF;CHANNEL7 OFF;
CHANNEL8 OFF;CHANNEL9 OFF;CHANNEL10 OFF;
CHANNEL11 OFF;CHANNEL12 OFF;
CHANNEL13 OFF;CHANNEL14 OFF;
CHANNEL15 OFF;CHANNEL16 OFF
```

:TRIGger:EOR:CHANnel<x>

Function Sets/queries the edge of each channel of the OR trigger.

Syntax
```
:TRIGger:EOR:CHANnel<x> {OFF|FALL|RISE}
:TRIGger:EOR:CHANnel<x>?
<x>=1 to 16
```

Example
```
(Below is an example on CH1.)
:TRIGGER:EOR:CHANNEL1 RISE
:TRIGGER:EOR:CHANNEL1?→:TRIGGER:EOR:
CHANNEL1 RISE
```

Description An error will occur, if there is no module installed or if there is a logic input module is installed at the channel (slot).

:TRIGger:HOLDoff

Function Sets/queries the holdoff time.

Syntax
```
:TRIGger:HOLDoff {<Time>}
:TRIGger:HOLDoff?
<Time>=0 to 1s (in steps of 500ns)
```

Example
```
:TRIGGER:HOLDOFF 500NS
:TRIGGER:HOLDOFF?→:TRIGGER:
HOLDOFF 0.5E-06
```

:TRIGger:MODE

Function Sets/queries the trigger mode.

Syntax
```
:TRIGger:MODE {AUTO|ALEVel|NORMal}
:TRIGger:MODE?
```

Example
```
:TRIGGER:MODE AUTO
:TRIGGER:MODE?→:TRIGGER:MODE AUTO
```

Description "ALEVel" can be specified only for simple triggers.

:TRIGger:SIMPle?

Function Queries all simple trigger settings.

Syntax
```
:TRIGger:SIMPle?
```

Example
```
:TRIGGER:SIMPLE?→:TRIGGER:SIMPLE:
SOURCE 1;SLOPE RISE;LEVEL 0.000E+03;
HYSTERESIS OFF
```

:TRIGger[:SIMPle]:HYSTeresis

Function Sets/queries the trigger hysteresis of the channel specified by "TRIGger[:SIMPle]:SOURce" of the simple trigger.

Syntax
```
:TRIGger[:SIMPle]:HYSTeresis {OFF|HIGH|
LOW}
:TRIGger[:SIMPle]:HYSTeresis?
```

Example
```
:TRIGGER:SIMPLE:HYSTERESIS HIGH
:TRIGGER:SIMPLE:HYSTERESIS?→:TRIGGER:
SIMPLE:HYSTERESIS HIGH
```

Description
- An error will occur, if the trigger source is LINE.
- An error will occur, if the voltage, the temperature, or the strain module is not installed.

:TRIGger[:SIMPle]:LEVel

Function Sets/queries the trigger level of the channel specified by "TRIGger[:SIMPle]:SOURce" of the simple trigger.

Syntax
```
:TRIGger[:SIMPle]:LEVel {<Voltage>|<NRf>}
:TRIGger[:SIMPle]:LEVel?
<Voltage>=Within the 8 div on the screen
          (for voltage modules)
<NRf>=-3000 to 3000(when set to °C on the
      temperature module)
     =-2000 to 2000 (×10^-6 strain, for
      strain module)
```

Example
```
:TRIGGER:SIMPLE:LEVEL 0V
:TRIGGER:SIMPLE:LEVEL?→:TRIGGER:SIMPLE:
LEVEL 0.000E+03
```

Description
- An error will occur, if the trigger source is LINE.
- An error will occur, if the voltage, the temperature, or the strain module is not installed.

:TRIGger[:SIMPle]:LOGic

Function Sets/queries the source bit, when the logic input module is installed at the trigger source channel (slot).

Syntax
```
:TRIGger[:SIMPle]:LOGic{A1|A2|A3|A4|A5|
A6|A7|A8|B1|B2|B3|B4|B5|B6|B7|B8}
:TRIGger[:SIMPle]:LOGic?
```

Example
```
:TRIGGER:SIMPLE:LOGIC A1
:TRIGGER:SIMPLE:LOGIC?→:TRIGGER:SIMPLE:
LOGIC A1
```

Description An error will occur, if the logic input module is not installed or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:TRIGger[:SIMPle]:SLOPe

Function Sets/queries the trigger slope for the simple trigger on the channel designated by "TRIGger[:SIMPle]:SOURce".

Syntax
```
:TRIGger[:SIMPle]:SLOPe {BOTH|FALL|RISE}
:TRIGger[:SIMPle]:SLOPe?
```

Example
```
:TRIGGER:SIMPLE:SLOPE RISE
:TRIGGER:SIMPLE:SLOPE?→:TRIGGER:SIMPLE:
SLOPE RISE
```

Description An error will occur, if the trigger source is LINE.

:TRIGger[:SIMPle]:SOURce

Function Sets/querier the trigger source for a simple trigger.

Syntax
```
:TRIGger[:SIMPle]:SOURce {<NRf>|EXTernal|
LINE}
:TRIGger[:SIMPle]:SOURce?
<NRf>=1 to 18 (However, 17 and 18
correspond to LOGIC1 and LOGIC2 of the
optional extended logic input.)
```

Example
```
:TRIGGER:SIMPLE:SOURCE 1
:TRIGGER:SIMPLE:SOURCE?→:TRIGGER:SIMPLE:
SOURCE 1
```

Description An error will occur, if there is no module installed at the channel (slot) or when x is greater than or equal to 17 and the optional extended logic input is not installed.

:TRIGger:TIMer?

Function Queries all settings relating to the timer trigger.
Syntax :TRIGger:TIMer?
Example :TRIGGER:TIMER?→:TRIGGER:TIMER:
DATE "99/01/01";TIME "00:00:00";
INTERVAL HOUR1

:TRIGger:TIMer:DATE

Function Sets/queries the trigger date of the timer trigger.
Syntax :TRIGger:TIMer:DATE <Character string>
:TRIGger:TIMer:DATE?
<Character string>=YY/MM/DD, refer to
IM701830-01J
Example :TRIGGER:TIMER:DATE "99/01/01"
:TRIGGER:TIMER:DATE→:TRIGGER:TIMER:
DATE "99/01/01"

:TRIGger:TIMer:INTerval

Function Sets/queries the trigger time interval of the timer trigger.
Syntax :TRIGger:TIMer:INTerval {MIN1|MIN2|MIN3|
MIN4|MIN5|MIN6|MIN7|MIN8|MIN9|MIN10|
MIN15|MIN20|MIN25|MIN30|MIN40|MIN45|
MIN50|HOUR1|HOUR2|HOUR3|HOUR4|HOUR5|
HOUR6|HOUR7|HOUR8|HOUR9|HOUR10|HOUR11|
HOUR12|HOUR18|HOUR24}
:TRIGger:TIMer:INTerval?
Example :TRIGGER:TIMER:INTERVAL HOUR1
:TRIGGER:TIMER:INTERVAL?→:TRIGGER:TIMER:
INTERVAL HOUR1

:TRIGger:TIMer:TIME

Function Sets/queries the trigger time of the timer trigger.
Syntax :TRIGger:TIMer:TIME <Character string>
:TRIGger:TIMer:TIME?
<Character string>=HH:MM:SS, refer to
IM701830-01J
Example :TRIGGER:TIMER:TIME "12:34:56"
:TRIGGER:TIMER:TIME→:TRIGGER:TIMER:TIME
"12:34:56"

:TRIGger:TYPE

Function Selects/queries the trigger type.
Syntax :TRIGger:TYPE {ABN|ADB|BGTime|BLTime|
BTOut|EOA|EOR|SIMPle|WINDow}
:TRIGger:TYPE?
Example :TRIGGER:TYPE SIMPLE
:TRIGGER:TYPE?→:TRIGGER:TYPE SIMPLE

:TRIGger:WINDow?

Function Queries all window trigger settings.
Syntax :TRIGger:WINDow?
Example :TRIGGER:WINDOW?→:TRIGGER:WINDOW:
SOURCE 1;CONDITION IN;HYSTERESIS OFF;
CENTER 0.00E+00;WIDTH 1.00E+00

:TRIGger:WINDow:CENTer

Function Sets/queries the window trigger center.
Syntax :TRIGger:WINDow:CENTer {<Voltage>|<NRf>}
:TRIGger:WINDow:CENTer?
<Voltage>=Within the 8 div on the screen
(for voltage modules)
<NRf>=-3000 to 3000 (when set to °C on
the temperature module)
=-2000 to 2000 ($\times 10^{-6}$ strain, for
strain module)
Example :TRIGGER:WINDOW:CENTER 5V
:TRIGGER:WINDOW:CENTER?→:TRIGGER:WINDOW:
CENTER 5.0E+00
Description An error will occur, if the voltage, the temperature, or the strain module is not installed.

:TRIGger:WINDow:CONDition

Function Sets/queries the window trigger condition.
Syntax :TRIGger:WINDow:CONDition {IN|OUT}
:TRIGger:WINDow:CONDition?
Example :TRIGGER:WINDOW:CONDITION IN
:TRIGGER:WINDOW:CONDITION?→:TRIGGER:
WINDOW:CONDITION IN

:TRIGger:WINDow:HYSTeresis

Function Sets/queries the trigger hysteresis of the window trigger.
Syntax :TRIGger:WINDow:HYSTeresis {OFF|HIGH|LOW}
:TRIGger:WINDow:HYSTeresis?
Example :TRIGGER:WINDOW:HYSTERESIS HIGH
:TRIGGER:WINDOW:HYSTERESIS?→:TRIGGER:
WINDOW:HYSTERESIS HIGH
Description An error will occur, if the voltage, the temperature, or the strain module is not installed.

:TRIGger:WINDow:SOURce

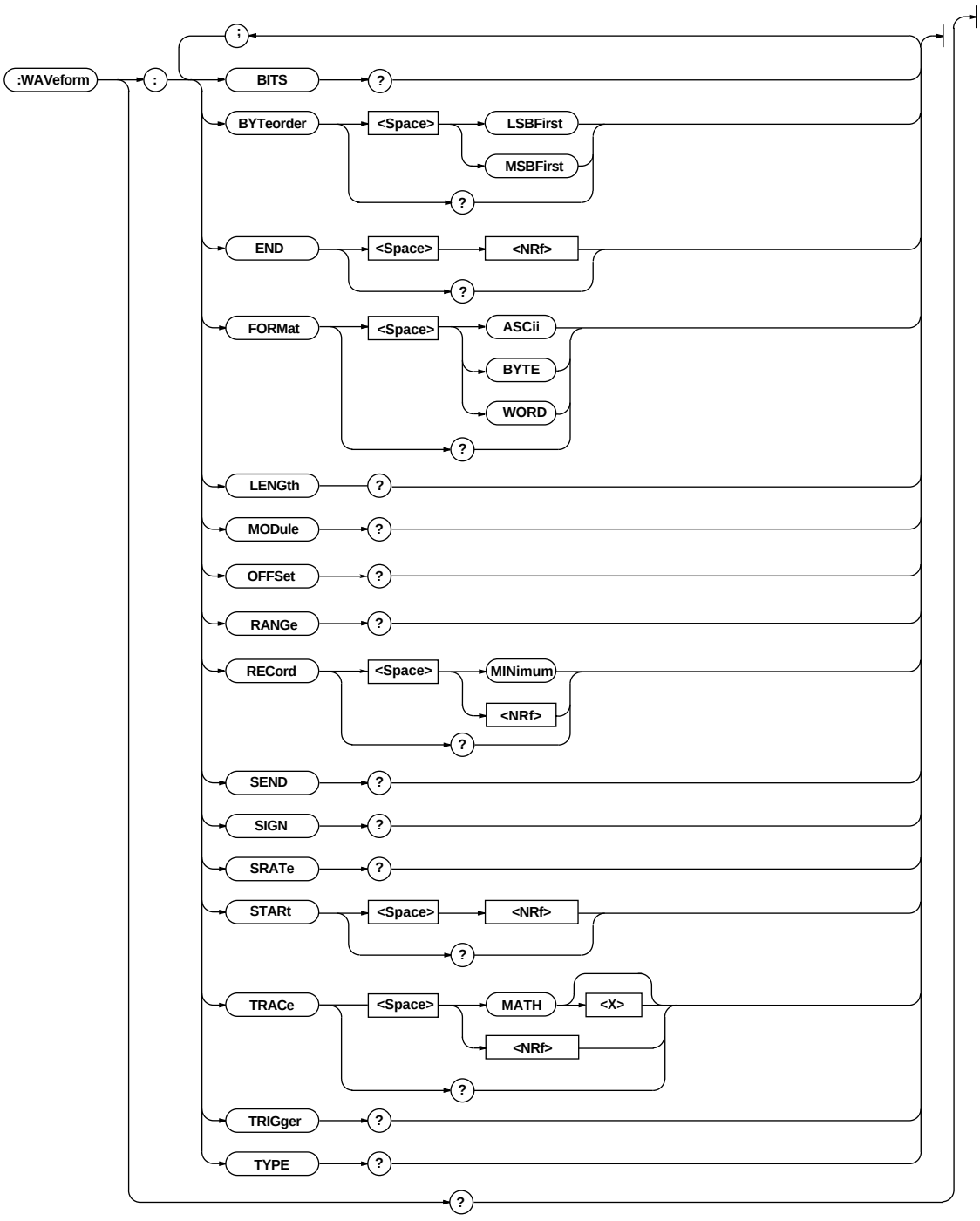
Function Sets/queries the trigger source of the window trigger.
Syntax :TRIGger:WINDow:SOURce {<NRf>}
:TRIGger:WINDow:SOURce?
<NRf>=1 to 16
Example :TRIGGER:WINDOW:SOURCE 1
:TRIGGER:WINDOW:SOURCE?→:TRIGGER:
WINDOW:SOURCE 1
Description An error will occur, if there is no module installed at the channel (slot).

:TRIGger:WINDow:WIDTh

Function Sets/queries the width of the window trigger.
Syntax :TRIGger:WINDow:WIDTh {<Voltage>|<NRf>}
:TRIGger:WINDow:WIDTh?
<Voltage>=Within the 8 div on the screen
(for voltage modules)
<NRf>=-3000 to 3000 (when set to °C on
the temperature module)
=-2000 to 2000 ($\times 10^{-6}$ strain, for
strain module)
Example :TRIGGER:WINDOW:WIDTH 5V
:TRIGGER:WINDOW:WIDTH?→:TRIGGER:WINDOW:
WIDTH 5.0E+00
Description An error will occur, if the voltage, the temperature, or the strain module is not installed.

4.27 WAVeform Group

The commands in the WAVeform group are used to make settings and queries about acquired waveform data. There is no front panel key for this function.



:WAVeform?

Function Queries all information relating to the waveform data.
Syntax :WAVeform?
Example :WAVEFORM?→:WAVEFORM:TRACE 1;
FORMAT WORD;BYTEORDER LSBFIRST;RECORD 0;
START 0;END 1001

:WAVeform:BITS?

Function Queries the bit length of the waveform data designated by "WAVeform:TRACe."
Syntax :WAVeform:BITS?
Example :WAVEFORM:BITS?→8

:WAVeform:BYTeorder

Function Sets/queries the order used to transmit words consisting of multiple bytes.
Syntax :WAVeform:BYTeorder {LSBFirst|MSBFirst}
:WAVeform:BYTeorder?
Example :WAVEFORM:BYTEORDER LSBFIRST
:WAVEFORM:BYTEORDER?→:WAVEFORM:
BYTEORDER LSBFIRST

:WAVeform:END

Function Sets/queries the point at which the last item of data of the waveform designated by "WAVeform:TRACe" is to be located.
Syntax :WAVeform:END {<NRf>}
:WAVeform:END?
<NRf>=1 to (total data points -1)
Example :WAVEFORM:END 0
:WAVEFORM:END?→:WAVEFORM:END 0
Description You can query total data length (points) with the ":WAVeform:LENgth" command.

:WAVeform:FORMat

Function Sets/queries the format in which the data is to be transmitted.
Syntax :WAVeform:FORMat {ASCii|BYTE|WORD}
:WAVeform:FORMat?
Example :WAVEFORM:FORMAT WORD
:WAVEFORM:FORMAT?→:WAVEFORM:FORMAT WORD
Description For information on the different formats selected with this command, refer to the Description section of "WAVeform:SEND?."

:WAVeform:LENGth?

Function Queries the number of records (total number of data points) in the waveform designated by "WAVeform:TRACe."
Syntax :WAVeform:LENGth?
Example :WAVEFORM:LENGTH?→1002
Description Total number of data points varies depending on the instrument settings. For details, refer to "Relationship between the Time Axis Setting, Sample Rate and Record Length" in the User's Manual IM701830-01E.

:WAVeform:MODule?

Function Queries the module of the waveform specified with "WAVeform:TRACe."
Syntax :WAVeform:MODule?
Example :WAVEFORM:MODULE?→M701856
Description The values returned by each module are as follows.

NOMODULE	No module
M701855	High-Speed Isolation Module
M701856	High-Speed Module
M701852	High-Resolution, High-Voltage Isolation Module
M701853	High-Resolution, Isolation Module
M701860	Temperature Module
M701870	Logic Input Module
M701880	Strain Module

:WAVeform:OFFSet?

Function Queries the offset value used in converting the waveform data specified with "WAVeform:TRACe" to physical values.
Syntax :WAVeform:OFFSet?
Example :WAVEFORM:OFFSET?→5.000E+00
Description This offset value is used when converting the <block data> output with the "WAVeform:SEND?" command to physical values.

:WAVeform:RANGe?

Function Queries the range value used in converting the waveform data specified with "WAVeform:TRACe" to physical values.
Syntax :WAVeform:RANGe?
Example :WAVEFORM:RANGE?→5.000E+00
Description This range value is used when converting the <block data> output with the "WAVeform:SEND?" command to physical values.

:WAVeform:RECord

Function Sets/queries the target record No. for the WAVeform group.
Syntax :WAVeform:RECord {MINimum|<NRf>}
:WAVeform:RECord?
<NRf>=0 to -999
Example :WAVEFORM:RECORD 0
:WAVEFORM:RECORD?→:WAVEFORM:RECORD 0
Description If "MINimum" is specified, the record number is set to the minimum value. The record numbers that can be selected vary depending on the extension memory and the acquisition settings. For details, refer to the User's Manual IM701830-01E.

:WAVeform:SEND?

Function Queries the waveform data of the waveform designated by "WAVeform:TRACe."

Syntax :WAVeform:SEND?

Example :WAVeform:SEND?→#9 (number of bytes [9-digit value])(data byte string)
or, <NRf>,<NRf>,...

Description The format that is output with the "WAVeform:SEND?" command varies according to the "WAVeform:FORMat" setting.

(1) If set to "ASCii"

The data is output as <voltage>,<voltage>,..., <voltage> for the voltage modules.

The data is output as <temperature>, <temperature>,...,<temperature> and the unit specified with "CHANnel<x>:TEMPerature:UNIT" for the temperature module.

The data is output as <NR1>,<NR1>,...,<NR1> for the logic input module.

<NR1> is the 16-bit bit pattern (MSB:Bits 8 to 1 of PodB, LSB:Bits 8 to 1 of PodA) converted to a decimal number.

The data is output as <strain>,<strain>,...,<strain> for the strain module.

The data is output as <NRf>,<NRf>,...,<NRf> for the math waveform.

(2) If set to "BYTE" or "WORD"

The <block data> format is used for the output on all modules.

The data includes a sign except for the logic input module.

For voltage modules, the data can be converted to a voltage value with the following equation.

$$\text{Voltage} = \frac{\text{Range} \times \text{Data} \times 8}{\text{Division}} + \text{Offset}$$

"BYTE" : Division =250
"WORD" : Division =64000
Range=Returned value of "WAVeform:RANGe?"
Offset=Returned value of "WAVeform:OFFSet?"

For temperature module, the data can be converted to a temperature value with the following equation.

Temperature (°C)=Data × Division
"BYTE" : Division=25.6
"WORD" : Division=0.1

For logic input module, the data is output in the following format.
"BYTE" : 8-bit bit pattern (Bits 8 to 1 of PodB)
"WORD" : 16-bit bit pattern (MSB side: Bits 8 to 1 of PodB, LSB side: Bits 8 to 1 of PodA)

For strain module, the data can be converted with the following equation.
Strain($\times 10^{-6}$)=Range×Data+Offset
Range=Returned value of "WAVeform:RANGe?"
Offset=Returned value of "WAVeform:OFFSet?"

For math waveforms, the data can be converted with the following equation.

$$\text{Computed value} = \frac{\text{Range} \times \text{Data} \times 8}{\text{Division}} + \text{Offset}$$

"BYTE" : Division=250
"WORD" : Division=64000
Range=Returned value of "WAVeform:RANGe?"
Offset=Returned value of "WAVeform:OFFSet?"

:WAVeform:SIGN?

Function Queries the presence of the sign, when querying the waveform specified with "WAVeform:TRACe" in block data form.

Syntax :WAVeform:SIGN?

Example :WAVEFORM:SIGN?→1

Description "0" is returned if the waveform is a logic input module waveform, otherwise "1" is returned.

:WAVeform:SRATe? (Sample RATE)

Function Queries the sampling rate for the record designated by "WAVeform:RECord".

Syntax :WAVeform:SRATe?

Example :WAVEFORM:SRATE?→0.1E+09

:WAVeform:STARt

Function Sets/queries the point at which the first item of data of the waveform designated by "WAVeform:TRACe" is to be located.

Syntax :WAVeform:STARt {<NRf>}
:WAVeform:STARt?
<NRf>=1 to (total data points -1)

Example :WAVEFORM:START 0
:WAVEFORM:START?→:WAVEFORM:START 0

Description Total number of data points can be queried with "WAVeform:LENGth?."

:WAVeform:TRACe

Function Sets/queries the target waveform for the WAVeform group.

Syntax :WAVeform:TRACe {<NRf>|MATH[<1-2>]}
:WAVeform:TRACe?
<x>=1,2
{<NRf>}=1 to 8

Example :WAVEFORM:TRACE 1
:WAVEFORM:TRACE?→:WAVEFORM:TRACE 1

Description An error will occur, if there is no module installed at the channel (slot).

:WAVeform:TRIGger?

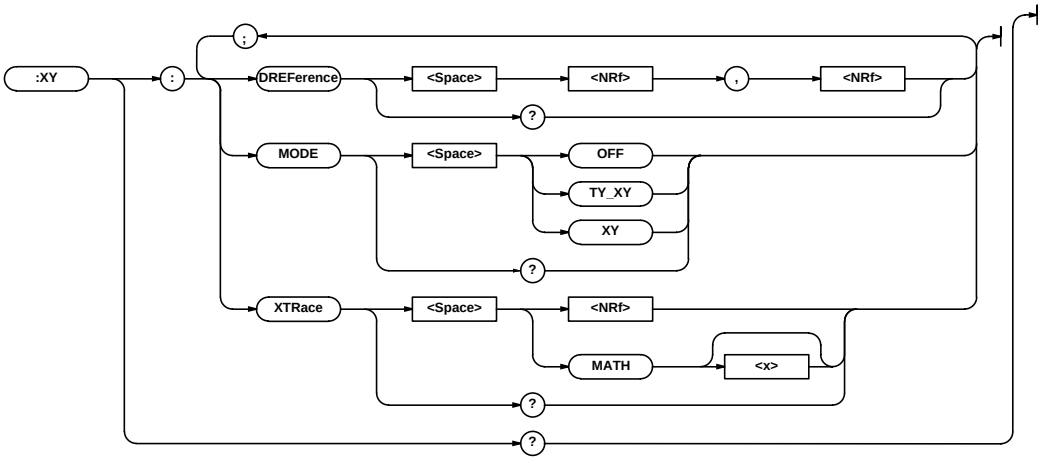
Function	Queries the trigger point for the record designated by "WAVeform:RECord".
Syntax	:WAVeform:TRIGger?
Example	:WAVEFORM:TRIGGER?→2000
Description	Returns the number of points between the record start and the trigger position.

:WAVeform:TYPE?

Function	Queries the acquisition mode for the waveform specified by the "WAVeform:TRACe" command.
Syntax	:WAVeform:TYPE?
Example	:WAVEFORM:TYPE?→NORMAL
Description	"AVERage," "ENVelope," or "NORMal" will be returned.

4.28 XY Group

The commands in the XY group are used to make settings and queries about the XY display. You can make the same settings that can make using the X-Y key (SHIFT+DISPLAY) on the front panel.



:XY?

Function	Queries all X/Y display settings.
Syntax	:XY?
Example	:XY?→:XY:MODE XY;XTRACE 1; DREFERENCE -5.00E+00,5.00E+00

:XY:DREFerence (Division REFERENCE)

Function	Sets/queries range for the T/Y waveform on the X/Y display.
Syntax	:XY:DREFerence {<NRf>,<NRf>} :XY:DREFerence? <NRf>=-5 to 5div(in steps of 10 div/ display-record-length)
Example	:XY:DREFERENCE -4,4 :XY:DREFERENCE?→:XY: DREFERENCE -4.00E+00,4.00E+00

:XY:MODE

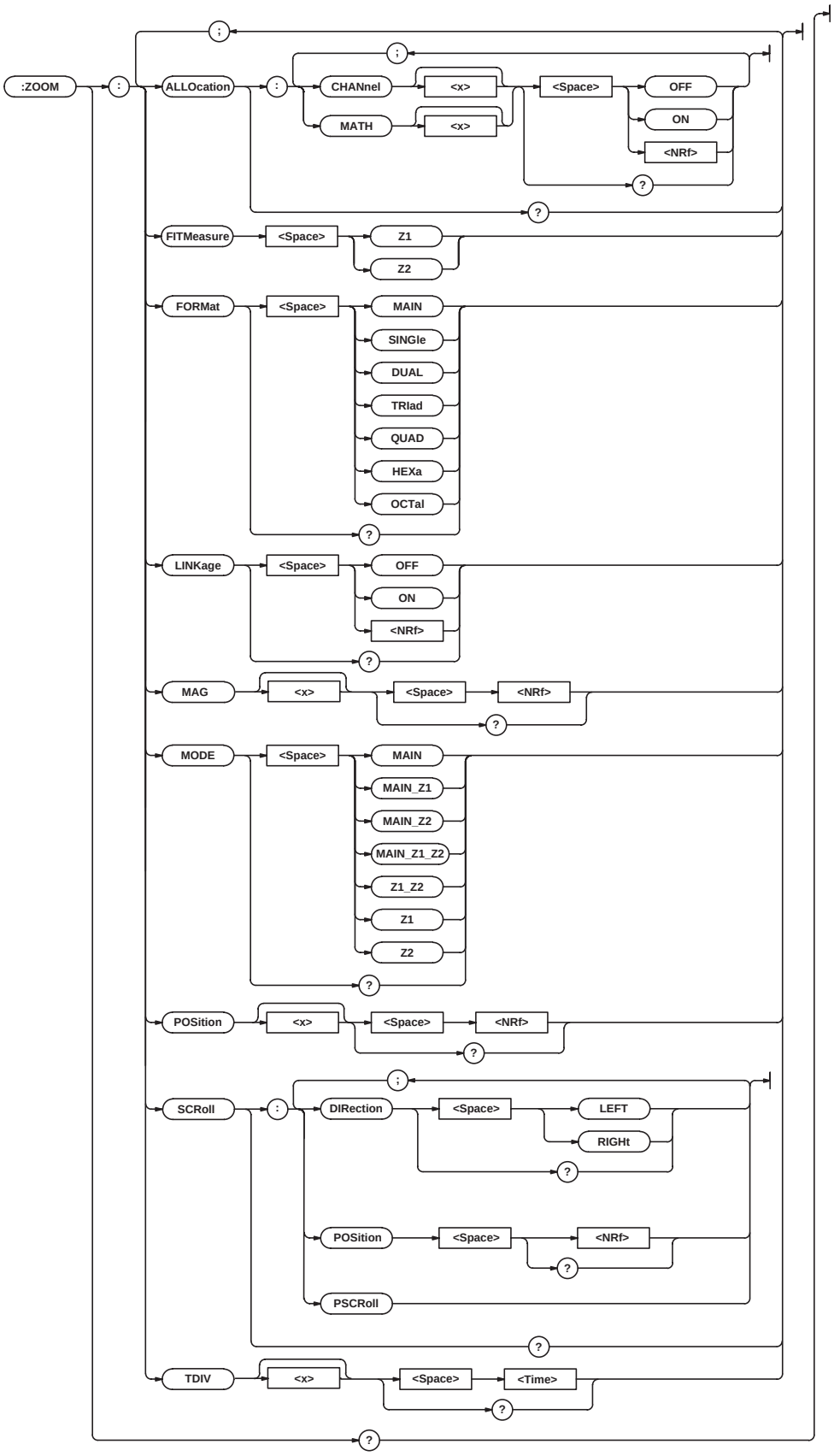
Function	Sets/queries the display mode.
Syntax	:XY:MODE {OFF|TY_XY|XY} :XY:MODE?
Example	:XY:MODE XY :XY:MODE?→:XY:MODE XY

:XY:XTRace

Function	Sets/queries the X-axis channel for the X/Y display.
Syntax	:XY:XTRace {<NRf>|MATH[<x>]} :XY:XTRace? <NRf>=1 to 8 <x>=1, 2
Example	:XY:XTRACE 0 :XY:XTRACE?→:XY:XTRACE 0
Description	An error will occur, if there is no module installed at the channel (slot).

4.29 ZOOM Group

The commands in the ZOOM group are used to make settings and queries about the zoom function. These settings can also be made using the ZOOM key on the front panel.



:ZOOM?

Function Queries all waveform zoom settings.
Syntax :ZOOM?
Example :ZOOM?→:ZOOM:MODE MAIN;FORMAT MAIN;
ALLOCATION:CHANNEL1 1;CHANNEL2 1;
CHANNEL3 1;CHANNEL4 1;CHANNEL5 1;
CHANNEL6 1;CHANNEL7 1;CHANNEL8 1;
CHANNEL9 1;CHANNEL10 1;CHANNEL11 1;
CHANNEL12 1;CHANNEL13 1;CHANNEL14 1;
CHANNEL15 1;CHANNEL16 1;CHANNEL17 1;
CHANNEL18 1;MATH1 0;MATH2 0;:ZOOM:
MAG1 2.5;MAG2 5.0;TDIV1 0.2E-06;
TDIV2 0.1E-06;POSITION1 0.00E+00;
POSITION2 0.00E+00;LINKAGE 0;SCROLL:
DIRECTION LEFT;POSITION 0.0000;
SPEED 0.0050

:ZOOM:ALLOcation?

Function Queries settings for the zoomed waveform.
Syntax :ZOOM:ALLOcation?
Example :ZOOM:ALLOCATION?→:ZOOM:ALLOCATION:
CHANNEL1 1;CHANNEL2 0;CHANNEL3 0;
CHANNEL4 0;CHANNEL5 0;CHANNEL6 0;
CHANNEL7 0;CHANNEL8 0;MATH1 0;MATH2 0

:ZOOM:ALLOcation:{CHANnel<x>|MATH<x>}

Function Selects/queries the zoomed waveform.
Syntax :ZOOM:ALLOcation:{CHANnel<x>|
MATH<x>} {<Boolean>}
:ZOOM:ALLOcation:{CHANnel<x>|MATH<x>}?
<x>(CHANnel)=1 to 8
<x>(MATH)=1, 2
Example :ZOOM:ALLOCATION:CHANNEL1 ON
:ZOOM:ALLOCATION:CHANNEL1?→:ZOOM:
ALLOCATION:CHANNEL1 1
Description An error will occur, if there is no module installed at the channel (slot).

:ZOOM:FITMeasure

Function Move the range of the automatic measurement of waveform parameters to the zoom waveform display frame.
Syntax :ZOOM:FITMeasure {Z1|Z2}
Example :ZOOM:FITMEASURE Z1

:ZOOM:FORMat

Function Sets/queries the zoom display format.
Syntax :ZOOM:FORMat {MAIN|SINGle|DUAL|TRIad|
QUAD|HEXa|OCTal}
:ZOOM:FORMat?
Example :ZOOM:FORMAT SINGLE
:ZOOM:FORMAT?→:ZOOM:FORMAT SINGLE

:ZOOM:LINKage

Function Enables/disables/queries zoom box linkage.
Syntax :ZOOM:LINKage {<Boolean>}
:ZOOM:LINKage?
Example :ZOOM:LINKAGE ON
:ZOOM:LINKAGE?→:ZOOM:LINKAGE 1

:ZOOM:MAG<x>

Function Sets/queries the zoom ratio.
Syntax :ZOOM:MAG<x> {<NRf>}
:ZOOM:MAG<x>?
<NRf>=magnification up to 10 points.
Refer to the User's Manual
IM701830-01E.
Example :ZOOM:MAG1 2.5
:ZOOM:MAG1?→:ZOOM:MAG1 2.5

:ZOOM:MODE

Function Sets/queries the zoom display mode.
Syntax :ZOOM:MODE {MAIN|MAIN_Z1|MAIN_Z2|
MAIN_Z1_Z2|Z1_Z2|Z1|Z2}
:ZOOM:MODE?
Example :ZOOM:MODE MAIN_Z1_Z2
:ZOOM:MODE?→:ZOOM:MODE MAIN_Z1_Z2

:ZOOM:POSition<x>

Function Sets/queries the zoom box position.
Syntax :ZOOM:POSition<x> {<NRf>}
:ZOOM:POSition<x>?
<x>=1, 2
<NRf>=-5 to 5div(in steps of 10 div/
display-record-length)
Example :ZOOM:POSITION1 2
:ZOOM:POSITION1?→:ZOOM:POSITION1 2.00E+00

:ZOOM:SCRoll?

Function Queries all settings relating to the auto scroll of the waveform.
Syntax :ZOOM:SCRoll?
Example :ZOOM:SCROLL?→:ZOOM:SCROLL:
DIRECTION LEFT;POSITION 0.0000

:ZOOM:SCRoll:ASCRoll

Function Executes auto scrolling of the waveform.
Syntax :ZOOM:SCRoll:ASCRoll
Example :ZOOM:SCROLL:ASCROLL
Description To stop the auto scrolling operation, use ":ZOOM:SCRoll:STOP."

:ZOOM:SCRoll:DIRection

Function Sets/queries the direction to auto scroll the waveform.
Syntax :ZOOM:SCRoll:DIRection {LEFT|RIGHt}
:ZOOM:SCRoll:DIRection?
Example :ZOOM:SCROLL:DIRECTION LEFT
:ZOOM:SCROLL:DIRECTION?→:ZOOM:SCROLL:
DIRECTION LEFT

:ZOOM:SCRoll:POSition

Function Sets/queries the display position of the waveform data on the screen.
Syntax :ZOOM:SCRoll:POSition {<NRf>}
:ZOOM:SCRoll:POSition?
<NRf>=0 to 100 (%, in steps of 0.1%)
Example :ZOOM:SCROLL:POSITION 10.0
:ZOOM:SCROLL:POSITION?→:ZOOM:SCROLL:
POSITION 10.0
Description If all the acquired data is being displayed on the screen, this command is irrelevant.

:ZOOM:SCRoll:PSCRoll

Function Execute the page scroll of the waveform.
Syntax :ZOOM:SCRoll:PSCRoll
Example :ZOOM:SCROLL:PSCROLL

:ZOOM:SCRoll:SPEed

Function Sets/queries the auto scrolling speed of the waveform.
Syntax :ZOOM:SCRoll:SPEed {<NRf>}
:ZOOM:SCRoll:SPEed?
<NRf>=0.0001,0.0002,0.0005,0.0010,0.0020,
0.0050,0.0100,0.0200,0.0500,0.1000,
0.2000,0.5000,1.0000,2.0000,5.0000
Example :ZOOM:SCROLL:SPEED 0.5000
:ZOOM:SCROLL:SPEED?→:ZOOM:SCROLL:
SPEED 0.5000

:ZOOM:SCRoll:STOP

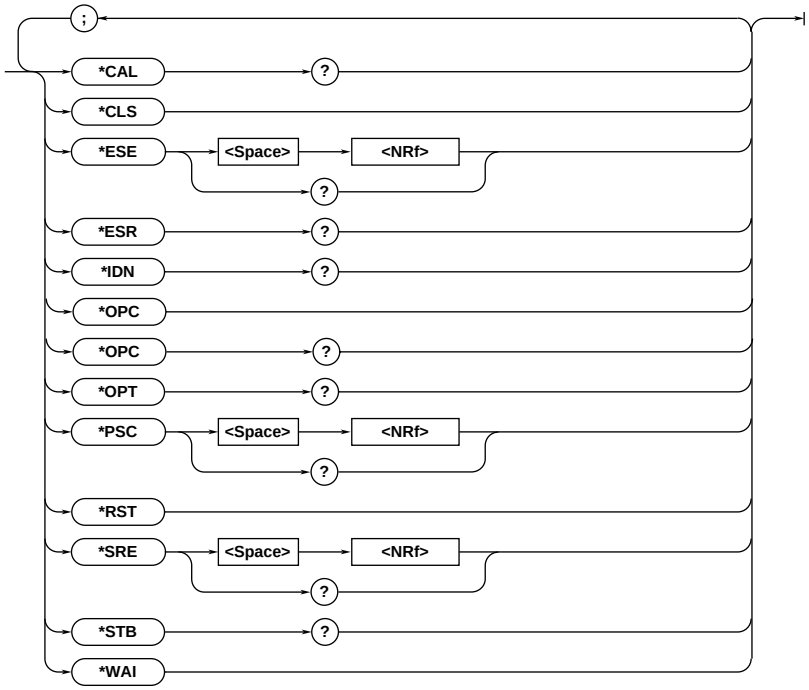
Function Stops the auto scrolling of the waveform.
Syntax :ZOOM:SCRoll:STOP
Example :ZOOM:SCROLL:STOP
:ZOOM:SCRoll:ASCRoll
:ZOOM:SCRoll:SPEed
Sets/queries the auto scrolling speed of
the waveform.
:ZOOM:SCRoll:STOP
Stops the auto scrolling of the waveform.

:ZOOM:TDIV<x>

Function Sets/queries the zoomed waveform's T/div value.
Syntax :ZOOM:TDIV<x> {<Time>}
:ZOOM:TDIV<x>?
<x>=1, 2
<Time>=T/div up to 10 points
Example :ZOOM:TDIV1 200NS
:ZOOM:TDIV1?→:ZOOM:TDIV1 0.2E-06

4.30 Common Command Group

The commands in the common command group are independent of the instrument's functions, and are specified in IEEE 488.2-1987. There is no front panel key that corresponds to this group.



*CAL? (CALibrate)

Function Performs calibration and queries about the result.

Syntax `*CAL?`

Example `*CAL?→0`

Description "0" will be returned if calibration is completed properly, and "1" will be returned if an abnormality has been detected during calibration.

*CLS (CLear Status)

Function Clears the standard event register, extended event register and error queue.

Syntax `*CLS`

Example `*CLS`

Description
- The output queue will also be cleared if a `*CLS` command is appended after the program message terminator.
- For details of the registers and queue, refer to Chapter 5.

*ESE (standard Event Status Enable register)

Function Sets the value for the standard event enable register/ queries about the current setting.

Syntax `*ESE {<NRf>}`
`*ESE?`
`<NRf>=0 to 255`

Example `*ESE 253`
`*ESE?→253`

Description
- <NRf> is the sum of the bits expressed as a decimal number.
- For example, if "`*ESE 253`" is set, the standard event enable register will be set to "11111101." This means that bit 2 of the standard event register is disabled so that bit 5 (ESB) of the status byte register will not be set to "1," even if an query error occurs.
- Default is "`*ESE 0`," i.e. all bits are disabled.
- The standard event enable register will not be cleared, even if an query is made using "`*ESE?`."
- For details of the standard event enable register, refer to page 5-3.

*ESR? (standard Event Status Register)

Function Queries about the value of the standard event register and clears it at the same time.

Syntax `*ESR?`

Example `*ESR?→32`

Description
- The sum of the bits is returned as a decimal value.
- It is possible to ascertain the type of event which has occurred, while SRQ is occurring.
- For example, if "32" is returned, this means that the standard event register is "00100000," i.e. the SRQ has occurred due to a command syntax error.
- If an query is made using "*ESR?," the standard event register will be cleared.
- For details of the standard event register, refer to page 5-3.

*IDN? (IDeNtify)

Function Queries about the instrument model.

Syntax `*IDN?`

Example `*IDN?→YOKOGAWA,701830,0,F1.10`

Description A reply consists of the following sequence: <Manufacturer>, <Model>, <Serial No.> and <Firmware version>. "0" is always returned as the <Serial No.>.

*LRN? (LeaRN)

Function Queries about all the current settings for the following command groups.
ACQuire, CHANnel<x>, TIMebase, TRIGger

Syntax `*LRN?`

Example
```
*LRN?→:ACQUIRE:RTOUT OFF;MODE NORMAL;
RLENGTH 1000;COUNT INFINITY;:CHANNEL1:
DISPLAY 1;LABEL "CH1 ";VOLTAGE:
COUPLING DC;POSITION 0.00;PROBE 10;
ZOOM 1.00;VDIV 0.50E+00;BWIDTH FULL;
INVERT 0;OFFSET 0.00E+00;LSCALE:MODE 0;:
CHANNEL3:DISPLAY 1;LABEL "CH2 ";
TEMPERATURE:TYPE J;UNIT C;
SCALE 1000.0,-200.0;BWIDTH FULL;RJC 1;:
CHANNEL5:DISPLAY 1;LABEL "CH5 ";LOGIC:
A1 1;A2 1;A3 1;A4 1;A5 1;A6 1;A7 1;A8 1;
B1 1;B2 1;B3 1;B4 1;B5 1;B6 1;B7 1;B8 1;
POSITION 0.00;ZOOM 1.00;BITMAP 1;:
TIMEBASE:SRATE 0.20E+06;TDIV 0.5E-03;:
TRIGGER:TYPE SIMPLE;MODE AUTO;
DELAY 0.0E+00;DREFERENCE 50;
HOLDOFF 0.1E-06;SIMPLE:SOURCE 1;
SLOPE RISE;LEVEL 0.50E+00;HYSTERESIS OFF
```

*OPC (OPeration Complete)

Function Sets bit 0 (OPC bit) of the standard event register to "1" when execution of an overlap command is completed.

Syntax `*OPC`

Example `*OPC`

Description
- For a description of the synchronization method using "*OPC," refer to page 3-7.
- Designation of an overlap command is performed using "COMMunicate:OPSE."
- Operation is not guaranteed if "*OPC" is not appended to the end of the message.

*OPC? (OPeration Complete)

Function After "*OPC?" is sent, "1" (ASCII) will be returned if execution of the designated overlap command has been completed.

Syntax `*OPC?`

Example `*OPC?→1`

Description
- For a description of the synchronization method using "*OPC?," refer to page 3-7.
- Designation of an overlap command is performed using "COMMunicate:OPSE."
- Operation is not guaranteed if "*OPC" is not appended to the end of the message.

*OPT? (OPTion)

Function Queries about installed options.

Syntax `*OPT?`

Example `*OPT?→8CHANNELS,HD,SCSI,USERDEFINE`

Description
- Indicates the number of channels, HD drive, and SCSI, and support or non-support of user defined expressions.
- This query must appear as the final query within the program message. Inclusion of a subsequent query within the same message will generate an error.

*PSC (Power-on Status Clear)

Function Selects/queries whether following registers are cleared when power is turned ON. However, they cannot be cleared if the parameter is "0" when rounded.
- Standard event enable register
- Extended event enable register
- Transit filter

Syntax
```
*PSC {<NRf>}
*PSC?
```
<NRf>=0(does not clear the registers), a value other than 0 (clears the registers)

Example
```
*PSC 1
*PSC?→1
```

Description For details of each register, refer to Chapter 5.

*RST (ReSet)

Function Resets the current setting for all the following command groups.
ACCumulate, ACQuire, CHANnel<x>, TIMedase, TRIGger

Syntax `*RST`

Example `*RST`

Description "`*OPC`" and "`*OPC?`" will also be reset.

*SRE (Service Request Enable register)

Function Sets the value of the service request enable register/ inquires about the current setting.

Syntax `*SRE <NRf>`
`*SRE?`
`<NRf>=0 to 255`

Example `*SRE 239`
`*SRE?→239`

Description
- <NRf> is the sum of the bits expressed as a decimal number. For example, if "`*ESE 239`" is set, the service request enable register will be set to "`11101111`." This means that bit 4 of the service request enable register is disabled, so that bit 5 (ESB) of the status byte register will not be set to "1," even if the output queue is not empty. However, bit 6 (MSS) of the status byte register is the MSS bit, so it will be ignored.
- Default is "`*SRE 0`," i.e. all bits are disabled.
- The service request enable register will not be cleared, even if a query is made using "`*SRE?`."
- For details of the service request enable register, refer to page 5-1.

*STB? (STatus Byte)

Function Queries about the value of the status byte register.

Syntax `*STB?`

Example `*STB?→4`

Description
- The sum of the bits expressed as a decimal value is returned.
- Bit 6 is MSS not RQS, since the register is read without serial polling. For example, if "`4`" is returned, the status byte register is set to "`00000100`," i.e. the error queue is not empty (an error has occurred).
- The status byte register will be cleared, even if a query is made using "`*STB?`."
- For details of the status byte register, refer to page 5-2.

*WAI (WAIt)

Function Waits for the command following "`*WAI`" until execution of the designated overlap command is completed.

Syntax `*WAI`

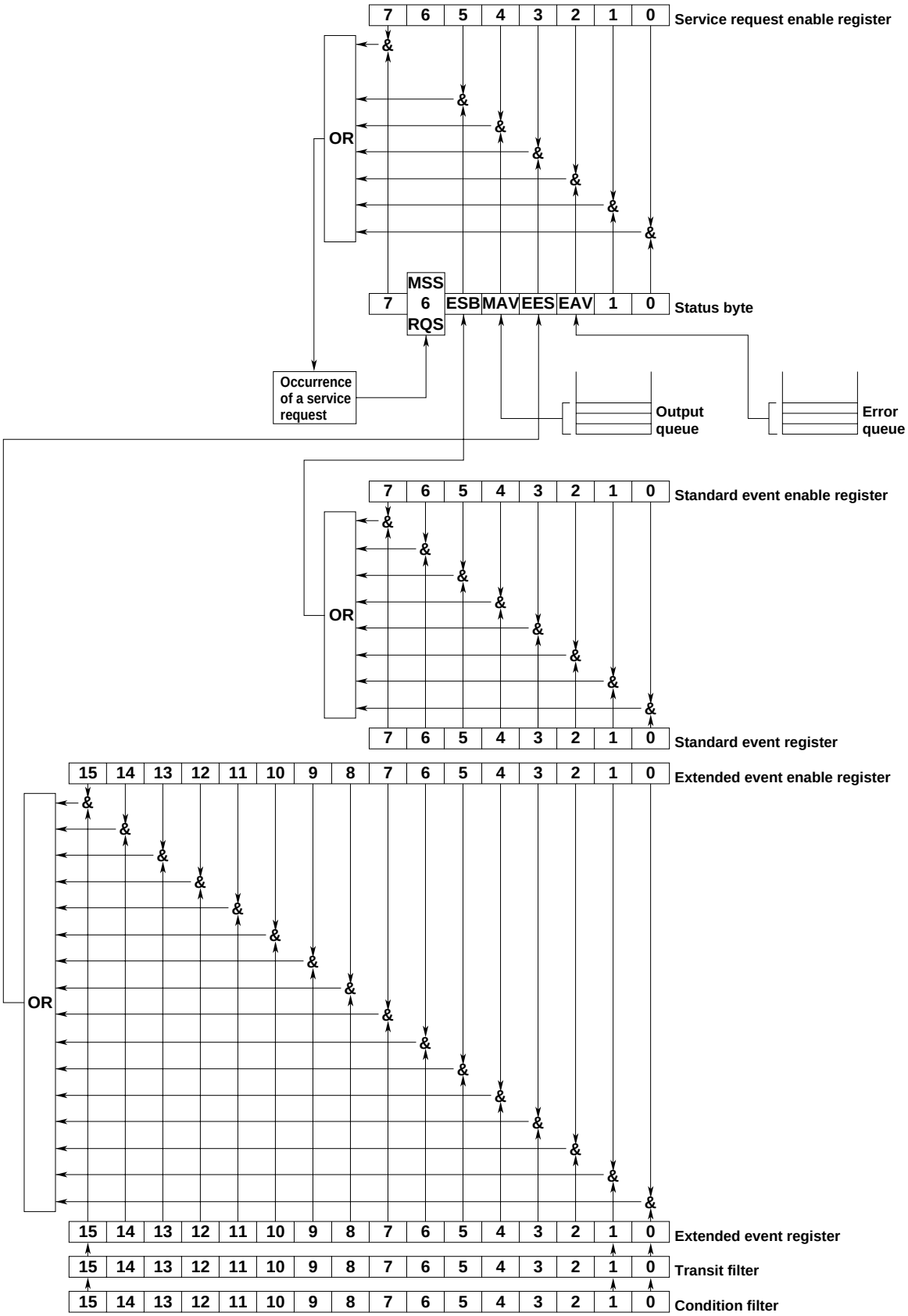
Example `*WAI`

Description
- For a description of the synchronization method using "`*WAI`," refer to page 3-7.
- Designation of overlap commands can be performed using "`COMMunicate:OPSE`."

5 Status Report

5.1 Overview of the Status Report

The figure below shows the status report which is read by a serial poll. This is an extended version of the one specified in IEEE 488.2-1992.



Overview of Registers and Queues

Name	Function	Writing	Reading
Status byte		—	Serial poll (RQS), *STB?(MSS)
Service request enable register	Masks status byte.	*SRE	*SRE?
Standard event register	Change in device status	—	*ESR?
Standard event enable register	Masks standard event register	*ESE	*ESE?
Extended event register	Change in device status	—	STATus:EESR?
Extended event enable register	Masks standard event register	STATus:EESE	STATus:EESE?
Condition register	Current instrument status	—	STATus:CONDition?
Transit filter	Extended event occurrence conditions	STATus:FILTer<x>	STATus:FILTer<x>?
Output queue	Stores response message to a query.	All executable queues	
Error queue	Stores error Nos. and messages.	—	STATus:ERRor?

Registers and Queues which Affect the Status Byte

Registers which affect each bit of the status byte are shown below.

Standard event register : Sets bit 5 (ESB) of status byte to "1" or "0."

Output queue : Sets bit 4 (MAV) of status byte to "1" or "0."

Extended event register : Sets bit 3 (EES) of status byte to "1" or "0."

Error queue : Sets bit 2 (EAV) of status byte to "1" or "0."

Enable Registers

Registers which mask a bit so that the bit does not affect the status byte, even if the bit is set to "1," are shown below.

Status byte : Masks bits using the service request enable register.

Standard event register : Masks bits using the standard event enable register.

Extended event register : Masks bits using the extended event enable register.

Writing/Reading from Registers

The *ESE command is used to set bits in the standard event enable register to "1" or "0," and the *ESE query is used to check whether bits in that register are set to "1" or "0." For details of these commands, refer to Chapter 4.

5.2 Status Byte

Overview of Status Byte

7	RQS 6 MSS	ESB	MAV	EES	EAV	1	0

Bits 0, 1 and 7

Not used (always "0")

Bit 2 EAV (Error Available)

Set to "1" when the error queue is not empty, i.e. when an error occurs. For details, refer to page 5-5.

Bit 3 EES (Extended Event Summary Bit)

Sets to "1" when the logical "AND" of an Extended Event Register bit and the corresponding Enable Register bit is equal to "1."—that is, when an event takes place in the instrument. Refer to page 5-4.

Bit 4 MAV (Message Available)

Set to "1" when the output queue is not empty, i.e. when there is data which is to be output when an query is made. Refer to page 5-5.

Bit 5 ESB (Event Summary Bit)

Set to "1" when the logical AND of the standard event register and the corresponding enable register is "1," i.e. when an event takes place in the instrument. Refer to page 5-3.

Bit 6 RQS (Request Status)/MSS (Master Summary Status)

Sets to "1" when the logical "AND" of any one of the Status Byte bits (other than bit 6) and the corresponding Service Request Enable Register bit becomes "1"—that is, when the instrument is requesting service from the controller.
RQS is set to "1" when MSS changes from "0" to "1," and is cleared when a serial poll is performed or when MSS changes to "0."

Bit Masking

To mask a bit in the status byte so that it does not cause an SRQ, set the corresponding bit of the service request enable register to "0."
For example, to mask bit 2 (EAV) so that no service will be requested, even if an error occurs, set bit 2 of the service request enable register to "0." This can be done using the *SRE command. To query whether each bit of the service request enable register is "1" or "0," use *SRE?. For details of the *SRE command, refer to Chapter 4.

Operation of the Status Byte

A service request is issued when bit 6 of the status byte becomes "1." Bit 6 becomes "1" when any of the other bits becomes "1" (or when the corresponding bit in the service request enable register becomes "1").
For example, if an event takes place and the logical OR of each bit of the standard event register and the corresponding bit in the enable register is "1," bit 5 (ESB) will be set to "1." In this case, if bit 5 of the service request enable register is "1," bit 6 (MSS) will be set to "1," thus requesting service from the controller.
It is also possible to check what type of event has occurred by reading the contents of the status byte.

Reading from the Status Byte

The following two methods are provided for reading the status byte.

- **Inquiry using the *STB? query**
  Making an query using the *STB? query sets bit 6 to MSS. This causes the MSS to be read. After completion of the read-out, none of the bits in the status byte will be cleared.

- **Serial poll**
  Execution of a serial poll changes bit 6 to RQS. This causes RQS to be read. After completion of the read-out, only RQS is cleared. Using a serial poll, it is not possible to read MSS.

Clearing the Status Byte

No method is provided for forcibly clearing all the bits in the status byte. Bits which are cleared are shown below.

- **When an query is made using the *STB? query**
  No bit is cleared.

- **When a serial poll is performed**
  Only the RQS bit is cleared.

- **When the *CLS command is received**
  When the *CLS command is received, the status byte itself is not cleared, but the contents of the standard event register (which affects the bits in the status byte) are cleared. As a result, the corresponding bits in the status byte are cleared, except bit 4 (MAV), since the output queue cannot be emptied by the *CLS command.
  However, the output queue will also be cleared if the *CLS command is received just after a program message terminator.

5.3 Standard Event Register

Overview of the Standard Event Register

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

Bit 7 PON (Power ON)

Bit 7 PON (Power ON) Set to "1" when power is turned ON

Bit 6 URQ (User Request)

Not used (always "0")

Bit 5 CME (Command Error)

Set to "1" when the command syntax is incorrect.
Examples: Incorrectly spelled command name; "9" used in octal data.

Bit 4 EXE (Execution Error)

Set to "1" when the command syntax is correct but the command cannot be executed in the current state.
Examples: Parameters are outside the setting range: an attempt is made to make a hard copy during acquisition.

Bit 3 DDE (Device Dependent Error)

Set to "1" when execution of the command is not possible due to an internal problem in the instrument that is not a command error or an execution error.
Example: The circuit breaker is reset.

Bit 2 QYE (Query Error)

Set to "1" if the output queue is empty or if the data is missing even after a query has been sent.
Examples: No response data; data is lost due to an overflow in the output queue.

Bit 1 RQC (Request Control)

Not used (always "0")

Bit 0 OPC (Operation Complete)

Set to "1" when the operation designated by the *OPC command has been completed. Refer to Chapter 4.

Bit Masking

To mask a bit in the standard event register so that it does not cause bit 5 (ESB) of the status byte to change, set the corresponding bit in the standard event enable register to "0."
For example, to mask bit 2 (QYE) so that ESB will not be set to "1," even if a query error occurs, set bit 2 of the standard event enable register to "0." This can be done using the *ESE command. To inquire whether each bit of the standard event enable register is "1" or "0," use the *ESE?. For details of the *ESE command, refer to Chapter 4.

Operation of the Standard Event Register

The standard event register is provided for eight different kinds of event which can occur inside the instrument. Bit 5 (ESB) of the status byte is set to "1" when any of the bits in this register becomes "1" (or when the corresponding bit of the standard event enable register becomes "1").

Examples

1. A query error occurs.
2. Bit 2 (QYE) is set to "1."
3. Bit 5 (ESB) of the status byte is set to "1" if bit 2 of the standard event enable register is "1."

It is also possible to check what type of event has occurred inside the instrument by reading the contents of the standard event register.

Reading from the Standard Event Register

The contents of the standard event register can be read by the `*ESR` command. After completion of the read-out, the register will be cleared.

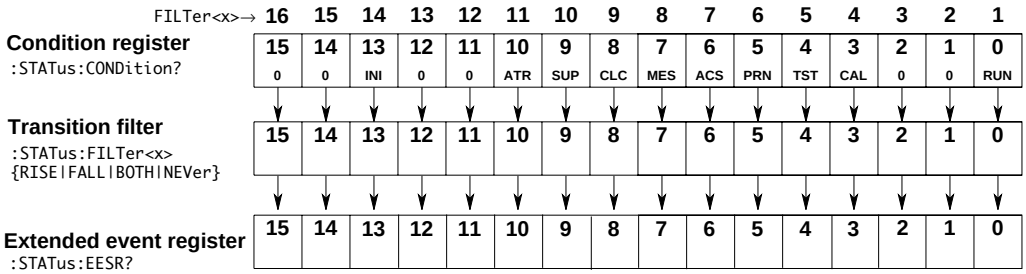
Clearing the Standard Event Register

The standard event register is cleared in the following three cases.

- When the contents of the standard event register are read using `*ESR?`
- When the `*CLS` command is received
- When power is turned ON again

5.4 Extended Event Register

Reading the extended event register tells you whether changes in the condition register (reflecting internal conditions) have occurred. A filter can be applied which allows you to decide which events are reported to the extended event register.



The meaning of each bit of the condition register is as follows.

Bit 0 RUN (Running)	Set to "1" during acquisition.
Bit 3 CAL (Calibrating)	Set to "1" during calibration.
Bit 4 TST (Testing)	Set to "1" during self-test.
Bit 5 PRN (Printing)	Set to "1" while the built-in printer is in operation.
Bit 6 ACS (Accessing)	Sets to "1" while floppy drive, HD drive, or external SCSI device is being accessed.
Bit 7 MES (Measuring)	Sets to "1" during automated measurment.
Bit 8 CLC (Calculating)	Sets to "1" while calculation is in progress.
Bit 9 SUP (Set-up)	Set to "1" during auto set-up.
Bit 10 ATR (Action On Trigger)	Sets to "1" during action on trigger.
Bit 13 INI (Initializing)	Sets to "1" during initialization.

The filter is applied to each bit of the condition register seperately, and can be selected from the following. Note that the numbering of the bits used in the filter setting differs from the actual bit number (1 to 16 vs. 0 to 15).

Rise	The bit of the extended event register becomes "1" when the bit of the condition register changes from "0" to "1."
Fall	The bit of the extended event register becomes "1" when the bit of the condition register changes from "1" to "0."
Both	The bit of the extended event register becomes "1" when the bit of the condition register changes from "0" to "1," or from "1" to "0."
Never	The bit of the extended event register is disabled and always "0."

5.5 Output Queue and Error Queue

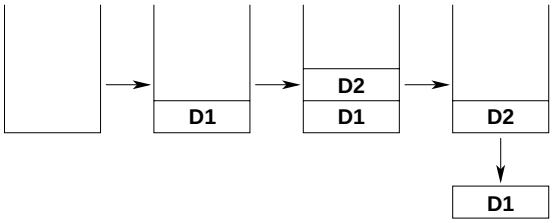
Overview of the Output Queue

The output queue is provided to store response messages to queries. For example, when the `WAVeform:SEND?` query is sent to request output of the acquired waveform, the response data will be stored in the output queue until it is read out.

The example below shows that data is stored record by record in the output queue, and is read out oldest item first, newest item last. The output queue is emptied in the following cases (in addition to when read-out is performed).

- When a new message is received from the controller
- When dead lock occurs (page 3-2)
- When a device clear command (DCL or SDC) is received
- When power is turned ON again

The output queue cannot be emptied using the `*CLS` command. To see whether the output queue is empty or not, check bit 4 (MAV) of the status byte.



Overview of the Error Queue

The error queue stores the error No. and message when an error occurs. For example, if the controller sends an incorrect program message, the number, “113, “Undefined header”,” and the error message are stored in the error queue, when the error is displayed.

The contents of the error queue can be read using the `STATus:ERRor?` query. As with the output queue, messages are read oldest first, newest last (refer to the previous page). If the error queue becomes full, the final message will be replaced by message “350, “Queue overflow”.”

The error queue is emptied in the following cases (in addition to when read-out is performed).

- When the `*CLS` command is received
- When power is turned ON again

To see whether the error queue is empty or not, check bit 2 (EAV) of the status byte.

6.1 Before Programming

Environment

Model: MS-DOS/V Computer
Language: Visual BASIC Ver5.0 Professional Edition or more

Setting up the Visual Basic

Compornent: MSComm
Standardmodule: Niglobal.bas
Vbib-32.bas

Setting up the DL716

- **Initialization**

  None of the sample programs given in this chapter include initialization of the DL716, so be sure to initialize the instrument using the Initialize menu before running the programs.

- **GPIB**

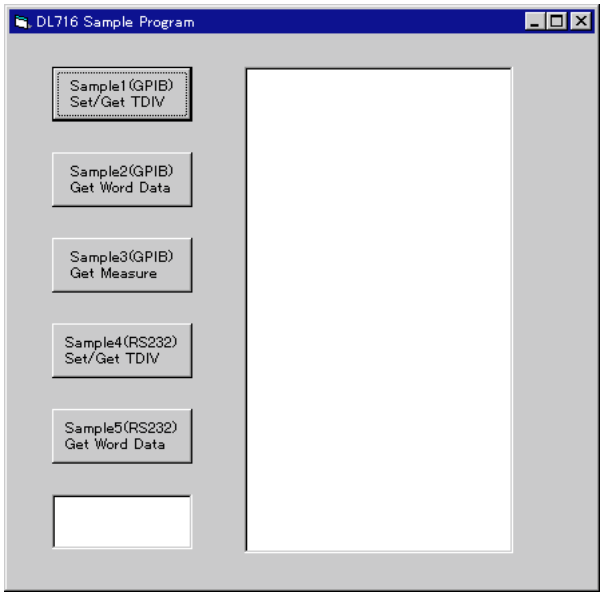
  All the sample programs given in this chapter use address 1 for the DL716, so be sure to assign the instrument to address 1 as described on page 1-5.

- **RS232**

  All the sample programs in this chapter assume the below settings, so be sure to set parameters as described on page 2-8.

Baud Rate	19200
Handshake	CTS-RTS
Parity Check	NO
Stop Bit	1
Character Length	8
Terminator	LF

6.2 Image of Sample Program



6.3 Initialize/Error/Execute

```
Option Explicit
Dim StartFlag As Integer                        'Start Flag
Dim TimerCount As Integer                       'Timeout(RS232)
Dim Addr As Integer                             'GPIB Address
Dim Timeout As Integer                          'Timeout
Dim Dev As Integer                              'Device ID(GPIB)
Dim CtsFlag As Integer                          'CTS Flag
Dim Term As String                              'Terminator
Dim Query(3) As String                          'Query String
Dim WaveBuffer(20070) As Integer                'WaveData Buffer(GPIB)
Dim Dummy As Integer
```

```
Private Function InitGpib() As Integer
    Dim eos As Integer                          'EOS
    Dim eot As Integer                          'EOI
    Dim brd As Integer                          'GPIB Board ID
    Dim sts As Integer

    eos = &HC0A                                 'Terminator = LF
    eot = 1                                     'EOI = Enable
    Term = Chr(10)
    Timeout = T10s                              'Timeout = 10s

    brd = ilfind("GPIB0")
    If (brd < 0) Then
        Call DisplayGPIBError(brd, "ilfind")
        InitGpib = 1
        Exit Function
    End If
    Dev = ildev(0, Addr, 0, Timeout, eot, eos)
    If (Dev < 0) Then
        Call DisplayGPIBError(Dev, "ildev")
        InitGpib = 1
        Exit Function
    End If
    sts = ilsic(brd)                            'Set IFC
    If (sts < 0) Then
        Call DisplayGPIBError(sts, "ilsic")
        InitGpib = 1
        Exit Function
    End If
    InitGpib = 0
End Function
```

```
Private Function InitSerial() As Integer
    Dim rat As String

    MSComm1.CommPort = 1                            'Port = COM1

    rat = "19200,N,8,1"                             'Rate = 19200, NoParity, 8Bit, 1Stopbit

    MSComm1.Settings = rat

    MSComm1.Handshaking = comRTS                    'Handshake = CTS-RTS
    MSComm1.RTSEnable = True                        'RTS = TRUE
    CtsFlag = 1
    Term = Chr(10)                                  'Terminator = LF
    Timeout = 10                                    'Timeout = 10s
    InitSerial = 0
End Function
```

```
Private Sub DisplayGPIBError(ByVal sts As Integer, ByVal msg As String)
    Dim wrn As String
    Dim ers As String
    Dim ern As Integer

    If (sts And TIMO) Then
        wrn = "Time out" + Chr(13)
    Else
        wrn = ""
    End If
    If (sts And EERR) Then
        ern = iberr
        If (ern = EDVR) Then
            ers = "EDVR:System error"
        ElseIf (ern = ECIC) Then
            ers = "ECIC:Function requires GPIB board to be CIC"
        ElseIf (ern = ENOL) Then
            ers = "ENOL:No Listeners on the GPIB"
        ElseIf (ern = EADR) Then
            ers = "EADR:GPIB board not addressed correctly"
        ElseIf (ern = EARG) Then
            ers = "EARG:Invalid argument to function call"
        ElseIf (ern = ESAC) Then
            ers = "ESAC:GPIB board not System Controller as required"
        ElseIf (ern = EABO) Then
            ers = "EABO:I/O operation aborted(timeout)"
        ElseIf (ern = ENEB) Then
            ers = "ENEB:Nonexistent GPIB board"
        ElseIf (ern = EDMA) Then
            ers = "EDMA:DMA error"
        ElseIf (ern = EOIP) Then
            ers = "EOIP:I/O operation started before previous operation completed"
        ElseIf (ern = ECAP) Then
            ers = "ECAP:No capability for intended operation"
        ElseIf (ern = EFSO) Then
            ers = "EFSO:File system operation error"
        ElseIf (ern = EBUS) Then
            ers = "EBUS:GPIB bus error"
        ElseIf (ern = ESTB) Then
            ers = "ESTB:Serial poll status byte queue overflow"
        ElseIf (ern = ESRQ) Then
            ers = "ESRQ:SRQ remains asserted"
        ElseIf (ern = ETAB) Then
            ers = "ETAB:The return buffer is full"
```

```
        ElseIf (ern = ELCK) Then
            ers = "ELCK:Address or board is locked"
        Else
            ers = ""
        End If
    Else
        ers = ""
    End If

    MsgBox ("Status No. " + Str(sts) + Chr(13) + wrn + "Error No. " + Str(ern) + Chr(13) +
ers + Chr(13) + msg), vbExclamation, "Error!"
    Call ibonl(Dev, 0)
    Dev = -1
End Sub
```

```
Private Sub DIsplayRS232Error(ByVal erm As String, Optional ByVal msg As String = "")
    MsgBox (erm + Chr(13) + msg), vbExclamation, "Error!"
End Sub
```

```
Private Sub Command1_Click()
    Dim sts As Integer

    If (StartFlag = 1) Then
        Exit Sub
    End If
    StartFlag = 1
    Text1.Text = "START"
    List1.Clear
    Dummy = DoEvents()
    sts = GpibTdiv                                  'Run Sample1(GPIB) Set/Get TDIV
    If (sts = 0) Then
        List1.AddItem Query(0)
    End If
    Text1.Text = "END"
    StartFlag = 0
End Sub
```

```
Private Sub Command2_Click()
    Dim sts As Integer

    If (StartFlag = 1) Then
        Exit Sub
    End If
    StartFlag = 1
    Text1.Text = "START"
    List1.Clear
    Dummy = DoEvents()
    sts = GpibGetWord                               'Run Sample2(GPIB) Get Word Data
    If (sts = 0) Then
        List1.AddItem "END"
    End If
    Text1.Text = "END"
    StartFlag = 0
End Sub
```

```
Private Sub Command3_Click()
    Dim sts As Integer

    If (StartFlag = 1) Then
        Exit Sub
    End If
    StartFlag = 1
    Text1.Text = "START"
    List1.Clear
    Dummy = DoEvents()
    sts = GpibGetMeasure                          'Run Sample3(GPIB) Get Measure
    If (sts = 0) Then
        List1.AddItem Query(0)
        List1.AddItem Query(1)
        List1.AddItem Query(2)
    End If
    Text1.Text = "END"
    StartFlag = 0
End Sub
```

```
Private Sub Command4_Click()
    Dim sts As Integer

    If (StartFlag = 1) Then
        Exit Sub
    End If
    StartFlag = 1
    Text1.Text = "START"
    List1.Clear
    sts = RS232Tdiv                               'Run Sample4(RS232) Set/Get TDIV
    If (sts = 0) Then
        List1.AddItem Query(0)
    End If
    Text1.Text = "END"
    StartFlag = 0
End Sub
```

```
Private Sub Command5_Click()
    Dim sts As Integer

    If (StartFlag = 1) Then
        Exit Sub
    End If
    StartFlag = 1
    Text1.Text = "START"
    List1.Clear
    sts = RS232GetWord                            'Run Sample5(RS232) Get Word Data
    If (sts = 0) Then
        List1.AddItem "END"
    End If
    Text1.Text = "END"
    StartFlag = 0
End Sub
```

```
Private Sub Form_Load()

    StartFlag = 0                                   'Clear Start Flag
    Dev = -1                                        'Clear device id
    Addr = 1                                        'GPIB Address = 1
    Timer1.Interval = 0
    Command1.Caption = "Sample1(GPIB)" + Chr(13) + "Set/Get TDIV"
    Command2.Caption = "Sample2(GPIB)" + Chr(13) + "Get Word Data"
    Command3.Caption = "Sample3(GPIB)" + Chr(13) + "Get Measure"
    Command4.Caption = "Sample4(RS232)" + Chr(13) + "Set/Get TDIV"
    Command5.Caption = "Sample5(RS232)" + Chr(13) + "Get Word Data"
    Text1.Text = ""

End Sub
```

```
Private Sub MSComm1_OnComm()
    Dim evt As Integer

    evt = MSComm1.CommEvent
    Select Case evt
    'Error
        Case comBreak
            Call DIsplayRS232Error("comBreak:Break received")
        Case comCDTO
            Call DIsplayRS232Error("comCDTO CD(RLSD) timeout")
        Case comCTSTO
            Call DIsplayRS232Error("comCTSTO:CTS timeout")
        Case comDSRTO
            Call DIsplayRS232Error("commDSRTO:DSR timeout")
        Case comFrame
            Call DIsplayRS232Error("comFrame:Frame error")
        Case comOverrun
            Call DIsplayRS232Error("comOverrun:Overrun")
        Case comRxOver
            Call DIsplayRS232Error("comRxOver:Receive buffer overflow")
        Case comRxParity
            Call DIsplayRS232Error("commRxParity:Parity error")
        Case comTxFull
            Call DIsplayRS232Error("comTxFull:Send buffer overflow")
    'Event
        Case comEvReceive
        Case comEvCD
        Case comEvCTS
        Case comEvDSR
        Case comEvRing
        Case comEvSend
    End Select

End Sub
```

```
Private Sub Timer1_Timer()
    TimerCount = TimerCount + 1
End Sub
```

6.4 Sets/Queries the T/Div

Sample1(GPIB) Set/Get TDIV

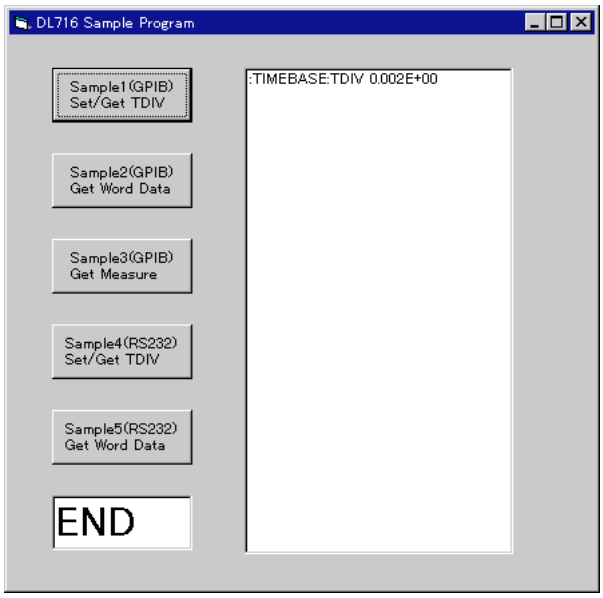
```
Private Function GpibTdiv() As Integer
    Dim msg As String                               'Command buffer
    Dim qry As String                               'Query biffer
    Dim sts As Integer

    msg = Space$(100)
    qry = Space$(100)

    sts = InitGpib                                  'Initialize GPIB
    If (sts <> 0) Then
        GpibTdiv = 1
        Exit Function
    End If

    msg = "TIMEBASE:TDIV 2ms" + Term                'Set T/div = 2ms
    sts = ilwrt(Dev, msg, Len(msg))                 'Send Command
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibTdiv = 1
        Exit Function
    End If
    msg = "TIMEBASE:TDIV?" + Term                   'Get T/div value
    sts = ilwrt(Dev, msg, Len(msg))                 'Send Command
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibTdiv = 1
        Exit Function
    End If
    sts = ilrd(Dev, qry, Len(qry))                  'Receive Query
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibTdiv = 1
        Exit Function
    End If
    Query(0) = Left(qry, ibcntl - 1)
    Call ibonl(Dev, 0)
    GpibTdiv = 0
End Function
```

Sample4(RS232) Set/Get TDIV

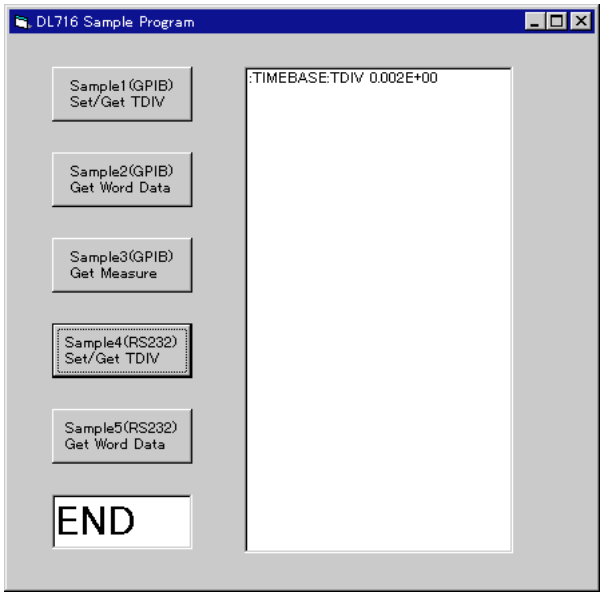
```
Private Function RS232Tdiv() As Integer
    Dim msg As String                               'Command buffer
    Dim qry As String                               'Query biffer
    Dim sts As Integer

    msg = Space$(100)
    qry = CStr(Empty)

    sts = InitSerial                                'Initialize RS232
    If (sts <> 0) Then
        Exit Function
    End If

    MSComm1.InputLen = 0                            'Receive All Data
    MSComm1.InputMode = comInputModeText            'Text Mode
    MSComm1.PortOpen = True                         'Port Open
    MSComm1.OutBufferCount = 0                      'Out Buffer Clear
    MSComm1.InBufferCount = 0                       'In Buffer Clear
    Timer1.Interval = 1000

    If CtsFlag = 1 Then                             'If CTS = FALSE
        TimerCount = 1                              'Wait until CTS = TRUE
        Do
            Dummy = DoEvents()
            If (TimerCount >= Timeout) Then
                Call DIsplayRS232Error("CTS Timeout")
                RS232Tdiv = 1
                GoTo finish
            End If
        Loop Until MSComm1.CTSHolding = True
    End If

    msg = "TIMEBASE:TDIV 2ms" + Term                'Set T/div = 2ms
    MSComm1.Output = msg                            'Send Command

    TimerCount = 1
    Do                                              'Wait until OutBufferCount = 0
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232Tdiv = 1
            GoTo finish
        End If
    Loop Until MSComm1.OutBufferCount = 0

    msg = "TIMEBASE:TDIV?" + Term                   'Get T/div value
    MSComm1.Output = msg                            'Send Command

    TimerCount = 1
    Do                                              'Wait until OutBufferCount = 0
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232Tdiv = 1
            GoTo finish
        End If
    Loop Until MSComm1.OutBufferCount = 0
```

```
    TimerCount = 1
    Do                                              'Receive Query
        qry = qry + MSComm1.Input                   'Wait until End Data = Terminator
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Receive Timeout", msg)
            RS232Tdiv = 1
            GoTo finish
        End If
    Loop Until Right$(qry, 1) = Term

    Query(0) = Left$(qry, Len(qry) - 1)
    RS232Tdiv = 0
finish:
    MSComm1.PortOpen = False                        'Port Close
    Timer1.Interval = 0

End Function
```

GP-IB



RS-232



6.5 Data Output in Word Format

Sample2(GPIB) Get Word Data

```
Private Function GpibGetWord() As Integer
    Dim msg As String                               'Command buffer
    Dim qry As String                               'Query biffer
    Dim sts As Integer
    Dim vdv As Variant                              'Vdiv value
    Dim ofs As Variant                              'Offset value
    Dim eos As Integer                              'EOS
    Dim hlg As Integer                              'Block Header Length
    Dim dlg As Integer                              'Block Data Length
    Dim dat As Variant                              'Data
    Dim i   As Integer

    msg = Space$(100)
    qry = Space$(100)

    sts = InitGpib                                  'Initialize GPIB
    If (sts <> 0) Then
        GpibGetWord = 1
        Exit Function
    End If

    msg = "STOP" + Term                             'Stop Acquisition
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    msg = "COMMUNICATE:HEADER OFF" + Term           'Query Header Off(for Get V/div)
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    msg = "WAVEFORM:TRACE 1" + Term                 'Trace = 1
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    msg = "WAVEFORM:RECORD 0" + Term                'Record number = 0
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    msg = "WAVEFORM:FORMAT WORD" + Term              'Data Format = WORD
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
```

```
    End If
    msg = "WAVEFORM:BYTEORDER LSBFIRST" + Term     'Data Byte order = LSB First(for Little
Endian)
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    msg = "WAVEFORM:START 0;END 1001" + Term       'START 0,END 1001(Length = 1002)
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    msg = "WAVEFORM:RANGE?" + Term                 'Get V/div value
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    sts = ilrd(Dev, qry, Len(qry))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    vdv = Val(qry)
    msg = "WAVEFORM:OFFSET?" + Term                 'Get Offset value
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    sts = ilrd(Dev, qry, Len(qry))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    ofs = Val(qry)

    eos = 0
    sts = ileos(Dev, eos)                          'Terminator = None(for Binary Data)
    If (sts < 0) Then
        Call DisplayGPIBError(sts, "ileos")
        GpibGetWord = 1
        Exit Function
    End If
    msg = "WAVEFORM:SEND?" + Term                  'Receive Waveform Data
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
```

```
    End If
    sts = ilrd(Dev, qry, 2)                         'Receive "#X"
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    hlg = Val(Mid$(qry, 2, 1))
    sts = ilrd(Dev, qry, hlg)                       'Receive Block Header
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    dlg = Val(Left$(qry, hlg))                      'dlg = Data Byte Length

    sts = ilrdi(Dev, WaveBuffer(), dlg + 1)         'Receive Waveform Data + LF
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If

    For i = 0 To (dlg / 2 - 1) Step 1
        dat = WaveBuffer(i) * vdv * 8 / 64000 + ofs
        List1.AddItem CStr(i) + ":" + CStr(dat)
    Next i

    eos = &HC0A
    sts = ileos(Dev, eos)                           'Terminator = LF
    If (sts < 0) Then
        Call DisplayGPIBError(sts, "ileos")
        GpibGetWord = 1
        Exit Function
    End If
    msg = "COMMUNICATE:HEADER ON" + Term            'Query Header On
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetWord = 1
        Exit Function
    End If
    Call ibonl(Dev, 0)
    GpibGetWord = 0
End Function
```

Sample5(RS232) Get Word Data

```
Private Function RS232GetWord() As Integer
    Dim msg As String                               'Command buffer
    Dim qry As String                               'Query biffer
    Dim sts As Integer
    Dim vdv As Variant                              'V/div value
    Dim ofs As Variant                              'Offset value
    Dim hlg As Integer                              'Block Header Length
    Dim dlg As Integer                              'Blocl Data Length
    Dim buf As Variant                              'temporary buffer
    Dim dat As Variant                              'data buffer
    Dim i   As Integer

    msg = Space$(100)
    qry = CStr(Empty)

    sts = InitSerial                                'Initialize RS232
    If (sts <> 0) Then
        Exit Function
    End If

    MSComm1.InputLen = 0                            'Receive All Data
    MSComm1.InputMode = comInputModeText            'Text Mode
    MSComm1.PortOpen = True                         'Port Open
    MSComm1.OutBufferCount = 0                      'Out Buffer Clear
    MSComm1.InBufferCount = 0                       'In Buffer Clear
    Timer1.Interval = 1000

    If CtsFlag = 1 Then                             'If CTS = FALSE
        TimerCount = 1                              'Wait unitl CTS = TRUE
        Do
            Dummy = DoEvents()
            If (TimerCount >= Timeout) Then
                Call DIsplayRS232Error("CTS Timeout")
                RS232GetWord = 1
                GoTo finish
            End If
        Loop Until MSComm1.CTSHolding = True
    End If

    msg = "STOP" + Term                             'Stop Acquisition
    MSComm1.Output = msg
    TimerCount = 1
    Do
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232GetWord = 1
            GoTo finish
        End If
    Loop Until MSComm1.OutBufferCount = 0

    msg = "COMMUNICATE:HEADER OFF" + Term           'Query Header Off(for Get V/div)
    MSComm1.Output = msg
    TimerCount = 1
    Do
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232GetWord = 1
```

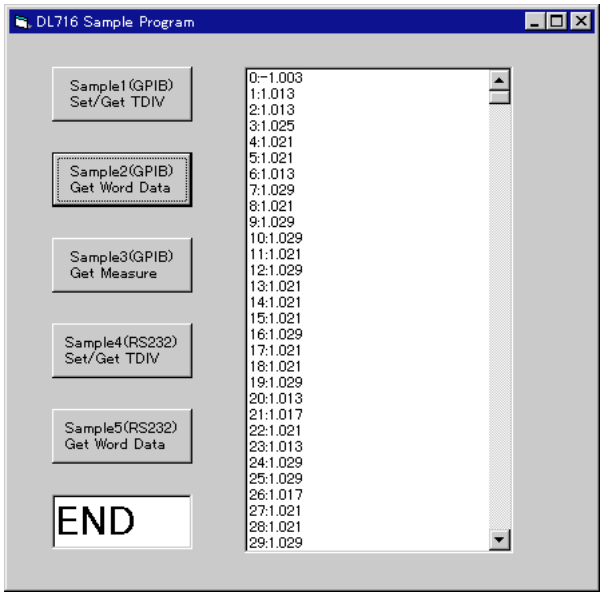
```
            GoTo finish
        End If
    Loop Until MSComm1.OutBufferCount = 0

    msg = "WAVEFORM:TRACE 1" + Term                'Trace = 1
    MSComm1.Output = msg
    TimerCount = 1
    Do
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232GetWord = 1
            GoTo finish
        End If
    Loop Until MSComm1.OutBufferCount = 0

    msg = "WAVEFORM:RECORD 0" + Term               'Record number = 0
    MSComm1.Output = msg
    TimerCount = 1
    Do
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232GetWord = 1
            GoTo finish
        End If
    Loop Until MSComm1.OutBufferCount = 0

    msg = "WAVEFORM:FORMAT WORD" + Term             'Data Format = WORD
    MSComm1.Output = msg
    TimerCount = 1
    Do
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232GetWord = 1
            GoTo finish
        End If
    Loop Until MSComm1.OutBufferCount = 0

    msg = "WAVEFORM:BYTEORDER LSBFIRST" + Term      'Data Byte order = LSB First(for Little
Endian)
    MSComm1.Output = msg
    TimerCount = 1
    Do
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232GetWord = 1
            GoTo finish
        End If
    Loop Until MSComm1.OutBufferCount = 0

    msg = "WAVEFORM:START 0;END 1001" + Term        'START 0,END 1001(Length = 1002)
    MSComm1.Output = msg
    TimerCount = 1
    Do
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Send Timeout", msg)
            RS232GetWord = 1
```

```
        GoTo finish
    End If
Loop Until MSComm1.OutBufferCount = 0

qry = CStr(Empty)
msg = "WAVEFORM:RANGE?" + Term                  'Get V/div value
MSComm1.Output = msg
TimerCount = 1
Do
    Dummy = DoEvents()
    If (TimerCount >= Timeout) Then
        Call DIsplayRS232Error("Send Timeout", msg)
        RS232GetWord = 1
        GoTo finish
    End If
Loop Until MSComm1.OutBufferCount = 0
TimerCount = 1
Do
    qry = qry + MSComm1.Input
    Dummy = DoEvents()
    If (TimerCount >= Timeout) Then
        Call DIsplayRS232Error("Receive Timeout", msg)
        RS232GetWord = 1
        GoTo finish
    End If
Loop Until Right$(qry, 1) = Term
vdv = Val(qry)

qry = CStr(Empty)
msg = "WAVEFORM:OFFSET?" + Term                  'Get Offset value
MSComm1.Output = msg
TimerCount = 1
Do
    Dummy = DoEvents()
    If (TimerCount >= Timeout) Then
        Call DIsplayRS232Error("Send Timeout", msg)
        RS232GetWord = 1
        GoTo finish
    End If
Loop Until MSComm1.OutBufferCount = 0
TimerCount = 1
Do
    qry = qry + MSComm1.Input
    Dummy = DoEvents()
    If (TimerCount >= Timeout) Then
        Call DIsplayRS232Error("Receive Timeout", msg)
        RS232GetWord = 1
        GoTo finish
    End If
Loop Until Right$(qry, 1) = Term
ofs = Val(qry)

msg = "WAVEFORM:SEND?" + Term                   'Receive Waveform Data
MSComm1.Output = msg

TimerCount = 1
Do
    Dummy = DoEvents()
    If (TimerCount >= Timeout) Then
        Call DIsplayRS232Error("Send Timeout", msg)
        RS232GetWord = 1
```

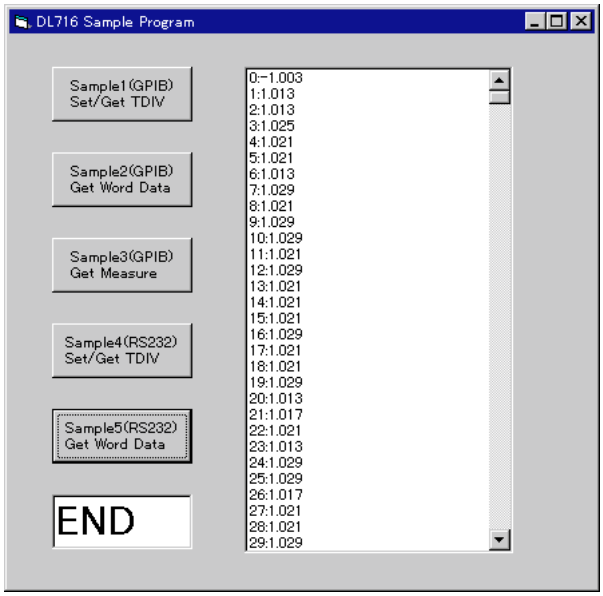
```
        GoTo finish
    End If
Loop Until MSComm1.OutBufferCount = 0

MSComm1.InputLen = 2                            'Receive "#X"
TimerCount = 1
Do Until MSComm1.InBufferCount >= 1
    Dummy = DoEvents()
    If (TimerCount >= Timeout) Then
        Call DIsplayRS232Error("Receive Timeout", msg)
        RS232GetWord = 1
        GoTo finish
    End If
Loop
qry = MSComm1.Input
hlg = Val(Mid$(qry, 2, 1))

MSComm1.InputLen = hlg                          'Receive Block Header
TimerCount = 1
Do Until MSComm1.InBufferCount >= hlg
    Dummy = DoEvents()
    If (TimerCount >= Timeout) Then
        Call DIsplayRS232Error("Receive Timeout", msg)
        RS232GetWord = 1
        GoTo finish
    End If
Loop
qry = MSComm1.Input
dlg = Val(Left$(qry, hlg))                      'leng% = Data Byte Length

MSComm1.InputMode = comInputModeBinary
MSComm1.InputLen = 2                            'Receive Waveform Data(2 Byte)

For i = 0 To (dlg / 2 - 1) Step 1               'Loop(dlg)
    TimerCount = 1
    Do Until MSComm1.InBufferCount >= 2
        Dummy = DoEvents()
        If (TimerCount >= Timeout) Then
            Call DIsplayRS232Error("Receive Timeout", msg)
            RS232GetWord = 1
            GoTo finish
        End If
    Loop
    buf = MSComm1.Input                         'Receive 1 Data(2 Byte)
    dat = buf(1) * 256 + buf(0)
    If (dat > 32767) Then
        dat = dat - 65536
    End If
    dat = dat * vdv * 8 / 64000 + ofs
    List1.AddItem CStr(i) + ":" + CStr(dat)
Next i

msg = "COMMUNICATE:HEADER ON" + Term            'Query Header On
MSComm1.Output = msg
TimerCount = 1
Do
    Dummy = DoEvents()
    If (TimerCount >= Timeout) Then
        Call DIsplayRS232Error("Send Timeout", msg)
        RS232GetWord = 1
        GoTo finish
```

```
        End If
    Loop Until MSComm1.OutBufferCount = 0

    RS232GetWord = 0
finish:
    MSComm1.PortOpen = False                    'Port Close
    Timer1.Interval = 0

End Function
```

GP-IB



RS-232



6.6 Sets/Queries Measure Value

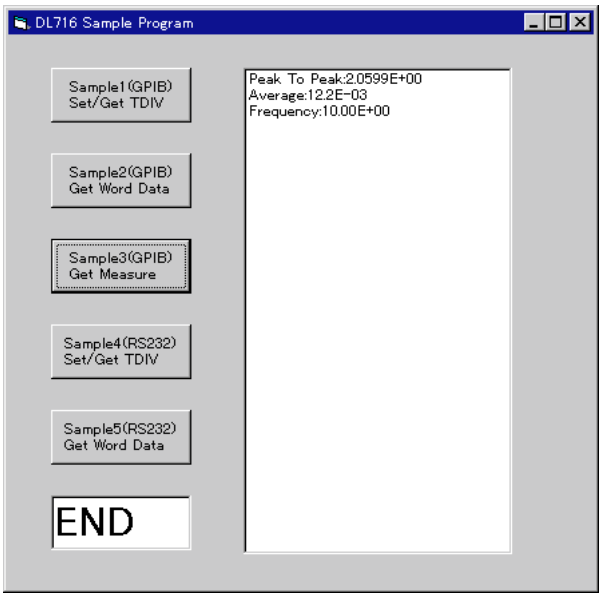
Sample3(GPIB) Get Measure value

```
Private Function GpibGetMeasure() As Integer
    Dim msg As String                               'Command buffer
    Dim qry As String                               'Query biffer
    Dim sts As Integer

    msg = Space$(100)
    qry = Space$(100)

    sts = InitGpib                                  'Initialize GPIB
    If (sts <> 0) Then
        GpibGetMeasure = 1
        Exit Function
    End If

    msg = "STOP" + Term                             'Acquisition = Stop
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetMeasure = 1
        Exit Function
    End If
    msg = "COMMUNICATE:HEADER OFF" + Term           'Query Header Off(for Get V/div)
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetMeasure = 1
        Exit Function
    End If
    msg = "CHANNEL:DISPLAY ON" + Term               'CH1 On
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetMeasure = 1
        Exit Function
    End If
    msg = "CHANNEL:VOLTAGE:PROBE 10" + Term         'CH1 Probe = 10:1
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetMeasure = 1
        Exit Function
    End If
    msg = "CHANNEL:VOLTAGE:VDIV 500mV" + Term       'CH1 V/div = 500mV
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetMeasure = 1
        Exit Function
    End If
    msg = "ACQUIRE:MODE NORMAL;RTOUT OFF;RLENGTH 1000" + Term
                                                    'Acquisition mode = NORMAL, length = 1000
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
```

```
    GpibGetMeasure = 1
    Exit Function
End If
msg = "TIMEBASE:TDIV 100ms" + Term              'T/div = 100ms
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
msg = "TRIGGER:SIMPLE:LEVEL 500mV" + Term       'Trigger level = 500mV
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
msg = "MEASURE:CHANNEL1:PTOPEAK:STATE ON" + Term
                                                'Measure P-P On
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
msg = "MEASURE:CHANNEL1:AVERAGE:STATE ON" + Term
                                                'Measure Average On
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
msg = "MEASURE:CHANNEL1:FREQUENCY:STATE ON" + Term
                                                'Measure Frequency On
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
msg = "MEASURE:HREFERENCE -5,5" + Term          'Measure Time Range -5,5
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
msg = "MEASURE:MODE ON" + Term                  'Measure On
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If

msg = "SSTART" + Term                           'Acquisition = Start(Single)
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
```

```
    Exit Function
End If

msg = "MEASURE:WAIT? 50" + Term                'Wait until stop Measure
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
sts = ilrd(Dev, qry, Len(qry))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If

msg = "MEASURE:CHANNEL1:PTOPEAK:VALUE?" + Term 'Get P-P value
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
sts = ilrd(Dev, qry, Len(qry))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
Query(0) = "Peak To Peak:" + Left$(qry, ibcntl - 1)

msg = "MEASURE:CHANNEL1:AVERAGE:VALUE?" + Term 'Get Average value
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
sts = ilrd(Dev, qry, Len(qry))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
Query(1) = "Average:" + Left$(qry, ibcntl - 1)

msg = "MEASURE:CHANNEL1:FREQUENCY:VALUE?" + Term 'Get Freq value
sts = ilwrt(Dev, msg, Len(msg))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
sts = ilrd(Dev, qry, Len(qry))
If (sts < 0) Then
    Call DisplayGPIBError(sts, msg)
    GpibGetMeasure = 1
    Exit Function
End If
```

```
    Query(2) = "Frequency:" + Left$(qry, ibcntl - 1)

    msg = "COMMUNICATE:HEADER ON" + Term           'Query Header On
    sts = ilwrt(Dev, msg, Len(msg))
    If (sts < 0) Then
        Call DisplayGPIBError(sts, msg)
        GpibGetMeasure = 1
        Exit Function
    End If
    Call ibonl(Dev, 0)
    GpibGetMeasure = 0
End Function
```

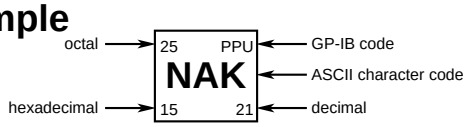


Appendix 1 ASCII Character Code

ASCII character codes are given below.

	0	1	2	3	4	5	6	7
0	0 **NUL** 0 0	20 **DEL** 10 16	40 0 **SP** 20 32	60 16 **0** 30 48	100 0 **@** 40 64	120 16 **P** 50 80	140 0 **‘** 60 96	160 16 **p** 70 112
1	1 GTL **SOH** 1 1	21 LLO **DC1** 11 17	41 1 **!** 21 33	61 17 **1** 31 49	101 1 **A** 41 65	121 17 **Q** 51 81	141 1 **a** 61 97	161 17 **q** 71 113
2	2 **STX** 2 2	22 **DC2** 12 18	42 2 **”** 22 34	62 18 **2** 32 50	102 2 **B** 42 66	122 18 **R** 52 82	142 2 **b** 62 98	162 18 **r** 72 114
3	3 **ETX** 3 3	23 **DC3** 13 19	43 3 **#** 23 35	63 19 **3** 33 51	103 3 **C** 43 67	123 19 **S** 53 83	143 3 **c** 63 99	163 19 **s** 73 115
4	4 SDC **EOT** 4 4	24 DCL **DC4** 14 20	44 4 **$** 24 36	64 20 **4** 34 52	104 4 **D** 44 68	124 20 **T** 54 84	144 4 **d** 64 100	164 20 **t** 74 116
5	5 PPC **ENQ** 5 5	25 PPU **NAK** 15 21	45 5 **%** 25 37	65 21 **5** 35 53	105 5 **E** 45 69	125 21 **U** 55 85	145 5 **e** 65 101	165 21 **u** 75 117
6	6 **ACK** 6 6	26 **SYN** 16 22	46 6 **&** 26 38	66 22 **6** 36 54	106 6 **F** 46 70	126 22 **V** 56 86	146 6 **f** 66 102	166 22 **v** 76 118
7	7 **BEL** 7 7	27 **ETB** 17 23	47 7 **’** 27 39	67 23 **7** 37 55	107 7 **G** 47 71	127 23 **W** 57 87	147 7 **g** 67 103	167 23 **w** 77 119
8	10 GET **BS** 8 8	30 SPE **CAN** 18 24	50 8 **(** 28 40	70 24 **8** 38 56	110 8 **H** 48 72	130 24 **X** 58 88	150 8 **h** 68 104	170 24 **x** 78 120
9	11 TCT **HT** 9 9	31 SPD **EM** 19 25	51 9 **)** 29 41	71 25 **9** 39 57	111 9 **I** 49 73	131 25 **Y** 59 89	151 9 **i** 69 105	171 25 **y** 79 121
A	12 **LF** A 10	32 **SUB** 1A 26	52 10 ***** 2A 42	72 26 **:** 3A 58	112 10 **J** 4A 74	132 26 **Z** 5A 90	152 10 **j** 6A 106	172 26 **z** 7A 122
B	13 **VT** B 11	33 **ESC** 1B 27	53 11 **+** 2B 43	73 27 **;** 3B 59	113 11 **K** 4B 75	133 27 **[** 5B 91	153 11 **k** 6B 107	173 27 **{** 7B 123
C	14 **FF** C 12	34 **FS** 1C 28	54 12 **,** 2C 44	74 28 **<** 3C 60	114 12 **L** 4C 76	134 28 **** 5C 92	154 12 **l** 6C 108	174 28 **|** 7C 124
D	15 **CR** D 13	35 **GS** 1D 29	55 13 **-** 2D 45	75 29 **=** 3D 61	115 13 **M** 4D 77	135 29 **]** 5D 93	155 13 **m** 6D 109	175 29 **}** 7D 125
E	16 **SO** E 14	36 **RS** 1E 30	56 14 **.** 2E 46	76 30 **>** 3E 62	116 14 **N** 4E 78	136 30 **^** 5E 94	156 14 **n** 6E 110	176 30 **~** 7E 126
F	17 **SI** F 15	37 **US** 1F 31	57 15 **/** 2F 47	77 UNL **?** 3F 63	117 15 **O** 4F 79	137 UNT **_** 5F 95	157 15 **o** 6F 111	177 **DEL (RUBOUT)** 7F 127
	Address Command	**Universal Command**	**Listener Address**		**Talker Address**		**Secondary Command**	

Example



Appendix 2 Error Messages

Error messages related to communications are given below.

- The instrument allows error messages to be displayed in either Japanese or English, however, they are shown only in English when they are displayed on a personal computer.
- When servicing is required, contact your nearest YOKOGAWA representative, given on the back cover of this manual.
- Only error messages relating to communications are given. For other error messages, refer to the User's Manual IM 701830-01E.

Errors in communication command (100 to 199)

Code	Message	Action	Reference Page
102	Syntax error	Incorrect syntax.	Chapter 3, 4
103	Invalid separator	Insert a comma between data items to separate them.	3-1
104	Data type error	Refer to pages 3-5 to 3-6 and enter using the correct data format.	3-5 to 3-6
105	GET not allowed	GET is not supported as response to an interface message.	1-9
108	Parameter not allowed	Check the number of parameters.	3-5, Chapter 4
109	Missing parameter	Enter required parameters.	3-5, Chapter 4
111	Header separator error	Insert a space between header and data to separate them.	3-1
112	Program mnemonic too long	Check the mnemonic (a character string consisting of letters and numbers).	Chapter 4
113	Undefined header	Check the header.	Chapter 4
114	Header suffix out of range	Check the header.	Chapter 4
120	Numeric data error	Numeric value must be entered for <NRf> format.	3-5
123	Exponent too large	Use a smaller exponent for <NR3> format.	3-5, Chapter 4
124	Too many digits	Limit the number of digits to 255 or less.	3-5, Chapter 4
128	Numeric data not allowed	Enter in a format other than <NRf> format.	3-5, Chapter 4
131	Invalid suffix	Check the unit for <Voltage>, <Time> and <Frequency>.	3-5
134	Suffix too long	Check the units for <Voltage>, <Time> and <Frequency>.	3-5
138	Suffix not allowed	No units are allowed other than <Voltage>, <Time> and <Frequency>.	3-5
141	Invalid character data	Enter one of the character strings in {...|...|...}.	Chapter 4
144	Character data too long	Check the character strings in {...|...|...}.	Chapter 4
148	Character data not allowed	Enter in a format other than in {...|...|...}.	Chapter 4
150	String data error	<Character string> must be enclosed by double quotation marks or single quotation marks.	3-6
151	Invalid string data	<Character string> is too long or contains characters which cannot be used.	Chapter 4
158	String data not allowed	Enter in a data format other than <Character string>.	Chapter 4
161	Invalid block data	<Block data> is not allowed.	3-6, Chapter 4
168	Block data not allowed	<Block data> is not allowed.	3-6, Chapter 4
171	Invalid expression	Equation is not allowed.	Chapter 4
178	Expression data not allowed	Equation is not allowed.	Chapter 4
181	Invalid outside macro definition	Does not conform to the macro function specified in IEEE488.2.	—

Error in communications execution (200 to 299)

Code	Message	Action	Reference Page
221	Setting conflict	Check the relevant setting.	Chapter 4
222	Data out of range	Check the setting range.	Chapter 4
223	Too much data	Check the data byte length.	Chapter 4
224	Illegal parameter value	Check the setting range.	Chapter 4
241	Hardware missing	Check availability of options.	—
260	Expression error	Equation is not allowed.	—
270	Macro error	Does not conform to the macro function specified in IEEE488.2.	—
272	Macro execution error	Does not conform to the macro function specified in IEEE488.2.	—
273	Illegal macro label	Does not conform to the macro function specified in IEEE488.2.	—
275	Macro definition too long	Does not conform to the macro function specified in IEEE488.2.	—
276	Macro recursion error	Does not conform to the macro function specified in IEEE488.2.	—
277	Macro redefinition not allowed	Does not conform to the macro function specified in IEEE488.2.	—
278	Macro header not found	Does not conform to the macro function specified in IEEE488.2.	—

Error in communications Query (400 to 499)

Code	Message	Action	Reference Page
410	Query INTERRUPTED	Check transmission/reception order.	3-2
420	Query UNTERMINATED	Check transmission/reception order.	3-2
430	Query DEADLOCKED	Limit the length of the program message including <PMT> to 1024 bytes or less.	3-2
440	Query UNTERMINATED after indefinite response	Do not enter any query after `*IDN?` and `*OPT?`.	—

Error in System Operation (912 to 914)

Code	Message	Action	Reference Page
912	Fatal error in Communication-driver	Servicing is required.	—
914	Time out occurs in Communication	Receive data within time-out time. The communications line may be faulty.	—
915	Can't detect listener	Check GP-IB connector Place the plotter or AG in listen-only mode then connect it.	—

Warning

Code	Message	Action	Reference Page
5	`*OPC/?` exists in message	Place the `*OPC` or `*OPC?` at the end of the program message.	—

Other errors (350 and 390)

Code	Message	Action	Reference Page
350	Queue overflow	Read the error queue. Code 350 occurs when the error queue is full up. This message is output only for the `STATus:ERRor?` query and is not displayed on the screen.	5-5
390	Overrun error (only RS-232)	Execute with a lower baud rate.	—

Note

Code 350 indicates overflow of error queue. This code is returned as a response to the "`STATus:ERRor?`" query; it does not appear on the screen.

Appendix 3 Overview of IEEE 488.2-1992

The GP-IB interface provided with DL716 conforms to IEEE 488.2-1992. This standard requires the following 23 points be stated in this document. This Appendix describes these points.

1 Subsets supported by IEEE 488.1 interface functions
Refer to Section 1.4 “GP-IB Interface Specifications”.

2 Operation of device when the device is assigned to an address other than addresses 0 to 30.
The DL716 does not allow assignment to an address other than 0 to 30.

3 Reaction when the user changes the address
The current address is changed when a new address is set using the MISC key. The newly set address is valid until another new address is set.

4 Device set-up at power ON. Commands which can be used at power ON
Basically, the previous settings (i.e. the settings which were valid when power was turned OFF) are valid. All commands are available at power ON.

5 Message transmission options
   a Input buffer size
   1024 bytes

   b Queries which return multiple response messages
   Refer to Chapter 4, “Command List”.

   c Queries which generate response data during analysis of the syntax
   Every query generates a response data when analysis of the syntax is completed.

   d Queries which generate response data during reception
   No query generates response data when the query is received by the controller.

   e Commands consisting of parameters which restrict one other
   Some commands, like the `CHANnel<x>:VOLTage:PROBe` and `VDIV`, have parameters which restrict unilaterally, but no commands have parameters which restrict bilaterally.

6 Options included in command function elements and composite header elements
Refer to Chapters 3 and 4.

7 Buffer size which affects transmission of block data
During transmission of block data, the output queue is extended according to the size of the data blocks.

8 List of program data elements which can be used in equations, and nesting limit
No equations can be used.

9 Syntax of response to queries
Refer to the description of the commands given in Chapter 4.

10 Communications between devices which do not follow the response syntax
Refer to Section 1.6 “Setting the Talk Only Mode (Data Output to the AG Series)” of this manual and Section 10.5 “Using the GP-IB/RS-232 Interface” of the User’s Manual IM 701830-01E.

11 Size of data block of response data
1 to 256512000(10002×128×1000×2) bytes

12 List of supported common commands
Refer to Section 4.30 “Common Command Group”.

13 Condition of device when calibration is successfully completed
Same as the one under which measurements are performed

14 Maximum length of block data which can be used for definition of *DDT trigger macro
Not supported

15 Maximum length of macro label used in definition of macro, maximum length of block data which can be used for definition of macro, processing when recursion is used in definition of macro
Macro functions are not supported.

16 Response to *IDN?
Refer to Section 4.30 “Common Command Group”.

17 Size of storage area for protected user data for PUD and *PUD?
*PUD and *PUD? are not supported.

18 Length of *RDT and *RDT? resource name
*RDT and *RDT? are not supported.

19 Change in status due to *RST, *LRN?, *RCL and *SAV
*RST, *LRN?
Refer to Section 4.30 “Common Command Group”.
*RCL, *SAV
These commands are not supported.

20 Execution range of self-test using the *TST?
All the memory tests (for each internal memory) given in the Self Test menu displayed using the MISC key can be executed.

21 Structure of extended return status
Refer to Chapter 5.

22 To find out whether each command is performed in parallel or sequentially
Refer to Section 3.5 “Synchronization with the Controller” and to Chapter 4.

23 Description of execution of each command
Refer to Chapter 4 of this manual and to the User’s Manual IM 701830-01E.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z