PZ4000
Power Analyzer Communication Interface
USER'S MANUAL

YOKOGAWA
Yokogawa Meters & Instruments Corporation

IM 253710-11E
2nd Edition

Introduction

Thank you for purchasing YOKOGAWA's PZ4000 Power Analyzer.
This Communication Interface User's Manual describes the functions and commands of the GP-IB and serial interfaces. To ensure proper use of the GP-IB/serial interfaces, please read this manual thoroughly.
Keep the manual in a safe place for quick reference whenever a question arises.
Two manuals are provided with the PZ4000 including this Communication Interface User's Manual.

Manual Name	Manual No.	Description
PZ4000 Power Analyzer User's Manual	IM 253710-01E	Describes all functions except for the communications functions and operation procedures of the instrument.
PZ4000 Power Analyzer Communication User's Manual	IM 253710-11E	Describes the communications functions of the GP-IB/serial interface.

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

- 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: April 1999
2nd Edition: April 2000

Disk No. BA32
2nd Edition: April 2000

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 Serial Interface
Describes the functions and specifications of serial.

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 Quick-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	[Comm Device]	Refers to a soft key displayed on the screen.

- **Symbols used in syntax descriptions**

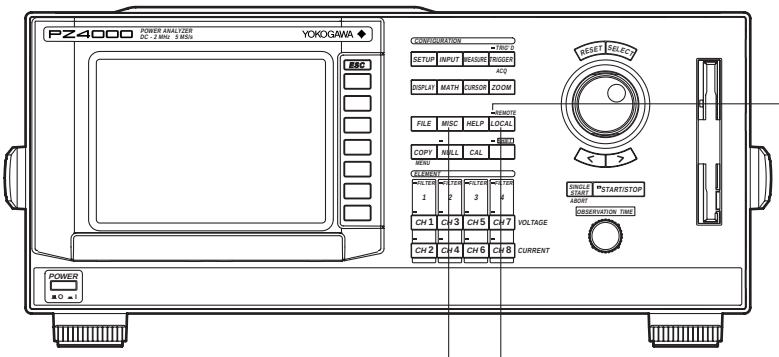
Symbols which are used in the syntax descriptions in Chapter 4 are shown below. These symbols are referred to as

Symbol	Description	Example	Example of Input
<>	Defined value	CHANnel <x> <x>=1 to 8	→CHANNEL2
{} |	One of the options in {} is selected. Exclusive OR	COUPling {AC|DC|GND}	→COUPLING AC
[]	Abbreviated	TRIGger [:SIMPle]:SLOPe	→TRIGger:SLOPer

Contents

1.1 Names of the Parts and Their Functions

Front Panel

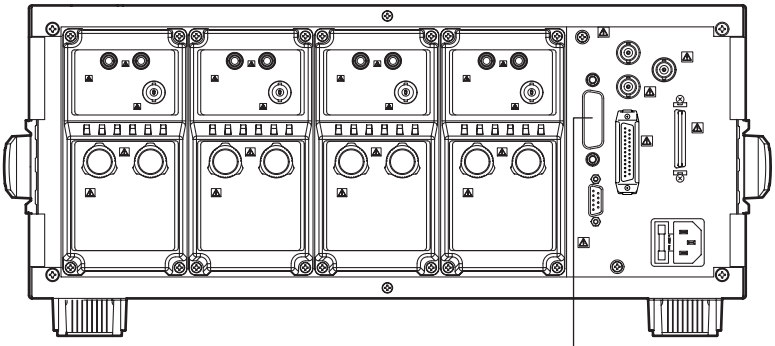


REMOTE indicator
Lit while remote mode is active via communications.

LOCAL key
Press this key to switch from remote mode to local mode so as to enable key operation.

MISC key
Press this key to set communication setting.

Rear Panel



GP-IB connector
Used to connect a controller (personal computer etc.) using a GP-IB cable.
For information on how to connect the GP-IB cable, refer to the following page.

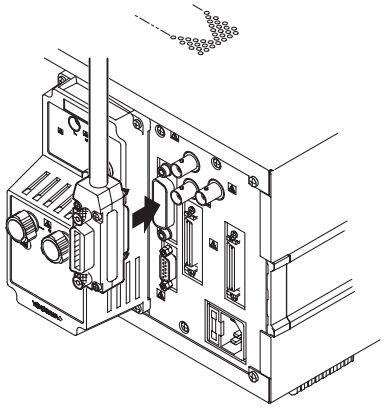
1.2 Connecting the GP-IB Cable

GP-IB Cable

The GP-IB connector on the side panel of the PZ4000 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.

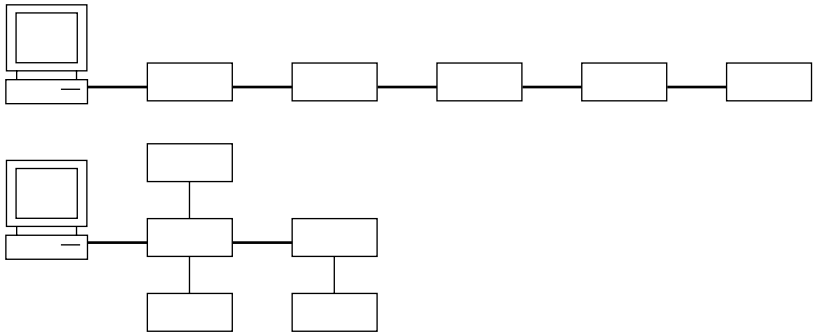
Connection Method

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.

Note

The talk-only, 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.
- All front panel keys except the LOCAL key can no longer be operated any more.
- Settings entered in local mode are retained.

When switched from Remote to Local Mode

Pressing the LOCAL 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 1-6).

- The REMOTE indicator 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 488.2-1987.
Code : ISO (ASCII) code
Mode : Addressable mode
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 LOCAL 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	T6	Basic talker capability, serial polling, untalk on MLA (My Listen Address), no 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	DT1	Device trigger capability
Controller	C0	No controller function
Electrical characteristic	E1	Open collector

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.

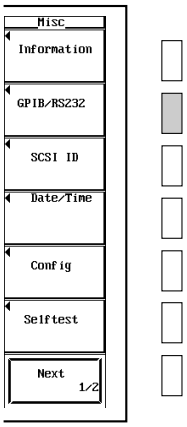
Note

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

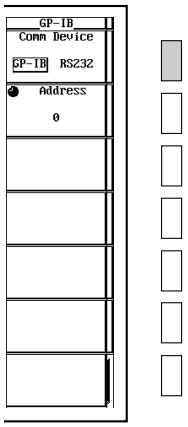
Operationg Procedure

1. Press the MISC key.
2. Press the "GP-IB/RS232" soft key.
3. Press the "Comm Device" soft key to select "GPIB."
4. Turn the jog shuttle to set the desired address.

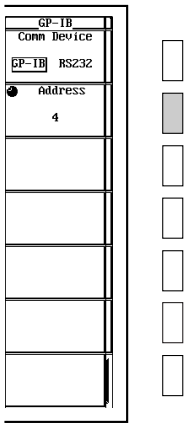
Step 1, 2



Step 3



Step 4



1.6 Response to Interface Messages

Response to Interface Messages

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 4-5).
*OPC and *OPC? will be disabled if they are currently being executed.
*WAI and COMMunicate:WAIT will be stopped immediately.

GET(Group Execute Trigger)

Operates in the sameway as the TRG command.

PPC (Parallel Poll Configure) and TCT (Take Control) are not supported

Response to a multi-line message (universal command)

LLO (Local Lockout)

Invalidates the LOCAL 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.

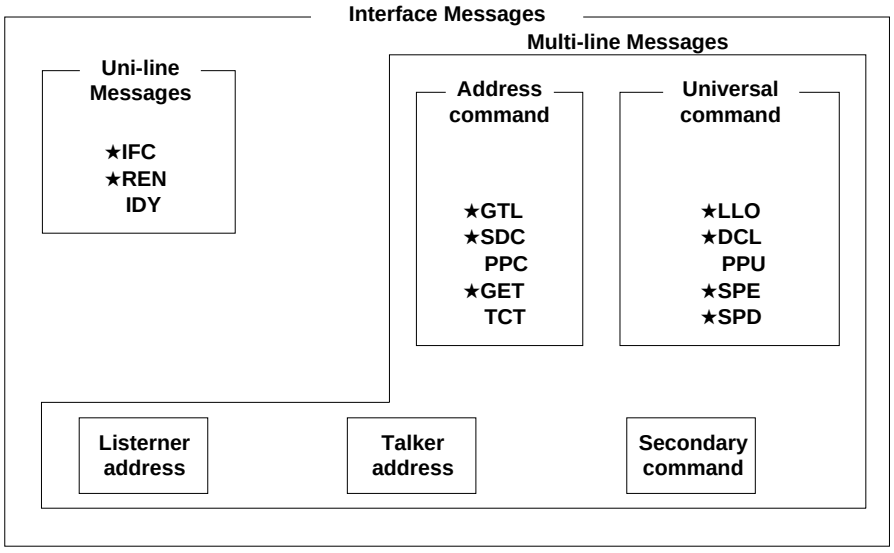
Commands valid for equipment designated as a listener
GTL (Go To Local)
SDC (Selected Device Clear)
PPC (Parallel Poll Configure)
GET (Group Execute Trigger)

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.



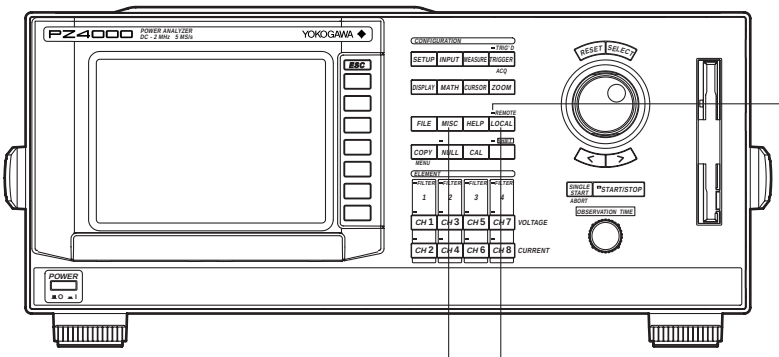
Messages marked with a "★" are interface messages supported by the PZ4000

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 Functions

Front Panel

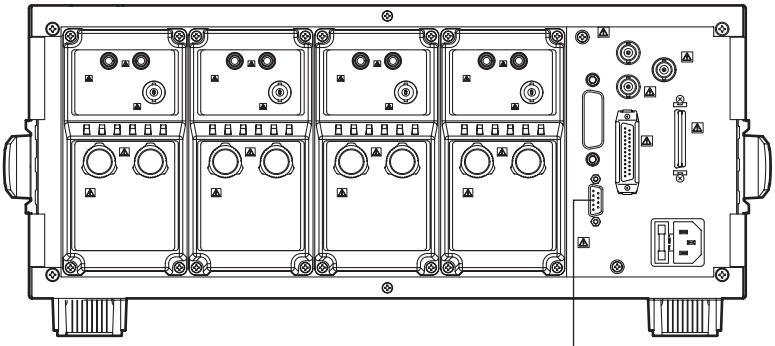


REMOTE indicator
Lit while remote mode is active via communications.

LOCAL key
Press this key to switch from remote mode to local mode so as to enable key operation.

MISC key
Press this key to set communication setting.

Rear Panel



Serial (RS-232) connector
Used to connect a controller (personal computer etc.) using a serial cable.
For information on how to connect the serial cable, refer to section 2.3.

2.2 Serial Interface Functions and Specifications

Receiving Function

It is possible to make the same settings via the serial 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.

Serial Interface Specifications

Electrical characteristics	: Complies with EIA-574 Standard (EIA-232 (RS-232) Standard for 9 pin)
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	: 256 bytes

Switching between Remote and Local Modes

when switched from Local to Remote Mode

Remote mode is activated when the “COMMunicate:REMote ON” command is received form a controller while local mode is active.

- REMOTE is displayed on.
- All front panel keys except the LOCAL key can no longer be operated any more.
- Settings entered in local mode are retained.

When switched from Remote to Local Mode

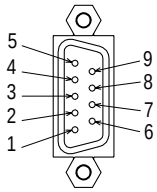
Pressing the LOCAL key in remote mode puts the instrument in local mode. However, this is not possible of Local Lockout (when the “COMMunicate:LOCKout ON” command is received) has been set by the controller (page 1-6).
Local mode is activated when the “COMMunicate:REMote OFF” command regardless of Local Lockout.

- The REMOTE indicator is turned off.
- All front panel keys are operative.
- Settings entered in remote mode are retained.

2.3 Connecting the Serial 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.

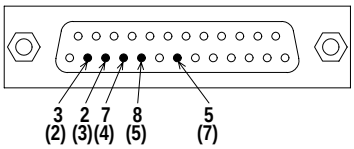
Connector and Signal Names



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 Serial interface.

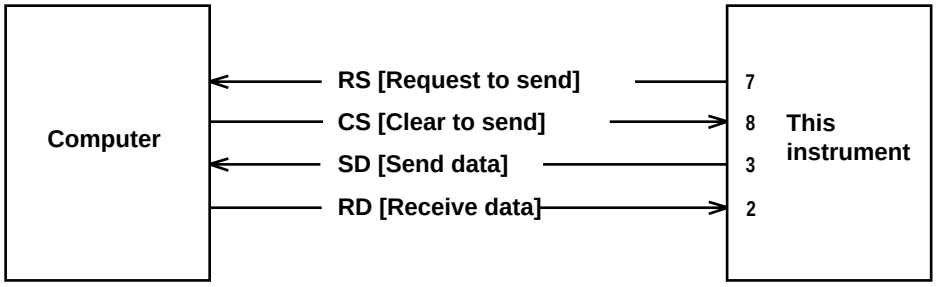


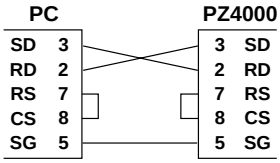
Table of Serial Standard Signals and their

Pin No. (9-pin connector)	Abbreviation			Description
	Serial (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

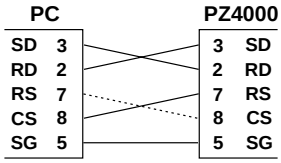
Signal line connection example

The pin numbers shown are that of 9-pin connectors.
In general, use a cross cable.

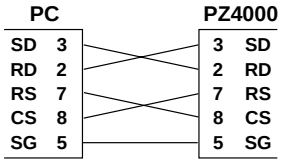
• OFF-OFF / XON-XON



• XON-RTS(XON-RS)



• CTS-RTS(CS-RS)



2.4 Handshaking

To use an serial 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

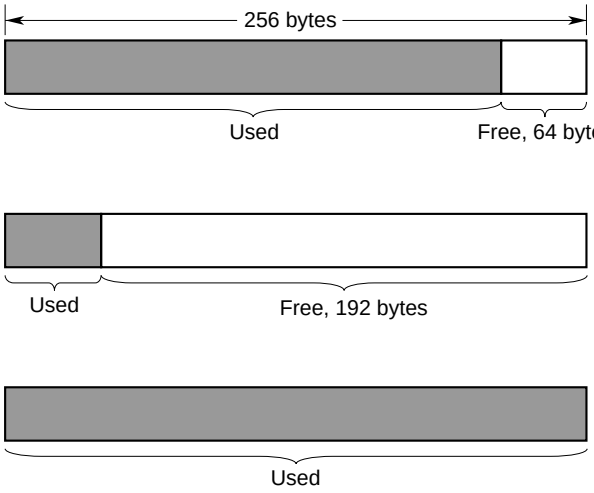
- **Transmission data control**
  A software handshake status is established between the instrument and host computer. The instrument will stop a data transmission when an X-OFF signal is received from the host computer, and will resume transmission when the next X-ON signal is received. A CS signal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



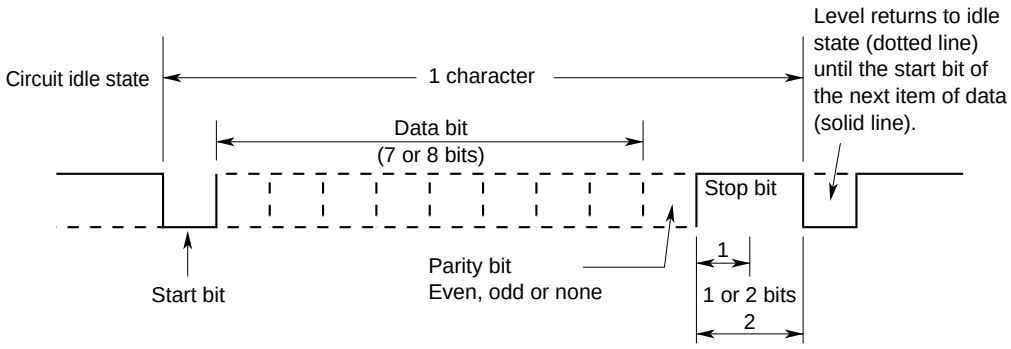
Data Receiving Control using Handshaking

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 serial 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

Operating Procedure

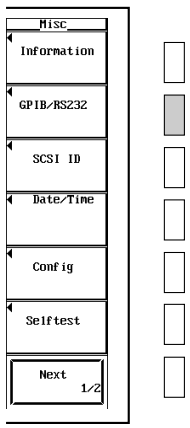
Displaying the Serial (RS-232) menu

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

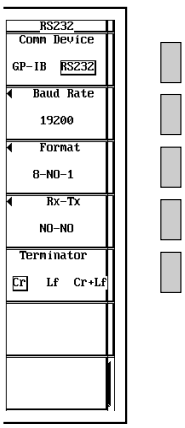
Selecting the baud rate, the data format and etc.

4. Press the “BaudRate” (baud rate), “Format” (data format), “Rx-Tx” (handshaking method), and the “Terminator” (terminator) soft keys individually, and set each item.

Step 1, 2



Step 3, 4



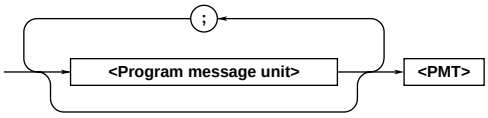
Chapter 3 Before Programming

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 message unit, 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 one 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 `:TRIGger:MODE AUTO;SOURCE 1<PMT>`
Unit Unit

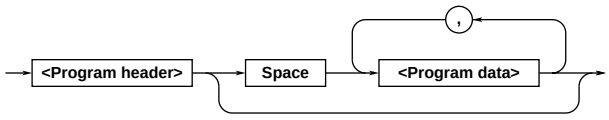
<PMT>

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.

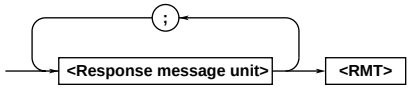
<Program data>

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 `:TRIGger:MODE AUTO<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.

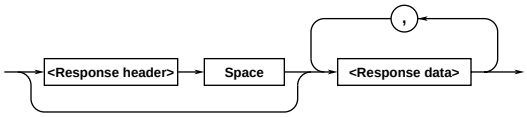
Example `:TRIGger:MODE AUTO;SOURCE 1<RMT>`
Unit Unit

<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.



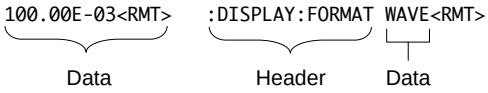
<Response header>

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 the transmission and reception buffer memories become full at the same time, the instrument will not be able to continue the communication operation. 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

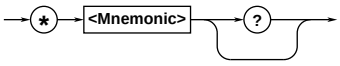
There are three types of command (program header) which can be sent from the controller to this instrument. They differ in the format of their program headers.

They are

- Common command header
- Compound header
- Simple header

Common Command Header

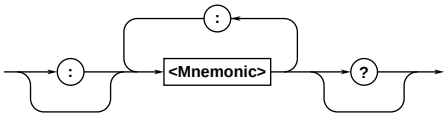
Commands defined in IEEE 488.2-1987 are called common commands. The header format of a common command is shown below. An asterisk (*) must always be attached to the beginning of a command.



An example of a common command
*CLS

Compound Header

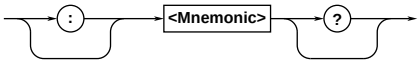
Commands designed to be used only with this instrument are classified and arranged in a hierarchy according to their function. The format of a compound header is illustrated below. A colon (:) must be used when specifying a lower-level header.



An example of a compound header
:ACQuire:DIVision

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?
:ACQuire:DIVision
:ACQuire:RLENgth
:ACQuire:TBASe
```

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:DIVision ON;TBASE INTernal
<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:DIVision ON;:DISPlay:FORMat
NUMeric<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:DIVision ON;: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:DIVision ON;*CLS;TBASe
INTernal<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:DIVision ON<PMT>:ACQuire:TBASe INTernal<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 `:TIMebase?<PMT>→:TIMEBASE:OBSERVE 100.00E-03;SRATE 1.000000E+06`

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 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 :DISPlay:FORMat?<PMT>→:DISPLAY:FORMAT WAVE<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 cmands can be returned after a header is attached to them.
  Example :CHANnel1:TYPE?<PMT>→VOLTAGE<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: Number of displayed digits for numerical data →SETup:RESolution 5)
<Voltage><Current> <Time><Frequency>	Physical value (Example: Waveform observation time →TIMebase:OBServe 100M)
<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: Measurement mode →SETup[:MODE] {NORMal|HARMonics})
<Boolean>	Indicates ON/OFF. Set to ON, OFF or value (Example: CH2 waveform display ON →CHANnel2:DISPlay ON)
<Character string data>	Arbitrary character string (Example: Waveform label of CH1 →CHANnel:LABel "CH1")
<Filename>	Gives the name of a file. (Example: Name of file to be saved →FILE:SAVE:WAVE[:EXECute] "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> E4 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>, <Current>, <Time>, <Frequency>

<Voltage>, <Current>, <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}

<Unit>

Units which can be used are shown below.

Symbol	Word	Description
V	Volt	Voltage
A	Ampere	Current
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
{NORMal|HARMonics}	NORMAL

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 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:RANGe] command, for example, is a sequential command. Assume that you set a new voltage range value and immediately request return of the new value, as follows:
:CHANnel1:VOLTage:RANGe 200V;RANGe?<PMT>
In this case, the oscilloscope always returns the newest setting ("200V"). This is because it always completes processing of the current sequential command (in this case, "RANGe 200V") before moving on to the next command ("RANGe?").
In contrast, assume that you begin a file load and then immediately query the voltage range value:
:FILE:LOAD:SETup "FILE1";:CHANnel1:VOLTage:RANGe?
Because "FILE:LOAD:SETup" is an overlapped command, the oscilloscope will advance to the "CHANNel1:VOLTage:RANGe?" command before it finishes the load. The returned voltage range 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 "FILE1";*WAI;:CHANnel1:VOLTage:<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:RANGe?",
"CHANnel1:VOLTage:RANGe?" 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
"FILE1";:CHANnel1:VOLTage:VOLTage:RANGe?<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" s a sequential command, ensuring that the "CHANnel1:VOLTage:RANGe?" 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 "FILE1";*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:RANGe?" 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
"FILE1";*OPC?<PMT>
(Response to *OPC? is decoded.)
:CHANnel1:VOLTage:RANGe?<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".

Chapter 4 Commands

4.1 Command Listing

Command	Function	Page
ABORt Group		
:ABORt	Aborts data acquisition.	4-11
ACQuire Group		
:ACQuire?	Queries all settings related to data acquisition.	4-11
:ACQuire:DIVision	Sets whether or not to divide the record length or queries the current setting.	4-11
:ACQuire:RLENgth	Sets the record length or queries the current setting.	4-11
:ACQuire:TBASe	Sets the sampling block or queries the current setting.	4-11
CHANnel Group		
:CHANnel<x>?	Queries all settings related to the vertical axis of each channel.	4-13
:CHANnel<x>:CURRent?	Queries all settings related to the current input channel.	4-13
:CHANnel<x>:CURRent:RANGe	Sets the current range of the current input channel or queries the current setting.	4-13
:CHANnel<x>:CURRent:SRATio	Sets the current sensor's scaling constant of the current input channel or queries the current setting.	4-13
:CHANnel<x>:CURRent:TERMinal	Sets the current measurement terminal of the current input channel or queries the current setting.	4-13
:CHANnel<x>:DISPlay	Turns ON/OFF the waveform display of each channel or queries the current setting.	4-13
:CHANnel<x>:LABel	Sets the waveform label of each channel or queries the current setting.	4-14
:CHANnel<x>:POSition	Sets the vertical position (the GND position) of each channel or queries the current setting.	4-14
:CHANnel<x>:SPEed?	Queries all settings related to the revolution sensor signal input channel.	4-14
:CHANnel<x>:SPEed:FRANge	Sets the frequency range of the revolution sensor signal input channel (pulse input) or queries the current setting.	4-14
:CHANnel<x>:SPEed:RANGe	Sets the input range of the revolution sensor signal input channel or queries the current setting.	4-14
:CHANnel<x>:SPEed:TYPE	Sets the input type of the revolution sensor signal input channel or queries the current setting.	4-15
:CHANnel<x>:TORQue?	Queries all settings related to the torque meter signal input channel.	4-15
:CHANnel<x>:TORQue:RANGe	Sets the input range of the torque meter signal input channel or queries the current setting.	4-15
:CHANnel<x>:TYPE?	Queries the input type of each channel.	4-15
:CHANnel<x>:VOLTage?	Queries all settings related to the voltage input channel.	4-15
:CHANnel<x>:VOLTage:RANGe	Sets the voltage range of the voltage input channel or queries the current setting.	4-15
:CHANnel<x>:VZoom	Sets the vertical zoom factor or queries the current setting.	4-15
COMMunicate Group		
:COMMunicate?	Queries all settings related to communications.	4-16
:COMMunicate:HEADer	Sets whether or not to attach headers to response data or queries the current setting.	4-16
:COMMunicate:LOCKout	Sets/releases local lockout.	4-16
:COMMunicate:OPSE	Sets the overlap commands for *OPC, *OPC?, and *WAI or queries the current setting.	4-17
:COMMunicate:OPSR?	Queries the operation pending status register.	4-17
:COMMunicate:OVERlap	Sets the commands to permit overlap operation or queries the current setting.	4-17
:COMMunicate:REMote	Switches between remote and local.	4-17
:COMMunicate:STATus?	Queries the line-specific status.	4-17
:COMMunicate:VERBose	Sets whether to use the full or abbreviated form for response data or queries the current setting.	4-17
:COMMunicate:WAIT	Waits for an extended event to occur.	4-17
:COMMunicate:WAIT?	Generates a response when one of the specified extended events occurs.	4-17
CURSor Group		
:CURSor?	Queries all settings related to cursor measurements.	4-19
:CURSor:HORizontal?	Queries all settings related to the H cursor.	4-19
:CURSor:HORizontal:DY?	Queries the Y-axis value between the H cursors.	4-19
:CURSor:HORizontal:POSition<x>	Sets the H cursor position or queries the current setting.	4-19

Command	Function	Page
:CURSor:HORizontal:TRACe	Sets the waveform on which to place the H cursor or queries the current setting.	4-19
:CURSor:HORizontal:Y<x>?	Queries the Y-axis value of the H cursor.	4-20
:CURSor:MARKer?	Queries all settings related to the marker.	4-20
:CURSor:MARKer:DX?	Queries the X-axis value between the marker.	4-20
:CURSor:MARKer:DY?	Queries the Y-axis value between the marker.	4-20
:CURSor:MARKer:FFT<x>	Sets the X-axis value of the marker position for the FFT result or queries the current setting.	4-20
:CURSor:MARKer:JUMP	Jumps to the zoomed waveform of the marker.	4-20
:CURSor:MARKer:PERDt?(1 PER Delta T)	Queries the 1/Δ value of the horizontal axis between the marker.	4-20
:CURSor:MARKer:POSition<x>	Sets the marker position or queries the current setting.	4-21
:CURSor:MARKer:TRACe<x>	Sets the waveform on which to place the marker or queries the current setting.	4-21
:CURSor:MARKer:X<x>?	Queries the X-axis value of the marker position.	4-21
:CURSor:MARKer:Y<x>?	Queries the Y-axis value of the marker position.	4-21
:CURSor:[TYPE]	Sets the cursor type or queries the current setting.	4-21
:CURSor:VERTical?	Queries all settings related to the V cursor.	4-21
:CURSor:VERTical:DX?	Queries the X-axis value between the V cursors.	4-21
:CURSor:VERTical:FFT<x>	Sets the V cursor position with respect to the FFT result.	4-21
:CURSor:VERTical:PERDt?	Queries the 1/Δ value of the horizontal axis between the V cursors.	4-22
:CURSor:VERTical:POSition<x>	Sets the V cursor position or queries the current setting.	4-22
:CURSor:VERTical:TRACe	Sets the waveform on which to place the V cursor or queries the current setting.	4-22
:CURSor:VERTical:X<x>?	Queries the X-axis value of the V cursor position.	4-22
:CURSor:XY?	Queries all settings related to the XY cursor.	4-22
:CURSor:XY:DX?	Queries the X-axis value between the XY cursors.	4-22
:CURSor:XY:POSition<x>	Sets the XY cursor position or queries the current setting.	4-22
:CURSor:XY:TRACe?	Queries the waveform on which the XY cursor is placed.	4-22
:CURSor:XY:X<x>?	Queries the X-axis value of the XY cursor position.	4-22
DISPlay Group		
:DISPlay?	Queries all settings related to the screen display.	4-25
:DISPlay:BAR?	Queries all settings related to the bar graph display.	4-25
:DISPlay:BAR:CURSor<x>	Sets the marker position (harmonic order) on the bar graph display or queries the current setting.	4-25
:DISPlay:BAR:ITEM<x>	Sets the bar graph display items (function, element) or queries the current setting.	4-25
:DISPlay:BAR:ORDer	Sets the start and end harmonic orders of the bar graph display or queries the current setting.	4-25
:DISPlay:DATE	Turns ON/OFF the date and time displays or queries the current setting.	4-26
:DISPlay:FORMat	Sets the display format or queries the current setting.	4-26
:DISPlay:NUMeric?	Queries all settings related to the numerical display.	4-26
:DISPlay[:NUMeric]:HARMonics?	Queries all settings related to the numerical display during harmonic measurement.	4-26
:DISPlay[:NUMeric]:HARMonics:IAMount	Sets the numerical display format during harmonic measurement or queries the current setting.	4-26
:DISPlay[:NUMeric]:HARMonics:ICURsor	Sets the cursor position of the numerical display during harmonic measurement or queries the current setting.	4-27
:DISPlay[:NUMeric]:HARMonics:ITEM<x>	Sets the numerical displayed items during harmonic measurement or queries the current setting.	4-27
:DISPlay[:NUMeric]:HARMonics:LCURsor	Sets the cursor position on the list display during harmonic measurement or queries the current setting.	4-27
:DISPlay[:NUMeric]:HARMonics:LIST<x>	Sets the list display items during harmonic measurement or queries the current setting.	4-27
:DISPlay[:NUMeric]:HARMonics:PRESet	Sets the numerical display items to a preset pattern during harmonic measurement.	4-27

Command	Function	Page
:DISPlay[:NUMeric]:NORMal?	Queries all settings related to the numerical display during normal measurement.	4-27
:DISPlay[:NUMeric]:NORMal:FCURsor	Sets the cursor position of the numerical display (All display) during normal measurement or queries the current setting.	4-28
:DISPlay[:NUMeric]:NORMal:IAMount	Sets the numerical display format during normal measurement or queries the current setting.	4-28
:DISPlay[:NUMeric]:NORMal:ICURsor	Sets the cursor position of the numerical display (split display) during normal measurement or queries the current setting.	4-28
:DISPlay[:NUMeric]:NORMal:ITEM<x>	Sets the numerical displayed item during normal measurement or queries the current setting.	4-28
:DISPlay[:NUMeric]:NORMal:PRESet	Sets the numerical display items to a preset pattern during normal measurement.	4-28
:DISPlay:VECTor?	Queries all settings related to the vector display.	4-28
:DISPlay:VECTor:IMAG	Sets the zoom factor of the current display during vector display or queries the current setting.	4-28
:DISPlay:VECTor:NUMeric	Turns ON/OFF the numerical data display during vector display or queries the current setting.	4-29
:DISPlay:VECTor:UMAG	Sets the zoom factor of the voltage display during vector display or queries the current setting.	4-29
:DISPlay:WAVE?	Queries all settings related to the waveform display.	4-29
:DISPlay:WAVE:{CHANnel<x>|MATH<x>}	Turns ON/OFF the channel/computed waveform display or queries the current setting.	4-29
:DISPlay:WAVE:FORMat	Sets the display format of the waveform or queries the current setting.	4-29
:DISPlay:WAVE:GRATicule	Sets the graticule type (grid) or queries the current setting.	4-29
:DISPlay:WAVE:INTerpolate	Sets the interpolation method of the waveform or queries the current setting.	4-29
:DISPlay:WAVE:MAPPing?	Queries all settings related to the waveform mapping to the split screen.	4-29
:DISPlay:WAVE:MAPPing:{CHANnel<x>|MATH<x>}	Sets the {channel waveform|MATH waveform} mapping to the split screen or queries the current setting.	4-29
:DISPlay:WAVE:MAPPing[:MODE]	Sets the waveform mapping method for the split screen or queries the current setting.	4-30
:DISPlay:WAVE:SVALue	Turns ON/OFF the scale value display or queries the current setting.	4-30
:DISPlay:WAVE:TLABel	Turns ON/OFF the waveform label display or queries the current setting.	4-30
:DISPlay:XY?	Queries all settings related to the X-Y display.	4-30
:DISPlay:XY:FFT	Sets the range of the FFT waveform to be displayed on the X-Y display or queries the current setting.	4-30
:DISPlay:XY:INTerpolate	Sets the interpolation method of the waveform or queries the current setting.	4-30
:DISPlay:XY:POSition	Sets the range of the T-Y waveform to be displayed on the X-Y display or queries the current setting.	4-31
:DISPlay:XY:XTRace	Sets the channel to assign to the X-axis of the X-Y display or queries the current setting.	4-31
FILE Group		
:FILE?	Queries all settings related to file operations.	4-34
:FILE:CDIRectory	Changes the current directory.	4-34
:FILE:DELete:IMAGe:{TIFF|BMP|PSCRipt}	Deletes a screen image data file.	4-34
:FILE:DELete:NUMeric:{ASCii|FLOat}	Deletes a numerical data file.	4-34
:FILE:DELete:SETup	Deletes a setup parameter file.	4-34
:FILE:DELete:WAVE:{BINary|ASCii|FLOat}	Deletes a waveform data file.	4-34
:FILE:DRIVe	Sets the drive (medium) setting.	4-34
:FILE:FORMat	Formats the floppy disk.	4-34
:FILE:FREE?	Queries the free space (bytes) on the drive.	4-34

Command	Function	Page
:FILE:LOAD:ABORt	Aborts loading a file.	4-34
:FILE:LOAD:SETup	Loads a setup parameter file.	4-35
:FILE:LOAD:WAVE	Loads a waveform data file.	4-35
:FILE:MDIRectory	Creates a directory.	4-35
:FILE:PATH?	Queries the absolute path of the current directory.	4-35
:FILE:SAVE?	Queries all settings related to saving a file.	4-35
:FILE:SAVE:ABORt	Aborts saving the file.	4-35
:FILE:SAVE:ANAMing	Sets whether or not to automatically assign file names or queries the current setting.	4-35
:FILE:SAVE:COMMent	Sets the comment that is attached to the file being saved or queries the current setting.	4-35
:FILE:SAVE:NUMeric?	Queries all settings related to saving the numerical data to a file.	4-35
:FILE:SAVE:NUMeric[:EXECute]	Saves the numerical data to a file.	4-35
:FILE:SAVE:NUMeric:LIST?	Queries all settings related to saving the numerical list data to a file during harmonic measurement.	4-35
:FILE:SAVE:NUMeric:LIST:ELEMent<x>	Turns ON/OFF the output of each element when saving numerical list data to a file during harmonic measurement or queries the current setting.	4-35
:FILE:SAVE:NUMeric:LIST:{<List-Function>|SIGMa}	Turns ON/OFF the output of each function when saving numerical list data to a file during harmonic measurement or queries the current setting.	4-35
:FILE:SAVE:NUMeric:TYPE	Sets the format of the numerical data being saved or queries the current setting.	4-36
:FILE:SAVE:SETup[:EXECute]	Saves the setup parameters to a file.	4-36
:FILE:SAVE:WAVE?	Queries all settings related to saving the waveform data to a file.	4-36
:FILE:SAVE:WAVE[:EXECute]	Saves the waveform data to a file.	4-36
:FILE:SAVE:WAVE:RANGe	Sets the range of the waveform to save to the file or queries the current setting.	4-36
:FILE:SAVE:WAVE:TRACe	Sets the waveform to save to the file or queries the current setting.	4-36
:FILE:SAVE:WAVE:TYPE	Sets the format of the waveform data being saved or queries the current setting.	4-36
HCOPy Group		
:HCOPy?	Queries all settings related to screen data output.	4-38
:HCOPy:ABORt	Aborts data output and paper feeding.	4-38
:HCOPy:CENTronics?	Queries all settings related to the external printer output.	4-38
:HCOPy:CENTronics:COLor	Sets the color (ON/OFF) of the external printer output or queries the current setting.	4-38
:HCOPy:CENTronics:FORMat	Sets the command format that is output to the printer or queries the current setting.	4-38
:HCOPy:COMMent	Sets the comment that is printed at the lower section of the screen or queries the current setting.	4-38
:HCOPy:DIRection	Sets the output destination of the data or queries the current setting.	4-38
:HCOPy:EXECute	Executes data output.	4-38
:HCOPy:FORMat	Sets the output data format or queries the current setting.	4-38
:HCOPy:PRINter:DLISt	Executes output of the numerical data list to the built-in printer.	4-38
:HCOPy:PRINter:FEED	Feeds the paper (built-in printer).	4-38
:HCOPy:SAVE?	Queries all settings related to saving the file.	4-38
:HCOPy:SAVE:ANAMing	Sets whether or not to automatically assign file names or queries the current setting.	4-38
:HCOPy:SAVE:COMMent	Sets the comment that is attached to the file being saved or queries the current setting.	4-39
:HCOPy:SAVE:NAME	Sets the file name or queries the current setting.	4-39
:HCOPy:{TIFF|BMP}?	Queries all settings related to the TIFF/BMP format.	4-39
:HCOPy:{TIFF|BMP}:COLor	Sets the color for the TIFF/BMP format or queries the current setting.	4-39
:HCOPy:{TIFF|BMP}:COMPression	Sets whether or not to compress the data in TIFF/BMP format or queries the current setting.	4-39
IMAGe Group		
:IMAGe?	Queries all settings related to the output of the screen image data.	4-40
:IMAGe:COLor	Sets the color of the screen image data being output or queries the current setting.	4-40
:IMAGe:FORMat	Sets the output format of the screen image data or queries the current setting.	4-40
:IMAGe:SEND?	Queries the screen image data.	4-40
INPut Group		
:INPut?	Queries all settings related to all input modules.	4-44

Command	Function	Page
MATH Group		
:MATH<x>?	Queries all settings related to computations.	4-53
:MATH<x>:EXECute	Executes computation.	4-53
:MATH<x>:EXPRession	Sets the equation or queries the current setting.	4-54
:MATH<x>:FFT?	Queries all settings related to the FFT.	4-54
:MATH<x>:FFT:POINt	Sets the number of points for the FFT or queries the current setting.	4-54
:MATH<x>:FFT:WINDow	Sets the window function for the FFT or queries the current setting.	4-54
:MATH<x>:FUNCtion	Enables/disables the computation function or queries the current setting.	4-54
:MATH<x>[:MODE]	Turns ON/OFF the computation or queries the current setting.	4-54
:MATH<x>:POINt	Sets the start and end points of the computation or queries the current setting.	4-54
:MATH<x>:SCALing?	Queries all settings related to scale converting.	4-54
:MATH<x>:SCALing:MODE	Sets the scale converting or queries the current setting.	4-55
:MATH<x>:SCALing:VALue	Sets the upper and lower limits for manual scaling or queries the current setting.	4-55
:MATH<x>:UNIT	Sets the unit to attach to the computed result or queries the current setting.	4-55
MEASure Group		
:MEASure?	Queries all settings related to measurements.	4-57
:MEASure:AVERaging?	Queries all settings related to averaging.	4-57
:MEASure:AVERaging:COUNt	Sets the number of averaging counts or queries the current setting.	4-57
:MEASure:AVERaging[:STATe]	Turns ON/OFF the averaging function or queries the current setting.	4-57
:MEASure:DMeasure	Sets the delta computation or queries the current setting.	4-58
:MEASure:FUNCtion<x>?	Queries all settings related to the user-defined function.	4-58
:MEASure:FUNCtion<x>:EXPRession	Sets the equation for the user-defined function or queries the current setting.	4-58
:MEASure:FUNCtion<x>:[:STATe]	Enable/disable the user-defined function or queries the current setting.	4-58
:MEASure:FUNCtion<x>:UNIT	Sets the unit to attach to the computed result of the user-defined function or queries the current setting.	4-58
:MEASure:HARMonics?	Queries all settings related to the measurement during harmonic measurement.	4-58
:MEASure:HARMonics:ORDer	Sets the minimum and maximum harmonic orders to be analyzed during harmonic measurement or queries the current setting.	4-58
:MEASure:HARMonics:THD	Sets the equation used to determine the THD (total harmonic distortion) during harmonic measurement or queries the current setting.	4-58
:MEASure[:MODE]	Turns ON/OFF the measurement computation or queries the current setting.	4-58
:MEASure:PC?	Queries all settings related to determination of Pc (Corrected Power).	4-59
:MEASure:PC:IEC	Sets the equation used to determine the Pc (Corrected Power) or queries the current setting.	4-59
:MEASure:PC:P<x>	Sets the parameters used to determine the Pc (Corrected Power) or queries the current setting.	4-59
:MEASure:PERiod?	Queries all settings related to the computation period.	4-59
:MEASure:PERiod:CURSor?	Queries all settings when specifying the computation period with the cursors.	4-59
:MEASure:PERiod:CURSor[:POSition]	Sets the computation period when specifying the period with the cursors or queries the current setting.	4-59
:MEASure:PERiod:ETRigger?	Queries all settings when using the external trigger signal to determine the computation period.	4-59
:MEASure:PERiod:ETRigger[:PATTern]	Sets the pattern that is used when determining the computation period with the external trigger signal or queries the current setting.	4-59
:MEASure:PERiod:EXECute	Executes the computation.	4-60
:MEASure:PERiod[:MODE]	Sets the method used to specify the computation period or queries the current setting.	4-60
:MEASure:PERiod:ZCRoss?	Queries all settings when using the zero crossing detection to determine the computation period.	4-60
:MEASure:PERiod:ZCRoss:SYNChronize?	Sets the synchronizing source for all elements when using the zero crossing detection to determine the computation period.	4-60

Command	Function	Page
:MEASure:PERiod:ZCRoss[:SYNChronize]:ELEMent<x>		
	Sets the synchronizing source for each element when using the zero crossing detection to determine the computation period.	4-60
:MEASure:PHASe	Sets the display format of the phase difference or queries the current setting.	4-60
:MEASure:SFORmula	Sets the equation used to determine S (apparent power) or queries the current setting.	4-60
NULL Group		
:NULL	Turns ON/OFF the NULL function or queries the current setting.	4-61
NUMeric Group		
:NUMeric?	Queries all settings related to the numerical data output.	4-63
:NUMeric:FORMat	Sets the format of the numerical data that are sent using the ":NUMeric:{NORMal|HARMonics|LIST}:VALue?" command or queries the current setting.	4-63
:NUMeric:HARMonics?	Queries all settings related to the numerical data output during harmonic measurement.	4-63
:NUMeric:HARMonics:CLEar	Clears the numerical data output items during harmonic measurement.	4-63
:NUMeric:HARMonics:ITEM<x>	Sets the numerical data output items during harmonic measurement or queries the current setting.	4-63
:NUMeric:HARMonics:NUMber	Sets the number of numerical data that are sent using the ":NUMeric:HARMonics:VALue?" command or queries the current setting.	4-63
:NUMeric:HARMonics:PRESet	Sets the numerical data output items to a preset pattern during harmonic measurement.	4-64
:NUMeric:HARMonics:VALue?	Queries the numerical data during harmonic measurement.	4-64
:NUMeric:LIST?	Queries all settings related to the output of the numerical list data during harmonic measurement.	4-64
:NUMeric:LIST:ITEM	Sets the output items of the numerical list data during harmonic measurement or queries the current setting.	4-64
:NUMeric:LIST:ORDer	Sets the maximum harmonic order of the numerical list data to output during harmonic measurement or queries the current setting.	4-64
:NUMeric:LIST:SELect	Sets the output components of the numerical list data during harmonic measurement or queries the current setting.	4-64
:NUMeric:LIST:VALue?	Queries the numerical list data during harmonic measurement.	4-64
:NUMeric:NORMal?	Queries all settings related to the numerical data output during normal measurement.	4-64
:NUMeric[:NORMal]:CLEar	Clears the numerical data output items during normal measurement.	4-65
:NUMeric[:NORMal]:ITEM<x>	Sets the numerical data output items during normal measurement or queries the current setting.	4-65
:NUMeric[:NORMal]:NUMber	Sets the number of numerical data during normal measurement or queries the current setting.	4-65
:NUMeric[:NORMal]:PRESet	Sets the numerical data output items to a preset pattern during normal measurement.	4-65
:NUMeric[:NORMal]:VALue?	Queries the numerical data during normal measurement.	4-65
SETup Group		
:SETup?	Queries all settings related to the measurement mode.	4-69
:SETup:INITialize	Initializes the settings.	4-69
:SETup[:MODE]	Sets the measurement mode or queries the current setting.	4-69
:SETup:PLLSource	Sets the PLL source during harmonic measurement or queries the current setting.	4-69
:SETup:RESolution	Sets the number of displayed digits for numerical data or queries the current setting.	4-69
:SETup:WIRing	Sets the wiring method or queries the current setting.	4-70
SSTart Group		
:SSTart	Executes single start.	4-70
STARt Group		
:STARt	Starts data acquisition.	4-70
STATus Group		
:STATus?	Queries all settings related to the communication status function.	4-71
:STATus:CONDition?	Queries the status register.	4-71
:STATus:EESE(Extended Event Status Enable register)		
	Sets the extended event enable register or queries the current setting.	4-71

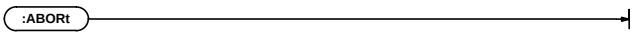
Command	Function	Page
:STATus:EESR?(Extended Event Status Register)		
	Queries and clears the extended event register.	4-71
:STATus:ERRor?	Queries the code and information of the error.	4-72
:STATus:FILTer<x>	Sets the transition filteror queries the current setting.	4-72
:STATus:QENable	Sets whether or not to store messages other than errors in the error queue or queries the current setting.	4-72
:STATus:QMESsage	Sets whether or not to attach a message to the "STATus:ERRor?" response or queries the current setting.	4-72
:STATus:SPOLl?(Serial Poll)	Executes serial polling.	4-72
STOP Group		
:STOP	Stops data acquisition.	4-72
SYSTem Group		
:SYSTem?	Queries all settings related to the system.	4-74
:SYSTem:DATE	Sets the date or queries the current setting.	4-74
:SYSTem:LANGuage	Sets the message language or queries the current setting.	4-74
:SYSTem:LCD?	Queries all settings related to the LCD monitor.	4-74
:SYSTem:LCD:BRIGhtness	Sets the brightness of the LCD monitor or queries the current setting.	4-74
:SYSTem:LCD:COLor?	Queries all settings related to the display colors of the LCD monitor.	4-74
:SYSTem:LCD:COLor:GRAPh?	Queries all settings related to the display color of graphic items.	4-74
:SYSTem:LCD:COLor:GRAPh:{BACKground|GRATicule|CURSor|CHANnel<x>|MATH<x>}		
	Queries the display color for the background/graticule/cursor/channel waveform/MATH waveform or queries the current setting.	4-74
:SYSTem:LCD:COLor:GRAPh:MODE	Sets the display color mode of graphic items or queries the current setting.	4-74
:SYSTem:LCD:COLor:TEXT?	Queries all settings related to the display color of text items.	4-74
:SYSTem:LCD:COLor:TEXT:{LETTer|BACKground|BOX|SUB|SELected}		
	Sets the display colors for characters (Menu Fore)/menu background (Menu Back)/selected menu (Select Box)/popup menu (Sub Menu)/selected key (Selected Key) or queries the current setting.	4-75
:SYSTem:LCD:COLor:TEXT:MODE	Sets the display color mode of text items or queries the current setting.	4-75
:SYSTem:SCSI?	Queries all settings related to the SCSI-ID.	4-75
:SYSTem:SCSI:INITialize	Initializes SCSI related settings.	4-75
:SYSTem:SCSI:OWNid	Sets the SCSI ID of this instrument or queries the current setting.	4-75
:SYSTem:TIME	Sets the time or queries the current setting.	4-75
TIMebase Group		
:TIMebase?	Queries all settings related to the time base (horizontal axis).	4-76
:TIMebase:OBServe	Sets the observation time of the waveform or queries the current setting.	4-76
:TIMebase:SRATe	Sets the sampling rate or queries the current setting.	4-76
TRIGger Group		
:TRIGger?	Queries all settings related to the trigger.	4-78
:TRIGger:ACTion?	Queries all settings related to action-on-trigger.	4-78
:TRIGger:ACTion:ACQCount	Sets the action count of action-on-trigger or queries the current setting.	4-78
:TRIGger:ACTion:HCOPy	Sets whether or not to output screen image data (ON/OFF) when an action is activated, or queries the current setting.	4-78
:TRIGger:ACTion:SAVE	Sets whether or not to save the waveform data to the storage medium (ON/OFF) when an action is activated, or queries the current setting.	4-78
:TRIGger:DELay	Sets the trigger delay or queries the current setting.	4-78
:TRIGger:DREFerence	Sets the trigger position or queries the current setting.	4-78
:TRIGger:EDGE?	Queries all settings related to the edge trigger.	4-78
:TRIGger:EDGE:LEVel	Sets the trigger level for the edge trigger or queries the current setting.	4-78
:TRIGger:EDGE:SLOPe	Sets the trigger slope for the edge trigger or queries the current setting.	4-78
:TRIGger:MODE	Sets the trigger mode or queries the current setting.	4-79
:TRIGger:SOURce	Sets the trigger source or queries the current setting.	4-79
:TRIGger:TYPE	Sets the trigger type or queries the current setting.	4-79
:TRIGger:WINDow?	Queries all settings related to the window trigger.	4-79

Command	Function	Page
:TRIGger:WINDow:CENTer	Sets the center level for the window trigger or queries the current setting.	4-79
:TRIGger:WINDow:CONDition	Sets the trigger condition for the window trigger or queries the current setting.	4-79
:TRIGger:WINDow:WIDTh	Sets the window width for the window trigger or queries the current setting.	4-79
WAVeform Group		
:WAVeform?	Queries all settings related to the waveform data.	4-80
:WAVeform:BYTeorder	Sets the byte order of the waveform data or queries the current setting.	4-80
:WAVeform:END	Sets the end point of the output of the waveform data or queries the current setting.	4-80
:WAVeform:FORMat	Sets the format of the waveform data or queries the current setting?	4-81
:WAVeform:LENGth?	Queries the total number of data points of the waveform.	4-81
:WAVeform:RANGe?	Queries the range value that is used to convert the waveform to physical data.	4-81
:WAVeform:SEND?	Queries the waveform data.	4-81
:WAVeform:SRATe?	Queries the sampling rate of the acquired data.	4-81
:WAVeform:STARt	Sets the start point of the output of the waveform data or queries the current setting.	4-82
:WAVeform:TDATe?	Queries the string containing the trigger date and time when the waveform was acquired.	4-82
:WAVeform:TRACe	Sets the waveform or queries the current setting.	4-82
:WAVeform:TRIGger?	Queries the trigger position of the acquired data.	4-82
:WAVeform:ZCRoss?	Queries zero crossing data of all channels.	4-82
ZOOM Group		
:ZOOM?	Queries all settings related to the zooming of the waveform.	4-83
:ZOOM:ALLOcation?	Queries all settings related to the zoomed waveform.	4-83
:ZOOM:ALLOcation:{CHANnel<x>|MATH<x>}	Sets whether or not to select the waveform to be zoomed or queries the current setting.	4-84
:ZOOM:FORMat	Sets the display format of the zoomed waveform or queries the current setting.	4-84
:ZOOM:MAG<x>	Sets the zoom factor or queries the current setting.	4-84
:ZOOM[:MODE]	Sets the the display mode of the zoomed waveform or queries the current setting.	4-84
:ZOOM:POSition<x>	Sets the position of the zoom box or queries the current setting.	4-84
Common Command Group		
*CAL?(CALibrate)	Performs calibration (zero level compensation) and queries the result.	4-85
*CLS(CLear Status)	Clears the standard event register, extended event register, and error queue.	4-85
*ESE(standard Event Status Enable register)	Sets the standard event enable register or queries the current setting.	4-85
*ESR?(standard Event Status Register)	Queries the standard event register and clears the register.	4-86
*IDN?(IDeNtify)	Queries the instrument model.	4-86
*OPC(OPeration Complete)	After the completion of the specified overlap command, sets the OPC event.	4-86
*OPC?(OPeration Complete)	Creates a response, after the completion of the specified overlap command.	4-86
*OPT?(OPTion)	Queries installed options.	4-86
*PSC(Power-on Status Clear)	Sets whether or not to clear each register on power up or queries the current setting.	4-86
*RST(ReSeT)	Initializes the command group settings.	4-86
*SRE(Service Request Enable register)	Sets the service request enable register or queries the current setting.	4-87
*STB?(STatus Byte)	Queries the status byte register.	4-87
*TRG(TRiGger)	Executes single start.	4-87
*TST(TeST)	Executes the self-test and queries the result.	4-87
*WAI(WAIt)	Waits until the execution of the specified overlap command completes before executing the commands that are specified after this command.	4-87

4.2 ABORt Group

The commands in the ABORt group are used to abort the data acquisition operation.

These commands can be used to make the same settings and inquiries as when the ABORT (SHIFT + SINGLE START) key on the front panel is pressed.



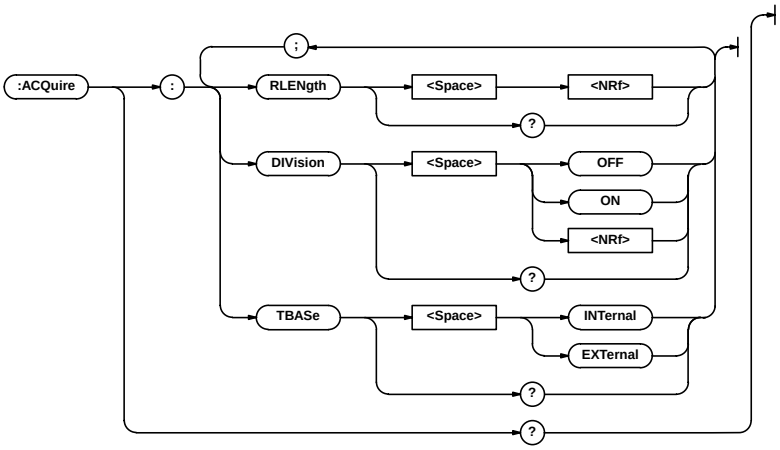
:ABORt

Function Aborts data acquisition.
Syntax `:ABORt`
Example `:ABORT`
Description For the details regarding the difference between the "`:ABORt`" and "`STOP`" commands, see the PZ4000 User's Manual.

4.3 ACQuire Group

The commands in the ACQuire Group deal with data acquisitions.

These commands can be used to make the same settings and inquiries as when the ACQ (SHIFT + TRIGGER) key on the front panel is pressed.



:ACQuire?

Function Queries all settings related to data acquisition.
Syntax `:ACQuire?`
Example `:ACQUIRE?→:ACQUIRE:RLENGTH 100000;DIVISION 0;TBASE INTERNAL`

:ACQuire:DIVision

Function Sets whether or not to divide the record length or queries the current setting.
Syntax `:ACQuire:DIVision {<Boolean>}`
`:ACQuire:DIVision?`
Example `:ACQUIRE:DIVISION OFF`
`:ACQUIRE:DIVISION?→:ACQUIRE:DIVISION 0`

:ACQuire:RLENgth

Function Sets the record length or queries the current setting.
Syntax `:ACQuire:RLENgth {<NRf>}`
`:ACQuire:RLENgth?`
`<NRf> = 100000,1000000,4000000`
Example `:ACQUIRE:RLENGTH 100000`
`:ACQUIRE:RLENGTH?→:ACQUIRE:RLENGTH 100000`
Description The record length that can be specified depends on the extended memory options.

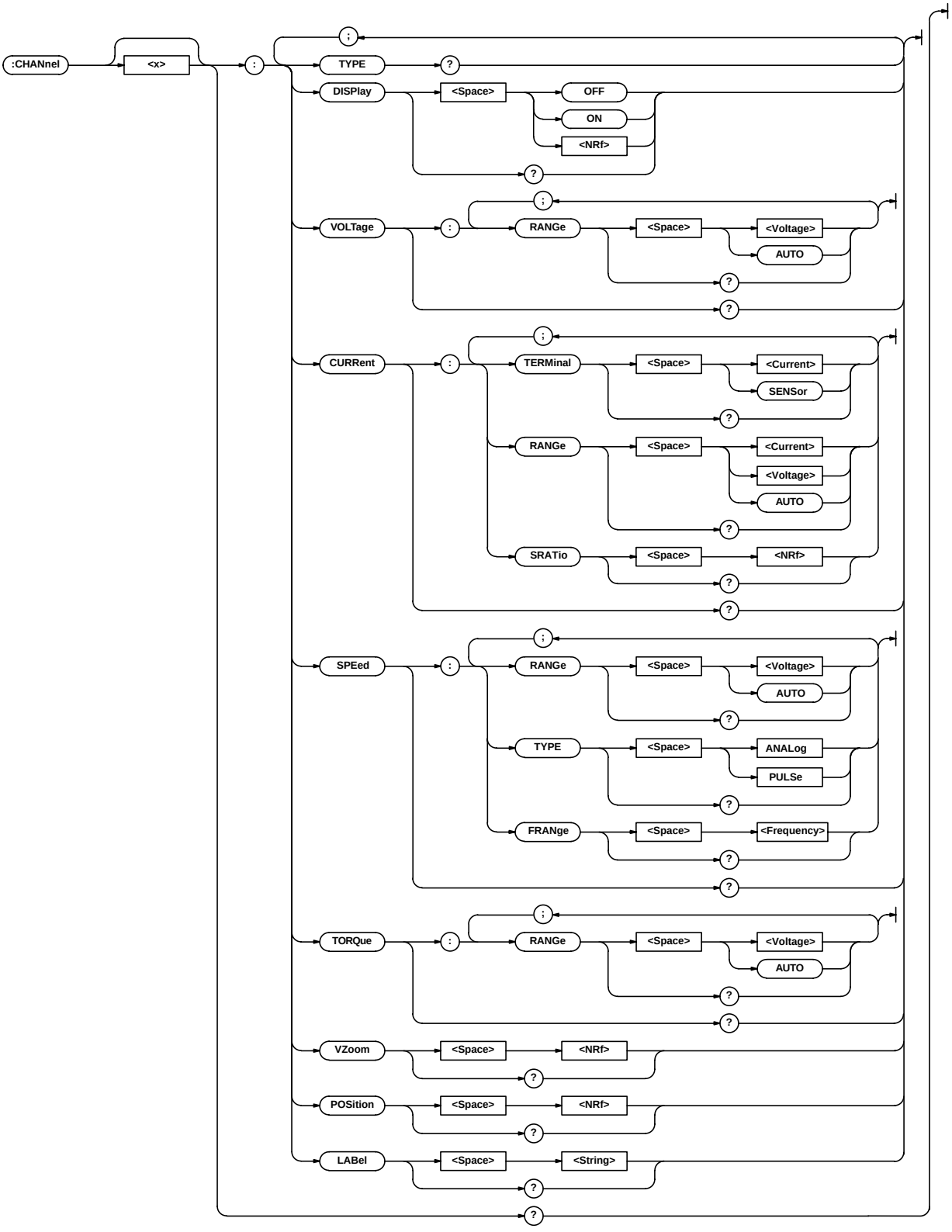
:ACQuire:TBASe

Function Sets the time base or queries the current setting.
Syntax `:ACQuire:TBASe {INTernal|EXTernal}`
`:ACQuire:TBASe?`
`Internal = Internal clock`
`External = External clock`
Example `:ACQUIRE:TBASE INTERNAL`
`:ACQUIRE:TBASE?→:ACQUIRE:TBASE INTERNAL`

4.4 CHANnel Group

The commands in the CHANnel Group deal with the vertical axis of each channel.

These commands can be used to make the same settings and inquiries as when the CH1 to CH8 keys on the front panel are pressed.



:CHANnel<x>?

Function Queries all settings related to the vertical axis of each channel.

Syntax
```
:CHANnel<x>?
<x> = 1 to 8
```

Example
```
:CHANNEL1?→:CHANNEL1:DISPLAY 1;VOLTAGE:
RANGE 2.00E+03;:CHANNEL1:VZOOM
1.00;POSITION 0.000;LABEL "CH1"
```

:CHANnel<x>:CURRent?

Function Queries all settings related to the current input channel.

Syntax
```
:CHANnel<x>:CURRent?
<x> = 1 to 8
```

Example
```
:CHANNEL2:CURRENT?→:CHANNEL2:CURRENT:
TERMINAL 5.0E+00;RANGE 10.0E+00
```

Description If you specify a channel that does not have the 253751/253752 power measurement module installed, an error will occur.

:CHANnel<x>:CURRent:RANGe

Function Sets the current range of the current input channel or queries the current setting.

Syntax
```
:CHANnel<x>:CURRent:RANGe {<current>|
<voltage>|AUTO}
:CHANnel<x>:CURRent:RANGe?
<x> = 1 to 8
<current> = 0.1, 0.2, 0.4, 1, 2, 4, 10(A)
            (when TERMinal = 5(A))
<current> = 1, 2, 4, 10, 20, 40, 100(A)
            (when TERMinal = 20(A))
<voltage> = 0.1, 0.2, 0.4, 1(V)
            (when TERMinal = SENSor)
AUTO      = Auto range
```

Example
```
:CHANNEL2:CURRENT:RANGE 10A
:CHANNEL2:CURRENT:RANGE?→:CHANNEL2:
CURRENT:RANGE 10.0E+00
```

Description
- The selectable range is determined by the setting of the current input terminal (`:CHANnel<x>:CURRent:TERMinal`).
- If you specify a channel that does not have the 253751/253752 power measurement module installed, an error will occur.
- The "`:INPut:POWer:CURRent:SRATio:ELEMent<x>` (where <x> is the element number)" command can be used to make the same settings and inquiries.

:CHANnel<x>:CURRent:SRATio

Function Sets the current sensor's transformation ratio of the current input channel or queries the current setting.

Syntax
```
:CHANnel<x>:CURRent:SRATio {NRf>}
<x> = 1 to 8
<NRf> = 0.0001 to 99999.9999
```

Example
```
:CHANnel<x>:CURRent:SRATio 10
:CHANnel<x>:CURRent:SRATio?
:CHANnel<x>:CURRent:SRATio 10.000
```

Description
- If you specify a channel that does not have the 253751/253752 power measurement module installed, an error will occur.
- The "`:INPut:POWer:CURRent:SRATio:ELEMent<x>` (where <x> is the element number)" command can be used to make the same settings and inquiries.

:CHANnel<x>:CURRent:TERMinal

Function Sets the current input terminal of the current input channel or queries the current setting.

Syntax
```
:CHANnel<x>:CURRent:TERMinal {<current>|
SENSor}
:CHANnel<x>:CURRent:TERMinal?
<x> = 1 to 8
<current> = 5(A) (for the 253751 power
            measurement module)
<current> = 5, 20(A)
            (for the 253752 power
            measurement module)
SENSor    = current sensor
```

Example
```
:CHANNEL2:CURRENT:TERMINAL 5A
:CHANNEL2:CURRENT:TERMINAL?→:
CHANNEL2:CURRENT:TERMINAL 5.0E+00
```

Description
- If you specify a channel that does not have the 253752/253752 power measurement module installed, an error will occur.
- The "`:INPut:POWer:CURRent:TERMinal:ELEMent<x>` (where <x> is the element number)" command can be used to make the same settings and inquiries.

:CHANnel<x>:DISPlay

Function Turns ON/OFF the waveform display of each channel or queries the current setting.

Syntax
```
:CHANnel<x>:DISPlay {<Boolean>}
:CHANnel<x>:DISPlay?
<x> = 1 to 8
```

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

Description The "`:DISPlay:WAVE:CHANnel<x>`" command can be used to make the same settings and inquiries.

:CHANnel<x>:LABel

Function Sets the waveform label of each channel or queries the current setting.

Syntax
```
:CHANnel<x>:LABel {<string>}
:CHANnel<x>:LABel?
<x> = 1 to 8
<string> = 8 characters or less
```

Example
```
:CHANNEL1:LABEL "CH1"
:CHANNEL1:LABEL?→:CHANNEL1:LABEL "CH1"
```

Description Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.
SPEed = Revolution sensor signal input
TORQue = Torque meter signal input

:CHANnel<x>:POSition

Function Sets the vertical position (the GND position) of each channel or queries the current setting.

Syntax
```
:CHANnel<x>:POSition {<NRf>}
:CHANnel<x>:POSition?
<x> = 1 to 8
<NRf> = -130.000 to 130.000(%)
```

Example
```
:CHANNEL1:POSition 0
:CHANNEL1:POSITION?→:CHANNEL1:
POSITION 0.000
```

:CHANnel<x>:SPEed?

Function Queries all settings related to the revolution sensor signal input channel.

Syntax
```
:CHANnel<x>:SPEed?
<x> = 7 (fixed)
```

Example
```
:CHANNEL7:SPEED?→:CHANNEL7:SPEED:
RANGE 50.0E+00;TYPE ANALOG
```

Description If the 253771 motor module is not installed, an error will occur.

:CHANnel<x>:SPEed:FRANge

Function Sets the frequency range of the revolution sensor signal input channel (pulse input) or queries the current setting.

Syntax
```
:CHANnel<x>:SPEed:FRANge {<frequency>|
AUTO}
:CHANnel<x>:SPEed:FRANge?
<x> = 7 (fixed)
<frequency> = 40(Hz): 1 to 40 Hz
            = 800(Hz): 16 to 800 Hz
            = 8k(Hz): 250 to 8 kHz
            = 200k(Hz): 2 k to 200 kHz
AUTO        = Auto range
```

Example
```
:CHANNEL7:SPEED:FRANGE 200KHZ
:CHANNEL7:SPEED:FRANGE?→:CHANNEL7:
SPEED:FRANGE 200.00E+03
```

Description
- Set the <frequency> to the maximum value within the frequency range.
- This command is valid when the input format of the revolution sensor signal (`:CHANnel<x>:SPEed:TYPE`) is set to "`PULSe (pulse input)`."
- If the 253771 motor module is not installed, an error will occur.
- The "`:INPut:MOTor:SPEed:FRANGe`" command can be used to make the same settings and inquiries.

:CHANnel<x>:SPEed:RANGe

Function Sets the input range of the revolution sensor signal input channel or queries the current setting.

Syntax
```
:CHANnel<x>:SPEed:RANGe {<voltage>|AUTO}
:CHANnel<x>:SPEed:RANGe?
<x> = 7 (fixed)
<voltage> = 1, 2, 5, 10, 20, and 50(V)
AUTO      = Auto range
```

Example
```
:CHANNEL7:SPEED:RANGE 50V
:CHANNEL7:SPEED:RANGE?→:CHANNEL7:SPEED:
RANGE 50.0E+00
```

Description
- When the input format of the revolution sensor signal (`:CHANnel<x>:SPEed:TYPE`) is set to "`PULSe (pulse input)`," it is fixed to 5 (V).
- If the 253771 motor module is not installed, an error will occur.
- The "`:INPut:MOTor:SPEed:RANGe`" command can be used to make the same settings and inquiries.

:CHANnel<x>:SPEed:TYPE

Function Sets the signal type of the revolution sensor signal input channel or queries the current setting.

Syntax
```
:CHANnel<x>:SPEed:TYPE {ANALog|PULSe}
:CHANnel<x>:SPEed:TYPE?
<x> = 7 (fixed)
```

Example
```
:CHANNEL7:SPEED:TYPE ANALOG
:CHANNEL7:SPEED:TYPE?→:CHANNEL7:SPEED:
TYPE ANALOG
```

Description
- If the 253771 motor module is not installed, an error will occur.
- The ":INPut:MOTor:SPEed:TYPE" command can be used to make the same settings and inquiries.

:CHANnel<x>:TORQue?

Function Queries all settings related to the torque meter signal input channel.

Syntax
```
:CHANnel<x>:TORQue?
<x> = 8 (fixed)
```

Example
```
:CHANNEL8:TORQUE?→:CHANNEL8:TORQUE:
RANGE 50.0E+00
```

Description If the 253771 motor module is not installed, an error will occur.

:CHANnel<x>:TORQue:RANGe

Function Sets the input range of the torque meter signal input channel or queries the current setting.

Syntax
```
:CHANnel<x>:TORQue:RANGe {<voltage>|
AUTO}
:CHANnel<x>:TORQue:RANGe?
<x> = 8 (fixed)
<voltage> = 1, 2, 5, 10, 20, and 50(V)
AUTO      = Auto range
```

Example
```
:CHANNEL8:TORQUE:RANGE 50V
:CHANNEL8:TORQUE:RANGE?→:CHANNEL8:
TORQUE:RANGE 50.0E+00
```

Description
- If the 253771 motor module is not installed, an error will occur.
- The ":INPut:MOTor:TORQue:RANGe" command can be used to make the same settings and inquiries.

:CHANnel<x>:TYPE?

Function Queries the input type of each channel.

Syntax
```
:CHANnel<x>:TYPE?
<x> = 1 to 8
```

Example
```
:CHANNEL1:TYPE?→VOLTAGE
```

Description The following responses are possible.
VOLTage = voltage input
CURRent = current input

:CHANnel<x>:VOLTage?

Function Queries all settings related to the voltage input channel.

Syntax
```
:CHANnel<x>:VOLTage?
<x> = 1 to 8
```

Example
```
:CHANNEL1:VOLTAGE?→:CHANNEL1:VOLTAGE:
RANGE 2.00E+03
```

:CHANnel<x>:VOLTage:RANGe

Function Sets the voltage range of the voltage input channel or queries the current setting.

Syntax
```
:CHANnel<x>:VOLTage:RANGe {<voltage>|
AUTO}
:CHANnel<x>:VOLTage:RANGe?
<x> = 1 to 8
<voltage> = 30,60,120,200,300,600,1200,
            2000(V)
AUTO      = AUTO RANGE
```

Example
```
:CHANNEL1:VOLTAGE:RANGE 2000V
:CHANNEL1:VOLTAGE:RANGE?→:
CHANNEL1:VOLTAGE:RANGE 2.00E+03
```

Description The ":INPut:POWer:VOLTage:RANGe:ELEMent<x> (where <x> is the element number)" command can be used to make the same settings and inquiries.

:CHANnel<x>:VZoom

Function Sets the vertical zoom factor or queries the current setting.

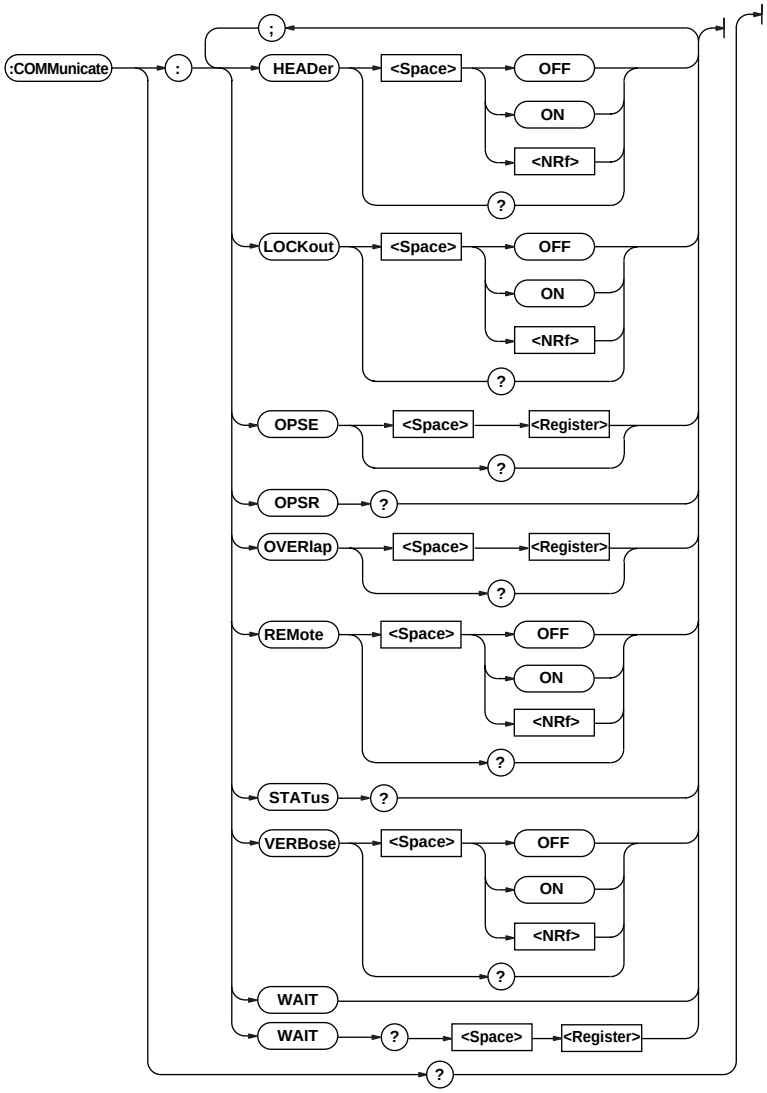
Syntax
```
:CHANnel<x>:VZoom {<NRf>}
:CHANnel<x>:VZoom?
<x> = 1 to 8
<NRf> = 0.1 to 100 (See the PZ4000
        User's Manual)
```

Example
```
:CHANNEL1:VZOOM 1
:CHANNEL1:VZOOM?→:CHANNEL1:VZOOM 1.00
```

4.5 COMMunicate Group

The commands in the COMMunicate Group deal with communications.

There are no front-panel keys that correspond to the commands in this group.



:COMMunicate?

Function Queries all settings related to communications.

Syntax `:COMMunicate?`

Example `:COMMUNICATE?→:COMMUNICATE:HEADER 1;OPSE 96;OVERLAP 96;VERBOSE 1`

:COMMunicate:HEADer

Function Sets whether or not to attach headers to response data or queries the current setting. (Example of a response with a header: `SETUP:MODE NORMAL`, example of a response without a header: `NORMAL`)

Syntax `:COMMunicate:HEADer {<Boolean>}`
`:COMMunicate:HEADer?`

Example `:COMMUNICATE:HEADER ON`
`:COMMUNICATE:HEADER?→:COMMUNICATE:HEADER 1`

:COMMunicate:LOCKout

Function Sets/releases local lockout.

Syntax `:COMMunicate:LOCKout {<Boolean>}`
`:COMMunicate:LOCKout?`

Example `:COMMUNICATE:LOCKOUT ON`
`:COMMUNICATE:LOCKOUT?→:COMMUNICATE:LOCKOUT 1`

Description This is a dedicated command for the serial interface. An interface message is available for the GP-IB interface.

:COMMunicate:OPSE

(Operation Pending Status Enable register)

Function Sets the overlap commands for *OPC, *OPC?, and *WAI or queries the current setting.

Syntax
```
:COMMunicate:OPSE <Register>
:COMMunicate:OPSE?
<Register> = 0 to 65535, See the diagram
             for the :COMMunicate:WAIT?
             command.
```

Example
```
:COMMUNICATE:OPSE 65535
:COMMUNICATE:OPSE?→:COMMUNICATE:OPSE 96
```

Description All bits are set to 1 in the above example to set all commands to overlap. However, bits that are fixed to 0 do not change, and therefore, only bits 5 and 6 are set to 1.

:COMMunicate:OPSR?

(Operation Pending Status Register)

Function Queries the operation pending status register.

Syntax
```
:COMMunicate:OPSR?
```

Example
```
:COMMUNICATE:OPSR?→0
```

Description For the operation pending registers, see the diagram for the :COMMunicate:WAIT? command.

:COMMunicate:OVERlap

Function Sets the commands to permit overlap operation or queries the current setting.

Syntax
```
:COMMunicate:OVERlap <Register>
:COMMunicate:OVERlap?
<Register> = 0 to 65535, See the diagram
             for the :COMMunicate:WAIT?
             command.
```

Example
```
:COMMUNICATE:OVERLAP 65535
:COMMUNICATE:OVERLAP?→:COMMUNICATE:
OVERLAP 96
```

Description
- All bits are set to 1 in the above example to set all commands to overlap. However, bits that are fixed to 0 do not change, and therefore, only bits 5 and 6 are set to 1.
- For the description regarding how to synchronize the program using the COMMunicate:OVERlap command, see page 3-7.
- Bits 5 and 6 are set to 1 in the above example to set all overlap commands (See the diagram for the :COMMunicate:WAIT? command.)

:COMMunicate:REMote

Function Switches between remote and local. ON is remote.

Syntax
```
:COMMunicate:REMote {<Boolean>}
:COMMunicate:REMote?
```

Example
```
:COMMUNICATE:REMOTE ON
:COMMUNICATE:REMOTE?→:COMMUNICATE:
REMOTE 1
```

Description This is a dedicated command for the serial interface. An interface message is available for the GP-IB interface.

:COMMunicate:STATus?

Function Queries the line-specific status.

Syntax
```
:COMMunicate:STATus?
```

Example
```
:COMMUNICATE:STATUS?→:COMMUNICATE:
STATUS 0
```

Description The meaning of each status bit is as follows:

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

The status bit is set when the causing event occurs and cleared when it is read.

:COMMunicate:VERBose

Function Sets whether to use the full (example: SETUP:MODE NORMAL) or abbreviated (example: SET NORM) form for response data or queries the current setting.

Syntax
```
:COMMunicate:VERBose {<Boolean>}
:COMMunicate:VERBose?
```

Example
```
:COMMUNICATE:VERBOSE ON
:COMMUNICATE:VERBOSE?→:COMMUNICATE:
VERBOSE 1
```

:COMMunicate:WAIT

Function Waits for one of the specified extended events to occur.

Syntax
```
:COMMunicate:WAIT <Register>
<Register> = 0 to 65535 (extended event
             register, see page 5-4.)
```

Example
```
:COMMUNICATE:WAIT 1
```

Description For the description regarding how to synchronize the program using COMMunicate:WAIT, see page 3-9.

: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, see 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	0	0	0	ACS	PRN	0	0	0	0	0

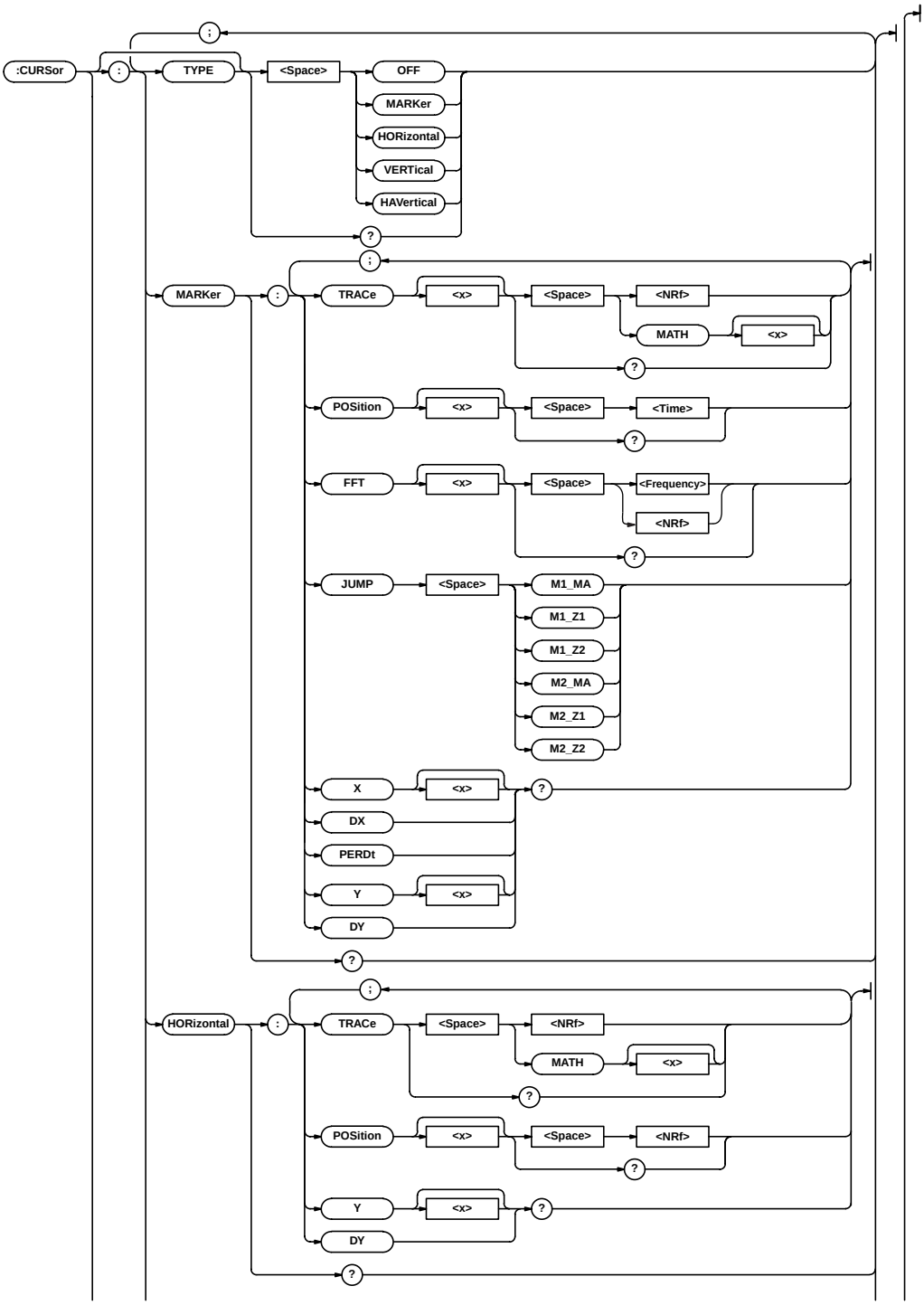
When bit 5 (PRN) = 1 :
Printer operation is not complete.

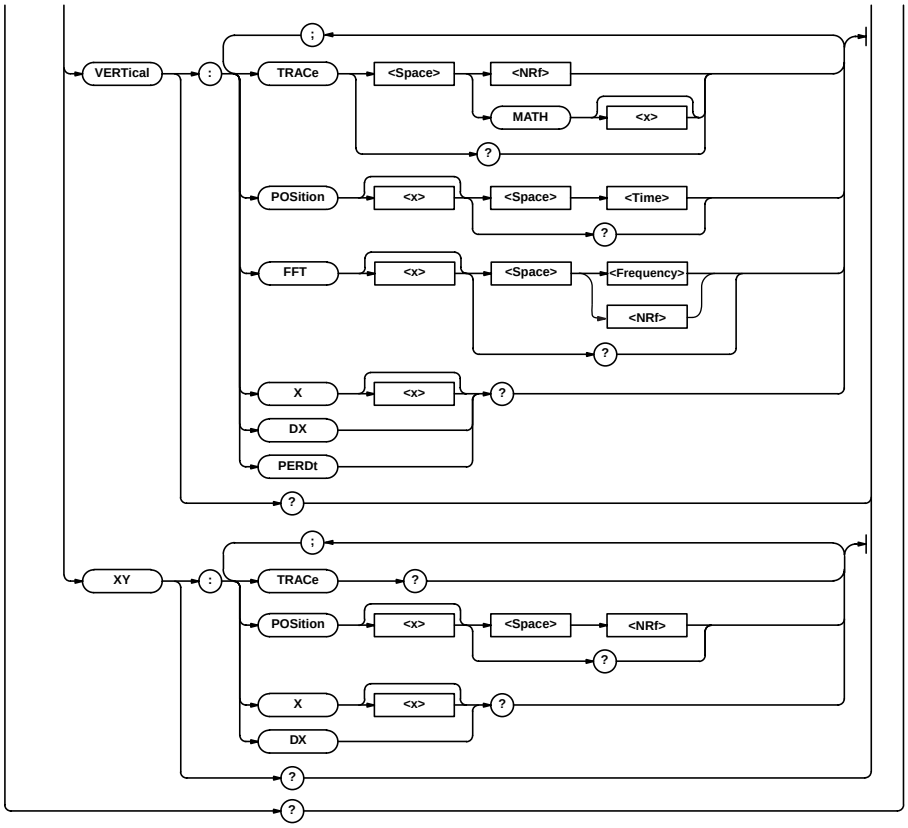
When bit 6 (ACS) = 1 :
Medium access is not complete.

4.6 CURSor Group

The commands in the CURSor Group deal with cursor measurements.

These commands can be used to make the same settings and inquiries as when the CURSOR key on the front panel is pressed.





:CURSor?

Function Queries all settings related to cursor measurements.

Syntax `:CURSor?`

Example `:CURSOR?→:CURSOR:TYPE HORIZONTAL;HORIZONTAL:TRACE 1; POSITION1 25.0;POSITION2 -25.0`

:CURSor:HORizontal?

Function Queries all settings related to the H cursor.

Syntax `:CURSor:HORizontal?`

Example `:CURSOR:HORIZONTAL?→:CURSOR:HORIZONTAL: TRACE 1;POSITION1 25.0;POSITION2 -25.0`

:CURSor:HORizontal:DY?

Function Queries the Y-axis value (physical value) between the H cursors.

Syntax `:CURSor:HORizontal:DY?`

Example `:CURSOR:HORIZONTAL:DY?→100.00E+00`

Description
- "NAN (Not A Number)" will be returned, if the ":CURSor:TYPE" is not set to Horizontal or HAVertical.
- "NAN (Not A Number)" will also be returned, if the ":DISPlay:FORMat" setting does not include the waveform display.

:CURSor:HORizontal:POSition<x>

Function Sets the H cursor position or queries the current setting.

Syntax `:CURSor:HORizontal:POSition<x> {<NRf>}`
`:CURSor:HORizontal:POSition<x>?`
`<NRf> = -100.0 to 100.0 (%)`
`(The resolution is 0.1%)`

Example `:CURSOR:HORIZONTAL:POSITION1 25`
`:CURSOR:HORIZONTAL:POSITION1?→:CURSOR:HORIZONTAL:POSITION1 25.0`

Description Set the position in terms of a percentage of the full scale value displayed on the screen.

:CURSor:HORizontal:TRACe

Function Sets the waveform on which to place the H cursor or queries the current setting.

Syntax `:CURSor:HORizontal:TRACe {<NRf>|MATH<x>}`
`:CURSor:HORizontal:TRACe?`
`<NRf> = 1 to 8 (channel)`
`<x> = 1, 2 (MATH)`

Example `:CURSOR:HORIZONTAL:TRACE 1`
`:CURSOR:HORIZONTAL:TRACE?→:CURSOR:HORIZONTAL:TRACE 1`

:CURSor:HORizontal:Y<x>?

Function H Queries the Y-axis value (physical value) of the H cursor.

Syntax `:CURSor:HORizontal:Y<x>?`

Example `:CURSOR:HORIZONTAL:Y1?→50.000E+00`

Description
- "NAN (Not A Number)" will be returned, if the ":CURSor:TYPE" is not set to Horizontal or HAVertical.
- "NAN (Not A Number)" will also be returned, if the ":DISPlay:FORMat" setting does not include the waveform display.

:CURSor:MARKer?

Function Queries all settings related to the marker.

Syntax `:CURSor:MARKer?`

Example `:CURSOR:MARKER?→:CURSOR:MARKER:TRACE1 1;TRACE2 1;POSITION1 20.000E-03;POSITION2 80.000E-03`

:CURSor:MARKer:DX?

Function Queries the X-axis value (physical value) between the markers.

Syntax `:CURSor:MARKer:DX?`

Example `:CURSOR:MARKER:DX?→60.000E-03`

:CURSor:MARKer:DY?

Function Queries the Y-axis value (physical value) between the markers.

Syntax `:CURSor:MARKer:DY?`

Example `:CURSOR:MARKER:DY?→157.26E+00`

Description
- "NAN (Not A Number)" will be returned, if the ":CURSor:TYPE" is not set to MARKer.
- "NAN (Not A Number)" will also be returned, if the ":DISPlay:FORMat" setting does not include the waveform display.

:CURSor:MARKer:FFT<x>

Function Sets the X-axis value of the marker position for the FFT result or queries the current setting.

Syntax `:CURSor:MARKer:FFT<x> {<frequency>|<NRf>}`
`:CURSor:MARKer:FFT<x>?`
`<x> = 1 to 2`
`<frequency> = 0 to 2.5 MHz (normal measurement, when Time Base = Internal)`
`<NRf> = 0 to 5000 (when Time Base = External or during harmonic measurement)`

Example `:CURSOR:MARKER:FFT1 200kHz`
`:CURSOR:MARKER:FFT1?→:CURSOR:MARKER:FFT1 200.0E+03`

Description
- This command is valid when ":CURSor:MARKer:TRACe<x>" is set to MATH<x> and the equation of MATH<x> is set to FFT.
- The range and resolution of <frequency> is determined from the sampling rate and the number of FFT points.
- <NRf> is set in terms of harmonic order. The range depends on the number of FFT points as follows. For the procedure to set the number of FFT points, see the ":MATH<x>:FFT:POINt" command.
  For 1000 points : 0 to 500
  For 2000 points : 0 to 1000
  For 10000 points : 0 to 5000

:CURSor:MARKer:JUMP

Function Jumps to a waveform of the marker.

Syntax `:CURSor:MARKer:JUMP {M1_MA|M1_Z1|M1_Z2|M2_MA|M2_Z1|M2_Z2}`

Example `:CURSOR:MARKER:JUMP M1_Z1`

Description The parameters "M1" and "M2" represent markers 1 and 2, respectively. "MA," "Z1," and "Z2" represent the main waveform, zoomed waveforms 1 and 2, respectively.

:CURSor:MARKer:PERDt?

Function Queries the 1/Δ value of the horizontal axis between the markers.

Syntax `:CURSor:MARKer:PERDt?`

Example `:CURSOR:MARKER:PERDT?→16.667E+00`

:CURSor:MARKer:POSition<x>

Function Sets the X-axis value (physical value) of the marker position or queries the current setting.

Syntax
```
:CURSor:MARKer:POSition<x> {<time>|<NRf>}
:CURSor:MARKer:POSition<x>?
<time> = 0 to (OBSERVATION TIME) (during the normal measurement mode, when Time Base = Internal)
<NRf> = 0 to Record length (when Time Base = Internal, or during the harmonic measurement mode)
```

Example
```
:CURSOR:MARKER:POSITION1 20MS
:CURSOR:MARKER:POSITION1?→:CURSOR:MARKER:POSITION1 20.000E-03
```

Description
- The range and resolution of <time> depends on the observation time.
- Specify <NRf> in terms of sampled data points. The range is from 0 to the record length.

:CURSor:MARKer:TRACe<x>

Function Sets the waveform on which to place the marker or queries the current setting.

Syntax
```
:CURSor:MARKer:TRACe<x> {<NRf>|MATH<x>}
:CURSor:MARKer:TRACe<x>?
TRACe<x>'s<x> = 1, 2
<NRf> = 1 to 8 (channel)
<x> = 1, 2 (MATH)
```

Example
```
:CURSOR:MARKER:TRACE1 1
:CURSOR:MARKER:TRACE1?→:CURSOR:MARKER:TRACE1 1
```

:CURSor:MARKer:X<x>?

Function Queries the X-axis value (physical value) of the marker position.

Syntax `:CURSor:MARKer:X<x>?`

Example `:CURSOR:MARKER:X1?→20.000E-03`

Description
- The ":CURSor:MARKer:POSition<x>?" command can be used to make the same inquiry.
- "NAN (Not A Number)" will also be returned, if the ":DISPlay:FORMat" setting does not include the waveform display.

:CURSor:MARKer:Y<x>?

Function Queries the Y-axis value (physical value) of the marker position.

Syntax `:CURSor:MARKer:Y<x>?`

Example `:CURSOR:MARKER:Y1?→78.628E+00`

Description "NAN (Not A Number)" will be returned, if the ":CURSor:TYPE" is not set to MARKer.

:CURSor[:TYPE]

Function Sets the marker/cursor type or queries the current setting.

Syntax
```
:CURSor[:TYPE] {OFF|MARKer|HORizontal|VERTical|HAVertical}
:CURSor:TYPE?
```

Example
```
:CURSOR:TYPE HORIZONTAL
:CURSOR:TYPE?→:CURSOR:TYPE HORIZONTAL
```

:CURSor:VERTical?

Function Queries all settings related to the V cursor.

Syntax `:CURSor:VERTical?`

Example
```
:CURSOR:VERTICAL?→:CURSOR:VERTICAL:TRACE 1;POSITION1 20.000E-03;
POSITION2 80.000E-03
```

:CURSor:VERTical:DX?

Function Queries the X-axis value (physical value) between the V cursors.

Syntax `:CURSor:VERTical:DX?`

Example `:CURSOR:VERTICAL:DX?→60.000E-03`

:CURSor:VERTical:FFT<x>

Function Sets the V cursor position with respect to the FFT result or queries the current setting.

Syntax
```
:CURSor:VERTical:FFT<x> {<frequency>|<NRf>}
:CURSor:VERTical:FFT<x>?
<x> = 1 to 2
<frequency> = 0 to 2.5MHz (during the normal measurement mode, when Time Base = Internal)
<NRf> = 0 to 5000 (when Time Base = External or during the harmonic measurement mode)
```

Example
```
:CURSOR:VERTICAL:FFT1 200kHz
:CURSOR:VERTICAL:FFT1?→:CURSOR:MARKER:FFT1 200.0E+03
```

Description
- This command is valid when ":CURSor:VERTical:TRACe<x>" is set to MATH<x> and the equation of MATH<x> is set to FFT.
- The range and resolution of <frequency> is determined from the sampling rate and the number of FFT points.
- <NRf> is set in terms of harmonic order. The range depends on the number of FFT points as follows. For the procedure to set the number of FFT points, see the ":MATH<x>:FFT:POINt" command.
  For 1000 points : 0 to 500
  For 2000 points : 0 to 1000
  For 10000 points : 0 to 5000

:CURSor:VERTical:PERDt?

Function Queries the 1/Δ value (physical value) of the horizontal axis between the V cursors.

Syntax `:CURSor:VERTical:PERDt?`

Example `:CURSOR:VERTICAL:PERDT?→16.667E+00`

:CURSor:VERTical:POSition<x>

Function Sets the V cursor position or queries the current setting.

Syntax
```
:CURSor:VERTical:POSition<x> {<time>|
<NRf>}
:CURSor:VERTical:POSition<x>?
<time> = 0 to (OBSERVATION TIME) (during
         the normal measurement mode,
         when Time Base = Internal)
<NRf> = 0 to Record length (when Time
        Base = Internal, or during the
        harmonic measurement mode)
```

Example
```
:CURSOR:VERTICAL:POSITION1 20MS
:CURSOR:VERTICAL:POSITION1?→:CURSOR:
VERTICAL:POSITION1 20.000E-03
```

Description
- The range and resolution of <time> depends on the observation time.
- Specify <NRf> in terms of sampled data points. The range is from 0 to the record length.

:CURSor:VERTical:TRACe

Function Sets the waveform on which to place the V cursor or queries the current setting.

Syntax
```
:CURSor:VERTical:TRACe {<NRf>|MATH<x>}
:CURSor:VERTical:TRACe?
<NRf> = 1 to 8(channel)
<x> = 1, 2(MATH)
```

Example
```
:CURSOR:VERTICAL:TRACE 1
:CURSOR:VERTICAL:TRACE?→:CURSOR:
VERTICAL:TRACE 1
```

:CURSor:VERTical:X<x>?

Function Queries the X-axis value (physical value) of the V cursor position.

Syntax `:CURSor:VERTical:X<x>?`

Example `:CURSOR:VERTICAL:X1?→20.000E-03`

Description The "`:CURSor:VERTical:POSition<x>?`" command can be used to make the same inquiry.

:CURSor:XY?

Function Queries all settings related to XY cursor.

Syntax `:CURSor:XY?`

Example
```
:CURSOR:XY?→:CURSOR:XY:
POSITION1 -25.0;POSITION2 25.0
```

:CURSor:XY:DX?

Function Queries the X-axis value (physical value) between the XY cursors.

Syntax `:CURSor:XY:DX?`

Example `:CURSOR:XY:DX?→150.00E+00`

:CURSor:XY:POSition<x>

Function Sets the XY cursor position or queries the current setting.

Syntax
```
:CURSor:XY:POSition<x> {<NRf>}
:CURSor:XY:POSition<x>?
<NRf> = -100.0 to 100.0(%) (The
        resolution is 0.1(%))
```

Example
```
:CURSOR:XY:POSITION1 -25
:CURSOR:XY:POSITION1?→:CURSOR:XY:
POSITION1 -25.0
```

Description Set the value in terms of a percentage of the full scale value displayed on the screen.

:CURSor:XY:TRACe?

Function Queries the waveform on which the XY cursor is placed.

Syntax `:CURSor:XY:TRACe?`

Example `:CURSOR:XY:TRACE?→:CURSOR:XY:TRACE 1`

Description The "`:DISPlay:XY:XTRace?`" command can be used to make the same inquiry.

:CURSor:XY:X<x>?

Function Queries the X-axis value (physical value) of the XY cursor position.

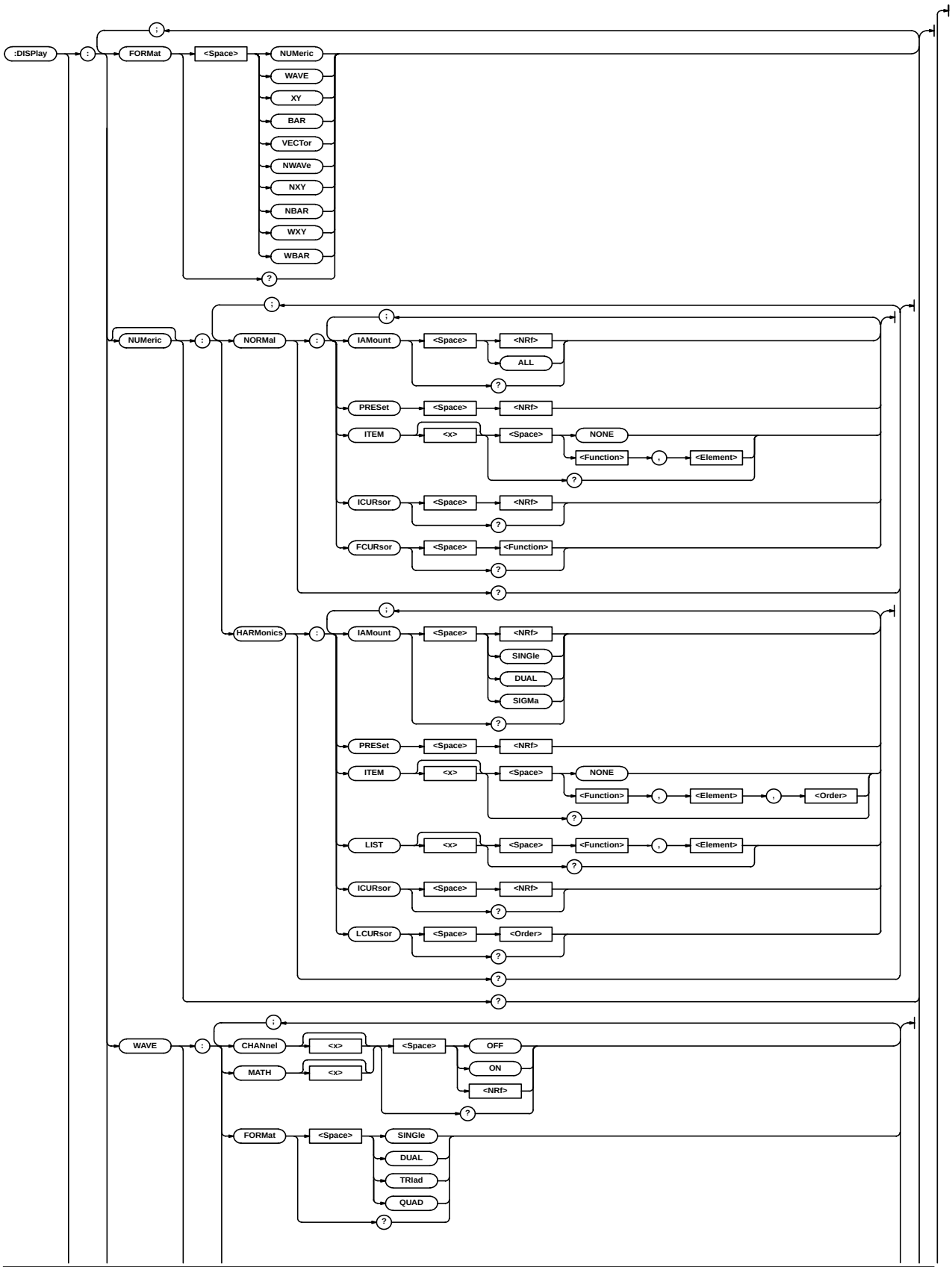
Syntax `:CURSor:XY:X<x>?`

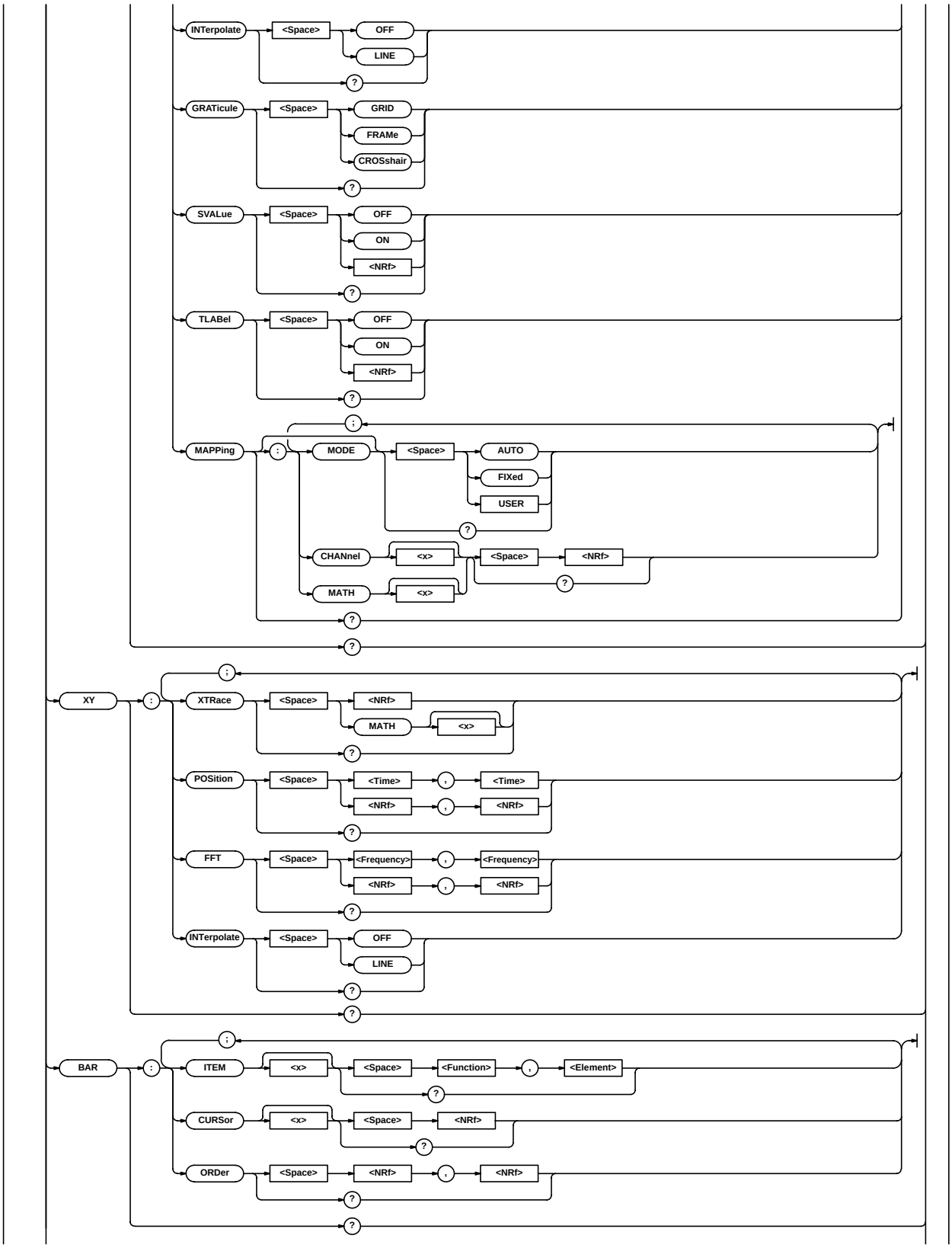
Example `:CURSOR:XY:X1?→-75.000E+00`

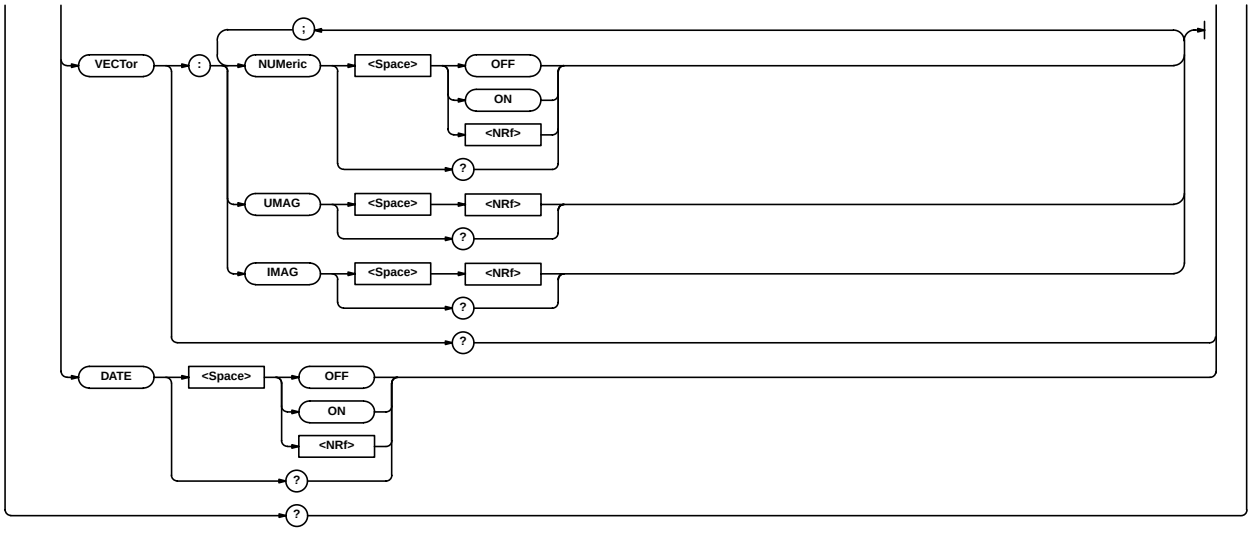
4.7 DISPlay Group

The commands in the DISPlay Group deal with the screen display

These commands can be used to make the same settings and inquiries as when the DISPLAY key on the front panel is pressed.




INTerpolate
<Space>
OFF
LINE
?
GRATicule
<Space>
GRID
FRAMe
CROSshair
?
SVALue
<Space>
OFF
ON
<NRf>
?
TLABel
<Space>
OFF
ON
<NRf>
?
MAPPing
:
;
MODE
<Space>
AUTO
FIXed
USER
?
CHANnel
<x>
<Space>
<NRf>
?
MATH
<x>
?
?
XY
:
;
XTRace
<Space>
<NRf>
MATH
<x>
?
POSition
<Space>
<Time>
,
<Time>
<NRf>
,
<NRf>
?
FFT
<Space>
<Frequency>
,
<Frequency>
<NRf>
,
<NRf>
?
INTerpolate
<Space>
OFF
LINE
?
?
BAR
:
;
ITEM
<x>
<Space>
<Function>
,
<Element>
?
CURSor
<x>
<Space>
<NRf>
?
ORDer
<Space>
<NRf>
,
<NRf>
?
?



:DISPlay?

Function Queries all settings related to the screen display.

Syntax `:DISPlay?`

Example Example when the display format (:DISPlay:FORMat) is set to "NWAVe"
`:DISPLAY?→:DISPLAY:FORMAT BOTH;(the response to ":DISPlay:NUMeric?" without the ":DISPLAY:" section);(the same response to ":DISPlay:WAVE?");:DISPLAY:DATE 1`

:DISPlay:BAR?

Function Queries all settings related to the bar graph display.

Syntax `:DISPlay:BAR?`

Example
```
:DISPLAY:BAR?→:DISPLAY:BAR:ITEM1 U,1;ITEM2 I,1;CURSOR1 1;CURSOR2 13;ORDER 1,100
```

:DISPlay:BAR:CURSor<x>

Function Sets the marker position (harmonic order) on the bar graph display or queries the current setting.

Syntax
```
:DISPlay:BAR:CURSor<x> {<NRf>}
:DISPlay:BAR:CURSor<x>?
<x> = 1, 2
<NRf> = 0 to 500 (To the end harmonic order of the bar graph display)
```

Example
```
:DISPLAY:BAR:CURSor 1
:DISPLAY:BAR:CURSor?→:DISPLAY:BAR:CURSor1 1
```

:DISPlay:BAR:ITEM<x>

Function Sets the bar graph display items (function, element) or queries the current setting.

Syntax
```
:DISPlay:BAR:ITEM<x> {<Function>,<Element>}
:DISPlay:BAR:ITEM<x>?
<x> = 1 to 2 (item number)
<Function> = {U|I|P|S|Q|LAMBda|...}
```
(See the function selection list on page 4-32 (3).)
```
<Element> = 1 to 4
```

Example
```
:DISPLAY:BAR:ITEM1 U,1
:DISPLAY:BAR:ITEM1?→:DISPLAY:BAR:ITEM1 U,1
```

:DISPlay:BAR:ORDer

Function Sets the start and end harmonic orders of the bar graph display or queries the current setting.

Syntax
```
:DISPlay:BAR:ORDer {<NRf>,<NRf>}
:DISPlay:BAR:ORDer?
First <NRf> = 0 to 490 (start harmonic order of the bar graph display)
Second <NRf> = 10 to 500 (end harmonic order of the bar graph diplay)
```

Example
```
:DISPLAY:BAR:ORDER 1,100
:DISPLAY:BAR:ORDER?→:DISPLAY:BAR:ORDER 1,100
```

Description
- Set the start harmonic order, then the end harmonic order.
- Set the end harmonic order so that it is greater than or equal to (start harmonic order + 10).

:DISPlay:DATE

Function Turns ON/OFF the date and time displays or queries the current setting.

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

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

:DISPlay:FORMat

Function Sets the display format or queries the current setting.

Syntax
```
:DISPlay:FORMat {NUMeric|WAVE|XY|BAR|
VECTor|NWAVe|NXY|NBAR|WXY|WBAR}
:DISPlay:FORMat?
```
```
NUMeric = Displays only the numerical
          values.
WAVE    = Displays only the waveforms.
XY      = Displays the X-Y display.
BAR     = Displays the bar graph.
VECTor  = Displays the vector graph.
NWAVe   = Displays both the numerical
          values and the waveforms.
NXY     = Displays both the numerical
          values and the X-Y display.
NBAR    = Displays both the numerical
          values and the bar graph.
WXY     = Displays both the waveforms and
          the X-Y display.
WBAR    = Displays both the waveforms and
          the bar graph.
```

Example
```
:DISPLAY:FORMAT NUMERIC
:DISPLAY:FORMAT?→:DISPLAY:
FORMAT NUMERIC
```

:DISPlay:NUMeric?

Function Queries all settings related to the numerical display.

Syntax
```
:DISPlay:NUMeric?
```

Example
- For normal measurement mode (when :SETup[:MODE] is set to "NORMal")
  DISPLAY:NUMERIC?→Same as the response for the ":DISPlay[NUMeric]:NORMal?" command.
- For harnomic analysis mode (when :SETup[:MODE] is set to "HARMonics")
  DISPLAY:NUMERIC?→Same as the response for the ":DISPlay[NUMeric]:HARMonics?" command.

:DISPlay[:NUMeric]:HARMonics?

Function Queries all settings related to the numerical display during harmonic measurement.

Syntax
```
:DISPlay[:NUMeric]:HARMonics?
```

Example
- Example when the numerical display format (:DISPlay[:NUMeric]:HARMonics:IAMount) is set to {8|16}
  ```
  :DISPLAY:NUMERIC:HARMONICS?→:
  DISPLAY:NUMERIC:HARMONICS:IAMOUNT 8;
  ITEM1 U,1,1;ITEM2 I,1,1;ITEM3 P,1,1;
  ...(abbreviated)...;ITEM255 NONE;
  ICURSOR 1
  ```
- Example when the numerical display format (:DISPlay[:NUMeric]:HARMonics:IAMount) is set to {SINGle|DUAL} list display
  ```
  :DISPLAY:NUMERIC:HARMONICS?→
  :DISPLAY:NUMERIC:HARMONICS:
  IAMOUNT SINGLE;LIST1 U,1;LIST2 I,1;
  LCURSOR 1
  ```

:DISPlay[:NUMeric]:HARMonics:IAMount

Function Queries all settings related to the numerical display during harmonic measurement.

Syntax
```
:DISPlay[:NUMeric]:HARMonics:
IAMount {<NRf>|SINGle|DUAL|SIGMa}
:DISPlay[:NUMeric]:HARMonics:IAMount?
<NRf> = 8, 16
```

Example
```
:DISPLAY:NUMERIC:HARMONICS:IAMOUNT 8:
DISPLAY:NUMERIC:HARMONICS:IAMOUNT?→:
DISPLAY:NUMERIC:HARMONICS:IAMOUNT 8
```

Description The harmonic measurement data information that is displayed depends on the selected numerical display format as follows.

```
<NRf>   = Displays the numerical display items
          in the order of item numbers. (<NRf>
          denotes the number of displayed
          items on one screen.)
SINGle  = Displays a list of display items in
          EVEN and ODD columns.
DUAL    = Displays two lists of display items in
          the order of harmonic order.
SIGMa   = Displays the numeric data of the main
          functions (U, I, P, S, Q, and λ...) and
          the phase difference (ϕ) between U
          and I for each element.
```

:DISPlay[:NUMeric]:HARMonics:ICURsor

Function Sets the cursor position of the numerical display during harmonic measurement or queries the current setting.

Syntax
```
:DISPlay[:NUMeric]:HARMonics:
ICURsor {<NRf>}
:DISPlay[:NUMeric]:HARMonics:ICURsor?
<NRf> = 1 to 300
```

Example
```
:DISPLAY:NUMERIC:HARMONICS:ICURSOR 1
:DISPLAY:NUMERIC:HARMONICS:ICURSOR?→:
DISPLAY:NUMERIC:HARMONICS:ICURSOR 1
```

Description
- The cursor position is specified using the item number.
- This command is valid when the numerical display format (`:DISPlay[:NUMeric]:HARMonics:IAMount`) is set to {8|16}.

:DISPlay[:NUMeric]:HARMonics:ITEM<x>

Function Sets the numerical displayed items during harmonic measurement or queries the current setting.

Syntax
```
:DISPlay[:NUMeric]:HARMonics:
ITEM<x> {NONE|<Function>,<Element>,
<Order>}
:DISPlay[:NUMeric]:HARMonics:ITEM<x>?
<x> = 1 to 255 (item number)
NONE = no display item
<Function> = {U|I|P|S|Q|...} (See the
             function selection list on
             page 4-32 (2).)
<Element> = {<NRf>|SIGMA|SIGMB}(<NRf> =
            1 to 4)
<Order> = {TOTal|DC|<NRf>}(<NRf> = 1 to
          500)
```

Example
```
:DISPLAY:NUMERIC:HARMONICS:ITEM1 U,1,1
:DISPLAY:NUMERIC:HARMONICS:ITEM1?→:
DISPLAY:NUMERIC:HARMONICS:ITEM1 U,1,1
```

Description This command is valid when the numerical display format (`:DISPlay[:NUMeric]:HARMonics:IAMount`) is set to {8|16}.

:DISPlay[:NUMeric]:HARMonics:LCURsor

Function Sets the cursor position on the list display during harmonic measurement or queries the current setting.

Syntax
```
:DISPlay[:NUMeric]:HARMonics:
LCURsor {<Order>}
:DISPlay[:NUMeric]:HARMonics:LCURsor?
<Order> = {TOTal|DC|<NRf>}(<NRf> = 1 to
          500)
```

Example
```
:DISPLAY:NUMERIC:HARMONICS:LCURSOR TOTAL
:DISPLAY:NUMERIC:HARMONICS:LCURSOR?→:
DISPLAY:NUMERIC:HARMONICS:LCURSOR TOTAL
```

Description
- The cursor position is specified using the harmonic order.
- This command is valid when the numerical display format (`:DISPlay[:NUMeric]:HARMonics:IAMount`) is set to {SINGle|DUAL|SIGMa} list display.

:DISPlay[:NUMeric]:HARMonics:LIST<x>

Function Sets the list display items during harmonic measurement or queries the current setting.

Syntax
```
:DISPlay[:NUMeric]:HARMonics:
LIST<x> {<Function>,<Element>}
:DISPlay[:NUMeric]:HARMonics:LIST<x>?
<x> = 1, 2(item number)
<Function> = {U|I|P|S|Q|LAMBda|...}
             (See the function selection
             list on page 4-32 (3).)
<Element> = {<NRf>|SIGMA|SIGMB}
            (<NRf> = 1 to 4)
```

Example
```
:DISPLAY:NUMERIC:HARMONICS:LIST1 U,1
:DISPLAY:NUMERIC:HARMONICS:LIST1?→:
DISPLAY:NUMERIC:HARMONICS:LIST1 U,1
```

Description This command is valid when the numerical display format (`:DISPlay[:NUMeric]:HARMonics:IAMount`) is set to {SINGle|DUAL} list display.

:DISPlay[:NUMeric]:HARMonics:PRESet

Function Sets the numerical display items to a preset pattern during harmonic measurement.

Syntax
```
:DISPlay[:NUMeric]:HARMonics:
PRESet {<NRf>}
<NRf> = 1 to 4
```

Example
```
:DISPLAY:NUMERIC:HARMONICS:PRESET 1
```

Description Regardless of what value (1 to 4) is specified for <NRf>, the display pattern (order) of the numerical display items will be the same as the display order when Reset Exec of the Display setting menu, which is displayed on the PZ4000 screen, is executed. For details related to the order of displayed items when reset is executed, see the PZ4000 User's Manual.

:DISPlay[:NUMeric]:NORMal?

Function Queries all settings related to the numerical display during normal measurement.

Syntax
```
:DISPlay[:NUMeric]:NORMal?
```

Example
- Example when the numerical display format (`:DISPlay[:NUMeric]:NORMal:IAMount`) is set to "<NRf>" (split display)
```
:DISPLAY:NUMERIC:NORMAL?→:DISPLAY:
NUMERIC:NORMAL:IAMOUNT 8;ITEM1 URMS,1;
ITEM2 UMN,1;ITEM3 UDC,1;...
(abbreviated)...;ITEM255 NONE;
ICURSOR 1
```
- Example when the numerical display format (`:DISPlay[:NUMeric]:NORMal:IAMount`) is set to "ALL"
```
:DISPLAY:NUMERIC:NORMAL?→:DISPLAY:
NUMERIC:NORMAL:IAMOUNT ALL;
FCURSOR URMS
```

:DISPlay[:NUMeric]:NORMal:FCURsor

Function Sets the cursor position of the numerical display (All display) during normal measurement or queries the current setting.

Syntax
```
:DISPlay[:NUMeric]:NORMal:
FCURsor {<Function>}
:DISPlay[:NUMeric]:NORMal:FCURsor?
<Function> = {URMS|UMN|UDC|UAC|IRMS|...}
             (See the function selection
             list on page 4-31 (1).)
```

Example
```
:DISPLAY:NUMERIC:NORMAL:FCURSOR URMS
:DISPLAY:NUMERIC:NORMAL:FCURSOR?→:
DISPLAY:NUMERIC:NORMAL:FCURSOR URMS
```

Description
- The cursor position is specified using the function.
- This command is valid when the numerical display format (`:DISPlay[:NUMeric]:HARMonics:IAMount`) is set to "`ALL`."

:DISPlay[:NUMeric]:NORMal:IAMount

Function Sets the numerical display format during normal measurement or queries the current setting.

Syntax
```
:DISPlay[:NUMeric]:NORMal:
IAMount {<NRf>|ALL}
:DISPlay[:NUMeric]:NORMal:IAMount?
<NRf> = 8, 16, 42, 78
```

Example
```
:DISPLAY:NUMERIC:NORMAL:IAMOUNT 8
:DISPLAY:NUMERIC:NORMAL:IAMOUNT?→:
DISPLAY:NUMERIC:NORMAL:IAMOUNT 8
```

Description The displayed measurement data depend on the selected numerical display format as follows.

`<NRf>` = Displays the numerical display items in the order of item numbers. (`<NRf>` denotes the number of displayed items on one screen.)

`ALL` = Displays all functions in order for each element.

:DISPlay[:NUMeric]:NORMal:ICURsor

Function Sets the cursor position of the numerical display (split display) during normal measurement or queries the current setting.

Syntax
```
:DISPlay[:NUMeric]:NORMal:
ICURsor {<NRf>}
:DISPlay[:NUMeric]:NORMal:ICURsor?
<NRf> = 1 to 300
```

Example
```
:DISPLAY:NUMERIC:NORMAL:ICURSOR 1
:DISPLAY:NUMERIC:NORMAL:ICURSOR?→:
DISPLAY:NUMERIC:NORMAL:ICURSOR 1
```

Description
- The cursor position is specified using the item number.
- This command is valid when the numerical display format (`:DISPlay[:NUMeric]:HARMonics:IAMount`) is set to `<NRF>` (split display).

:DISPlay[:NUMeric]:NORMal:ITEM<x>

Function Sets the numerical displayed item during normal measurement or queries the current setting.

Syntax
```
:DISPlay[:NUMeric]:NORMal:
ITEM<x> {NONE|<Function>,<Element>}
:DISPlay[:NUMeric]:NORMal:ITEM<x>?
<x> = 1 to 255(item number)
NONE = no display item
<Function> = {URMS|UMN|UDC|UAC|IRMS|...}
             (See the function selection
             list on page 4-31 (1).)
<Element> = {<NRf>|SIGMA|SIGMB}(<NRf>=1
             to 4)
```

Example
```
:DISPLAY:NUMERIC:NORMAL:ITEM1 URMS,1
:DISPLAY:NUMERIC:NORMAL:ITEM1?→:
DISPLAY:NUMERIC:NORMAL:ITEM1 URMS,1
```

Description This command is valid when the numerical display format (`:DISPlay[:NUMeric]:HARMonics:IAMount`) is set to `<NRF>` (split display).

:DISPlay[:NUMeric]:NORMal:PRESet

Function Sets the numerical display items to a preset pattern during normal measurement.

Syntax
```
:DISPlay[:NUMeric]:NORMal:PRESet {<NRf>}
<NRf> = 1 to 4
```

Example
```
:DISPLAY:NUMERIC:NORMAL:PRESET 1
```

Description Regardless of what value (1 to 4) is specified for `<NRf>`, the display pattern (order) of the numerical display items will be the same as the display order when Reset Exec of the Display setting menu, which is displayed on the PZ4000 screen, is executed. For details related to the order of displayed items when reset is executed, see the PZ4000 User's Manual.

:DISPlay:VECTor?

Function Queries all settings related to the vector display.

Syntax
```
:DISPlay:VECTor?
```

Example
```
:DISPLAY:VECTOR?→:DISPLAY:VECTOR:
NUMERIC 1;UMAG 1.000;IMAG 1.000
```

:DISPlay:VECTor:IMAG

Function Sets the zoom factor of the current display during vector display or queries the current setting.

Syntax
```
:DISPlay:VECTor:IMAG {<NRf>}
:DISPlay:VECTor:IMAG?
<NRf> = 0.100 to 100,000
```

Example
```
:DISPLAY:VECTOR:IMAG 1
:DISPLAY:VECTOR:IMAG?→:DISPLAY:VECTOR:
IMAG 1.000
```

:DISPlay:VECTor:NUMeric

Function Turns ON/OFF the numerical data display during vector display or queries the current setting.

Syntax
```
:DISPlay:VECTor:NUMeric {<Boolean>}
:DISPlay:VECTor:NUMeric?
```

Example
```
:DISPLAY:VECTOR:NUMERIC ON
:DISPLAY:VECTOR:NUMERIC?→:DISPLAY:
VECTOR:NUMERIC 1
```

:DISPlay:VECTor:UMAG

Function Sets the zoom factor of the voltage display during vector display or queries the current setting.

Syntax
```
:DISPlay:VECTor:UMAG {<NRf>}
:DISPlay:VECTor:UMAG?
<NRf> = 0.100 to 100,000
```

Example
```
:DISPLAY:VECTOR:UMAG 1
:DISPLAY:VECTOR:UMAG?→:DISPLAY:VECTOR:
UMAG 1.000
```

:DISPlay:WAVE?

Function Queries all settings related to the waveform display.

Syntax
```
:DISPlay:WAVE?
```

Example
```
:DISPLAY:WAVE?→:DISPLAY:WAVE:
CHANNEL1 1;CHANNEL2 1;CHANNEL3 1;
CHANNEL4 1;CHANNEL5 1;CHANNEL6 1;
CHANNEL7 1;CHANNEL8 1;MATH1 0;MATH2 0;
FORMAT SINGLE;INTERPOLATE LINE;
GRATICULE GRID;SVALUE 0;TLABEL 1;
MAPPING:MODE AUTO
```

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

Function Turns ON/OFF the channel/computed waveform display or queries the current setting.

Syntax
```
:DISPlay:WAVE:{CHANnel<x>|
MATH<x>} {<Boolean>}
:DISPlay:WAVE:{CHANnel<x>|MATH<x>}?
```

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

Description The ":CHANnel<x>:DISPlay" and ":MATH<x>:FUNCtion" commands can be used to make the same settings and inquiries.

:DISPlay:WAVE:FORMat

Function Sets the display format of the waveform or queries the current setting.

Syntax
```
:DISPlay:WAVE:FORMat {SINGle|DUAL|TRIad|
QUAD}
:DISPlay:WAVE:FORMat?
```

Example
```
:DISPLAY:WAVE:FORMAT SINGLE
:DISPLAY:WAVE:FORMAT?→:DISPLAY:WAVE:
FORMAT SINGLE
```

:DISPlay:WAVE:GRATicule

Function Sets the graticule type (grid) or queries the current setting.

Syntax
```
:DISPlay:WAVE:GRATicule {GRID|FRAMe|
CROSshair}
:DISPlay:WAVE:GRATicule?
```

Example
```
:DISPLAY:WAVE:GRATICULE GRID
:DISPLAY:WAVE:GRATICULE?→:DISPLAY:WAVE:
GRATICULE GRID
```

:DISPlay:WAVE:INTerpolate

Function Sets the interpolation method of the waveform or queries the current setting.

Syntax
```
:DISPlay:WAVE:INTerpolate {OFF|LINE}
:DISPlay:WAVE:INTerpolate?
```

Example
```
:DISPLAY:WAVE:INTERPOLATE LINE
:DISPLAY:WAVE:INTERPOLATE?→:DISPLAY:
WAVE:INTERPOLATE LINE
```

:DISPlay:WAVE:MAPPing?

Function Queries all settings related to the waveform mapping to the split screen.

Syntax
```
:DISPlay:WAVE:MAPPing?
```

Example
```
:DISPLAY:WAVE:MAPPING?→:DISPLAY:WAVE:
MAPPING:MODE USER;CHANNEL1 0;CHANNEL2 0;
CHANNEL3 1;CHANNEL4 1;CHANNEL5 2;
CHANNEL6 2;CHANNEL7 3;CHANNEL8 3;
MATH1 0;MATH2 1
```

:DISPlay:WAVE:MAPPing:{CHANnel<x>|MATH<x>}

Function Sets the {channel waveform|MATH waveform} mapping to the split screen or queries the current setting.

Syntax
```
:DISPlay:WAVE:MAPPing:
{CHANnel<x>|MATH<x>} {<NRf>}
:DISPlay:WAVE:MAPPing:
{CHANnel<x>|MATH<x>}?
The <x> in CHANnel<x> = 1 to 8
The <x> in MATH<x> = 1 or 2.
<NRf> = 0 to 3
```

Example
```
:DISPLAY:WAVE:MAPPING:CHANNEL1 0
:DISPLAY:WAVE:MAPPING:CHANNEL1?→:
DISPLAY:WAVE:MAPPING:CHANNEL1 0
```

Description This command is valid when the waveform mapping method (:DISPlay:WAVE:MAPPing[:MODE]) is set to "USER."

:DISPlay:WAVE:MAPPing[:MODE]

Function Sets the waveform mapping method for the split screen or queries the current setting.

Syntax
```
:DISPlay:WAVE:MAPPing[:MODE] {AUTO|FIXed|USER}
:DISPlay:WAVE:MAPPing:MODE?
```

Example
```
:DISPLAY:WAVE:MAPPING:MODE AUTO
:DISPLAY:WAVE:MAPPING:MODE?→:DISPLAY:WAVE:MAPPING:MODE AUTO
```

:DISPlay:WAVE:SVALue (Scale VALue)

Function Turns ON/OFF the scale value display or queries the current setting.

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

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

:DISPlay:WAVE:TLABel (Trace LABel)

Function Turns ON/OFF the waveform label display or queries the current setting.

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

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

Description The waveform labels can be set using the "`:CHANnel<x>:LABel`" command.

:DISPlay:XY?

Function Queries all settings related to the X-Y display.

Syntax
```
:DISPlay:XY?
```

Example
```
:DISPLAY:XY?→:DISPLAY:XY:XTRACE 1;POSITION 20.000E-03,80.000E-03;INTERPOLATE LINE
```

:DISPlay:XY:FFT

Function Sets the range of the FFT waveform to be displayed on the X-Y display or queries the current setting.

Syntax
```
:DISPlay:XY:FFT {<frequency>,<frequency>|<NRf>,<NRf>}:DISPlay:XY:FFT?
<frequency> = 0 to 2.5MHz (during the normal measurement mode, when Time Base = Internal)
<NRf> = 0 to 5000 (when Time Base = External or during the harmonic measurement mode)
```

Example
```
:DISPLAY:XY:FFT 0,200KHZ
:DISPLAY:XY:FFT?→:DISPLAY:XY:FFT 0.000E+00,200.0E+03
```

Description
- Set the start point first and then the end point.
- This command is valid when "`:DISPlay:XY:XTRace`" is set to "`MATH<x>`" and the equation of "`MATH<x>`" is set to FFT.
- The range and resolution of <Frequency> is determined from the sampling rate and the number of FFT points.
- <NRf> is set in terms of harmonic order. The range depends on the number of FFT points as follows. For the procedure to set the number of FFT points, see the "`:MATH<x>:FFT:POINt`" command.
  For 1000 points: 0 to 500
  For 2000 points: 0 to 1000
  For 10000 points: 0 to 5000

:DISPlay:XY:INTerpolate

Function Sets the interpolation method of the waveform or queries the current setting.

Syntax
```
:DISPlay:XY:INTerpolate {OFF|LINE}
:DISPlay:XY:INTerpolate?
```

Example
```
:DISPLAY:XY:INTERPOLATE LINE
:DISPLAY:XY:INTERPOLATE?→:DISPLAY:XY:INTERPOLATE LINE
```

Description The "`:DISPlay:WAVE:INTerpolate`" command can be used to make the same settings and inquiries.

:DISPlay:XY:POSition

Function Sets the range of the T-Y waveform to be displayed on the X-Y display or queries the current setting.

Syntax
```
:DISPlay:XY:POSition {<time>,
<time>|<NRf>,<NRf>}:DISPlay:XY:POSition?
<time> = 0 to (OBSERVATION TIME) (during
         the normal measurement mode,
         when Time Base = Internal)
<NRf> = 0 to (Record Length) (when Time
        Base = External or during the
        harmonic measurement mode)
```

Example
```
:DISPLAY:XY:POSITION 0,80MS
:DISPLAY:XY:POSITION?→:DISPLAY:XY:
POSITION 0.000E-03,80.000E-03
```

Description
- Set the start point first and then the end point.
- The range and resolution of <time> depend on the observation time.
- When using <NRf>, specify using the number of sampling data points. The range is from 0 to (record length).

:DISPlay:XY:XTRace

Function Sets the channel to assign to the X-axis of the X-Y display or queries the current setting.

Syntax
```
:DISPlay:XY:XTRace {<NRf>|MATH<x>}
:DISPlay:XY:XTRace?
<NRf> = 1 to 8 (channel)
<x> = 1, 2 (MATH)
```

Example
```
:DISPLAY:XY:XTRACE 1
:DISPLAY:XY:XTRACE?→:DISPLAY:XY:
XTRACE 1
```

* Function selection (<Function>) list

(1) Functions in the normal measurement mode
Applicable commands
```
:DISPlay[:NUMeric]:NORMal:FCURsor
:DISPlay[:NUMeric]:NORMal:ITEM<x>
```

Selection used in communications	:	Function name used in the menu (numerical display header name)
URMS	:	Urms
UMN	:	Umean
UDC	:	Udc
UAC	:	Uac
IRMS	:	Irms
IMN	:	Imean
IDC	:	Idc
IAC	:	Iac
P	:	P
S	:	S
Q	:	Q
LAMBda	:	λ
PHI	:	ϕ
FU	:	FreqU (fU)
FI	:	FreqI (fI)
UPPeak	:	U+peak (U+pk)
UMPeak	:	U-peak (U-pk)
IPPeak	:	I+peak (I+pk)
IMPeak	:	I-peak (I-pk)
CFU	:	CfU
CFI	:	CfI
FFU	:	FfU
FFI	:	FfI
Z	:	Z
RS	:	Rs
XS	:	Xs
RP	:	Rp
XP	:	Xp
PC	:	Pc
ETA	:	η
SETA	:	1/η
F1	:	F1
F2	:	F2
F3	:	F3
F4	:	F4
DURMS	:	ΔUrms
DUMN	:	ΔUmean
DUDC	:	ΔUdc
DUAC	:	ΔUac
DIRMS	:	ΔIrms
DIMN	:	ΔImean
DIDC	:	ΔIdc
DIAC	:	ΔIac
SPEed	:	Speed
TORQue	:	Torque
SYNC	:	SyncSpd
SLIP	:	Slip
PM	:	Pm

MAETa : ηmA
MBETa : ηmB
* SPEed, TORQue, SYNC, SLIP, PM, MAETa, and MBETa are applicable when the motor module is installed.

(2) Functions in the harmonic measurement mode
Applicable commands
`:DISPlay[:NUMeric]:HARMonics:ITEM<x>`

Selection used in communications	:	Function name used in the menu (numerical display header name)
U	:	U
I	:	I
P	:	P
S	:	S
Q	:	Q
LAMBda	:	λ
PHI	:	ϕ
PHIU	:	ϕU
PHII	:	ϕI
FU	:	FreqU (fU)
FI	:	FreqI (fI)
Z	:	Z
RS	:	Rs
XS	:	Xs
RP	:	Rp
XP	:	Xp
UHDF	:	Uhdf
IHDF	:	Ihdf
PHDF	:	Phdf
SHDF	:	Shdf
QHDF	:	Qhdf
UTHD	:	Uthd
ITHD	:	Ithd
PTHD	:	Pthd
STHD	:	Sthd
QTHD	:	Qthd
UTHF	:	Uthf
ITHF	:	Ithf
UTIF	:	Utif
ITIF	:	Itif
HVF	:	hvf
HCF	:	hcf
ETA	:	η
SETA	:	1/η
PHI_U1U2	:	ηU1-U2
PHI_U1U3	:	ηU1-U3
PHI_U1I1	:	ηU1-I1
PHI_U1I2	:	ηU1-I2
PHI_U1I3	:	ηU1-I3
F1	:	F1
F2	:	F2
F3	:	F3
F4	:	F4
SPEed	:	Speed
TORQue	:	Torque
SYNC	:	SyncSpd
SLIP	:	Slip
PM	:	Pm
MAETa	:	ηmA
MBETa	:	ηmB

* SPEed, TORQue, SYNC, SLIP, PM, MAETa, and MBETa are applicable when the motor module is installed.

(3) Functions in the harmonic measurement mode (list display)
Applicable commands
`:DISPlay[:NUMeric]:HARMonics:LIST<x>`
`:DISPlay:BAR:ITEM<x>`
`:FILE:SAVE:NUMeric:LIST`

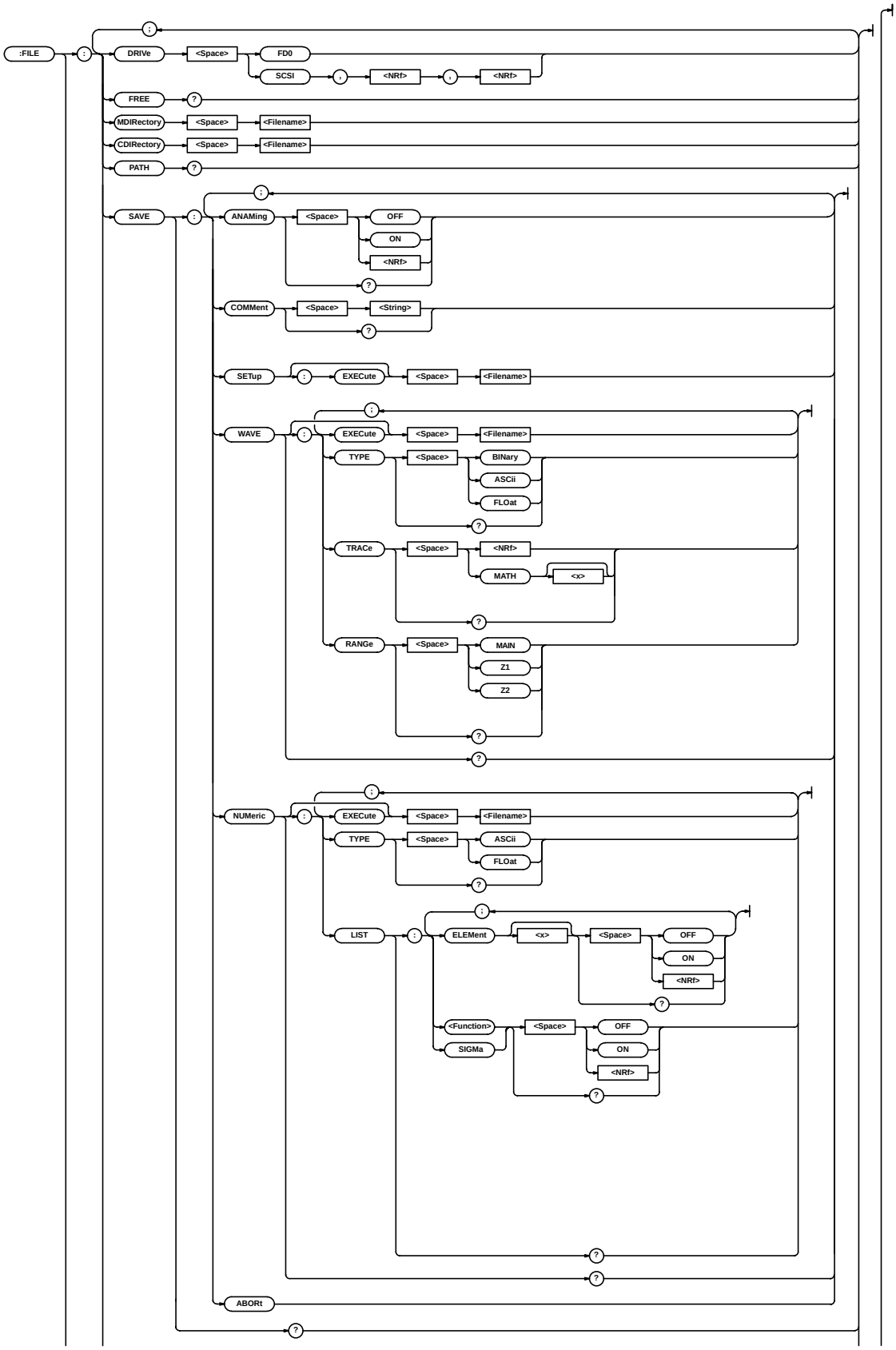
Selection used in communications	:	Function name used in the menu (numerical display header name)
U	:	U
I	:	I
P	:	P
S	:	S
Q	:	Q
LAMBda	:	λ
PHI	:	ϕ
PHIU	:	ϕU
PHII	:	ϕI
Z	:	Z
RS	:	Rs
XS	:	Xs
RP	:	Rp
XP	:	Xp
TORQue	:	Torque

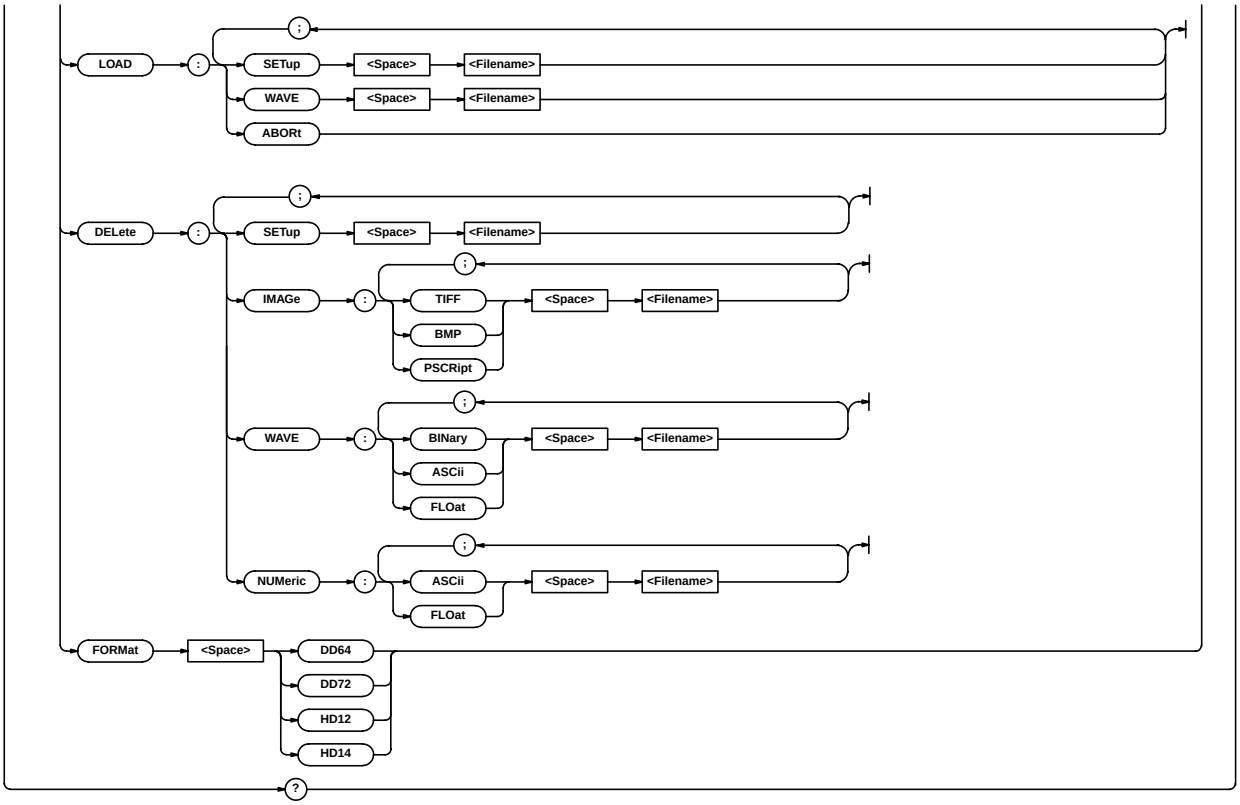
* TORQue is applicable when the motor module is installed.

4.8 FILE Group

The commands in the FILE Group deal with file operations.

These commands can be used to make the same settings and inquiries as when the FILE key on the front panel is pressed.





:FILE?

Function Queries all settings related to file operations.
Syntax `:FILE?`
Example `:FILE?→:FILE:SAVE:ANAMING 1;COMMENT "CASE1";WAVE:TYPE BINARY;RANGE MAIN;:FILE:SAVE:NUMERIC:TYPE FLOAT`

:FILE:CDIRectory

Function Changes the current directory.
Syntax `:FILE:CDIRectory {<Filename>}`
`<Filename> = directory name`
Example `:FILE:CDIRECTORY "IMAGE"`
Description Specify "`..`" to move to a higher directory.

:FILE:DELete:IMAGe:{TIFF|BMP|PSCRipt}

Function Deletes a screen image data file.
Syntax `:FILE:DELete:IMAGe:{TIFF|BMP|PSCRipt} {<Filename>}`
Example `:FILE:DELETE:IMAGE:TIFF "IMAGE1"`

:FILE:DELete:NUMeric:{ASCii|FLOat}

Function Deletes a numerical data file.
Syntax `:FILE:DELete:NUMeric:{ASCii|FLOat} {<Filename>}`
Example `:FILE:DELETE:NUMERIC:ASCII "NUM1"`

:FILE:DELete:SETup

Function Deletes a setup parameter file.
Syntax `:FILE:DELete:SETup {<Filename>}`
Example `:FILE:DELETE:SETUP "SETUP1"`

:FILE:DELete:WAVE:{BINary|ASCii|FLOat}

Function Deletes a waveform data file.
Syntax `:FILE:DELete:WAVE:{BINary|ASCii|FLOat} {<Filename>}`
Example `:FILE:DELETE:WAVE:BINARY "WAVE1"`

:FILE:DRIVe

Function Sets the drive (medium) setting.
Syntax `:FILE:DRIVe {FD0|SCSI,<NRf>[,<NRf>]}`
`First <NRf> = SCSI address (0 to 7)`
`Second <NRf> = partition (0 to 5)`
Example `:FILE:DRIVE FD0`
Description If you are using a drive that has no partitions set, omit the second `<NRf>`.

:FILE:FORMat

Function Formats the floppy disk.
Syntax `:FILE:FORMat {DD64|DD72|HD12|HD14}`
Example `:FILE:FORMAT HD14`

:FILE:FREE?

Function Queries the free space (bytes) on the drive.
Syntax `:FILE:FREE?`
Example `:FILE:FREE?→163840`

:FILE:LOAD:ABORt

Function Aborts loading a file.
Syntax `:FILE:LOAD:ABORt`
Example `:FILE:LOAD:ABORT`

:FILE:LOAD:SETup

Function Loads a setup parameter file.
Syntax `:FILE:LOAD:SETup {<Filename}`
Example `:FILE:LOAD:SETUP "SETUP1"`

:FILE:LOAD:WAVE

Function Loads a waveform data file.
Syntax `:FILE:LOAD:WAVE {<Filename>}`
Example `:FILE:LOAD:WAVE "WAVE1"`
Description Only waveform data in binary format can be loaded.

:FILE:MDIRectory

Function Creates a directory.
Syntax `:FILE:MDIRectory {<Filename>}`
`<Filename> = directory name`
Example `:FILE:MDIRECTORY "TEST"`

:FILE:PATH?

Function Queries the absolute path of the current directory.
Syntax `:FILE:PATH?`
Example `:FILE:PATH?→"FD0\IMAGE"`

:FILE:SAVE?

Function Queries all settings related to saving a file.
Syntax `:FILE:SAVE?`
Example `:FILE:SAVE?→:FILE:SAVE:ANAMING 1;COMMENT "CASE1";WAVE:TYPE BINARY;RANGE MAIN;:FILE:SAVE:NUMERIC:TYPE FLOAT`

:FILE:SAVE:ABORt

Function Aborts saving the file.
Syntax `:FILE:SAVE:ABORt`
Example `:FILE:SAVE:ABORT`

:FILE:SAVE:ANAMing

Function Sets whether or not to automatically assign file names or queries the current setting.
Syntax `:FILE:SAVE:ANAMing {<Boolean>}`
`:FILE:SAVE:ANAMing?`
Example `:FILE:SAVE:ANAMING ON`
`:FILE:SAVE:ANAMING?→:FILE:SAVE:ANAMING 1`

:FILE:SAVE:COMMent

Function Sets the comment that is attached to the file being saved or queries the current setting.
Syntax `:FILE:SAVE:COMMent {<string>}`
`:FILE:SAVE:COMMent?`
`<string> = 25 characters or less`
Example `:FILE:SAVE:COMMENT "CASE1"`
`:FILE:SAVE:COMMENT?→:FILE:SAVE:COMMENT "CASE1"`

:FILE:SAVE:NUMeric?

Function Queries all settings related to saving the numerical data to a file.
Syntax `:FILE:SAVE:NUMeric?`
Example `:FILE:SAVE:NUMERIC?→:FILE:SAVE:NUMERIC:TYPE FLOAT`

:FILE:SAVE:NUMeric[:EXECute]

Function Saves the numerical data to a file.
Syntax `:FILE:SAVE:NUMeric[:EXECute] {<Filename>}`
Example `:FILE:SAVE:NUMERIC:EXECUTE "NUM1"`

:FILE:SAVE:NUMeric:LIST?

Function Queries all settings related to saving the numerical list data to a file during harmonic measurement.
Syntax `:FILE:SAVE:NUMeric:LIST?`
Example `:FILE:SAVE:NUMERIC:LIST?→:FILE:SAVE:NUMERIC:LIST:ELEMENT1 1;ELEMENT2 0;ELEMENT3 0;ELEMENT4 0;U 1;I 0;P 0;S 0;Q 0;LAMBDA 0;PHI 0;PHIU 0;PHII 0;Z 0;RS 0;XS 0;RP 0;XP 0;SIGMA 0`

:FILE:SAVE:NUMeric:LIST:ELEMent<x>

Function Turns ON/OFF the output of each element when saving numerical list data to a file during harmonic measurement or queries the current setting.
Syntax `:FILE:SAVE:NUMeric:LIST:ELEMent<x> {<Boolean>}`
`:FILE:SAVE:NUMeric:LIST:ELEMent<x>?`
`<x> = 1 to 4`
Example `:FILE:SAVE:NUMERIC:LIST:ELEMENT1 ON`
`:FILE:SAVE:NUMERIC:LIST:ELEMENT1?→:FILE:SAVE:NUMERIC:LIST:ELEMENT1 1`

:FILE:SAVE:NUMeric:LIST:{<List-Function>|SIGMa}

Function Turns ON/OFF the output of each function when saving numerical list data to a file during harmonic measurement or queries the current setting.
Syntax `:FILE:SAVE:NUMeric:LIST:{<List-Function>|SIGMa} {<Boolean>}`
`:FILE:SAVE:NUMeric:LIST:{<List-Function>|SIGMa}?`
`List-Function`
`= {U|I|P|S|Q|LAMBda|...}` (See the function selection list on page 4-32 (3).)
Example `:FILE:SAVE:NUMERIC:LIST:U ON`
`:FILE:SAVE:NUMERIC:LIST:U?→:FILE:SAVE:NUMERIC:LIST:U 1`

:FILE:SAVE:NUMeric:TYPE

Function Sets the format of the numerical data being saved or queries the current setting.

Syntax `:FILE:SAVE:NUMeric:TYPE {ASCii|FLOat}`
`:FILE:SAVE:NUMeric:TYPE?`

Example `:FILE:SAVE:NUMERIC:TYPE FLOAT`
`:FILE:SAVE:NUMERIC:TYPE?→:FILE:SAVE:NUMERIC:TYPE FLOAT`

:FILE:SAVE:SETup[:EXECute]

Function Saves the setup parameters to a file.

Syntax `:FILE:SAVE:SETup[:EXECute] {<Filename>}`

Example `:FILE:SAVE:SETUP:EXECUTE "SETUP1"`

:FILE:SAVE:WAVE?

Function Queries all settings related to saving the waveform data to a file.

Syntax `:FILE:SAVE:WAVE?`

Example `:FILE:SAVE:WAVE?→:FILE:SAVE:WAVE:TYPE BINARY;RANGE MAIN`

:FILE:SAVE:WAVE[:EXECute]

Function Saves the waveform data to a file.

Syntax `:FILE:SAVE:WAVE[:EXECute] {<Filename>}`

Example `:FILE:SAVE:WAVE:EXECUTE "WAVE1"`

:FILE:SAVE:WAVE:RANGe

Function Sets the range of the waveform to save to the file or queries the current setting.

Syntax `:FILE:SAVE:WAVE:RANGe {MAIN|Z1|Z2}`
`:FILE:SAVE:WAVE:RANGe?`

Example `:FILE:SAVE:WAVE:RANGE MAIN`
`:FILE:SAVE:WAVE:RANGE?→:FILE:SAVE:WAVE:RANGE MAIN`

:FILE:SAVE:WAVE:TRACe

Function Sets the waveform to save to the file or queries the current setting.

Syntax `:FILE:SAVE:WAVE:TRACe {<NRf>|MATH<x>}`
`:FILE:SAVE:WAVE:TRACe?`
`<NRf> = 1 to 8(channel)`
`<x> = 1, 2(MATH)`

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

Description This command is valid when the format of the waveform data being saved (`:FILE:SAVE:WAVE:TYPE`) is set to `{FLOat}`. If it is set to `{BINary|ASCii}`, then all waveforms that are turned ON will be selected.

:FILE:SAVE:WAVE:TYPE

Function Sets the format of the waveform data being saved or queries the current setting.

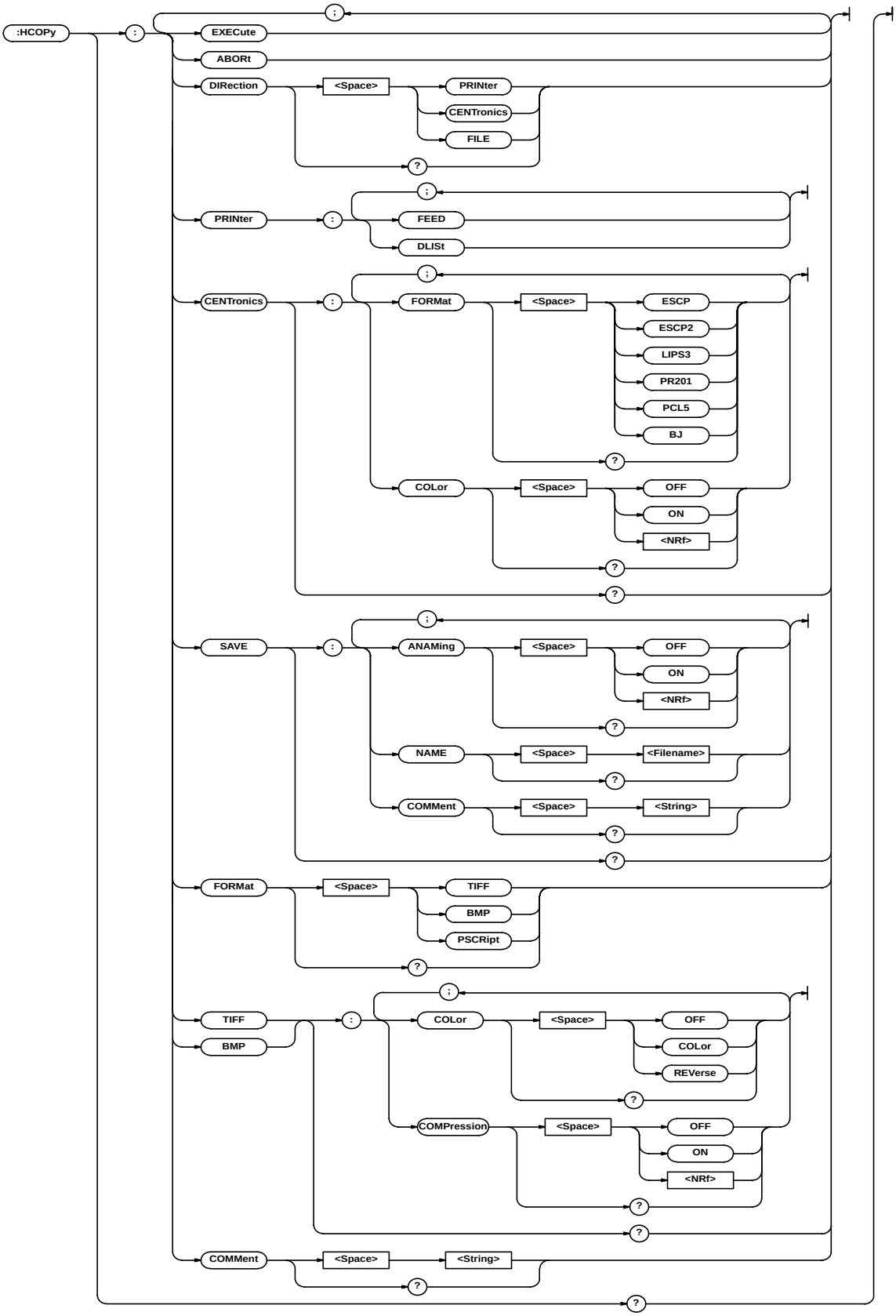
Syntax `:FILE:SAVE:WAVE:TYPE {BINary|ASCii|FLOat}`
`:FILE:SAVE:WAVE:TYPE?`

Example `:FILE:SAVE:WAVE:TYPE BINARY`
`:FILE:SAVE:WAVE:TYPE?→:FILE:SAVE:WAVE:TYPE BINARY`

Description Waveform data files that are saved in binary format can be loaded by this instrument.

4.9 HCOPy Group

The commands in the HCOPy Group deal with the output of screen data to the built-in printer (option) or other devices. These commands can be used to make the same settings and inquiries as when the COPY or MENU (SHIFT+COPY) key on the front panel is pressed.



:HCOPy?

Function Queries all settings related to screen data output.
Syntax :HCOPy?
Example :HCOPY?→:HCOPY:DIRECTION PRINTER;
COMMENT "THIS IS TEST."

:HCOPy:ABORt

Function Aborts data output and paper feeding.
Syntax :HCOPy:ABORt
Example :HCOPY:ABORT

:HCOPy:CENTronics?

Function Queries all settings related to the external printer output.
Syntax :HCOPy:CENTronics?
Example :HCOPY:CENTRONICS?→:HCOPY:CENTRONICS:
FORMAT ESCP2;COLOR 0

:HCOPy:CENTronics:COLor

Function Sets the color (ON/OFF) of the external printer output or queries the current setting.
Syntax :HCOPy:CENTronics:COLor {<Boolean>}
:HCOPy:CENTronics:COLor?
Example :HCOPY:CENTRONICS:COLOR OFF
:HCOPY:CENTRONICS:COLOR?→:HCOPY:
CENTRONICS:COLOR 0

:HCOPy:CENTronics:FORMat

Function Sets the command format that is output to the printer or queries the current setting.
Syntax :HCOPy:CENTronics:FORMat {ESCP|ESCP2|
LIPS3|PR201|PCL5|BJ}
:HCOPy:CENTronics:FORMat?
Example :HCOPY:CENTRONICS:FORMAT ESCP2
:HCOPY:CENTRONICS:FORMAT?→:HCOPY:
CENTRONICS:FORMAT ESCP2

:HCOPy:COMMent

Function Sets the comment that is printed at the lower section of the screen or queries the current setting.
Syntax :HCOPy:COMMent {<string>}
:HCOPy:COMMent?
<string> = 25 characters or less
Example :HCOPY:COMMENT "THIS IS TEST."
:HCOPY:COMMENT?→:HCOPY:
COMMENT "THIS IS TEST."
Description Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.

:HCOPy:DIRection

Function Sets the output destination of the data or queries the current setting.
Syntax :HCOPy:DIRection {PRINter|CENTronics|
FILE}
:HCOPy:DIRection?
Example :HCOPY:DIRECTION PRINTER
:HCOPY:DIRECTION?→:HCOPY:
DIRECTION PRINTER
Description "PRINter" is an option.

:HCOPy:EXECute

Function Executes data output. This is an overlap command.
Syntax :HCOPy:EXECute
Example :HCOPY:EXECUTE

:HCOPy:FORMat

Function Sets the output data format or queries the current setting.
Syntax :HCOPy:FORMat {TIFF|BMP|PSCRipt}
:HCOPy:FORMat?
Example :HCOPY:FORMAT TIFF
:HCOPY:FORMAT?→:HCOPY:FORMAT TIFF
Description This command is void when the data output destination (:HCOPy:DIRection) is set to {PRINter|CENTronics}.

:HCOPy:PRINter:DLISt

Function Outputs of the numerical data list to the built-in printer. This is an overlap command.
Syntax :HCOPy:PRINter:DLISt
Example :HCOPY:PRINTER:DLIST

:HCOPy:PRINter:FEED

Function Feeds the paper (built-in printer). This is an overlap command.
Syntax :HCOPy:PRINter:FEED
Example :HCOPY:PRINTER FEED

:HCOPy:SAVE?

Function Queries all settings related to saving the file.
Syntax :HCOPy:SAVE?
Example :HCOPY:SAVE?→:HCOPY:SAVE:ANAMING 1;
NAME "DATA1";COMMENT "CASE1"

:HCOPy:SAVE:ANAMing

Function Sets whether or not to automatically assign file names or queries the current setting.
Syntax :HCOPy:SAVE:ANAMing {<Boolean>}
:HCOPy:SAVE:ANAMing?
Example :HCOPY:SAVE:ANAMING ON
:HCOPY:SAVE:ANAMING?→:HCOPY:SAVE:
ANAMING 1

:HCOPy:SAVE:COMMent

Function Sets the comment that is attached to the file being saved or queries the current setting.

Syntax
```
:HCOPy:SAVE:COMMent {<string>}
:HCOPy:SAVE:COMMent?
<string> = 25 characters or less
```

Example
```
:HCOPY:SAVE:COMMENT "CASE1"
:HCOPY:SAVE:COMMENT?→:HCOPY:SAVE:
COMMENT "CASE1"
```

Description Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.

:HCOPy:SAVE:NAME

Function Sets the file name or queries the current setting.

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

Example
```
:HCOPY:SAVE:NAME "DATA1"
:HCOPY:SAVE:NAME?→:HCOPY:SAVE:
NAME "DATA1"
```

Description The save destination of the screen data is specified using:

- the “`:FILE:DRIVe`” command for the drive.
- the “`:FILE:CDIRectory`” command for the directory.

The save destination path can be queried using the “`:FILE:PATH?`” command.

:HCOPy:{TIFF|BMP}?

Function Queries all settings related to the TIFF/BMP format.

Syntax
```
:HCOPy:{TIFF|BMP}?
```

Example
```
:HCOPY:TIFF?→:HCOPY:TIFF:COLOR COLOR;
COMPRESSION 0
```

:HCOPy:{TIFF|BMP}:COLor

Function Sets the color for the TIFF/BMP format or queries the current setting.

Syntax
```
:HCOPy:{TIFF|BMP}:COLor {OFF|COLor|
REVerse}
:HCOPy:{TIFF|BMP}:COLor?
```

Example
```
:HCOPY:TIFF:COLOR COLOR
:HCOPY:TIFF:COLOR?→:HCOPY:TIFF:
COLOR COLOR
```

:HCOPy:{TIFF|BMP}:COMPression

Function TIFF/BMP Sets whether or not to compress the data in TIFF/BMP format or queries the current setting.

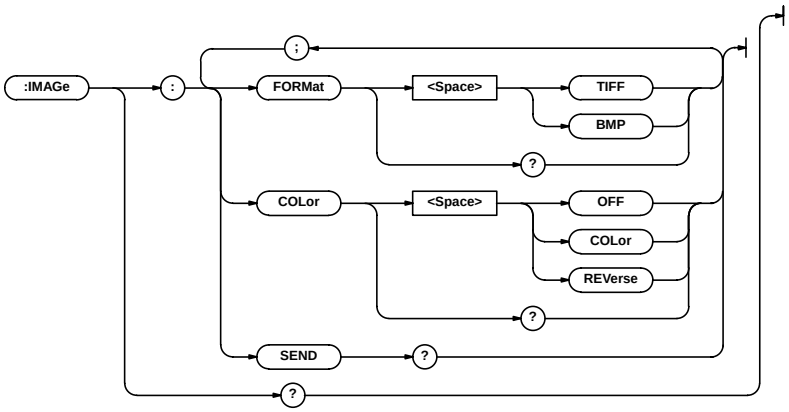
Syntax
```
:HCOPy:{TIFF|BMP}:
COMPression {<Boolean>}
:HCOPy:{TIFF|BMP}:COMPression?
```

Example
```
:HCOPY:TIFF:COMPRESSION OFF
:HCOPY:TIFF:COMPRESSION?→:HCOPY:TIFF:
COMPRESSION 0
```

Description This command is valid when the color (“`:HCOPy:{TIFF|BMP}:COLor`”) is set to `{COLor|REVerse}`.

4.10 IMAGe Group

The commands in the IMAGe Group deal with the output of screen image data.
There are no front-panel keys that correspond to the commands in this group.



:IMAGe?

Function Queries all settings related to the output of the screen image data.

Syntax `:IMAGe?`

Example `:IMAGE?→:IMAGE:FORMAT TIFF;COLOR OFF`

:IMAGe:COLor

Function Sets the color of the screen image data being output or queries the current setting.

Syntax `:IMAGe:COLor {OFF|COLor|REVerse}`
`:IMAGe:COLor?`

Example `:IMAGE:COLOR OFF`
`:IMAGE:COLOR?→:IMAGE:COLOR OFF`

:IMAGe:FORMat

Function Sets the output format of the screen image data or queries the current setting.

Syntax `:IMAGe:FORMat {TIFF|BMP}`
`:IMAGe:FORMat?`

Example `:IMAGE:FORMAT TIFF`
`:IMAGE:FORMAT?→:IMAGE:FORMAT TIFF`

:IMAGe:SEND?

Function Queries the screen image data.

Syntax `:IMAGe:SEND?`

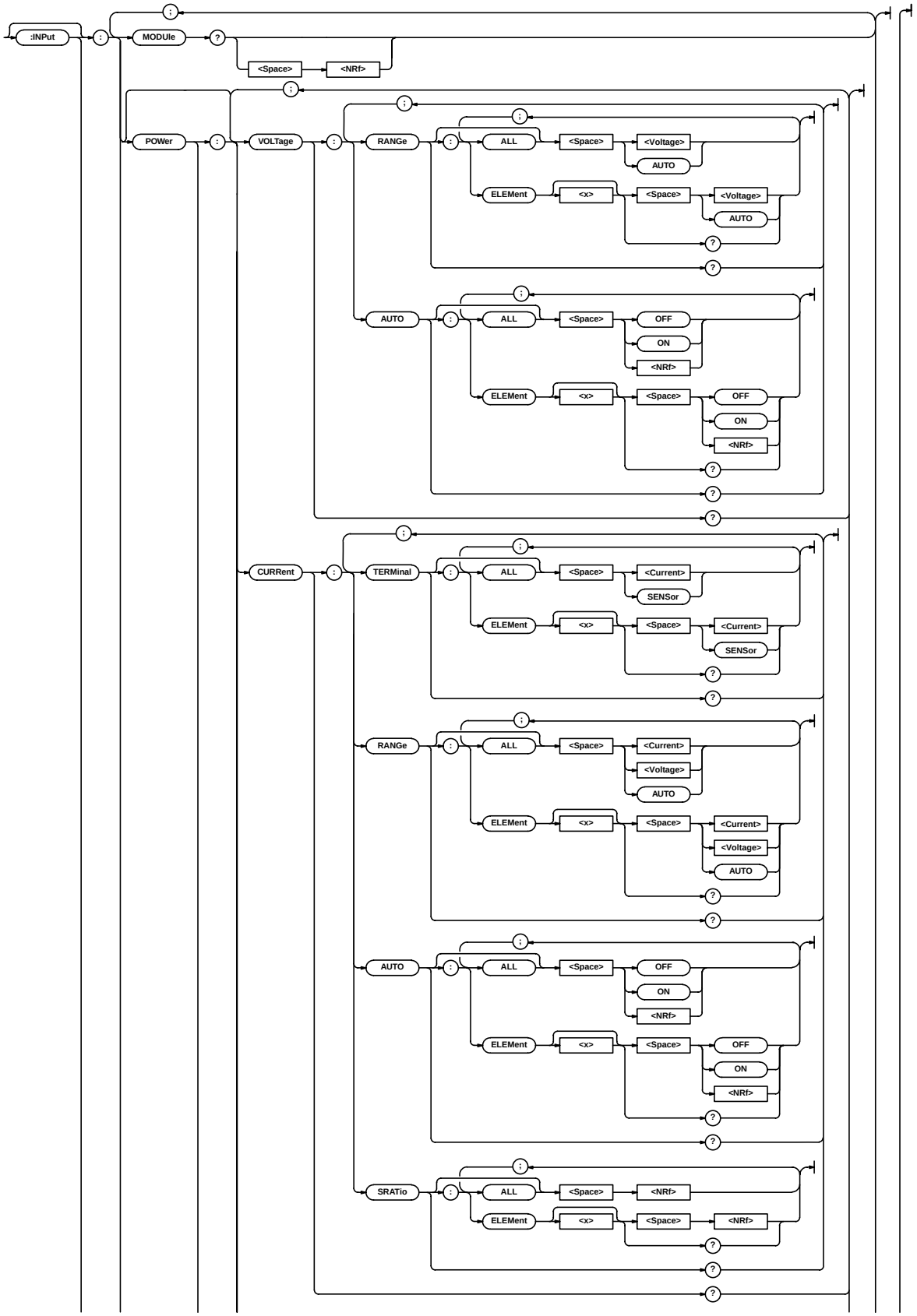
Example `:IMAGE:SEND?→#6 (Number of bytes, 6 digits) (Series of data bytes)`

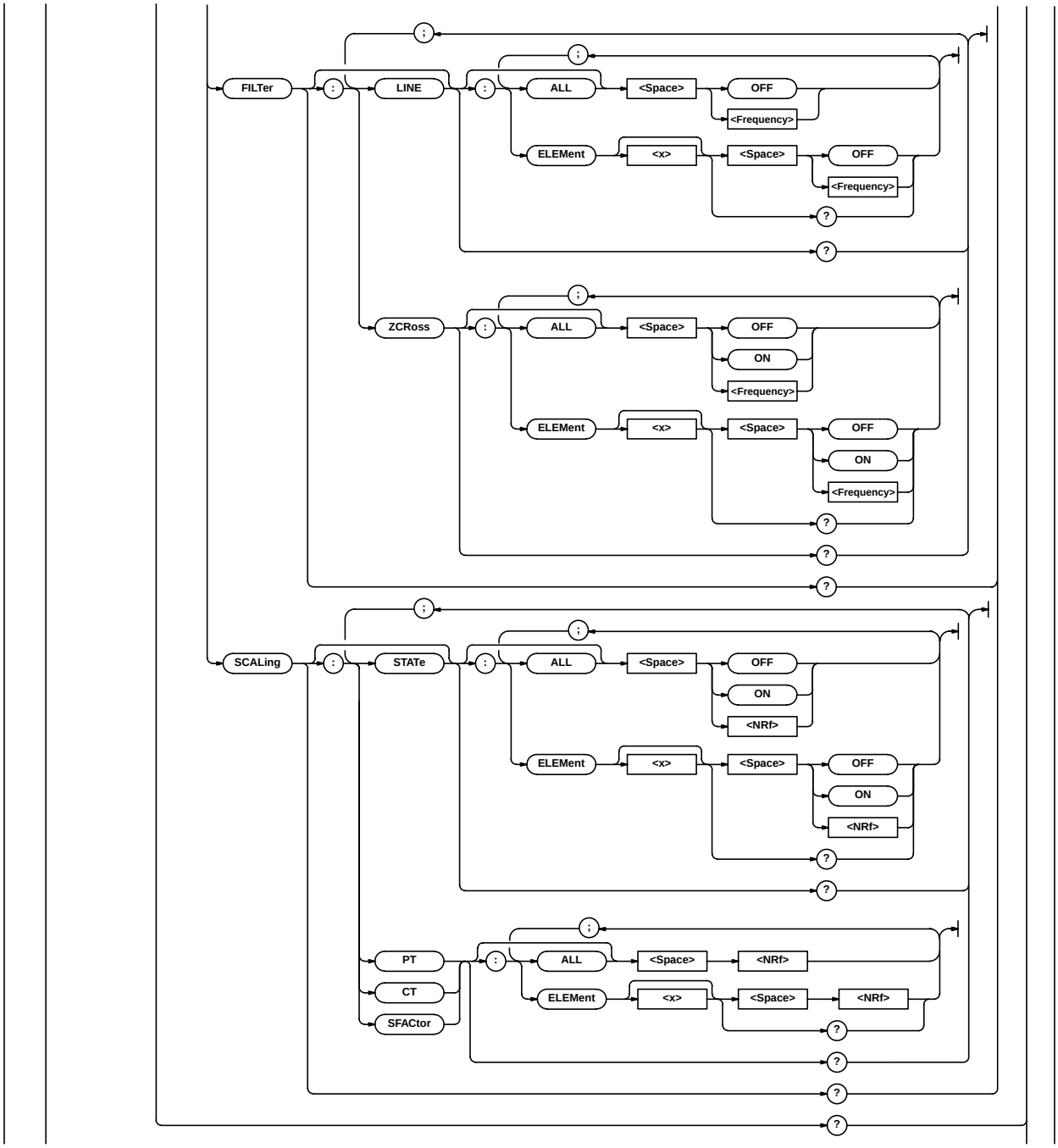
Description
- The number of bytes in <block data> is {2 + 6 + number of data + 1 (delimiter)}.
- For information about block data, see page 3-6.

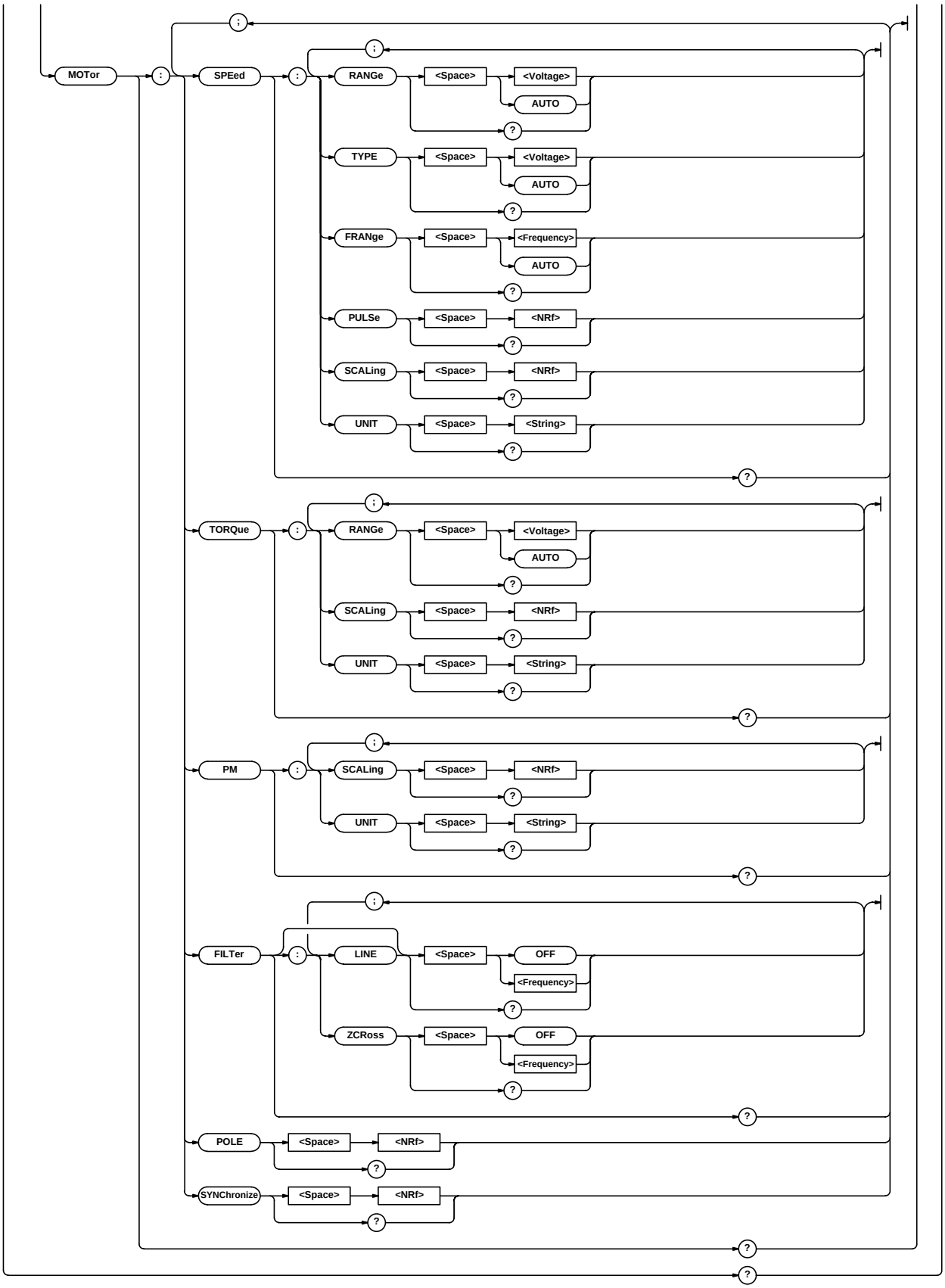
4.11 INPut Group

The commands in the INPut Group deal with the measurement conditions of each input module.

These commands can be used to make the same settings and inquiries as when the INPUT key on the front panel is pressed.




FILTer
LINE
ALL
<Space>
OFF
<Frequency>
ELEMent
<x>
<Space>
OFF
<Frequency>
ZCRoss
ALL
<Space>
OFF
ON
<Frequency>
ELEMent
<x>
<Space>
OFF
ON
<Frequency>
SCALing
STATe
ALL
<Space>
OFF
ON
<NRf>
ELEMent
<x>
<Space>
OFF
ON
<NRf>
PT
CT
SFACtor
ALL
<Space>
<NRf>
ELEMent
<x>
<Space>
<NRf>
:
;
?


MOTor
SPEed
RANGe
<Space>
<Voltage>
AUTO
TYPE
FRANge
<Frequency>
PULSe
<NRf>
SCALing
UNIT
<String>
TORQue
PM
FILTer
LINE
OFF
ZCRoss
POLE
SYNChronize

:INPut?

Function Queries all settings related to all input modules.

Syntax `:INPut?`

Example
- When the motor module is not installed
  `:INPUT?→(Same as the response to "[:INPUT]:POWer?")`
- When the motor module is installed
  `:INPUT?→(Same as the response to "[:INPUT]:POWer?");(Same as the response to "[:INPUT]:MOTor?")`

[:INPut]:MODUle?

Function Queries the model name of each input module.

Syntax
```
[:INPut]:MODUle? {|<NRf>}
[:INPut]:MODUle?
<NRf> = 1 to 4 (element)
```

Example
```
:INPUT:MODULE? 1→253751
:INPUT:MODULE?→253751, 253752, 253752, 253771
```

Description
- The following responses are possible.
  253751 = Power measurement module (1000 V/5 A)
  253752 = Power measurement module (1000 V/20&5 A)
  253771 = Motor module (Speed/Torque)
  0 = no module
- If the parameter is omitted, the model name of each input module is returned for all elements in order starting with element 1.

[:INPut]:MOTor?

Function Queries all settings related to the motor module.

Syntax `[:INPut]:MOTor?`

Example
```
:INPUT:MOTOR?→:INPUT:MOTOR:SPEED:RANGE 50.0E+00;TYPE ANALOG;FRANGE 200.00E+03;PULSE 60;SCALING 1.0000;UNIT "rpm";:INPUT:MOTOR:TORQUE:RANGE 50.0E+00;SCALING 1.0000;UNIT "Nm";:INPUT:MOTOR:PM:SCALING 1.0000;UNIT "W";:INPUT:MOTOR:FILTER:LINE OFF;ZCROSS OFF;:INPUT:MOTOR:POLE 2;SYNCHRONIZE 2
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:FILTer?

Function Queries all settings related to the filter for the motor module.

Syntax `[:INPut]:MOTor:FILTer?`

Example
```
:INPUT:MOTOR:FILTER?→:INPUT:MOTOR:FILTER:LINE OFF;ZCROSS OFF
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:FILTer[:LINE]

Function Sets the line filter for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:FILTer[:LINE] {OFF|<frequency>}
[:INPut]:MOTor:FILTer:LINE?
OFF = Line filter OFF
<frequency> = 100Hz, 500Hz (line filter ON, cutoff frequency)
```

Example
```
:INPUT:MOTOR:FILTER:LINE OFF
:INPUT:MOTOR:FILTER:LINE?→:INPUT:MOTOR:FILTER:LINE OFF
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:FILTer:ZCRoss

Function Sets the zero crossing filter for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:FILTer:ZCRoss {OFF|<frequency>}
[:INPut]:MOTor:FILTer:ZCRoss?
OFF = zero crossing filter OFF
<frequency> = 100Hz, 500Hz (zero crossing filter ON, cutoff frequency)
```

Example
```
:INPUT:MOTOR:FILTER:ZCROSS OFF
:INPUT:MOTOR:FILTER:ZCROSS?→:INPUT:MOTOR:FILTER:ZCROSS OFF
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:PM?

Function Queries all settings related to the motor output for the motor module.

Syntax `[:INPut]:MOTor:PM?`

Example
```
:INPUT:MOTOR:PM?→:INPUT:MOTOR:PM:SCALING 1.0000;UNIT "W"
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:PM:SCALing

Function Sets the scaling factor used during motor output computation on the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:PM:SCALing {<NRf>}
[:INPut]:MOTor:PM:SCALing?
<NRf> = 0.0001 to 99999.9999
```

Example
```
:INPUT:MOTOR:PM:SCALING 1
:INPUT:MOTOR:PM:SCALING?→:INPUT:MOTOR:PM:SCALING 1.0000
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:PM:UNIT

Function Sets the unit to add to the motor output computation result or queries the current setting.

Syntax
```
[:INPut]:MOTor:PM:UNIT {<string>}
[:INPut]:MOTor:PM:UNIT?
<string> = 8 characters or less
```

Example
```
:INPUT:MOTOR:PM:UNIT "W"
:INPUT:MOTOR:PM:UNIT?→:INPUT:MOTOR:PM:
UNIT "W"
```

Description
- Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.
- This command never affects the computation result.
- If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:POLE

Function Sets the motor's number of poles for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:POLE {<NRf>}
[:INPut]:MOTor:POLE?
<NRf> = 1 to 99
```

Example
```
:INPUT:MOTOR:POLE 2
:INPUT:MOTOR:POLE?→:INPUT:MOTOR:POLE 2
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:SPEed?

Function Queries all settings related to the revolution sensor signal input for the motor module.

Syntax
```
[:INPut]:MOTor:SPEed?
```

Example
```
:INPUT:MOTOR:SPEED?→:INPUT:MOTOR:SPEED:
RANGE 50.0E+00;TYPE ANALOG;
FRANGE 200.00E+03;PULSE 60;
SCALING 1.0000;UNIT "Nm"
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:SPEed:FRANge

Function Sets the frequency range of the revolution sensor signal input (pulse input) for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:SPEed:FRANge {<frequency>
|AUTO}
[:INPut]:MOTor:SPEed:FRANge?
<Frequency> = 40(Hz): 1 to 40 Hz
            = 800(Hz): 16 to 800 Hz
            = 8k(Hz): 250 to 8 kHz
            = 200k(Hz): 2 k to 200 kHz
AUTO        = Auto range
```

Example
```
:INPUT:MOTOR:SPEED:FRANGE 200KHZ
:INPUT:MOTOR:SPEED:FRANGE?→:INPUT:
MOTOR:SPEED:FRANGE 200.00E+03
```

Description
- Set the <Frequency> to the maximum value within the frequency range.
- This command is valid when the input format of the revolution sensor signal ([:INPut]:MOTor:SPEed:TYPE) is set to "PULSe (pulse input)."
- If the 253771 motor module is not installed, an error will occur.
- The ":CHANnel7:SPEed:FRANGe" command can be used to make the same settings and inquiries.

[:INPut]:MOTor:SPEed:PULSe

Function Sets the pulse count of the revolution sensor signal input (pulse input) for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:SPEed:PULSe {<NRf>}
[:INPut]:MOTor:SPEed:PULSe?
<NRf> = 1 to 9999
```

Example
```
:INPUT:MOTOR:SPEED:PULSE 60
:INPUT:MOTOR:SPEED:PULSE?→:INPUT:MOTOR:
SPEED:PULSE 60
```

Description
- This command is valid when the input format of the revolution sensor signal ([:INPut]:MOTor:SPEed:TYPE) is set to "PULSe (pulse input)."
- If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:SPEed:RANGe

Function Sets the voltage range of the revolution sensor signal input for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:SPEed:RANGe {<voltage>|AUTO}
[:INPut]:MOTor:SPEed:RANGe?
<voltage> = 1, 2, 5, 10, 20, and 50(V)
AUTO = Auto range
```

Example
```
:INPUT:MOTOR:SPEED:RANGE 50V
:INPUT:MOTOR:SPEED:RANGE?→:INPUT:MOTOR:SPEED:RANGE 50.0E+00
```

Description
- When the input format of the revolution sensor signal ([:INPut]:MOTor:SPEed:TYPE) is set to "PULSe (pulse input)," it is fixed to 5 (V).
- If the 253771 motor module is not installed, an error will occur.
- The ":CHANnel7:SPEed:RANGe" command can be used to make the same settings and inquiries.

[:INPut]:MOTor:SPEed:SCALing

Function Sets the scaling factor used during rotating speed computation on the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:SPEed:SCALing {<NRf>}
[:INPut]:MOTor:SPEed:SCALing?
<NRf> = 0.0001 to 99999.9999
```

Example
```
:INPUT:MOTOR:SPEED:SCALING 1
:INPUT:MOTOR:SPEED:SCALING?→:INPUT:MOTOR:SPEED:SCALING 1.0000
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:SPEed:TYPE

Function Sets the input type of the revolution sensor signal input for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:SPEed:TYPE {ANALog|PULSe}
[:INPut]:MOTor:SPEed:TYPE?
```

Example
```
:INPUT:MOTOR:SPEED:TYPE ANALOG
:INPUT:MOTOR:SPEED:TYPE?→:INPUT:MOTOR:SPEED:TYPE ANALOG
```

Description
- If the 253771 motor module is not installed, an error will occur.
- The ":CHANnel7:SPEed:TYPE" command can be used to make the same settings and inquiries.

[:INPut]:MOTor:SPEed:UNIT

Function Sets the unit to add to the rotating speed computation result or queries the current setting.

Syntax
```
[:INPut]:MOTor:SPEed:UNIT {<string>}
[:INPut]:MOTor:SPEed:UNIT?
<string> = 8 characters or less
```

Example
```
:INPUT:MOTOR:SPEED:UNIT "rpm"
:INPUT:MOTOR:SPEED:UNIT?→:INPUT:MOTOR:SPEED:UNIT "rpm"
```

Description
- Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.
- This command never affects the computation result.
- If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:SYNChronize

Function Sets the frequency measurement source for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:SYNChronize {<NRf>}
[:INPut]:MOTor:SYNChronize?
<NRf> = 1 to 8
```

Example
```
:INPUT:MOTOR:SYNCHRONIZE 2
:INPUT:MOTOR:SYNCHRONIZE?→:INPUT:MOTOR:SYNCHRONIZE 2
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:TORQue?

Function Queries all settings related to the torque meter signal input for the motor module.

Syntax
```
[:INPut]:MOTor:TORQue?
```

Example
```
:INPUT:MOTOR:TORQUE?→:INPUT:MOTOR:TORQUE:RANGE 50.0E+00;SCALING 1.0000;UNIT "Nm"
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:TORQue:RANGe

Function Sets the voltage range of the torque meter signal input for the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:TORQue:RANGe {<voltage>|AUTO}
[:INPut]:MOTor:TORQue:RANGe?
<voltage> = 1, 2, 5, 10, 20, and 50(V)
AUTO = Auto range
```

Example
```
:INPUT:MOTOR:TORQUE:RANGE 50V
:INPUT:MOTOR:TORQUE:RANGE?→:INPUT:MOTOR:TORQUE:RANGE 50.0E+00
```

Description
- If the 253771 motor module is not installed, an error will occur.
- The ":CHANnel8:TORQue:RANGe" command can be used to make the same settings and inquiries.

[:INPut]:MOTor:TORQue:SCALing

Function Sets the scaling factor used during torque computation on the motor module or queries the current setting.

Syntax
```
[:INPut]:MOTor:TORQue:SCALing {<NRf>}
[:INPut]:MOTor:TORQue:SCALing?
<NRf> = 0.0001 to 99999.9999
```

Example
```
:INPUT:MOTOR:TORQUE:SCALING 1
:INPUT:MOTOR:TORQUE:SCALING?→:INPUT:
MOTOR:TORQUE:SCALING 1.0000
```

Description If the 253771 motor module is not installed, an error will occur.

[:INPut]:MOTor:TORQue:UNIT

Function Sets the unit to add to the torque computation result or queries the current setting.

Syntax
```
[:INPut]:MOTor:TORQue:UNIT {<string>}
[:INPut]:MOTor:TORQue:UNIT?
<string> = 8 characters or less
```

Example
```
:INPUT:MOTOR:TORQUE:UNIT "Nm"
:INPUT:MOTOR:TORQUE:UNIT?→:INPUT:MOTOR:
TORQUE:UNIT "Nm"
```

Description
- Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.
- This command never affects the computation result.
- If the 253771 motor module is not installed, an error will occur.

[:INPut]:POWer?

Function Queries all settings related to the power measurement module.

Syntax
```
[:INPut]:POWer?
```

Example
```
:INPUT:POWER?→:INPUT:POWER:VOLTAGE:
RANGE:ELEMENT1 2.00E+03;
ELEMENT2 2.00E+03;ELEMENT3 2.00E+03;:
INPUT:POWER:CURRENT:TERMINAL:
ELEMENT1 5.0E+00;ELEMENT2 5.0E+00;
ELEMENT3 5.0E+00;ELEMENT4 5.0E+00;:
INPUT:POWER:CURRENT:RANGE:
ELEMENT1 10.0E+00;ELEMENT2 10.0E+00;
ELEMENT3 10.0E+00;ELEMENT4 10.0E+00;:
INPUT:POWER:CURRENT:SRATIO:
ELEMENT1 10.0000;ELEMENT2 10.0000;
ELEMENT3 10.0000;ELEMENT4 10.0000;:
INPUT:POWER:FILTER:LINE:ELEMENT1 OFF;
ELEMENT2 OFF;ELEMENT3 OFF;ELEMENT4 OFF;:
INPUT:POWER:FILTER:ZCROSS:ELEMENT1 OFF;
ELEMENT2 OFF;ELEMENT3 OFF;ELEMENT4 OFF;:
INPUT:POWER:SCALING:STATE:ELEMENT1 0;
ELEMENT2 0;ELEMENT3 0;ELEMENT4 0;:
INPUT:POWER:SCALING:PT:ELEMENT1 1.0000;
ELEMENT2 1.0000;ELEMENT3 1.0000;
ELEMENT4 1.0000;:INPUT:POWER:SCALING:
CT:ELEMENT1 1.0000;ELEMENT2 1.0000;
ELEMENT3 1.0000;ELEMENT4 1.0000;:INPUT:
POWER:SCALING:SFACTOR:ELEMENT1 1.0000;
ELEMENT2 1.0000;ELEMENT3 1.0000;
ELEMENT4 1.0000
```

[:INPut][:POWer]:CURRent?

Function Queries all settings related to the current measurement on the power measurement module.

Syntax
```
[:INPut][:POWer]:CURRent?
```

Example
```
:INPUT:POWER:CURRENT?→:INPUT:POWER:
CURRENT:TERMINAL:ELEMENT1 5.0E+00;
ELEMENT2 5.0E+00;ELEMENT3 5.0E+00;
ELEMENT4 5.0E+00;:INPUT:POWER:CURRENT:
RANGE:ELEMENT1 10.0E+00;
ELEMENT2 10.0E+00;ELEMENT3 10.0E+00;
ELEMENT4 10.0E+00;:INPUT:POWER:CURRENT:
SRATIO:ELEMENT1 10.0000;
ELEMENT2 10.0000;ELEMENT3 10.0000;
ELEMENT4 10.0000
```

[:INPut][:POWer]:CURRent:AUTO?

Function Queries the ON/OFF state of the current auto range function of all elements with the power measurement modules.

Syntax `[:INPut][:POWer]:CURRent:AUTO?`

Example `:INPUT:POWER:CURRENT:AUTO?→:INPUT:POWER:CURRENT:AUTO:ELEMENT1 0;ELEMENT2 0;ELEMENT3 0;ELEMENT4 0`

[:INPut][:POWer]:CURRent:AUTO[:ALL]

Function Turns ON/OFF the current auto range function of all elements with the power measurement modules.

Syntax `[:INPut][:POWer]:CURRent:AUTO[:ALL] {<Boolean>}`

Example `:INPUT:POWER:CURRENT:AUTO:ALL ON`

[:INPut][:POWer]:CURRent:AUTO:ELEMent<x>

Function Turns ON/OFF the current auto range function of each element with power measurement module or queries the current setting.

Syntax `[:INPut][:POWer]:CURRent:AUTO:ELEMent<x> {<Boolean>}`
`[:INPut][:POWer]:CURRent:AUTO:ELEMent<x>?`
`<x> = 1 to 4`

Example `:INPUT:POWER:CURRENT:AUTO:ELEMENT1 ON`
`:INPUT:POWER:CURRENT:AUTO:ELEMENT1?→:INPUT:POWER:CURRENT:AUTO:ELEMENT1 1`

[:INPut][:POWer]:CURRent:RANGe?

Function Queries the current range of all elements with the power measurement modules.

Syntax `[:INPut][:POWer]:CURRent:RANGe?`

Example `:INPUT:POWER:CURRENT:RANGE?→:INPUT:POWER:CURRENT:RANGE:ELEMENT1 10.0E+00;ELEMENT2 10.0E+00;ELEMENT3 10.0E+00;ELEMENT4 10.0E+00`

[:INPut][:POWer]:CURRent:RANGe[:ALL]

Function Sets the current range of all elements with the power measurement modules.

Syntax `[:INPut][:POWer]:CURRent:RANGe[:ALL] {<current>|<voltage>|AUTO}`
`<current> = 0.1, 0.2, 0.4, 1, 2, 4, 10(A)`
`(when TERMinal = 5(A))`
`<current> = 1, 2, 4, 10, 20, 40, 100(A)`
`(when TERMinal = 20(A))`
`<voltage> = 0.1, 0.2, 0.4, 1(V)`
`(when TERMinal = SENSor)`
`AUTO = AUTO RANGE`

Example `:INPUT:POWER:CURRENT:RANGE:ALL 10A`

Description
- The selectable range is determined by the current input terminal setting of element 1 (`[:INPut][:POWer]:CURRent:TERMinal:ELEMent1`). Therefore, only elements that have the same current measurement terminal setting as element 1 are set.

[:INPut][:POWer]:CURRent:RANGe:ELEMent<x>

Function Sets the current range of each element with the power measurement module or queries the current setting.

Syntax `[:INPut][:POWer]:CURRent:RANGe:ELEMent<x> {<current>|<voltage>|AUTO}`
`[:INPut][:POWer]:CURRent:RANGe:ELEMent<x>?`
`<x> = 1 to 4`
`<current> = 0.1, 0.2, 0.4, 1, 2, 4, 10(A)`
`(when TERMinal = 5(A))`
`<current> = 1, 2, 4, 10, 20, 40, 100(A)`
`(when TERMinal = 20(A))`
`<voltage> = 0.1, 0.2, 0.4, 1(V)`
`(when TERMinal = SENSor)`
`AUTO = AUTO RANGE`

Example `:INPUT:POWER:CURRENT:RANGE:ELEMENT1 10A`
`:INPUT:POWER:CURRENT:RANGE:ELEMENT1?→:INPUT:POWER:CURRENT:RANGE:ELEMENT1 10.0E+00`

Description
- The selectable range is determined by the current input terminal setting of the specified element.
- The "`:CHANnel<x>:CURRent:RANGe` (where <x> is the channel number)" command can be used to make the same settings and inquiries.
- Setting "`AUTO`" using this command is equivalent to specifying "`ON`" using the "`[:INPut][:POWer]:CURRent:AUTO:ELEMent<x>`" command.

[:INPut][:POWer]:CURRent:SRATio?

Function Queries the current sensor's transformation ratio of all elements with the power measurement modules.

Syntax `[:INPut][:POWer]:CURRent:SRATio?`

Example `:INPUT:POWER:CURRENT:SRATIO?→:INPUT:POWER:CURRENT:SRATIO:ELEMENT1 10.0000;ELEMENT2 10.0000;ELEMENT3 10.0000;ELEMENT4 10.0000`

[:INPut][:POWer]:CURRent:SRATio[:ALL]

Function Sets the current sensor transformation ratio of all elements with the power measurement modules.

Syntax `[:INPut][:POWer]:CURRent:SRATio[:ALL] {<NRf>}`
`<NRf> = 0.0001 to 99999.9999`

Example `:INPUT:POWER:CURRENT:SRATIO:ALL 10`

[:INPut][:POWer]:CURRent:SRATio:ELEMent<x>

Function Sets the current sensor transformation ratio of each element with the power measurement module or queries the current setting.

Syntax `[:INPut][:POWer]:CURRent:SRATio:ELEMent<x> {<NRf>}`
`[:INPut][:POWer]:CURRent:SRATio:ELEMent<x>?`
`<x> = 1 to 4`
`<NRf> = 0.0001 to 99999.9999`

Example `:INPUT:POWER:CURRENT:SRATIO:ELEMENT1 10`
`:INPUT:POWER:CURRENT:SRATIO:ELEMENT1?→:INPUT:POWER:CURRENT:SRATIO:ELEMENT1 10.0000`

[:INPut][:POWer]:CURRent:TERMinal?

Function Queries the current input terminals of all elements with the power measurement modules.

Syntax `[:INPut][:POWer]:CURRent:TERMinal?`

Example `:INPUT:POWER:CURRENT:TERMINAL?→:INPUT:POWER:CURRENT:TERMINAL:ELEMENT1 5.0E+00;ELEMENT2 5.0E+00;ELEMENT3 5.0E+00;ELEMENT4 5.0E+00`

[:INPut][:POWer]:CURRent:TERMinal[:ALL]

Function Sets the current input terminals of all elements with the power measurement modules.

Syntax `[:INPut][:POWer]:CURRent:TERMinal[:ALL] {<current>|SENSor}`
`<current> = 5, 20(A)`
`SENSor = current sensor`

Example `:INPUT:POWER:CURRENT:TERMINAL:ALL 5A`

Description
- For elements that have 253751 power measurement modules (1000V/5A) installed, 20(A) setting will not be carried out.
- For elements that do not have 253751/253752 power measurement modules installed, current measurement terminal settings will not be carried out.

[:INPut][:POWer]:CURRent:TERMinal:ELEMent<x>

Function Sets the current input terminals of each element with the power measurement module or queries the current setting.

Syntax `[:INPut][:POWer]:CURRent:TERMinal:ELEMent<x> {<current>|SENSor}`
`[:INPut][:POWer]:CURRent:TERMinal:ELEMent<x>?`
`<x> = 1 to 4`
`<current> = 5(A)`
(for 253751 power measurement modules)
`<current> = 5, 20(A)`
(for 253752 power measurement modules)
`SENSor = current sensor`

Example `:INPUT:POWER:CURRENT:TERMINAL:ELEMENT1 5A`
`:INPUT:POWER:CURRENT:TERMINAL:ELEMENT1?→:INPUT:POWER:CURRENT:TERMINAL:ELEMENT1 5.0E+00`

Description
- If the 253752/253752 power measurement module is not installed, an error will occur.
- The "`:CHANnel<x>:CURRent:TERMinal` (where <x> is the channel number)" command can be used to make the same settings and inquiries.

[:INPut][:POWer]:FILTer?

Function Queries all settings related to the filter for the power measurement module.

Syntax `[:INPut][:POWer]:FILTer?`

Example `:INPUT:POWER:FILTER?→:INPUT:POWER:FILTER:LINE:ELEMENT1 OFF;ELEMENT2 OFF;ELEMENT3 OFF;ELEMENT4 OFF;:INPUT:POWER:FILTER:ZCROSS:ELEMENT1 OFF;ELEMENT2 OFF;ELEMENT3 OFF;ELEMENT4 OFF`

[:INPut][:POWer]:FILTer:LINE?

Function Queries the line filter setting of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:FILTer:LINE?
```

Example
```
:INPUT:POWER:FILTER:LINE?→:INPUT:POWER:
FILTER:LINE:ELEMENT1 OFF;ELEMENT2 OFF;
ELEMENT3 OFF;ELEMENT4 OFF
```

[:INPut][:POWer]:FILTer[:LINE][:ALL]

Function Sets the line filter setting of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:FILTer[:LINE][:
ALL] {OFF|<frequency>}
OFF = Line filter OFF
<frequency> = 500Hz, 20kHz, 1MHz (line
              filter ON, cut-off
              frequency)
```

Example
```
:INPUT:POWER:FILTER:LINE:ALL OFF
```

[:INPut][:POWer]:FILTer[:LINE]:ELEMent<x>

Function Sets the line filter setting of each element with the power measurement module or queries the current setting.

Syntax
```
[:INPut][:POWer]:FILTer[:LINE]:
ELEMent<x> {OFF|<frequency>}
[:INPut][:POWer]:FILTer[:LINE]:
ELEMent<x>?
<x> = 1 to 4
OFF = Line filter OFF
<frequency> = 500Hz, 20kHz, 1MHz (line
              filter ON, cut-off
              frequency)
```

Example
```
:INPUT:POWER:FILTER:LINE:ELEMENT1 OFF
:INPUT:POWER:FILTER:LINE:ELEMENT1?→:
INPUT:POWER:FILTER:LINE:ELEMENT1 OFF
```

[:INPut][:POWer]:FILTer:ZCRoss?

Function Queries the zero crossing filter of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:FILTer:ZCRoss?
```

Example
```
:INPUT:POWER:FILTER:ZCROSS?→:
INPUT:POWER:FILTER:ZCROSS:ELEMENT1 OFF;
ELEMENT2 OFF;ELEMENT3 OFF;ELEMENT4 OFF
```

[:INPut][:POWer]:FILTer:ZCRoss[:ALL]

Function Sets the zero crossing filter of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:FILTer:ZCRoss[:
ALL] {OFF|<frequency>}
OFF = Zero crossing filter OFF
<frequency> = 500Hz, 20kHz (zero
              crossing filter ON, cut-
              off frequency)
```

Example
```
:INPUT:POWER:FILTER:ZCROSS:ALL OFF
```

[:INPut][:POWer]:FILTer:ZCRoss:ELEMent<x>

Function Sets the zero crossing filter of each element with the power measurement module or queries the current setting.

Syntax
```
[:INPut][:POWer]:FILTer:ZCRoss:
ELEMent<x> {OFF|<frequency>}
[:INPut][:POWer]:FILTer:ZCRoss:
ELEMent<x>?
<x> = 1 to 4
OFF = Zero crossing filter OFF
<frequency> = 500Hz, 20kHz (zero
              crossing filter ON, cut-
              off frequency)
```

Example
```
:INPUT:POWER:FILTER:ZCROSS:ELEMENT1 OFF
:INPUT:POWER:FILTER:ZCROSS:ELEMENT1?→:
INPUT:POWER:FILTER:ZCROSS:ELEMENT1 0
```

[:INPut][:POWer]:SCALing?

Function Queries all settings related to scaling for the power measurement module.

Syntax
```
[:INPut][:POWer]:SCALing?
```

Example
```
:INPUT:POWER:SCALING?→:INPUT:POWER:
SCALING:STATE:ELEMENT1 0;ELEMENT2 0;
ELEMENT3 0;ELEMENT4 0;:INPUT:POWER:
SCALING:PT:ELEMENT1 1.0000;
ELEMENT2 1.0000;ELEMENT3 1.0000;
ELEMENT4 1.0000;:INPUT:POWER:SCALING:CT:
ELEMENT1 1.0000;ELEMENT2 1.0000;
ELEMENT3 1.0000;ELEMENT4 1.0000;:INPUT:
POWER:SCALING:SFACTOR:ELEMENT1 1.0000;
ELEMENT2 1.0000;ELEMENT3 1.0000;
ELEMENT4 1.0000
```

[:INPut][:POWer]:SCALing:{PT|CT|SFACtor}?

Function Queries the PT ratio/CT ratio/power coefficient of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:SCALing:{PT|CT|
SFACtor}?
```

Example
```
:INPUT:POWER:SCALING:PT?→:INPUT:POWER:
SCALING:PT:ELEMENT1 1.0000;
ELEMENT2 1.0000;ELEMENT3 1.0000;
ELEMENT4 1.0000
```

[:INPut][:POWer]:SCALing:{PT|CT|SFACtor}[:ALL]

Function Sets the PT ratio/CT ratio/power coefficient of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:SCALing:{PT|CT|SFACtor}
[:ALL] {<NRf>}
<NRf> = 0.0001 to 99999.9999
```

Example
```
:INPUT:POWER:SCALING:PT:ALL 1
```

[:INPut][:POWer]:SCALing:{PT|CT|SFACtor}:ELEMent<x>

Function Sets the PT ratio/CT ratio/power coefficient of each element with the power measurement module or queries the current setting.

Syntax
```
[:INPut][:POWer]:SCALing:
{PT|CT|SFACtor}:ELEMent<x> {<NRf>}
[:INPut][:POWer]:SCALing:
{PT|CT|SFACtor}:ELEMent<x>?
<x> = 1 to 4
<NRf> = 0.0001 to 99999.9999
```

Example
```
:INPUT:POWER:SCALING:PT:ELEMENT1 1
:INPUT:POWER:SCALING:PT:ELEMENT1?→:
INPUT:POWER:SCALING:PT:ELEMENT1 1.0000
```

[:INPut][:POWer]:SCALing:STATe?

Function Queries the ON/OFF state of the scaling function of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:SCALing:STATe?
```

Example
```
:INPUT:POWER:SCALING:STATE?→:INPUT:
POWER:SCALING:STATE:ELEMENT1 0;
ELEMENT2 0;ELEMENT3 0;ELEMENT4 0
```

[:INPut][:POWer]:SCALing[:STATe][:ALL]

Function Turns ON/OFF the scaling function of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:SCALing[:STATe][:
ALL] {<Boolean>}
```

Example
```
:INPUT:POWER:SCALING:STATE:ALL OFF
```

[:INPut][:POWer]:SCALing[:STATe]:ELEMent<x>

Function Turns ON/OFF the scaling function of each element with the power measurement module or queries the current setting.

Syntax
```
[:INPut][:POWer]:SCALing[:STATe]:
ELEMent<x> {<Boolean>}
[:INPut][:POWer]:SCALing[:STATe]:
ELEMent<x>?
```

Example
```
:INPUT:POWER:SCALING:STATE:ELEMENT1 OFF
:INPUT:POWER:SCALING:STATE:ELEMENT1?→:
INPUT:POWER:SCALING:STATE:ELEMENT1 0
```

[:INPut][:POWer]:VOLTage?

Function Queries all settings related to the voltage measurement for power measurement modules.

Syntax
```
[:INPut][:POWer]:VOLTage?
```

Example
```
:INPUT:POWER:VOLTAGE?→:INPUT:POWER:
VOLTAGE:RANGE:ELEMENT1 2.00E+03;
ELEMENT2 2.00E+03;ELEMENT3 2.00E+03;
ELEMENT4 2.00E+03
```

[:INPut][:POWer]:VOLTage:AUTO?

Function Queries the ON/OFF state of the voltage auto range function of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:VOLTage:AUTO?
```

Example
```
:INPUT:POWER:VOLTAGE:AUTO?→:INPUT:
POWER:VOLTAGE:AUTO:ELEMENT1 0;
ELEMENT2 0;ELEMENT3 0;ELEMENT4 0
```

[:INPut][:POWer]:VOLTage:AUTO[:ALL]

Function Turns ON/OFF the voltage auto range function of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:VOLTage:AUTO[:
ALL] {<Boolean>}
```

Example
```
:INPUT:POWER:VOLTAGE:AUTO:ALL ON
```

[:INPut][:POWer]:VOLTage:AUTO:ELEMent<x>

Function Turns ON/OFF the voltage auto range function of each element with the power measurement module or queries the current setting.

Syntax
```
[:INPut][:POWer]:VOLTage:AUTO:
ELEMent<x> {<Boolean>}
[:INPut][:POWer]:VOLTage:AUTO:
ELEMent<x>?
<x> = 1 to 4
```

Example
```
:INPUT:POWER:VOLTAGE:AUTO:ELEMENT1 ON
:INPUT:POWER:VOLTAGE:AUTO:ELEMENT1?→:
INPUT:POWER:VOLTAGE:AUTO:ELEMENT1 1
```

[:INPut][:POWer]:VOLTage:RANGe?

Function Queries the voltage range of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:VOLTage:RANGe?
```

Example
```
:INPUT:POWER:VOLTAGE:RANGE?→:INPUT:
POWER:VOLTAGE:RANGE:ELEMENT1 2.00E+03;
ELEMENT2 2.00E+03;ELEMENT3 2.00E+03;
ELEMENT4 2.00E+03
```

[:INPut][:POWer]:VOLTage:RANGe[:ALL]

Function Sets the voltage range of all elements with the power measurement modules.

Syntax
```
[:INPut][:POWer]:VOLTage:RANGe[:
ALL] {<voltage>|AUTO}
<voltage> = 30, 60, 120, 200, 300, 600,
            1200, 2000(V)
AUTO = AUTO RANGE
```

Example
```
:INPUT:POWER:VOLTAGE:RANGE:ALL 2000V
```

Description Setting "AUTO" using this command is equivalent to executing "[:INPut][:POWer]:VOLTage:AUTO[:ALL] ON."

[:INPut][:POWer]:VOLTage:RANGe:ELEMent<x>

Function Sets the voltage range of each element with the power measurement module or queries the current setting.

Syntax
```
[:INPut][:POWer]:VOLTage:RANGe:
ELEMent<x> {<voltage>|AUTO}
[:INPut][:POWer]:VOLTage:RANGe:
ELEMent<x>?
<x> = 1 to 4
<voltage> = 30, 60, 120, 200, 300, 600,
            1200, 2000(V)
AUTO = AUTO RANGE
```

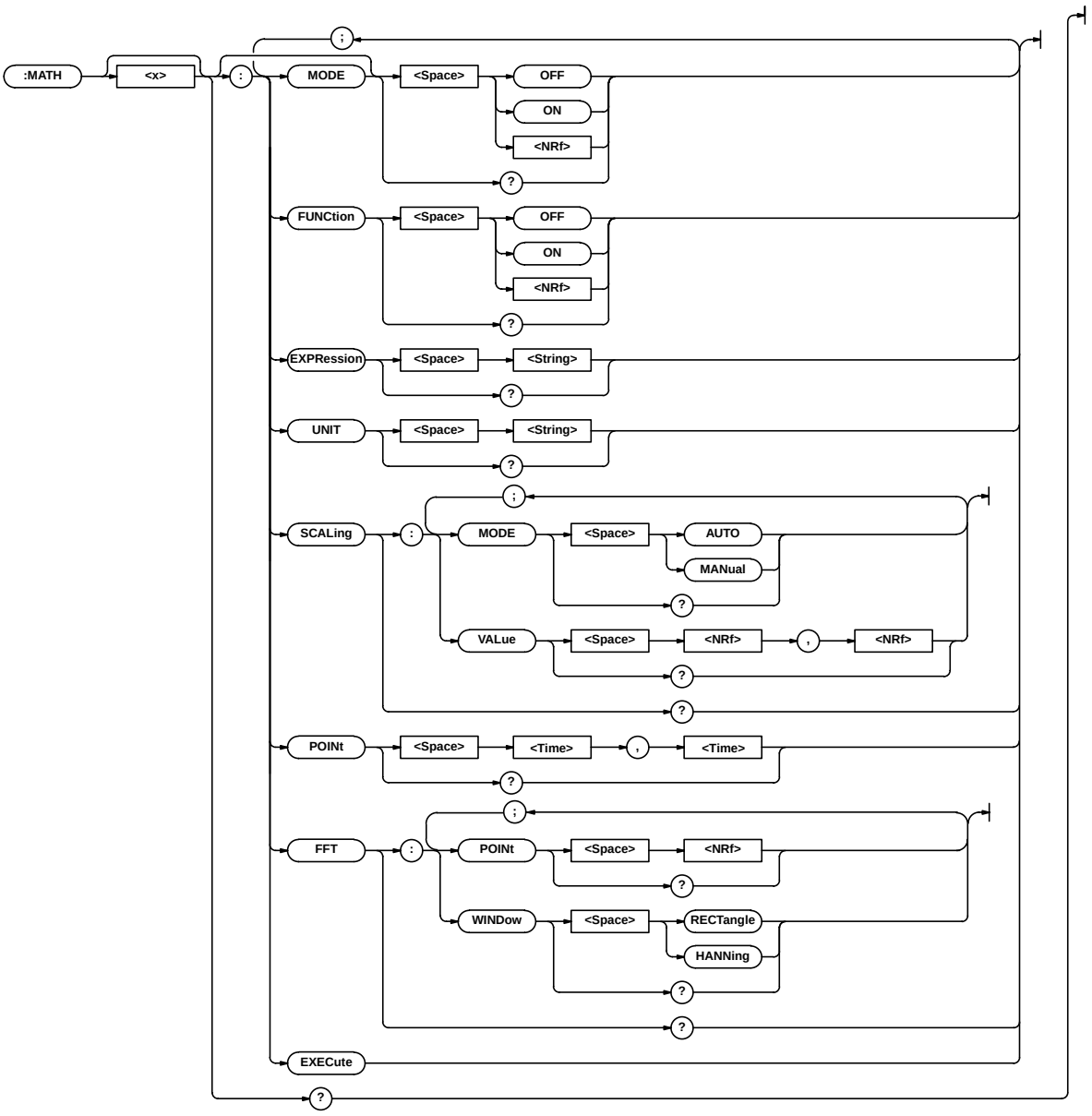
Example
```
:INPUT:POWER:VOLTAGE:RANGE:
ELEMENT1 2000V
:INPUT:POWER:VOLTAGE:RANGE:ELEMENT1?→:
INPUT:POWER:VOLTAGE:RANGE:
ELEMENT1 2.00E+03
```

Description
- The “`:CHANnel<x>:VOLTage:RANGe` (where <x> is the channel number)” command can be used to make the same setting and inquiries.
- Setting “`AUTO`” using this command is equivalent to specifying “`ON`” using the “`[:INPut][:POWer]:VOLTage:AUTO:ELEMent<x>`” command.

4.12 MATH Group

The commands in the MATH Group deal with computations.

These commands can be used to make the same settings and inquiries as when the MATH key on the front panel is pressed.



:MATH<x>?

Function Queries all settings related to computations.

Syntax
```
:MATH<x>?
<x> = 1, 2
```

Example
```
:MATH1?→:MATH:MODE 1;FUNCTION 0;
EXPRESSION "C1";UNIT "";SCALING:
MODE AUTO;VALUE 1.0000E+02,
-1.0000E+02;:MATH1:POINT 10.000E-03,
90.000E-03;FFT:POINT 1000;
WINDOW RECTANGLE
```

:MATH<x>:EXECute

Function Executes computation.

Syntax
```
:MATH<x>:EXECute
<x> = 1, 2
```

Example
```
:MATH1:EXECUTE
```

Description This command is applicable to both MATH1 and MATH2. Specifying <x> has no meaning.

:MATH<x>:EXPRession

Function Sets the equation or queries the current setting.

Syntax
```
:MATH<x>:EXPRession {<string>}
:MATH<x>:EXPRession?
<x> = 1, 2
<string> = 50 characters or less
```

Example
```
:MATH1:EXPRESSION "C1"
:MATH1:EXPRESSION?→:MATH1:
EXPRESSION "C1"
```

Description Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.

:MATH<x>:FFT?

Function Queries all settings related to the FFT.

Syntax
```
:MATH<x>:FFT?
<x> = 1, 2
```

Example
```
:MATH1:FFT?→:MATH1:FFT:POINT 1000;
WINDOW RECTANGLE
```

Description This command is applicable to both MATH1 and MATH2. Specifying <x> has no meaning.

:MATH<x>:FFT:POINt

Function Sets the number of points for the FFT or queries the current setting.

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
```

Description This command is applicable to both MATH1 and MATH2. Specifying <x> has no meaning.

:MATH<x>:FFT:WINDow

Function Sets the window function for the FFT or queries the current setting.

Syntax
```
:MATH<x>:FFT:WINDow {RECTangle|HANNing}
:MATH<x>:FFT:WINDow?
<x> = 1, 2
```

Example
```
:MATH1:FFT:WINDOW RECTANGLE
:MATH1:FFT:WINDOW?→:MATH1:FFT:
WINDOW RECTANGLE
```

Description This command is applicable to both MATH1 and MATH2. Specifying <x> has no meaning.

:MATH<x>:FUNCtion

Function Enables/disables the computation function or queries the current setting.

Syntax
```
:MATH<x>:FUNCtion {<Boolean>}
:MATH<x>:FUNCtion?
<x> = 1, 2
```

Example
```
:MATH1:FUNCTION ON
:MATH1:FUNCTION?→:MATH1:FUNCTION 1
```

Description The ":DISPlay:WAVE:MATH<x>" command can be used to make the same settings and inquiries.

:MATH<x>[:MODE]

Function Turns ON/OFF the computation or queries the current setting.

Syntax
```
:MATH<x>[:MODE] {<Boolean>}
:MATH<x>:MODE?
<x> = 1, 2
```

Example
```
:MATH1:MODE ON
:MATH1:MODE?→:MATH1:MODE 1
```

Description This command is applicable to both MATH1 and MATH2. Specifying <x> has no meaning.

:MATH<x>:POINt

Function Sets the start and end points of the computation or queries the current setting.

Syntax
```
:MATH<x>:POINt {<time>,<time>|<NRf>,
<NRf>}
:MATH<x>:POINt?
<x> = 1, 2
<time> = 0 to (OBSERVATION TIME) (during
         the normal measurement mode,
         when Time Base = Internal)
<NRf> = 0 to Record length (when Time
        Base = Internal, or during the
        harmonic measurement mode)
```

Example
```
:MATH1:POINT 10MS,90MS
:MATH1:POINT?→:MATH1:POINT 10.000E-03,
90.000E-03
```

Description
- Set the start point, then the end point.
- The range and resolution of <time> depends on the observation time.
- This command is applicable to both MATH1 and MATH2. Specifying <x> has no meaning.
- Specify <NRf> in terms of sampled data points. The range is from 0 to the record length.

:MATH<x>:SCALing?

Function Queries all settings related to converting the scale.

Syntax
```
:MATH<x>:SCALing?
<x> = 1, 2
```

Example
```
:MATH1:SCALING?→:MATH1:SCALING:
MODE AUTO;VALUE 0.1000,0.0000
```

:MATH<x>:SCALing:MODE

Function Sets the converting the scale or queries the current setting.

Syntax
```
:MATH<x>:SCALing:MODE {AUTO|MANual}
:MATH<x>:SCALing:MODE?
<x> = 1, 2
```

Example
```
:MATH1:SCALING:MODE AUTO
:MATH1:SCALING:MODE?→:MATH1:SCALING:
MODE AUTO
```

:MATH<x>:SCALing:VALue

Function Sets the upper and lower limits for manual scaling or queries the current setting.

Syntax
```
:MATH<x>:SCALing:VALue {<NRf>,<NRf>}
:MATH<x>:SCALing:VALue?
<x> = 1, 2
<NRf> = -9.9999E+30 to 9.9999E+30
```

Example
```
:MATH1:SCALING:VALUE 100,-100
:MATH1:SCALING:VALUE?→:MATH1:SCALING:
VALUE 1.0000E+02,-1.0000E+02
```

Description Set the upper limit, then the lower limit.

:MATH<x>:UNIT

Function Sets the unit to attach to the computed result or queries the current setting.

Syntax
```
:MATH<x>:UNIT {<string>}
:MATH<x>:UNIT?
<x> = 1, 2
<string> = 8 characters or less
```

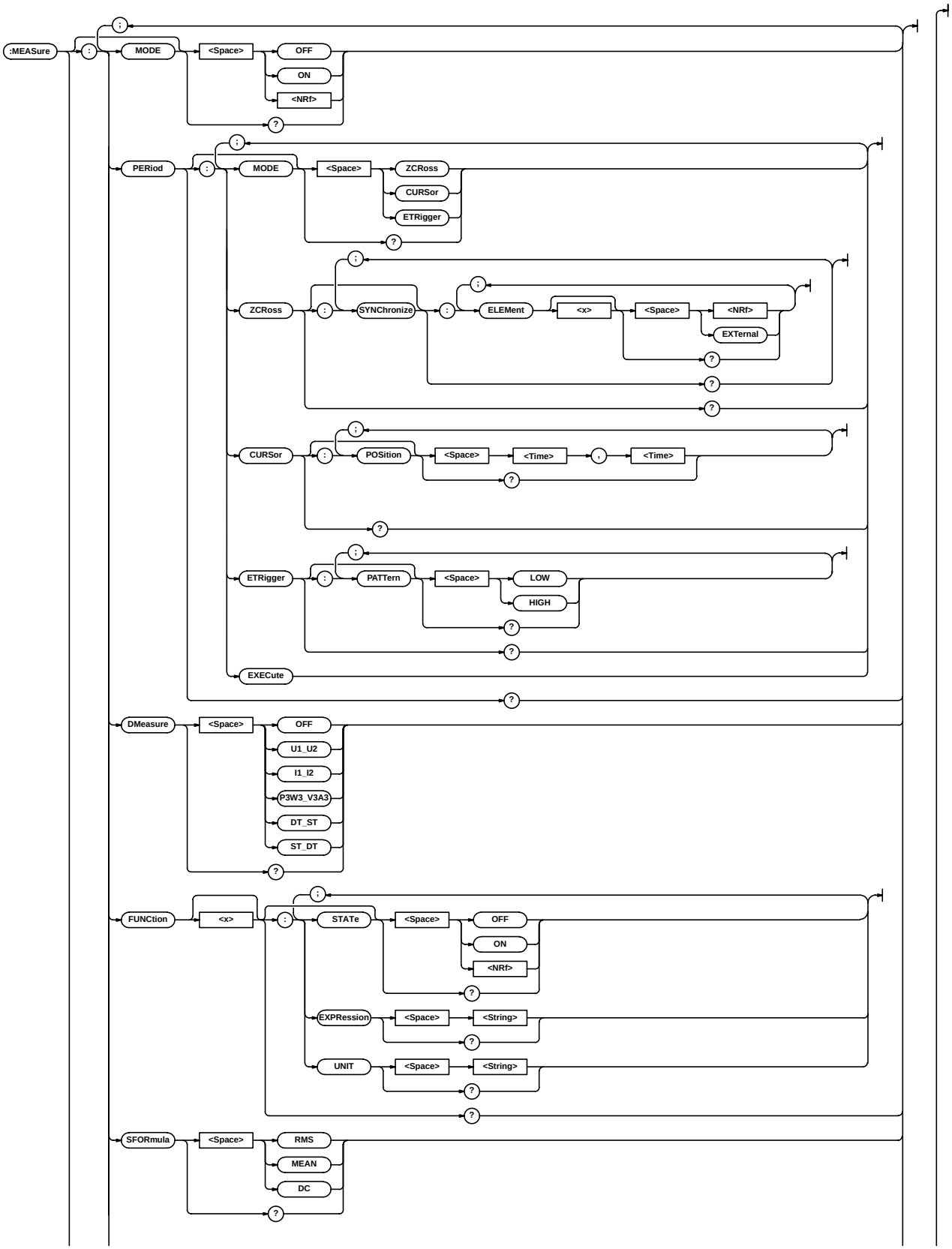
Example
```
:MATH1:UNIT ""
:MATH1:UNIT?→:MATH1:UNIT ""
```

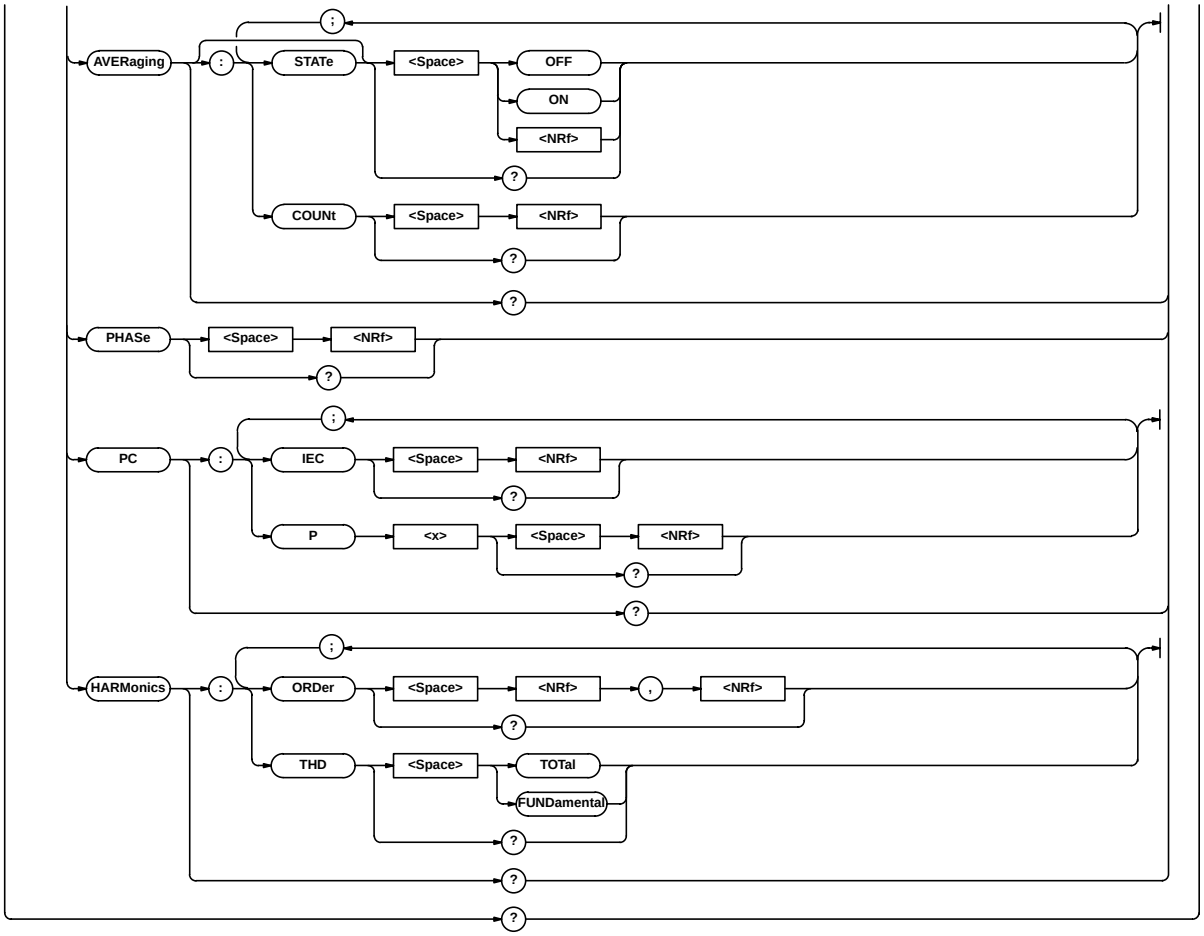
Description
- Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.
- This command does not affect the computation results in any way.

4.13 MEASure Group

The commands in the MEASure Group deal with measurements.

These commands can be used to make the same settings and inquiries as when the MEASURE key on the front panel is pressed.





:MEASure?

Function Queries all settings related to measurements.

Syntax :MEASure?

Example :MEASURE?→:MEASURE:MODE 1;PERIOD:
MODE ZCROSS;ZCROSS:SYNCHRONIZE:
ELEMENT1 2;ELEMENT2 4;ELEMENT3 6;
ELEMENT4 8;:MEASURE:DMEASURE OFF;
FUNCTION1:STATE 0;EXPRESSION "URMS(E1)";
UNIT "";:MEASURE:FUNCTION2:STATE 0;
EXPRESSION "URMS(E2)";UNIT "";:MEASURE:
FUNCTION3:STATE 0;EXPRESSION "URMS(E3)";
UNIT "";:MEASURE:FUNCTION4:STATE 0;
EXPRESSION "URMS(E4)";UNIT "";:MEASURE:
SFORMULA RMS;AVERAGING:STATE 1;COUNT 4;:
MEASURE:PHASE 180;PC:IEC 1976;P1 0.5000;
P2 0.5000

:MEASure:AVERaging?

Function Queries all settings related to averaging.

Syntax :MEASure:AVERaging?

Example :MEASURE:AVERAGING?→:MEASURE:AVERAGING:
STATE 1;COUNT 4

:MEASure:AVERaging:COUNt

Function Sets the number of averaging counts or queries the current setting.

Syntax :MEASure:AVERaging:COUNt {<NRf>}
:MEASure:AVERaging:COUNt?
<NRf> = 2, 4, 8, 16, 32, 64

Example :MEASURE:AVERAGING:COUNT 4
:MEASURE:AVERAGING:COUNT?→:MEASURE:
AVERAGING:COUNT 4

:MEASure:AVERaging[:STATe]

Function Turns ON/OFF the averaging function or queries the current setting.

Syntax :MEASure:AVERaging[:STATe] {<Boolean>}
:MEASure:AVERaging:STATe?

Example :MEASURE:AVERAGING:STATE ON
:MEASURE:AVERAGING:STATE?→:MEASURE:
AVERAGING:STATE 1

:MEASure:DMeasure

Function Sets the delta computation or queries the current setting.

Syntax
```
:MEASure:DMeasure {OFF|U1_U2|I1_I2|
P3W3_V3A3|DT_ST|ST_DT}
:MEASure:DMeasure?
```

Example
```
:MEASURE:DMEASURE OFF
:MEASURE:DMEASURE?→:MEASURE:
DMEASURE OFF
```

Description The following selection are available.

OFF	= Does not perform delta computation.
U1_U2	= u1-u2
I1_I2	= i1-i2
P3W3_V3A3	= 3P3W-to-3V3A transformation
DT_ST	= Delta-to-Star transformation
ST_DT	= Star to Delta transformation

:MEASure:FUNCtion<x>?

Function Queries all settings related to the user-defined function.

Syntax
```
:MEASure:FUNCtion<x>?
<x> = 1 to 4
```

Example
```
:MEASURE:FUNCTION1?→:MEASURE:FUNCTION1:
STATE 1;EXPRESSION "URMS(E1)";UNIT ""
```

:MEASure:FUNCtion<x>:EXPRession

Function Sets the equation for the user-defined function or queries the current setting.

Syntax
```
:MEASure:FUNCtion<x>:
EXPRession {<string>}
:MEASure:FUNCtion<x>:EXPRession?
<x> = 1 to 4
<string> = 50 characters or less
```

Example
```
:MEASURE:FUNCTION1:EXPRESSION "URMS(E1)"
:MEASURE:FUNCTION1:EXPRESSION?→:
MEASURE:FUNCTION1:EXPRESSION "URMS(E1)"
```

Description Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.

:MEASure:FUNCtion<x>[:STATE]

Function Enable/disable the user-defined function or queries the current setting.

Syntax
```
:MEASure:FUNCtion<x>:[:
STATE] {<Boolean>}
:MEASure:FUNCtion<x>:STATE?
<x> = 1 to 4
```

Example
```
:MEASURE:FUNCTION1:STATE ON
:MEASURE:FUNCTION1:STATE?→:MEASURE:
FUNCTION1:STATE 1
```

:MEASure:FUNCtion<x>:UNIT

Function Sets the unit to attach to the computed result of the user-defined function or queries the current setting.

Syntax
```
:MEASure:FUNCtion<x>:UNIT {<string>}
:MEASure:FUNCtion<x>:UNIT?
<x> = 1 to 4
<string> = 8 characters or less
```

Example
```
:MEASURE:FUNCTION1:UNIT ""
:MEASURE:FUNCTION1:?→:MEASURE:
FUNCTION1:UNIT ""
```

Description
- Characters and symbols other than the ones displayed on the keyboard on the screen cannot be used.
- This command does not affect the computation results in any way.

:MEASure:HARMonics?

Function Queries all settings related to the measurement during harmonic measurement.

Syntax
```
:MEASure:HARMonics?
```

Example
```
:MEASURE:HARMONICS?→:MEASURE:HARMONICS:
ORDER 0,100;THD TOTAL
```

:MEASure:HARMonics:ORDer

Function Sets the minimum and maximum harmonic orders to be analyzed during harmonic measurement or queries the current setting.

Syntax
```
:MEASure:HARMonics:ORDer {<NRf>,<NRf>}
:MEASure:HARMonics:ORDer?
First <NRf> = 0, 1 (minimum harmonic
              order under analysis)
Second <NRf> = 1 to 500 (maximum
               harmonic order under
               analysis)
```

Example
```
:MEASURE:HARMONICS:ORDER 0,100
:MEASURE:HARMONICS:ORDER?→:MEASURE:
HARMONICS:ORDER 0,100
```

:MEASure:HARMonics:THD

Function Sets the equation used to determine the THD (total harmonic distortion) during harmonic measurement or queries the current setting.

Syntax
```
:MEASure:HARMonics:THD {TOTal|
FUNDamental}
:MEASure:HARMonics:THD?
```

Example
```
:MEASURE:HARMONICS:THD TOTAL
:MEASURE:HARMONICS:THD?→:MEASURE:
HARMONICS:THD TOTAL
```

:MEASure[:MODE]

Function Turns ON/OFF the measurement/computation or queries the current setting.

Syntax
```
:MEASure[:MODE] {<Boolean>}
:MEASure:MODE?
```

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

:MEASure:PC?

Function Queries all settings related to determination of Pc (Corrected Power).

Syntax :MEASure:PC?

Example :MEASURE:PC?→:MEASURE:PC:IEC 1976;P1 0.5000;P2 0.5000

:MEASure:PC:IEC

Function Sets the equation used to determine the Pc (Corrected Power) or queries the current setting.

Syntax :MEASure:PC:IEC {<NRf>}
:MEASure:PC:IEC?
<NRf> = 1976, 1993

Example :MEASURE:PC:IEC 1976
:MEASURE:PC:IEC?→:MEASURE:PC:IEC 1976

Description Specifies the year of the issue of the IEC76-1 in which the equation used to determine the Pc is given.

:MEASure:PC:P<x>

Function Pc(Corrected Power) Sets the parameters used to determine the Pc (Corrected Power) or queries the current setting.

Syntax :MEASure:PC:P<x> {<NRf>}
:MEASure:PC:P<x>?
<x> = 1, 2
<NRf> = 0.0001 to 9.9999

Example :MEASURE:PC:P1 0.5
:MEASURE:PC:P1?→:MEASURE:PC:P1 0.5000

Description This parameter is used when ":MEASure:PC:IEC" is set to "1976(IEC76-1(1976), IEEE C57.12.90-1993)."

:MEASure:PERiod?

Function Queries all settings related to the computation period.

Syntax :MEASure:PERiod?

Example :MEASURE:PERIOD?→:MEASURE:PERIOD:MODE ZCROSS;ZCROSS:SYNCHRONIZE:ELEMENT1 2;ELEMENT2 4;ELEMENT3 6;ELEMENT4 8

:MEASure:PERiod:CURSor?

Function Queries all settings when specifying the computation period with the cursors.

Syntax :MEASure:PERiod:CURSor?

Example :MEASURE:PERIOD:CURSOR?→:MEASURE:PERIOD:CURSOR:POSITION 0.000E-03,90.000E-03

:MEASure:PERiod:CURSor[:POSition]

Function Sets the computation period when specifying the period with the cursors or queries the current setting.

Syntax :MEASure:PERiod:CURSor[:POSition] {<time>,<time>|<NRf>,<NRf>}
:MEASure:PERiod:CURSor:POSition?
<time> = 0 to (OBSERVATION TIME)(During the normal measurement mode, when Time Base=Internal)
<NRf> = 0 to (Record Length)(During the normal measurement mode, when Time Base=External)
<NRf> = 0 to (Record Length-8192)(During the harmonic measurement mode)

Example :MEASURE:PERIOD:CURSOR:POSITION 0,90MS
:MEASURE:PERIOD:CURSOR:POSITION?→:MEASURE:PERIOD:CURSOR:POSITION 0.000E-03,90.000E-03

Description
- Set the start point, then the end point.
- Set only the start point during the harmonic measurement mode. (The end point cannot be specified since it is fixed to start point +8192.)
- The range and resolution of <time> depends on the observation time.
- Specify <NRf> in terms of sampled data points. The range is from 0 to the record lengh. The record length varies depending on the extended memory options.
- The range of the start point of computation (<NRf>) is from 0 to the record length-8192 during the harmonic measurement mode.

:MEASure:PERiod:ETRigger?

Function Queries all settings when using the external trigger signal to determine the computation period.

Syntax :MEASure:PERiod:ETRigger?

Example :MEASURE:PERIOD:ETRIGGER?→:MEASURE:PERIOD:ETRIGGER:PATTERN LOW

:MEASure:PERiod:ETRigger[:PATTern]

Function Sets the pattern that is used when determining the computation period with the external trigger signal or queries the current setting.

Syntax :MEASure:PERiod:ETRigger[:PATTern] {LOW|HIGH}
:MEASure:PERiod:ETRigger:PATTern?

Example :MEASURE:PERIOD:ETRIGGER:PATTERN LOW
:MEASURE:PERIOD:ETRIGGER:PATTERN?→:MEASURE:PERIOD:ETRIGGER:PATTERN LOW

:MEASure:PERiod:EXECute

Function Executes the computation.
Syntax :MEASure:PERiod:EXECute
Example :MEASURE:PERIOD:EXECUTE

:MEASure:PERiod[:MODE]

Function Sets the method used to specify the computation period or queries the current setting.
Syntax :MEASure:PERiod[:MODE] {ZCRoss|CURSor|ETRigger}
:MEASure:PERiod:MODE?
Example :MEASURE:PERIOD:MODE ZCROSS
:MEASURE:PERIOD:MODE?→:MEASURE:PERIOD:MODE ZCROSS
Description This command is valid during the normal measreument mode. It is fixed to CURSor during the harmonic measurement mode.

:MEASure:PERiod:ZCRoss?

Function Queries all settings when using the zero crossing detection to determine the computation period.
Syntax :MEASure:PERiod:ZCRoss?
Example :MEASURE:PERIOD:ZCROSS?→:MEASURE:PERIOD:ZCROSS:SYNCHRONIZE:ELEMENT1 2;ELEMENT2 4;ELEMENT3 6;ELEMENT4 8

:MEASure:PERiod:ZCRoss:SYNChronize?

Function Sets the synchronizing source for all elements when using the zero crossing detection to determine the computation period.
Syntax :MEASure:PERiod:ZCRoss:SYNChronize?
Example :MEASURE:PERIOD:ZCROSS:SYNCHRONIZE?→:MEASURE:PERIOD:ZCROSS:SYNCHRONIZE:ELEMENT1 2;ELEMENT2 4;ELEMENT3 6;ELEMENT4 8

:MEASure:PERiod:ZCRoss[:SYNChronize]:ELEMent<x>

Function Sets the synchronizing source for each element when using the zero crossing detection to determine the computation period.
Syntax :MEASure:PERiod:ZCRoss[:SYNChronize]:ELEMent<x> {<NRf>|EXTernal}
:MEASure:PERiod:ZCRoss[:SYNChronize]:ELEMent<x>?
<x> = 1 to 4
<NRf> = 1 to 8
EXTernal = External Clock
Example :MEASURE:PERIOD:ZCROSS:SYNCHRONIZE:ELEMENT1 2
:MEASURE:PERIOD:ZCROSS:SYNCHRONIZE:ELEMENT1?→:MEASURE:PERIOD:ZCROSS:SYNCHRONIZE:ELEMENT1 2

:MEASure:PHASe

Function Sets the display format of the phase difference or queries the current setting.
Syntax :MEASure:PHASe {<NRf>}
:MEASure:PHASe?
<NRf> = 180, 360
Example :MEASURE:PHASE 180
:MEASURE:PHASE?→:MEASURE:PHASE 180
Description “180” and “360” denote 0 to ±180° (Lead/Lag) and 0 to 360°, respectively.

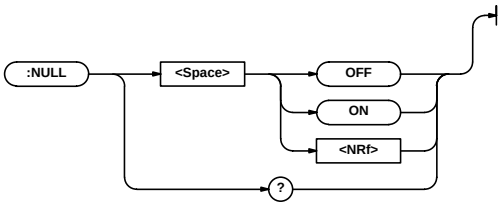
:MEASure:SFORmula

Function Sets the equation used to determine S (apparent power) or queries the current setting.
Syntax :MEASure:SFORmula {RMS|MEAN|DC}
:MEASure:SFORmula?
Example :MEASURE:SFORMULA RMS
:MEASURE:SFORMULA?→:MEASURE:SFORMULA RMS
Description The equation corresponding to each selection is as follows:
RMS : S = Urms * Irms
MEAN : S = Umean * Imean
DC : S = Udc * Idc

4.14 NULL Group

The commands in the NULL Group deal with the NULL function.

These commands can be used to make the same settings and inquiries as when the NULL key on the front panel is pressed.



:NULL

Function Turns ON/OFF the NULL function or queries the current setting.

Syntax
```
:NULL {<Boolean>}
:NULL?
```

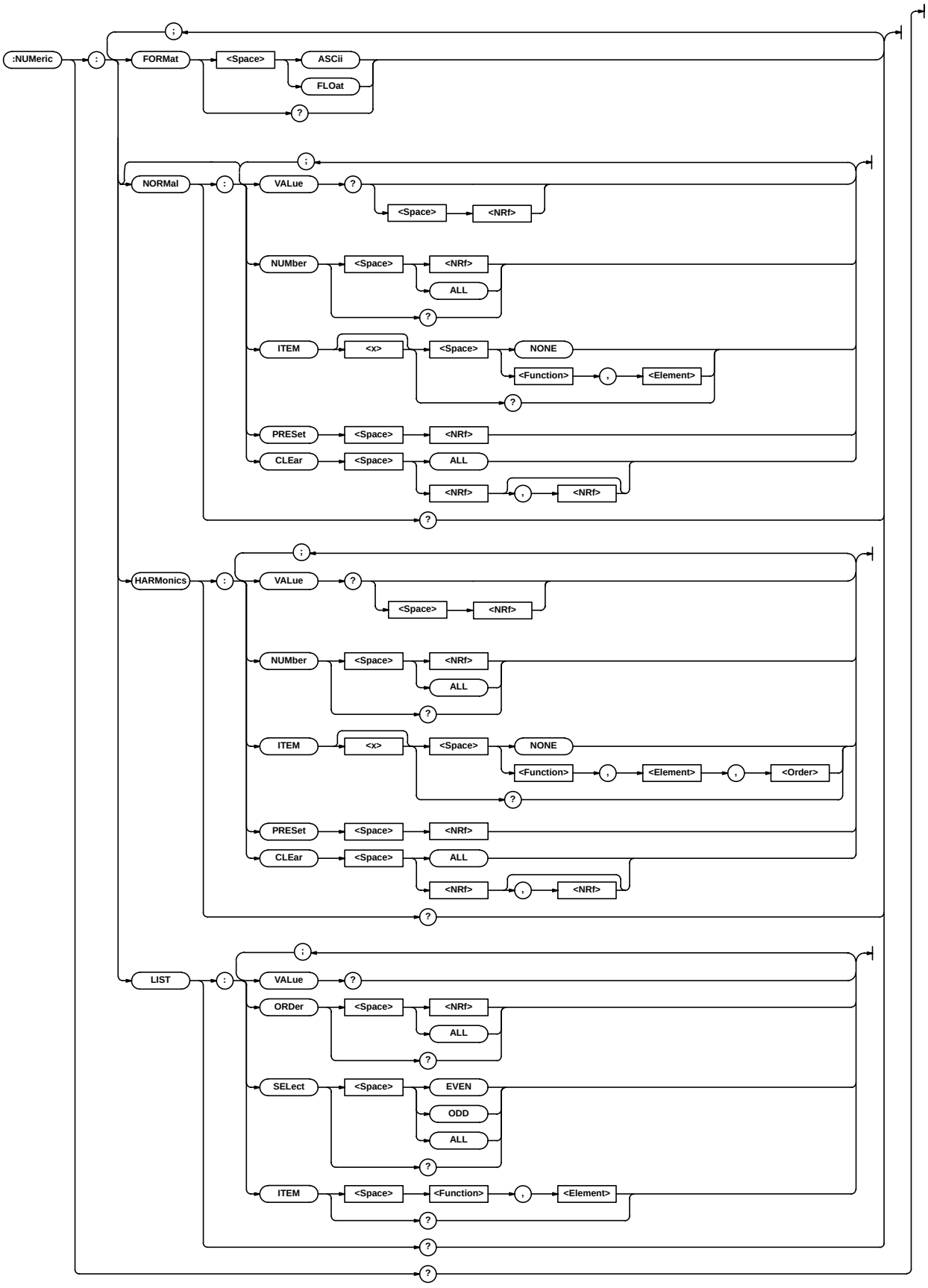
Example
```
:NULL ON
:NULL?→:NULL 1
```

Description When turn ON, the applied voltage/current is set as the reference (0) and all succeeding measured values will be based on this reference.

4.15 NUMeric Group

The commands in the NUMeric Group deal with the output of numerical data.

There are no front-panel keys that correspond to the commands in this group.



:NUMeric?

Function Queries all settings related to the numerical data output.

Syntax :NUMeric?

Example :NUMERIC?→:NUMERIC:FORMAT ASCII;NORMAL:NUMBER 8;ITEM1 URMS,1;ITEM2 UMN,1;ITEM3 UDC,1;ITEM4 UAC,1;ITEM5 IRMS,1;ITEM6 IMN,1;ITEM7 IDC,1;ITEM8 IAC,1

:NUMeric:FORMat

Function Sets the format of the numerical data that are sent using the ":NUMeric:{NORMal|HARMonics|LIST}:VALue?" command or queries the current setting.

Syntax :NUMeric:FORMat {ASCii|FLOat}
:NUMeric:FORMat?

Example :NUMERIC:FORMAT ASCII
:NUMERIC:FORMAT?→:NUMERIC:FORMAT ASCII

Description The format of the numerical data that is output depends on the ":NUMeric:FORMat" setting.

(1) When set to "ASCii"
The physical values are output in <NR3> format. Each item of data is separated by a comma.

(2) When set to "FLOat"
A 6-byte header ("#40060" for example) is added to begginning of numeric data block. The header is followed by physical values in IEEE single precision floating point format (4 bytes).

The byte order of each item of data is MSB First.

:NUMeric:HARMonics?

Function Queries all settings related to the numerical data output during harmonic measurement.

Syntax :NUMeric:HARMonics?

Example :NUMERIC:HARMONICS?→:NUMERIC:HARMONICS:NUMBER 5;ITEM1 U,1,1;ITEM2 I,1,1;ITEM3 P,1,1;ITEM4 S,1,1;ITEM5 Q,1,1

:NUMeric:HARMonics:CLEar

Function Clears the numerical data output items during harmonic measurement (sets them to NONE).

Syntax :NUMeric:HARMonics:CLEar {ALL|<NRf>[,<NRf>]}
First <NRf> = 1 to 255 (First item number to clear)
Second <NRf> = 1 to 255 (Last item number to clear)

Example :NUMERIC:HARMONICS:CLEAR ALL

Description If the second <NRf> is omitted, output items from the first item number to the end item (255) are cleared.

:NUMeric:HARMonics:ITEM<x>

Function Sets the numerical data output items during harmonic measurement or queries the current setting.

Syntax :NUMeric:HARMonics:ITEM<x> {NONE|<Function>,<Element>,<Order>}
:NUMeric:HARMonics:ITEM<x>?
<x> = 1 to 255(item number)
NONE = no display item
<Function> = {U|I|P|S|Q|...} (See the function selection list on pate 4-32 (2))
<Element> = {<NRf>|SIGMA|SIGMB} (<NRf> = 1 to 4)
<Order> = {TOTal|DC|<NRf>} (<NRf> = 1 to 500)

Example :NUMERIC:HARMONICS:ITEM1 U,1,1
:NUMERIC:HARMONICS:ITEM1?→:NUMERIC:HARMONICS:ITEM1 U,1,1

:NUMeric:HARMonics:NUMber

Function Sets the number of numerical data that are sent using the ":NUMeric:HARMonics:VALue?" command or queries the current setting.

Syntax :NUMeric:HARMonics:NUMber {<NRf>|ALL}
:NUMeric:HARMonics:NUMber?
<NRf> = 1 to 255(ALL)

Example :NUMERIC:HARMONICS:NUMBER 8
:NUMERIC:HARMONICS:NUMBER→:NUMERIC:HARMONICS:NUMBER 8

Description If the parameter is omitted in the ":NUMeric:HARMonics:VALue?" command, 1 to (the specified value) of numerical data are output in order.

:NUMeric:HARMonics:PRESet

Function Sets the numerical data output items to a preset pattern during harmonic measurement.

Syntax `:NUMeric:HARMonics:PRESet {<NRf>}`
`<NRf> = 1 to 4`

Example `:NUMERIC:HARMONICS:PRESET 1`

Description For information related to the output items that are set to preset values, see "A list of numerical data output items that are preset" on page 4-67.

:NUMeric:HARMonics:VALue?

Function Queries the numerical data during harmonic measurement.

Syntax `:NUMeric:HARMonics:VALue? {|<NRf>}`
`<NRf> = 1 to 255(item number)`

Example
Example when <NRf> is specified
`:NUMERIC:HARMONICS:VALUE? 1→104.75E+00`
Example when <NRf> is omitted
`:NUMERIC:HARMONICS:VALUE?→104.75E+00, 0.9584E+00, 72.01E+00, (omit), 50.086E+00`
When ":NUMeric:FORMat" is set to FLOat
`:NUMeric:NORMAL:VALUE?→#4 (Number of bytes, 4 digits) (Series of data bytes)`

Description
- When <NRf> is specified, only the numerical data of that item number are output.
- When <NRf> is omitted, the numerical data from 1 to the item number specified using the ":NUMeric:HARMonics:NUMber" command are output in order.
- If the item of the specified number is set to "NONE" or if no numerical data exist, the item will output error data. "NAN" (Not A Number) is returned when ":NUMeric:FORMat" is set to "ASCii." 9.91E+37 is returned if it is set to "FLOat."
- In addition, if the numerical data are erroneous (the display is "Error" or "--OF--"), "INF" (infinity) is returned when ":NUMeric:FORMat" is set to "ASCii." 9.9E+37 is returned if it is set to "FLOat."

:NUMeric:LIST?

Function Queries all settings related to the output of the numerical list data during harmonic measurement.

Syntax `:NUMeric:LIST?`

Example `:NUMERIC:LIST?→:NUMERIC:LIST:ORDER 100;SELECT ALL;ITEM U,1`

:NUMeric:LIST:ITEM

Function Sets the output items of the numerical list data during harmonic measurement or queries the current setting.

Syntax `:NUMeric:LIST:ITEM {<Function>,<Element>}`
`:NUMeric:LIST:ITEM?`
`<Function> = {U|I|P|S|Q|LAMBda|PHI|PHIU|PHII}` (See the function selection list on page 4-32 (3))
`<Element> = {<NRf>|SIGMA|SIGMB}` (<NRf>=1 to 4)

Example `:NUMERIC:LIST:ITEM U,1`
`:NUMERIC:LIST:ITEM?→:NUMERIC:LIST:ITEM U,1`

:NUMeric:LIST:ORDer

Function Sets the maximum harmonic order of the numerical list data to output during harmonic measurement or queries the current setting.

Syntax `:NUMeric:LIST:ORDer {<NRf>|ALL}`
`:NUMeric:LIST:ORDer?`

Example `:NUMERIC:LIST:ORDER 100`
`:NUMERIC:LIST:ORDER?→:NUMERIC:LIST:ORDER 100`

:NUMeric:LIST:SELect

Function Sets the output components of the numerical list data during harmonic measurement or queries the current setting.

Syntax `:NUMeric:LIST:SELect {EVEN|ODD|ALL}`
`:NUMeric:LIST:SELect?`

Example `:NUMERIC:LIST:SELECT ALL`
`:NUMERIC:LIST:SELECT?→:NUMERIC:LIST:SELECT ALL`

:NUMeric:LIST:VALue?

Function Queries the numerical list data during harmonic measurement.

Syntax `:NUMeric:LIST:VALue?`

Example `:NUMERIC:LIST:VALUE?→103.58E+00, 0.00E+00,103.53E+00,0.09E+00,2.07E+00, 0.04E+00,(omit),0.01E+00,0.01E+00`

Description The numerical data of TOTal, DC, and 1st order to ":NUMeric:LIST:ORDer" are output.

:NUMeric:NORMal?

Function Queries all settings related to the numerical data output during normal measurement.

Syntax `:NUMeric:NORMal?`

Example `:NUMERIC:NORMAL?→:NUMERIC:NORMAL:NUMBER 8;ITEM1 URMS,1;ITEM2 UMN,1;ITEM3 UDC,1;ITEM4 UAC,1;ITEM5 IRMS,1;ITEM6 IMN,1;ITEM7 IDC,1;ITEM8 IAC,1`

:NUMeric[:NORMal]:CLEar

Function Clears the numerical data output items during normal measurement (Sets them to "NONE").

Syntax
```
:NUMeric[:NORMal]:CLEar {ALL|
<NRf>[,<NRf>]}
First <NRf> = 1 to 255
              (First item number to
              clear)
Second <NRf> = 1 to 255
               (Last item number to
               clear)
```

Example
```
:NUMERIC:NORMAL:CLEAR ALL
```

Description If the second <NRf> is omitted, output items from the first item number to the end item (255) are cleared.

:NUMeric[:NORMal]:ITEM<x>

Function Sets the numerical data output items during normal measurement or queries the current setting.

Syntax
```
:NUMeric[:NORMal]:ITEM<x> {NONE|
<Function>,<Element>}
:NUMeric[:NORMal]:ITEM<x>?
<x> = 1 to 255(item number)
NONE = No output items
<Function> = {URMS|UMN|UDC|UAC|IRMS|...}
             (See the function selection
             list on page 4-31 (1))
<Element>  = {<NRf>|SIGMA|SIGMB}(<NRf> =
             1 to 4)
```

Example
```
:NUMERIC:NORMAL:ITEM1 URMS,1
:NUMERIC:NORMAL:ITEM1?→:NUMERIC:NORMAL:
ITEM1 URMS,1
```

:NUMeric[:NORMal]:NUMber

Function Sets the number of numerical data that are sent using the ":NUMeric:NORMal:VALue?" command or queries the current setting.

Syntax
```
:NUMeric[:NORMal]:NUMber {<NRf>|ALL}
:NUMeric[:NORMal]:NUMber?
<NRf> = 1 to 255(ALL)
```

Example
```
:NUMERIC:NORMAL:NUMBER 8
:NUMERIC:NORMAL:NUMBER→:NUMERIC:NORMAL:
NUMBER 8
```

Description If the parameter is omitted in the ":NUMeric:HARMonics:VALue?" command, 1 to (the specified value) of numerical data are output in order.

:NUMeric[:NORMal]:PRESet

Function Sets the numerical data output items to a preset pattern during normal measurement.

Syntax
```
:NUMeric[:NORMal]:PRESet {<NRf>}
<NRf> = 1 to 4
```

Example
```
:NUMERIC:NORMAL:PRESET 1
```

Description For information related to the output items that are set to preset values, see "A list of numerical data output items that are preset" on next page.

:NUMeric[:NORMal]:VALue?

Function Queries the numerical data during normal measurement.

Syntax
```
:NUMeric[:NORMal]:VALue? {<NRf>}
<NRf> = 1 to 255(item number)
```

Example
```
Example when <NRf> is specified
:NUMERIC:NORMAL:VALUE? 1→104.75E+00
Example when <NRf> is omitted
:NUMERIC:NORMAL:VALUE?→104.75E+00,
105.02E+00, -0.38E+00, (omit),
49.868E+00
When ":NUMeric:FORMat" is set to FLOat
:NUMeric:NORMAL:VALUE?→#4 (Number of
bytes, 4 digits) (Series of data bytes)
```

Description
- When <NRf> is specified, only the numerical data of that item number are output.
- When <NRf> is omitted, the numerical data from 1 to the item number specified using the ":NUMeric:HARMonics:NUMber" command are output in order.
- If the item of the specified number is set to "NONE" or if no numerical data exist, the item will output error data. "NAN" (Not A Number) is returned when ":NUMeric:FORMat" is set to "ASCii." 9.91E+37 is returned if it is set to "FLOat."
- In addition, if the numerical data are erroneous (the display is "Error" or "--OF--"), "INF" (infinity) is returned when ":NUMeric:FORMat" is set to "ASCii." 9.9E+37 is returned if it is set to "FLOat."
- If the output item is PHI (ϕ), the result is returned in the range from 0 to 360° regardless of the display format of the phase difference specified by MEASure:PHASe.

*** A list of numerical data output items that are preset**

(1) Preset pattern of normal measurement numerical data output items

Applicable command ":NUMeric[:NORMal]:PRESet"

Pattern 1

ITEM<x>	<Function>,	<Element>
1	URMS,	1
2	IRMS,	1
3	P,	1
4	S,	1
5	Q,	1
6	LAMBda,	1
7	PHI,	1
8	FU,	1
9	FI,	1
10	NONE	
11 to 19	URMS to FI,	2
20	NONE	
21 to 29	URMS to FI,	3
30	NONE	
31 to 39	URMS to FI,	4
40	NONE	
41 to 49	URMS to FI,	SIGMA
50	NONE	
51 to 59	URMS to FI,	SIGMB
60	NONE	
61 to 255	NONE	

Pattern 2

ITEM<x>	<Function>,	<Element>
1	URMS,	1
2	UMN,	1
3	UDC,	1
4	UAC,	1
5	IRMS,	1
6	IMN,	1
7	IDC,	1
8	IAC,	1
9	P,	1
10	S,	1
11	Q,	1
12	LAMBda,	1
13	PHI,	1
14	FU,	1
15	FI,	1
16 to 30	URMS to FI,	2
31 to 45	URMS to FI,	3
46 to 60	URMS to FI,	4
61 to 75	URMS to FI,	SIGMA
76 to 90	URMS to FI,	SIGMB
91 to 255	NONE	

Pattern 3

ITEM<x>	<Function>,	<Element>
1	URMS,	1
2	UMN,	1
3	UDC,	1
4	UAC,	1
5	IRMS,	1
6	IMN,	1
7	IDC,	1
8	IAC,	1
9	P,	1
10	S,	1
11	Q,	1
12	LAMBda,	1
13	PHI,	1
14	FU,	1
15	FI,	1
16	UPPeak,	1
17	UMPeak,	1
18	IPPeak,	1
19	IMPeak,	1
20	NONE	
21 to 39	URMS to IMPeak,	2
40	NONE	
41 to 59	URMS to IMPeak,	3
60	NONE	
61 to 79	URMS to IMPeak,	4
80	NONE	
81 to 99	URMS to IMPeak,	SIGMA
100	NONE	
101 to 119	URMS to IMPeak,	SIGMB
120	NONE	
121 to 255	NONE	

Pattern 4

ITEM<x>	<Function>,	<Element>
1	URMS,	1
2	UMN,	1
3	UDC,	1
4	UAC,	1
5	IRMS,	1
6	IMN,	1
7	IDC,	1
8	IAC,	1
9	P,	1
10	S,	1
11	Q,	1
12	LAMBda,	1
13	PHI,	1
14	FU,	1
15	FI,	1
16	UPPeak,	1
17	UMPeak,	1
18	IPPeak,	1
19	IMPeak,	1
20	CFU,	1
21	CFI,	1
22	FFU,	1
23	FFI,	1
24	Z,	1
25	RS,	1
26	XS,	1

27	RP,	1
28	XP,	1
29	PC,	1
30	ETA,	1
31 to 60	URMS to ETA,	2
61 to 90	URMS to ETA,	3
91 to 120	URMS to ETA,	4
121 to 150	URMS to ETA,	SIGMA
151 to 180	URMS to ETA,	SIGMB
181 to 255	NONE	

(2) Preset pattern of harmonic measurement numerical data output items

Applicable command ":NUMeric:HARMonics:PRESet"

181 to 255 NONE

Pattern 1

ITEM<x>	<Function>,	<Element>,	<Order>
1	U,	1,	TOTal
2	I,	1,	TOTal
3	P,	1,	TOTal
4	Q,	1,	TOTal
5	U,	1,	1
6	I,	1,	1
7	P,	1,	1
8	Q,	1,	1
9	FU,	1,	(1)
10	FI,	1,	(1)
11 to 20	U to FI,	2,	TOTal to 1
21 to 30	U to FI,	3,	TOTal to 1
31 to 40	U to FI,	4,	TOTal to 1
41 to 50	U to FI,	SIGMA,	TOTal to 1
51 to 60	U to FI,	SIGMB,	TOTal to 1
61 to 255	NONE		

Pattern 2

ITEM<x>	<Function>,	<Element>,	<Order>
1	U,	1,	TOTal
2	I,	1,	TOTal
3	P,	1,	TOTal
4	S,	1,	TOTal
5	Q,	1,	TOTal
6	LAMBda,	1,	TOTal
7	U,	1,	1
8	I,	1,	1
9	P,	1,	1
10	S,	1,	1
11	Q,	1,	1
12	LAMBda,	1,	1
13	PHI,	1,	1
14	FU,	1,	(1)
15	FI,	1,	(1)
16 to 30	U to FI,	2,	TOTal to 1
31 to 45	U to FI,	3,	TOTal to 1
46 to 60	U to FI,	4,	TOTal to 1
61 to 75	U to FI,	SIGMA,	TOTal to 1
76 to 90	U to FI,	SIGMB,	TOTal to 1
91 to 255	NONE		

Pattern 3

ITEM<x>	<Function>,	<Element>,	<Order>
1	U,	1,	TOTal
2	I,	1,	TOTal
3	P,	1,	TOTal
4	S,	1,	TOTal
5	Q,	1,	TOTal
6	LAMBda,	1,	TOTal
7	U,	1,	DC(0)
8	I,	1,	DC(0)
9	P,	1,	DC(0)
10	S,	1,	DC(0)
11	Q,	1,	DC(0)
12	U,	1,	1
13	I,	1,	1
14	P,	1,	1
15	S,	1,	1
16	Q,	1,	1
17	LAMBda,	1,	1
18	PHI,	1,	1
19	FU,	1,	(1)
20	FI,	1,	(1)
21 to 40	U to FI,	2,	TOTal to 1
41 to 60	U to FI,	3,	TOTal to 1
61 to 80	U to FI,	4,	TOTal to 1
81 to 100	U to FI,	SIGMA,	TOTal to 1
101 to 120	U to FI,	SIGMB,	TOTal to 1
121 to 255	NONE		

Pattern 4

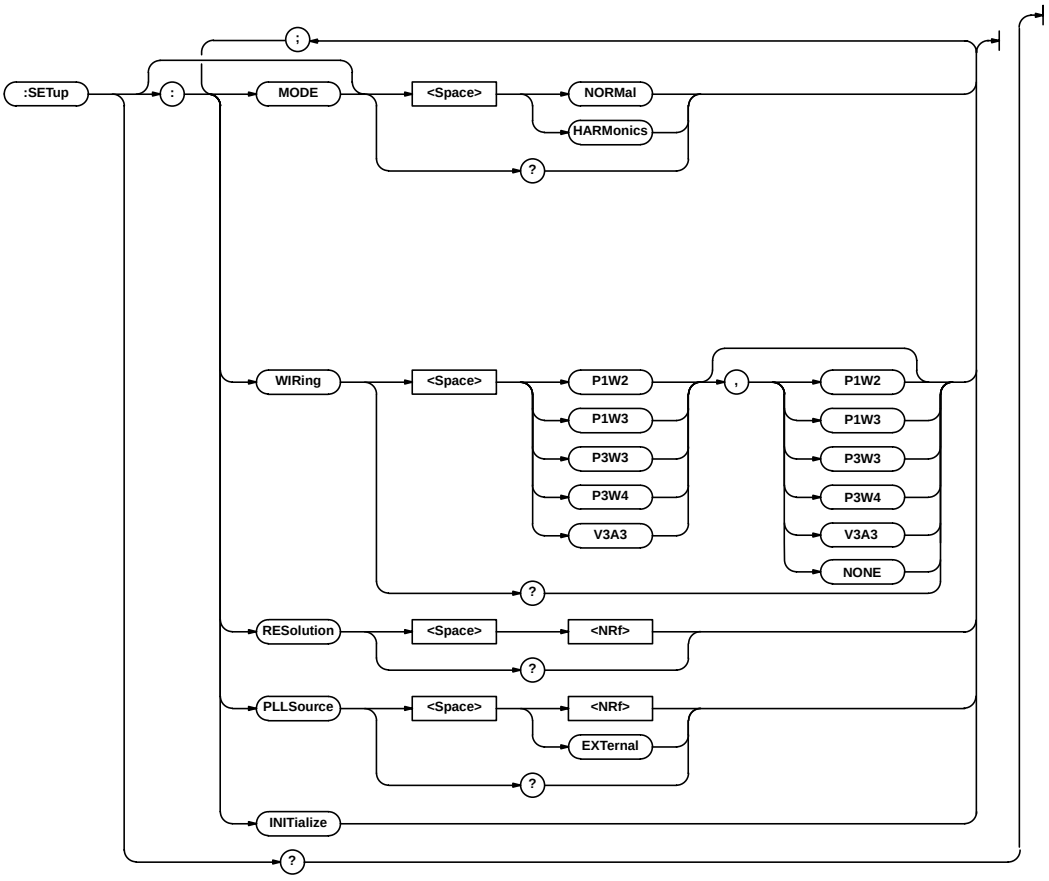
ITEM<x>	<Function>,	<Element>,	<Order>
1	U,	1,	TOTal
2	I,	1,	TOTal
3	P,	1,	TOTal
4	S,	1,	TOTal
5	Q,	1,	TOTal
6	LAMBda,	1,	TOTal
7	U,	1,	DC(0)
8	I,	1,	DC(0)
9	P,	1,	DC(0)
10	S,	1,	DC(0)
11	Q,	1,	DC(0)
12	U,	1,	1
13	I,	1,	1
14	P,	1,	1
15	S,	1,	1
16	Q,	1,	1
17	LAMBda,	1,	1
18	PHI,	1,	1
19	FU,	1,	(1)
20	FI,	1,	(1)
21	Z,	1,	1
22	RS,	1,	1
23	XS,	1,	1
24	RP,	1,	1
25	XP,	1,	1
26	UTHD,	1,	(1)

27	ITHD,	1,	(1)
28	PTHD,	1,	(1)
29	STHD,	1,	(1)
30	QTHD,	1,	(1)
31 to 60	U to QTHD,	2,	TOTal to 1
61 to 90	U to QTHD,	3,	TOTal to 1
91 to 120	U to QTHD,	4,	TOTal to 1
121 to 150	U to QTHD,	SIGMA,	TOTal to 1
151 to 180	U to QTHD,	SIGMB,	TOTal to 1
181 to 255	NONE		

4.16 SETup Group

The commands in the SETup Group deal with setting the measurement mode.

These commands can be used to make the same settings and inquiries as when the SETUP key on the front panel is pressed.



:SETup?

Function Queries all settings related to the measurement mode.

Syntax `:SETup?`

Example `:SETUP?→:SETUP:MODE NORMAL;`
`WIRING P1W2,P1W2;RESOLUTION 5`

:SETup:INITialize

Function Initializes the settings.

Syntax `:SETup:INITialize`

Example `:SETUP:INITIALIZE`

Description Resets all setup parameters except communication settings to factory default values.

:SETup[:MODE]

Function Sets the measurement mode or queries the current setting.

Syntax `:SETup[:MODE] {NORMal|HARMonics}`
`:SETup:MODE?`

Example `:SETUP:MODE NORMAL`
`:SETUP:MODE?→:SETUP:MODE NORMAL`

:SETup:PLLSource

Function Sets the PLL source during harmonic measurement or queries the current setting.

Syntax `:SETup:PLLSource {<NRf>|EXTernal}`
`:SETup:PLLSource?`
`<NRf> = 1 to 8`
`EXTernal = External clock`

Example `:SETUP:PLLSOURCE 1`
`:SETUP:PLLSOURCE?→:SETUP:PLLSOURCE 1`

:SETup:RESolution

Function Sets the number of displayed digits for numerical data or queries the current setting.

Syntax `:SETup:RESolution {<NRf>}`
`:SETup:RESolution?`
`<NRf> = 5, 6`

Example `:SETUP:RESOLUTION 5`
`:SETUP:RESOLUTION?→:SETUP:RESOLUTION 5`

:SETup:WIRing

Function Sets the wiring method or queries the current setting.

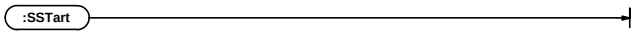
Syntax
```
:SETup:WIRing {(P1W2|P1W3|P3W3|P3W4|
[,(P1W2|P1W3|P3W3|P3W4|V3A3|NONE)]}
:SETup:WIRing?
P1W2 = single-phase two-wire system
P1W3 = single-phase three-wire system
P3W3 = three-phase three-wire system
P3W4 = three-phase four-wire system
V3A3 = three-voltage three-current system
NONE = No wiring
```

Example
```
:SETUP:WIRING P1W2,P1W2
:SETUP:WIRING?→:SETUP:WIRING P1W2,P1W2
```

Description
- Set Wiring-A, then Wiring-B.
- Wiring-B can be omitted for combinations in which Wiring-B cannot be set.
- Depending on the model, some combinations of wiring methods cannot be selected.
- For a single-phase model, Wiring-A is fixed to P1W2 and Wiring-B cannot be set.

4.17 SSTart Group

The commands in the SSTart Group are used to execute single start measurement.
This command can be used to execute the same operation as when the SINGLE START key on the front panel is pressed.



:SSTart

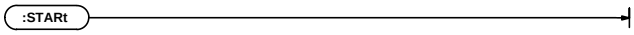
Function Executes single start.

Syntax
```
:SSTart
```

Example
```
:SSTART
```

4.18 STARt Group

The commands in the STARt Group are used to start the data acquisition operation.
This command can be used to execute the same operation as when the START/STOP key on the front panel is pressed.



:STARt

Function Starts data acquisition.

Syntax
```
:STARt
```

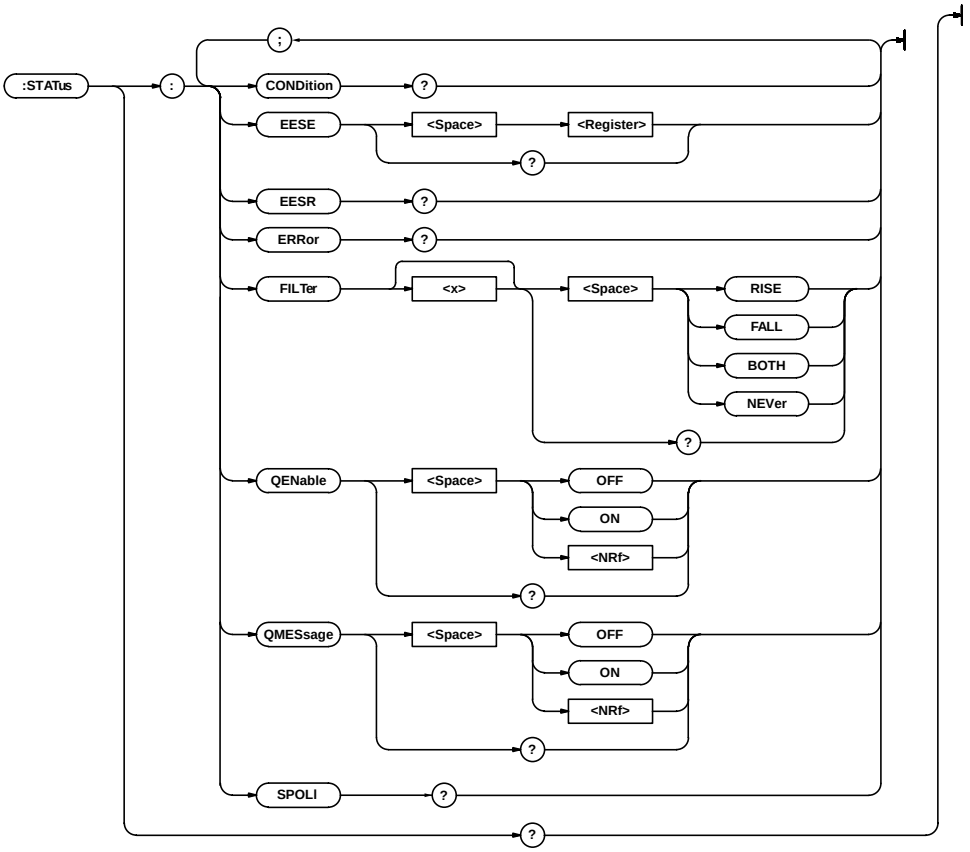
Example
```
:START
```

Description Use the ":STOP" command to stop the data acquisition.

4.19 STATus Group

The commands in the STATus Group are used to set and query the status report.

There are no front-panel keys that correspond to the commands in this group. For the status report, see chapter 5.



:STATus?

Function Queries all settings related to the communication 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 status register.

Syntax `:STATus:CONDition?`

Example `:STATUS:CONDITION?→16`

Description For the description regarding how to synchronize the program using the `:STATus:CONDition` command, see page 3-8.

:STATus:EESE

(Extended Event Status Enable register)

Function Sets the extended event enable register or queries the current setting.

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 and clears the extended event register.

Syntax `:STATus:EESR?`

Example `:STATUS:EESR?→0`

:STATus:ERRor?

Function Queries the code and information of the error (top of the error queue).

Syntax `:STATus:ERRor?`

Example `:STATUS:ERROR?→113,"Underfined Header"`

Description
- "0" (No error) is returned, if there is no error.
- The messages cannot be returned in Japanese.
- You can set whether or not to attach the messages to the error using the "STATus:QMESsage" command.

:STATus:FILTer<x>

Function Sets the transition filter or queries the current setting.

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 Sets how the bits in the status register must change in order to set the event. If it is set to "Rise", an event is set when the value changes from "0" to "1."

:STATus:QENable

Function Sets whether or not to store messages other than errors in the error queue or queries the current setting.

Syntax `:STATus:QENable {<Boolean>}`
`:STATus:QENable?`

Example `:STATUS:QENABLE ON`
`:STATUS:QENABLE?→:STATUS:QENABLE 1`

:STATus:QMESsage

Function Sets whether or not to attach a message to the "STATus:ERRor?" response or queries the current setting.

Syntax `:STATus:QMESsage {<Boolean>}`
`:STATus:QMESsage?`

Example `:STATUS:QMESSAGE ON`
`:STATUS:QMESSAGE?→:STATUS:QMESSAGE 1`

:STATus:SPOLl? (Serial Poll)

Function Executes serial polling.

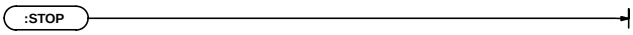
Syntax `:STATus:SPOLl?`

Example `:STATUS:SPOLL?→:STATUS:SPOLL 0`

Description This is a dedicated command for the serial interface. An interface message is available for the GP-IB interface.

4.20 STOP Group

The commands in the STOP Group are used to stop the data acquisition operation.
This command can be used to execute the same operation as when the START/STOP key on the front panel is pressed.



:STOP

Function Stops data acquisition.

Syntax `:STOP`

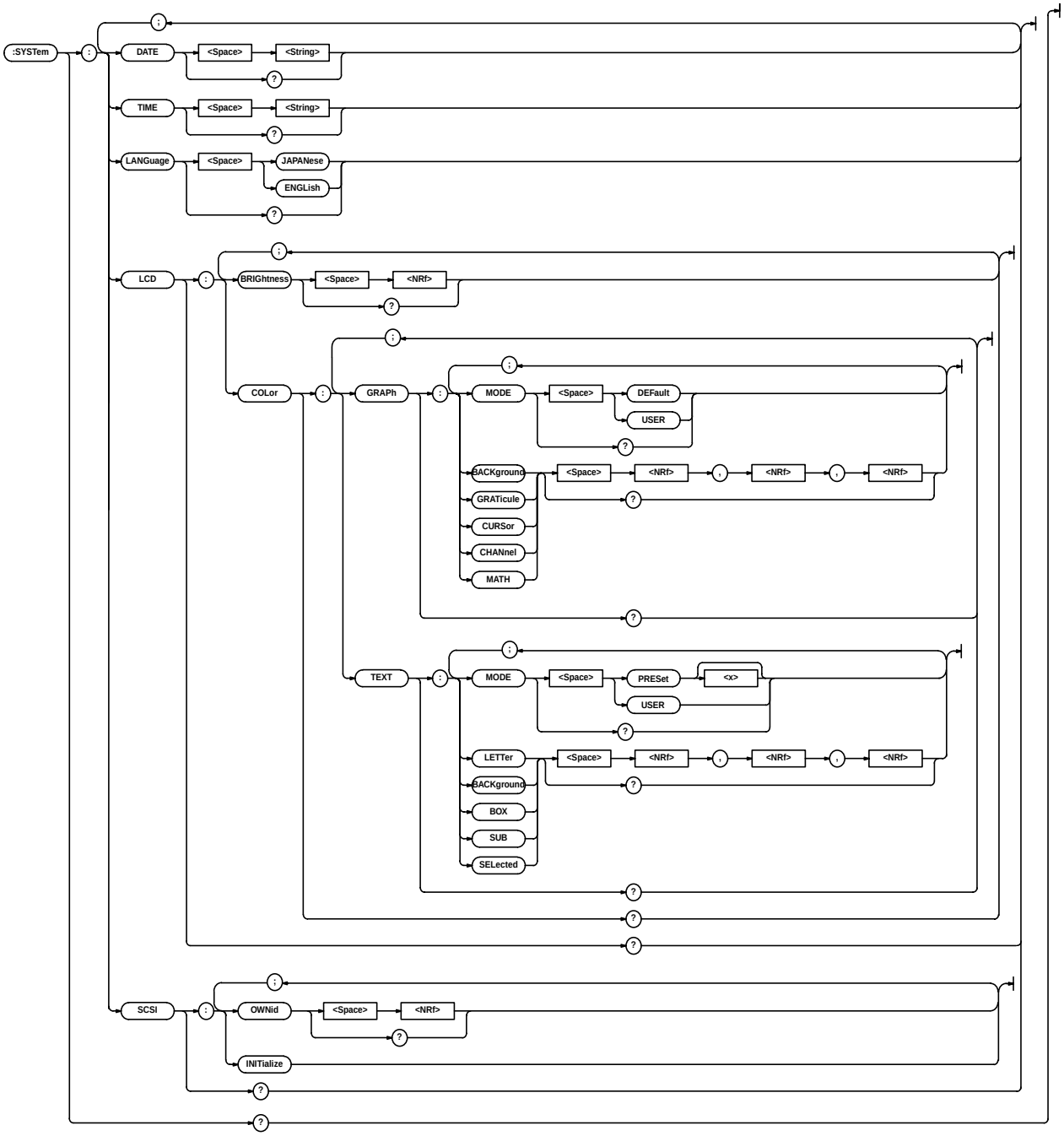
Example `:STOP`

Description Use the ":STARt" command to start the data acquisition.

4.21 SYSTem Group

The commands in the SYSTem Group deal with cursor measurements.

These commands can be used to make the same settings and inquiries as when the MISC key on the front panel is pressed.



:SYSTem?

Function Queries all settings related to the system.
Syntax :SYSTem?
Example :SYSTEM?→:SYSTEM:LANGUAGE JAPANESE;LCD:BRIGHTNESS 2;COLOR:GRAPH:MODE DEFAULT;:SYSTEM:LCD:COLOR:TEXT:MODE PRESET1;:SYSTEM:SCSI:OWNID 6

:SYSTem:DATE

Function Sets the date or queries the current setting.
Syntax :SYSTem:DATE {<string>}
:SYSTem:DATE?
<string> = "YY/MM/DD"
(YY = year, MM = month, DD = day)
Example :SYSTEM:DATE "99/01/01"
:SYSTEM:DATE?→"99/01/01"
Description The lower two digits are displayed for the year.

:SYSTem:LANGuage

Function Sets the message language or queries the current setting.
Syntax :SYSTem:LANGuage {JAPANese|ENGLish}
:SYSTem:LANGuage?
Example :SYSTEM:LANGUAGE JAPANESE
:SYSTEM:LANGUAGE?→:SYSTEM:LANGUAGE JAPANESE

:SYSTem:LCD?

Function Queries all settings related to the LCD monitor.
Syntax :SYSTem:LCD?
Example :SYSTEM:LCD?→:SYSTEM:LCD:BRIGHTNESS 2;COLOR:GRAPH:MODE DEFAULT;:SYSTEM:LCD:COLOR:TEXT:MODE PRESET1

:SYSTem:LCD:BRIGhtness

Function Sets the brightness of the LCD monitor or queries the current setting.
Syntax :SYSTem:LCD:BRIGhtness {<NRf>}
:SYSTem:LCD:BRIGhtness?
<NRf> = -1 to 3
Example :SYSTEM:LCD:BRIGHTNESS 2
:SYSTEM:LCD:BRIGHTNESS?→:SYSTEM:LCD:BRIGHTNESS 2

:SYSTem:LCD:COLor?

Function LCD Queries all settings related to the display colors of the LCD monitor.
Syntax :SYSTem:LCD:COLor?
Example :SYSTEM:LCD:COLOR?→:SYSTEM:LCD:COLOR:GRAPH:MODE DEFAULT;:SYSTEM:LCD:COLOR:TEXT:MODE PRESET1

:SYSTem:LCD:COLor:GRAPh?

Function Queries all settings related to the display color of graphic items.
Syntax :SYSTem:LCD:COLor:GRAPh?
Example :SYSTEM:LCD:COLOR:GRAPH?→:SYSTEM:LCD:COLOR:GRAPH:MODE USER;BACKGROUND 0,0,0;GRATICULE 6,6,6;CURSOR 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;MATH1 0,4,7;MATH2 5,5,5

:SYSTem:LCD:COLor:GRAPh:{BACKground|GRATicule|CURSor|CHANnel<x>|MATH<x>}

Function Queries the display color for the background/graticule/cursor/channel waveform/MATH waveform or queries the current setting.
Syntax :SYSTem:LCD:COLor:GRAPh:{BACKground|GRATicule|CURSor|CHANnel<x>|MATH<x>} {<NRf>,<NRf>,<NRf>}
:SYSTem:LCD:COLor:GRAPh:{BACKground|GRATicule|CURSor|CHANnel<x>|MATH<x>}?
<x> of the CHANnel<x> = 1 to 8
<x> of the MATH<x> = 1, 2
<NRf> = 0 to 7
Example :SYSTEM:LCD:COLOR:GRAPH:BACKGROUND 0,0,0
:SYSTEM:LCD:COLOR:GRAPH:BACKGROUND?→:SYSTEM:LCD:COLOR:GRAPH:BACKGROUND 0,0,0
Description Set the color in the order R, G, and B.

:SYSTem:LCD:COLor:GRAPh:MODE

Function Sets the display color mode of graphic items or queries the current setting.
Syntax :SYSTem:LCD:COLor:GRAPh:MODE {DEFault|USER}
:SYSTem:LCD:COLor:GRAPh:MODE?
Example :SYSTEM:LCD:COLOR:GRAPH:MODE DEFAULT
:SYSTEM:LCD:COLOR:GRAPH:MODE?→:SYSTEM:LCD:COLOR:GRAPH:MODE DEFAULT

:SYSTem:LCD:COLor:TEXT?

Function Queries all settings related to the display color of text items.
Syntax :SYSTem:LCD:COLor:TEXT?
Example :SYSTEM:LCD:COLOR:TEXT?→:SYSTEM:LCD:COLOR:TEXT:MODE USER;LETTER 7,7,7;BACKGROUND 2,2,6;BOX 0,0,7;SUB 3,3,3;SELECTED 0,4,7

:SYSTem:LCD:COLor:TEXT:{LETTer|BACKground|BOX|SUB|SELected}

Function Sets the display colors for characters (Menu Fore)/menu background (Menu Back)/selected menu (Select Box)/popup menu (Sub Menu)/selected key (Selected Key) or queries the current setting.

Syntax
```
:SYSTem:LCD:COLor:TEXT:{LETTer|BACKground|BOX|SUB|SELected}{<NRf>,<NRf>,<NRf>}
:SYSTem:LCD:COLor:TEXT:{LETTer|BACKground|BOX|SUB|SELected}?
<NRf> = 0 to 7
```

Example
```
:SYSTEM:LCD:COLOR:TEXT:LETTER 7,7,7
:SYSTEM:LCD:COLOR:TEXT:LETTER?→:SYSTEM:LCD:COLOR:TEXT:LETTER 7,7,7
```

Description Set the color in the order R, G, and B.

:SYSTem:LCD:COLor:TEXT:MODE

Function Sets the display color mode of text items or queries the current setting.

Syntax
```
:SYSTem:LCD:COLor:TEXT:MODE {PRESet<x>|USER}
:SYSTem:LCD:COLor:TEXT:MODE?
<x> = 1 to 3
```

Example
```
:SYSTEM:LCD:COLOR:TEXT:MODE PRESET1
:SYSTEM:LCD:COLOR:TEXT:MODE?→:SYSTEM:LCD:COLOR:TEXT:MODE PRESET1
```

:SYSTem:SCSI?

Function Queries all settings related to the SCSI-ID.

Syntax
```
:SYSTem:SCSI?
```

Example
```
:SYSTEM:SCSI?→:SYSTEM:SCSI:OWNID 6
```

Description If the SCSI (option) is not installed, an error occurs.

:SYSTem:SCSI:INITialize

Function Initializes SCSI related settings.

Syntax
```
:SYSTem:SCSI:INITialize
```

Example
```
:SYSTEM:SCSI:INITIALIZE
```

Description
- If the SCSI (option) is not installed, an error occurs.
- Make sure to execute this command, if the SCSI-ID of this instrument is changed using the "`:SYSTem:SCSI:OWNid`" command.

:SYSTem:SCSI:OWNid

Function Sets the SCSI ID of this instrument or queries the current setting.

Syntax
```
:SYSTem:SCSI:OWNid {<NRf>}
:SYSTem:SCSI:OWNid?
<NRf> = 0 to 7
```

Example
```
:SYSTEM:SCSI:OWNID 6
:SYSTEM:SCSI:OWNID?→:SYSTEM:SCSI:OWNID 6
```

Description If the SCSI (option) is not installed, an error occurs.

:SYSTem:TIME

Function Sets the time or queries the current setting.

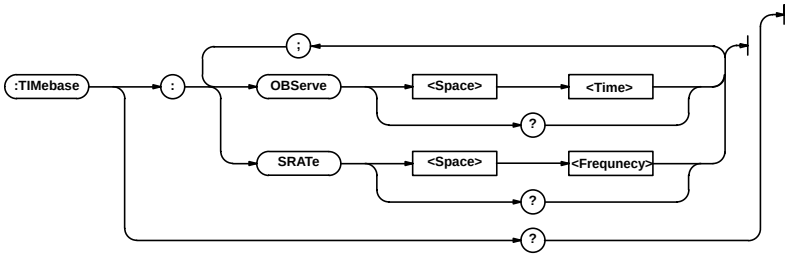
Syntax
```
:SYSTem:TIME {<string>}
:SYSTem:TIME?
<string> = "HH:MM:SS" (HH = hour, MM = minute, SS = second)
```

Example
```
:SYSTEM:TIME "14:30:00"
:SYSTEM:TIME?→"14:30:00"
```

4.22 TIMebase Group

The commands in the TIMebase Group deal with the time base (horizontal axis).

These commands can be used to make the same settings and inquiries as when the OBSERVATION TIME knob on the front panel is pressed.



:TIMebase?

Function Queries all settings related to the time base (horizontal axis).

Syntax `:TIMebase?`

Example
```
:TIMEBASE?→:TIMEBASE:
OBSERVE 10000E-03;SRATE 1.00000E+06
```

:TIMebase:OBServe

Function Sets the observation time of the waveform or queries the current setting.

Syntax
```
:TIMebase:OBServe {<time>}
:TIMebase:OBServe?
<time> = 10us to 1ks
         (See the PZ4000 User's Manual.)
```

Example
```
:TIMEBASE:OBSERVE 100MS
:TIMEBASE:OBSERVE?→:TIMEBASE:
OBSERVE 100.00E-03
```

:TIMebase:SRATe

Function Sets the sampling rate or queries the current setting.

Syntax
```
:TIMebase:SRATe {<frequency>}
:TIMebase:SRATe?
<frequency> = 50Hz to 5MHz
              (See the PZ4000 User's
              Manual.)
```

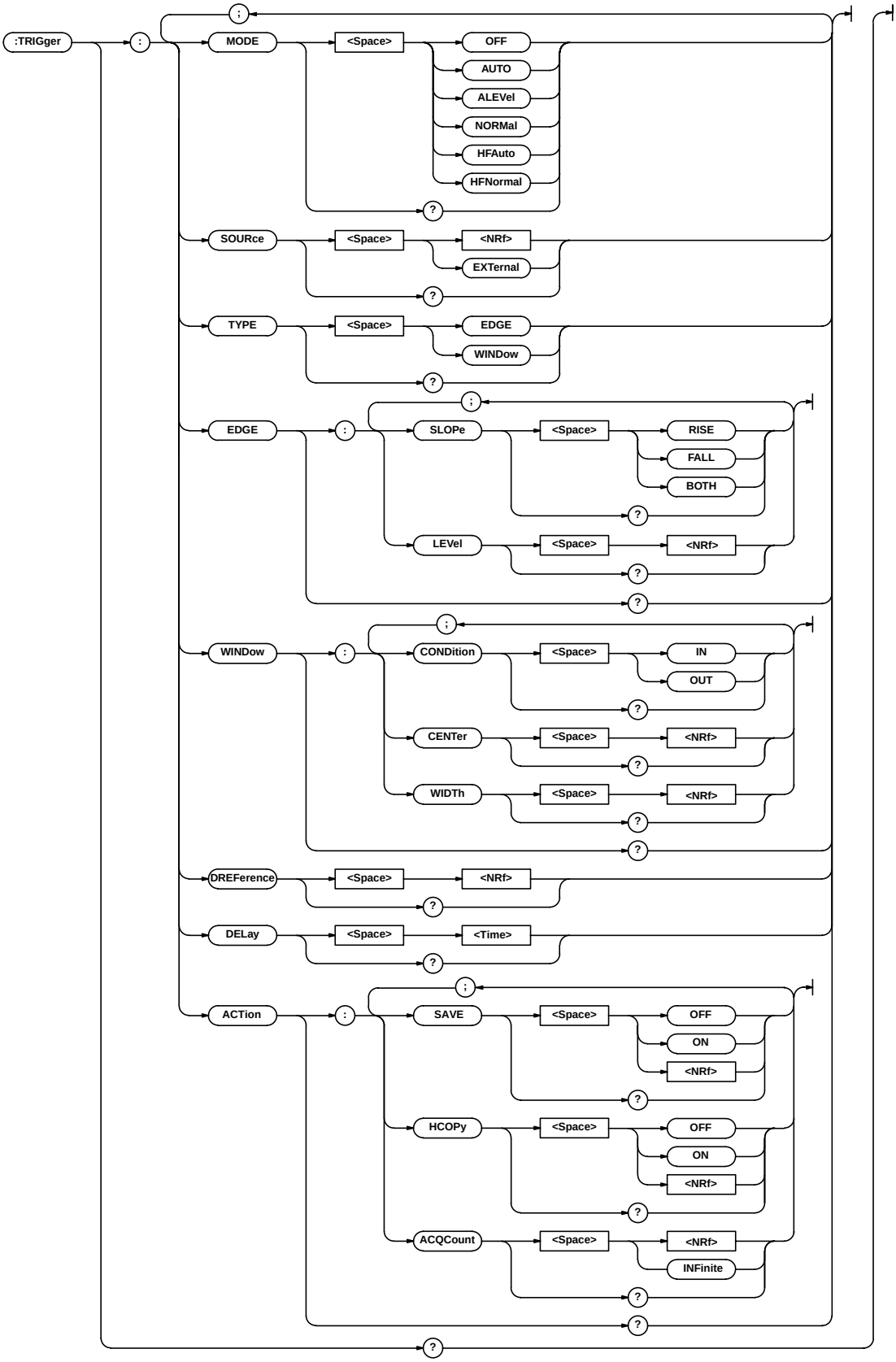
Example
```
:TIMEBASE:SRATE 1MHz
:TIMEBASE:SRATE?→:TIMEBASE:
SRATE 1.00000E+06
```

Description The observation time is set to the optimal setting (longest range possible) depending on the specified sampling rate.

4.23 TRIGger Group

The commands in the TRIGger Group deal with the trigger.

These commands can be used to make the same settings and inquiries as when the TRIGGER key on the front panel is pressed.



:TRIGger?

Function Queries all settings related to the trigger.
Syntax :TRIGger?
Example
```
:TRIGGER?→:TRIGGER:MODE AUTO;SOURCE 1;
TYPE EDGE;EDGE:SLOPE RISE;
LEVEL 1.000E+03;:TRIGGER:DREFERENCE 10;
DELAY 0.0E+00;ACTION:SAVE 0;HCOPY 0;
ACQCOUNT INFINITE
```

:TRIGger:ACTion?

Function Queries all settings related to action-on-trigger.
Syntax
```
:TRIGger:ACTion?
```
Example
```
:TRIGGER:ACTION?→:TRIGGER:ACTION:
SAVE 0;HCOPY 0;ACQCOUNT INFINITE
```

:TRIGger:ACTion:ACQCount

Function Sets the action count of action-on-trigger or queries the current setting.
Syntax
```
:TRIGger:ACTion:ACQCount {<NRf>|
INFinite}
:TRIGger:ACTion:ACQCount?
<NRf> = 1 to 65536
```
Example
```
:TRIGGER:ACTION:ACQCOUNT 10
:TRIGGER:ACTION:ACQCOUNT?→:TRIGGER:
ACTION:ACQCOUNT 10
```

:TRIGger:ACTion:HCOPy

Function Sets whether or not to output screen image data (ON/OFF) when an action is activated, or queries the current setting.
Syntax
```
:TRIGger:ACTion:HCOPy {<Boolean>}
:TRIGger:ACTion:HCOPy?
```
Example
```
:TRIGGER:ACTION:HCOPY ON
:TRIGGER:ACTION:HCOPY?→:TRIGGER:
ACTION:HCOPY 1
```

:TRIGger:ACTion:SAVE

Function Sets whether or not to save the waveform data to the storage medium (ON/OFF) when an action is activated, or queries the current setting.
Syntax
```
:TRIGger:ACTion:SAVE {<Boolean>}
:TRIGger:ACTion:SAVE?
```
Example
```
:TRIGGER:ACTION:SAVE ON
:TRIGGER:ACTION:SAVE?→:TRIGGER:ACTION:
SAVE 1
```

:TRIGger:DELay

Function Sets the trigger delay or queries the current setting.
Syntax
```
:TRIGger:DELay {<time>}
:TRIGger:DELay?
<time> = 0 to 1s (The resolution is
         0.5ms)
```
Example
```
:TRIGGER:DELAY 0
:TRIGGER:DELAY?→:TRIGGER:DELAY 0.0E+00
```
Description The trigger delay is set to the time from the trigger point to the trigger position on this instrument.

:TRIGger:DREFerence (Delay REFerence)

Function Sets the trigger position or queries the current setting.
Syntax
```
:TRIGger:DREFerence {<NRf>}
:TRIGger:DREFerence?
<NRf> = 0 to 100(%)
```
Example
```
:TRIGGER:DREFERENCE 10
:TRIGGER:DREFERENCE?→:TRIGGER:
DREFERENCE 10
```

:TRIGger:EDGE?

Function Queries all settings related to the edge trigger.
Syntax
```
:TRIGger:EDGE?
```
Example
```
:TRIGGER:EDGE?→:TRIGGER:EDGE:
SLOPE RISE;LEVEL 0.0
```

:TRIGger:EDGE:LEVel

Function Sets the trigger level for the edge trigger or queries the current setting.
Syntax
```
:TRIGger:EDGE:LEVel {<NRf>}
:TRIGger:EDGE:LEVel?
<NRf> = -100.0 to 100.0
        (The resolution is 0.1(%))
```
Example
```
:TRIGGER:EDGE:LEVEL 1000V
:TRIGGER:EDGE:LEVEL?→:TRIGGER:EDGE:
LEVEL 0.0
```
Description Set the level in terms of a percentage of the full scale value displayed on the screen.

:TRIGger:EDGE:SLOPe

Function Sets the trigger slope for the edge trigger or queries the current setting.
Syntax
```
:TRIGger:EDGE:SLOPe {RISE|FALL|BOTH}
:TRIGger:EDGE:SLOPe?
```
Example
```
:TRIGGER:EDGE:SLOPE RISE
:TRIGGER:EDGE:SLOPE?→:TRIGGER:EDGE:
SLOPE RISE
```

:TRIGger:MODE

Function Sets the trigger mode or queries the current setting.

Syntax
```
:TRIGger:MODE {OFF|AUTO|ALEVel|NORMal|
HFAuto|HFNormal}
:TRIGger:MODE?
```

Example
```
:TRIGGER:MODE AUTO
:TRIGGER:MODE?→:TRIGGER:MODE AUTO
```

:TRIGger:SOURce

Function Sets the trigger source or queries the current setting.

Syntax
```
:TRIGger:SOURce {<NRf>|EXTernal}
:TRIGger:SOURce?
<NRf> = 1 to 8
EXTernal = External trigger
```

Example
```
:TRIGGER:SOURCE 1
:TRIGGER:SOURCE?→:TRIGGER:SOURCE 1
```

:TRIGger:TYPE

Function Sets the trigger type or queries the current setting.

Syntax
```
:TRIGger:TYPE {EDGE|WINDow}
:TRIGger:TYPE?
```

Example
```
:TRIGGER:TYPE EDGE
:TRIGGER:TYPE?→:TRIGGER:TYPE EDGE
```

:TRIGger:WINDow?

Function Queries all settings related to the window trigger.

Syntax
```
:TRIGger:WINDow?
```

Example
```
:TRIGGER:WINDOW?→:TRIGGER:WINDOW:
CONDITION IN;CENTER 0.0;WIDTH 25.0
```

:TRIGger:WINDow:CENTer

Function Sets the center level for the window trigger or queries the current setting.

Syntax
```
:TRIGger:WINDow:CENTer {<NRf>}
:TRIGger:WINDow:CENTer?
<NRf> = -100.0 to 100.0
        (The resolution is 0.1(%))
```

Example
```
:TRIGGER:WINDOW:CENTER
:TRIGGER:WINDOW:CENTER?→:TRIGGER:
WINDOW:CENTER 0.0
```

Description Set the center level in terms of a percentage of the full scale value displayed on the screen.

:TRIGger:WINDow:CONDition

Function Sets the trigger condition for the window trigger or queries the current setting.

Syntax
```
:TRIGger:WINDow:CONDition {IN|OUT}
:TRIGger:WINDow:CONDition?
```

Example
```
:TRIGGER:WINDOW:CONDITION IN
:TRIGGER:WINDOW:CONDITION?→:TRIGGER:
WINDOW:CONDITION IN
```

:TRIGger:WINDow:WIDTh

Function Sets the window width for the window trigger or queries the current setting.

Syntax
```
:TRIGger:WINDow:WIDTh {<NRf>}
:TRIGger:WINDow:WIDTh?
<NRf> = -100.0 to 100.0
        (The resolution is 0.1(%))
```

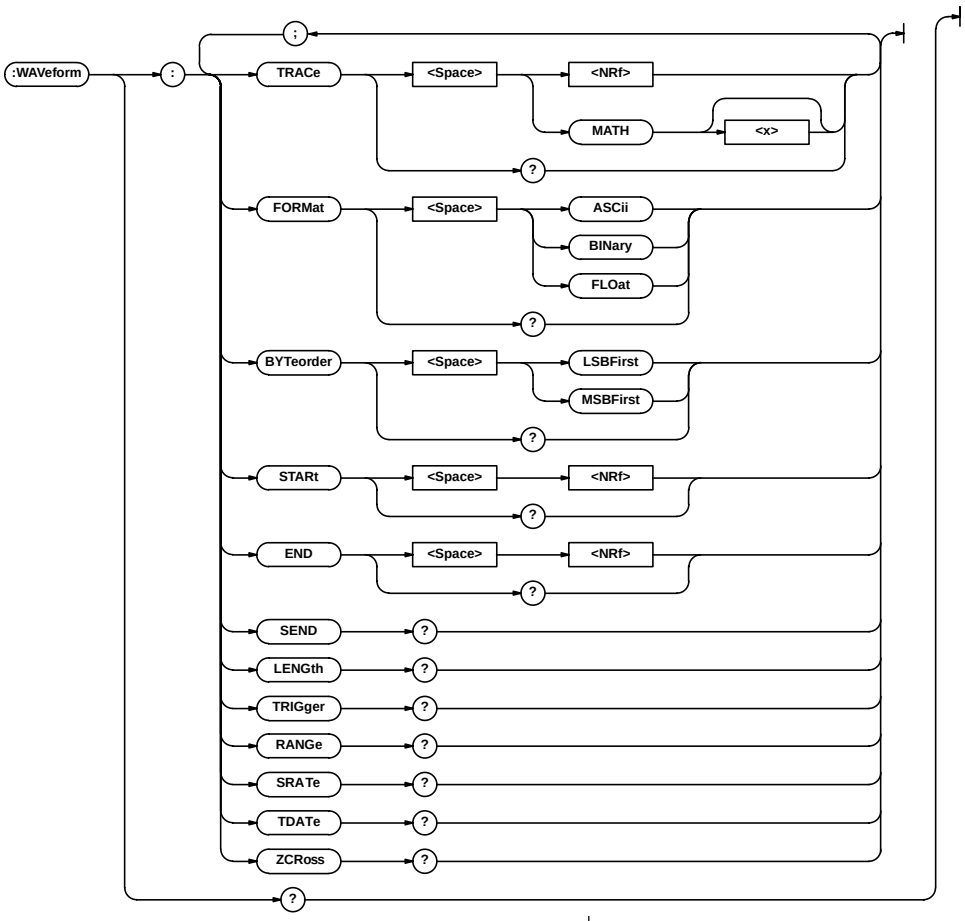
Example
```
:TRIGGER:WINDOW:WIDTH 25
:TRIGGER:WINDOW:WIDTH?→:TRIGGER:
WINDOW:WIDTH 25.0
```

Description Set the width in terms of a percentage of the full scale value displayed on the screen.

4.24 WAVeform Group

The commands in the WAVeform Group deal with the acquired waveform data.

There are no front-panel keys that correspond to the commands in this group.



:WAVeform?

Function Queries all settings related to the waveform data.

Syntax `:WAVeform?`

Example
```
:WAVEFORM?→:WAVEFORM:TRACE 1;
FORMAT BINARY;BYTEORDER LSBFIRST;
START 0;END 100
```

:WAVeform:BYTeorder

Function Sets the byte order of the waveform data that are sent using the "`:WAVeform:SEND?`" command or queries the current setting.

Syntax
```
:WAVeform:BYTeorder {LSBFirst|MSBFirst}
:WAVeform:BYTeorder?
```

Example
```
:WAVEFORM:BYTEORDER LSBFIRST
:WAVEFORM:BYTEORDER?→:WAVEFORM:
BYTEORDER LSBFIRST
```

Description This setting is valid when "`:WAVeform:FORMat`" is set to {`BINary|FLOat`}.

:WAVeform:END

Function Sets the end point of the output of the waveform data that are sent using the "`:WAVeform:SEND?`" command or queries the current setting.

Syntax
```
:WAVeform:END {<NRf>}
:WAVeform:END?
<NRf> = 0 to (total number of data
        points -1)
```

Example
```
:WAVEFORM:END 100
:WAVEFORM:END?→:WAVEFORM:END 100
```

Description (The total number of data points) can be queried using the "`:WAVeform:LENGth?`" command.

:WAVeform:FORMat

Function Sets the format of the waveform data that are sent using the ":WAVeform:SEND?" command or queries the current setting?

Syntax `:WAVeform:FORMat {ASCii|BINary|FLOat}`
`:WAVeform:FORMat?`

Example `:WAVEFORM:FORMAT BINARY`
`:WAVEFORM:FORMAT?→:WAVEFORM:FORMAT BINARY`

Description For the differences in the waveform data output depending on the format setting, see the description for the ":WAVeform:SEND?" command.

:WAVeform:LENGth?

Function Queries the total number of data points of the waveform that is specified using the ":WAVeform:TRACe" command.

Syntax `:WAVeform:LENGth?`

Example `:WAVEFORM:LENGTH?→100001`

Description
- The total number of data points varies depending on the record length (ON/OFF state of dividing the record length) and observation time (sampling rate).
- For details, see the PZ4000 User's Manual.

:WAVeform:RANGe?

Function Queries the range value that is used to convert the waveform specified using the ":WAVeform:TRACe" command to physical data.

Syntax `:WAVeform:RANGe?`

Example `:WAVEFORM:RANGE?→250.00E+00`

Description
- This range value is used when converting the waveform to physical values when the ":WAVeform:FORMat" is set to BINary.
- "0" is returned when ":WAVeform:TRACe" is set to MATH<x>.

:WAVeform:SEND?

Function Queries the waveform data that are specified using the ":WAVeform:TRACe" command.

Syntax `:WAVeform:SEND?`

Example
- When ":WAVeform:FORMat" is set to {ASCii}
  `:WAVEFORM:SEND?→<NR3>,<NR3>,...`
- When ":WAVeform:FORMat" is set to {BINary|FLOat}
  `:WAVEFORM:SEND?→#8(Number of bytes, 8 digits)(Series of data bytes)`

Description The format of the numerical data that is output depends on the ":NUMeric:FORMat" setting.

(1) When set to "ASCii"
The physical values are output in <NR3> format. Each item of data is separated by a comma.

(2) When set to "BINary"
The A/D value before it is converted to a physical value is output in WORD format (2 bytes, 0 to FFFH, unsigned).
The output byte order of each data point follows the order that is set using the ":WAVeform:BYTeorder" command.
The equation used to convert to a physical value is
Physical value
= (WORD data - 2048)/2048 X the range value.
Binary output is not possible when ":WAVeform:TRACe" is set to MATH<x>. All 0s are returned. Inquire using the FLOat format in this case.

(3) When set to "FLOat"
The physical values are output in IEEE single precision floating point format (4 bytes).
The output byte order of each data point follows the order that is set using the ":WAVeform:BYTeorder" command.

:WAVeform:SRATe?

Function Queries the sampling rate of the acquired data.

Syntax `:WAVeform:SRATe?`

Example `:WAVEFORM:SRATE?→1.00000E+06`

:WAVeform:STARt

Function Sets the start point of the output of the waveform data that are sent using the ":WAVeform:SEND?" command or queries the current setting.

Syntax
```
:WAVeform:STARt {<NRf>}
:WAVeform:STARt?-
<NRf> = 0 to (Total number of data
        points-1)
```

Example
```
:WAVEFORM:START 0
:WAVEFORM:START?→:WAVEFORM:START 0
```

Description (The total number of data points) can be queried using the ":WAVeform:LENGth?" command.

:WAVeform:TDATe?

Function Queries the string containing the trigger date and time when the waveform was acquired.

Syntax
```
:WAVeform:TDATe?
```

Example
```
:WAVEFORM:TDATE?→"1999/12/23 12:34:56"
```

Description The date and time is separated by one space character.

:WAVeform:TRACe

Function Sets the target waveform in the waveform group or queries the current setting.

Syntax
```
:WAVeform:TRACe {<NRf>|MATH<x>}
:WAVeform:TRACe?
<NRf> = 1 to 8(channel)
<x> = 1, 2(MATH)
```

Example
```
:WAVEFORM:TRACE 1
:WAVEFORM:TRACE?→:WAVEFORM:TRACE 1
```

:WAVeform:TRIGger?

Function Queries the trigger position of the acquired data.

Syntax
```
:WAVeform:TRIGger?
```

Example
```
:WAVEFORM:TRIGGER?→10000
```

Description Queries the number of points from the beginning of the record length to the trigger position.

:WAVeform:ZCRoss?

Function Queries zero crossing data of all channels.

Syntax
```
:WAVeform:ZCRoss?
```

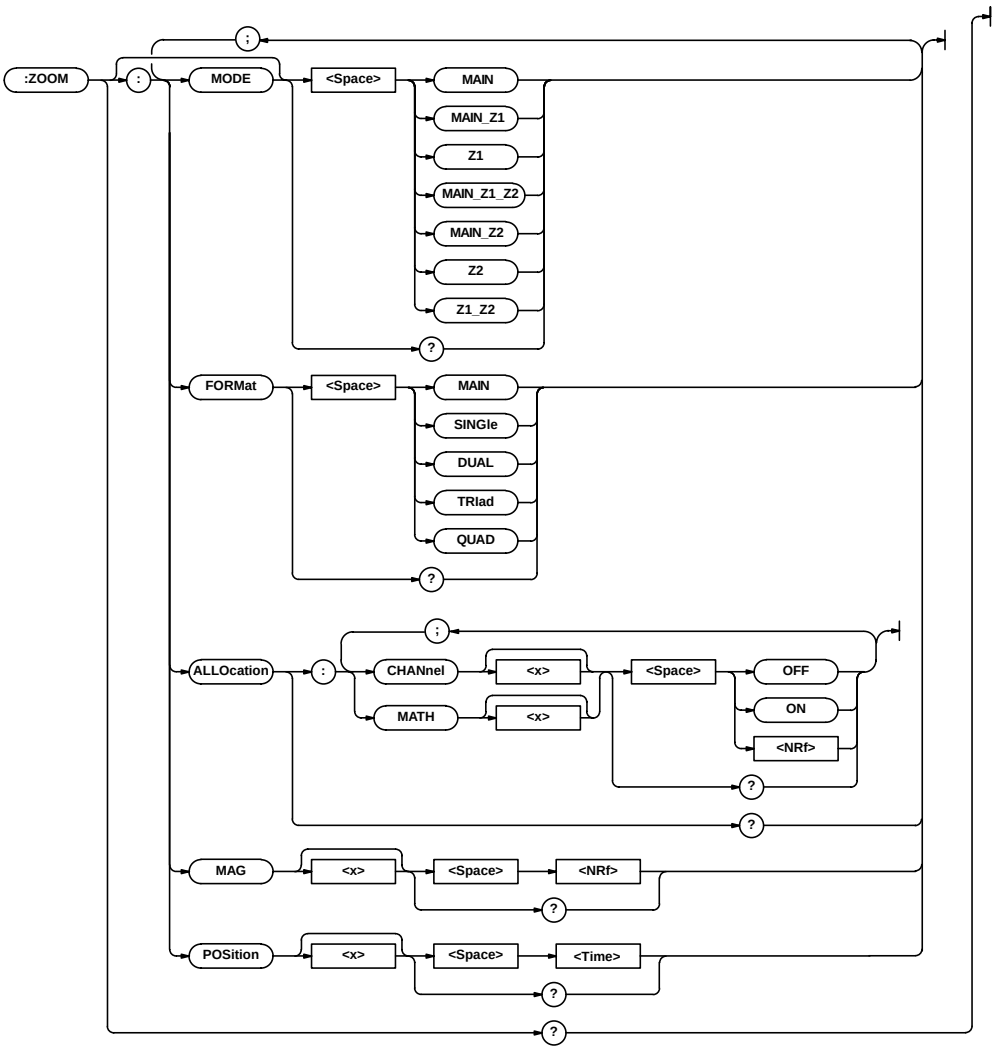
Example
```
:WAVEFORM:ZCROSS?→#8(Number of bytes, 8
digits)(Series of data bytes)
```

Description
- The output start and end points of zero crossing data are specified using the ":WAVeform:{STARt|END}" command in the same fashion as for the waveform data.
- The data format of each output point is fixed to WORD (2-byte) format.
- The output byte order follows the order that is set using the ":WAVeform:BYTeorder" command.

4.25 ZOOM Group

The commands in the ZOOM Group deal with the zooming of the waveform.

These commands can be used to make the same settings and inquiries as when the ZOOM key on the front panel is pressed.



:ZOOM?

Function Queries all settings related to the zooming of the waveform.

Syntax :ZOOM?

Example :ZOOM?→:ZOOM:MODE MAIN_Z1_Z2;
FORMAT SINGLE;ALLOCATION:CHANNEL1 1;
CHANNEL2 0;CHANNEL3 0;CHANNEL4 0;
CHANNEL5 0;CHANNEL6 0;CHANNEL7 0;
CHANNEL8 0;MATH1 0;MATH2 0;:ZOOM:MAG1 2;
MAG2 2;POSITION1 25.000E-03;
POSITION2 75.000E-03

:ZOOM:ALLOcation?

Function Queries all settings related to 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 Sets whether or not to select the waveform to be zoomed or queries the current setting.

Syntax
```
:ZOOM:ALLOcation:{CHANnel<x>|
MATH<x>} {<Boolean>}
:ZOOM:ALLOcation:{CHANnel<x>|MATH<x>}?
<x> of CHANnel<x> = 1 to 8
<x> of MATH<x> = 1, 2
```

Example
```
:ZOOM:ALLOCATION:CHANNEL1 ON
:ZOOM:ALLOCATION:CHANNEL1?→:ZOOM:
ALLOCATION:CHANNEL1 1
```

:ZOOM:FORMat

Function Sets the display format of the zoomed waveform or queries the current setting.

Syntax
```
:ZOOM:FORMat {MAIN|SINGle|DUAL|TRIad|
QUAD}
:ZOOM:FORMat?
```

Example
```
:ZOOM:FORMAT SINGLE
:ZOOM:FORMAT?→:ZOOM:FORMAT SINGLE
```

:ZOOM:MAG<x>

Function Sets the zoom factor or queries the current setting.

Syntax
```
:ZOOM:MAG<x> {<NRf>}
:ZOOM:MAG<x>?
<x> = 1, 2
<NRf> = 2 to 100000 (See the PZ4000
        User's Manual)
```

Example
```
:ZOOM:MAG1 2
:ZOOM:MAG1?→:ZOOM:MAG1 2
```

Description The selectable zoom factor varies depending on the measurement mode, the observation time, the record length, and the record length division settings.

:ZOOM[:MODE]

Function Sets the the display mode of the zoomed waveform or queries the current setting.

Syntax
```
:ZOOM[:MODE] {MAIN|MAIN_Z1|Z1|
MAIN_Z1_Z2|MAIN_Z2|Z2|Z1_Z2}
:ZOOM:MODE?
```

Example
```
:ZOOM:MODE MAIN_Z1_Z2
:ZOOM:MODE?→:ZOOM:MODE MAIN_Z1_Z2
```

:ZOOM:POSition<x>

Function Sets the position of the zoom box or queries the current setting.

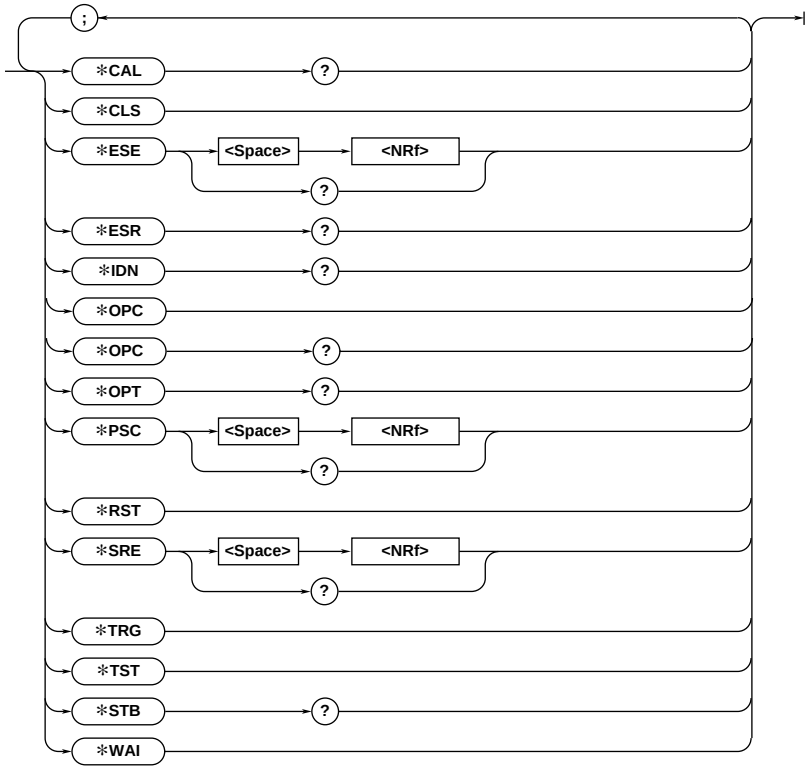
Syntax
```
:ZOOM:POSition<x> {<time>|<NRf>}
:ZOOM:POSition<x>?
<x> = 1, 2
<time> = 0 to (OBSERVATION TIME) (during
         the normal measurement mode,
         when Time Base = Internal)
<NRf>  = 0 to Record length (when Time
         Base = Internal, or during the
         harmonic measurement mode)
```

Example
```
:ZOOM:POSITION1 25MS
:ZOOM:POSITION1?→:ZOOM:
POSITION1 25.000E-03
```

Description
- The range and resolution of <time> depends on the observation time.
- Specify <NRf> in terms of sampled data points. The range is from 0 to the record length.

4.26 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 (zero level compensation, same operation as pressing the CAL key) and queries the result.

Syntax `*CAL?`

Example `*CAL?→0`

Description "0" is returned when the calibration completes properly. "1" i s returned if there is an abnormality.

*CLS (CLear Status)

Function Clears the standard event register, extended event register, and error queue.

Syntax `*CLS`

Example `*CLS`

Description
- If the `*CLS` command is immediately after the program message terminator, the output queue is also cleared.
- For details related to the registers and queues, see chapter 5.

*ESE (standard Event Status Enable register)

Function Sets the standard event enable register or queries the current setting.

Syntax `*ESE {<NRf>}`
`*ESE?`
`<NRf> = 0 to 255`

Example `*ESE 251`
`*ESE?→251`

Description
- Set the value using a decimal sum of each bit.
- For example, if "`*ESE 251`" is set, the standard event enable register is set to "`11111011.`" This means that bit 2 of the standard event register is disabled so that bit 5 (ESB) of the status register will not be set to "`1,`" even if a query error occurs.
- The default setting is "`*ESE 0`" (all bits disabled).
- The standard event enable register is not cleared even if an inquiry is made using `*ESE?.`
- For details related to the standard event enable register, see page 5-3.

*ESR? (standard Event Status Register)

Function Queries the standard event register and clears the register.

Syntax `*ESR?`

Example `*ESR?→32`

Description
- Returns the sum of each bit expressed as a decimal number.
- You can determine what type of event occurred when SRQ occurred.
- For example, if "32" is returned, it indicates that the standard event register is set to "00100000." This means that SRQ occurred because a "command syntax error" error occurred.
- The standard event register is cleared if an inquiry is made using *ESR?.
- For details related to the standard event register, see page 5-3.

*IDN? (IDeNtify)

Function Queries the instrument model.

Syntax `*IDN?`

Example `*IDN?→YOKOGAWA,253710,0,F1.10`

Description The response is returned in the following format: <Maker>, <Model>, <Serial No.>, <Firmware version>.
The <Serial No. > is always set to 0.

*OPC (OPeration Complete)

Function Sets bit 0 of the standard event register (OPC bit) to 1 when the specified overlap command completes.

Syntax `*OPC`

Example `*OPC`

Description
- For the description regarding how to synchronize the program using the *OPC command, see page 3-7.
- The "COMMunicate:OPSE" command is used to specify the overlap commands.
- If the *OPC command is not placed at the end of the message, the operation is not guaranteed.

*OPC? (OPeration Complete)

Function If the specified overlap command has been completed when *OPC? is sent, ASCII code "1" is returned.

Syntax `*OPC?`

Example `*OPC?→1`

Description
- For the description regarding how to synchronize the program using the *OPC? command, see page 3-13.
- The "COMMunicate:OPSE" command is used to specify the overlap commands.
- If the *OPC? command is not placed at the end of the message, the operation is not guaranteed.

*OPT? (OPTion)

Function Queries installed options.

Syntax `*OPT?`

Example `*OPT?→M1,PRINTER,SCSI`

Description
- Returns whether or not the following items exist: <Extended memory>, <Built-in printer>, <SCSI>
- The "*OPT?" command must be the last query in a program message. Otherwise, an error occurs.

*PSC (Power-on Status Clear)

Function Sets whether or not to clear the following registers on power up or queries the current setting. The registers are cleared if the value that is rounded to an integer is a non-zero number.
- Standard event enable register
- Extended event enable register
- Transition filter

Syntax
```
*PSC {<NRf>}
*PSC?
<NRf> = 0 (does not clear the
        registers), other than 0 (clears
        the registers)
```

Example
```
*PSC 1
*PSC?→1
```

Description For details regarding the registers, see chapter 5.

*RST (ReSeT)

Function Initializes the settings.

Syntax `*RST`

Example `*RST`

Description
- *OPC and *OPC? that were sent earlier are also reset.
- Resets all setup parameters except communication settings to factory default values.

*SRE (Service Request Enable register)

Function Sets the service request enable register or queries the current setting.

Syntax
```
*SRE <NRf>
*SRE?
<NRf> = 0 to 255
```

Example
```
*SRE 239
*SRE?→175 (because bit 6 (MSS) is ignored)
```

Description
- Set the value using a decimal sum of each bit.
- For example, if "*SRE 239" is set, the service request enable register is set to "11101111." This means that bit 4 of the standard event register is disabled so that bit 4 (MAV) of the status register will not be set to "1," even if the "output queue is not empty."
- However, bit 6 of the status byte is the MSS bit, so it is ignored.
- The default setting is "*SRE 0" (all bits disabled).
- The service request enable register is not cleared even if an inquiry is made using *SRE?.
- For details related to the service request enable register, see page 5-1.

*STB? (STatus Byte)

Function Queries the status byte register.

Syntax
```
*STB?
```

Example
```
*STB?→4
```

Description
- Returns the sum of each bit expressed as a decimal number.
- Since the register is read without serial polling, bit 6 is the MSS bit, not RQS.
- For example, if "4" is returned, it indicates that the standard event register is set to "00000100." This means that SRQ occurred because the "error queue is not empty."
- The status byte register is not cleared, even if an inquiry is made using *STB?.
- For details related to the status byte register, see page 5-2.

*TRG (TRiGger)

Function Executes single start (the same as pressing the SINGLE START key).

Syntax
```
*TRG
```

Exampl
```
*TRG
```

Description The multi-line message GET (Group Execute Trigger) operates in the same way as this command.

*TST? (TeST)

Function Executes the self-test and queries the result.

Syntax
```
*TST?
```

Example
```
*TST?→0
```

Description
- The self-test involves the testing of the internal memories.
- "0" is returned if the self-test is successful. "1" is returned otherwise.

*WAI (WAIt)

Function Waits until the execution of the specified overlap command completes before executing the commands that are specified after this command.

Syntax
```
*WAI
```

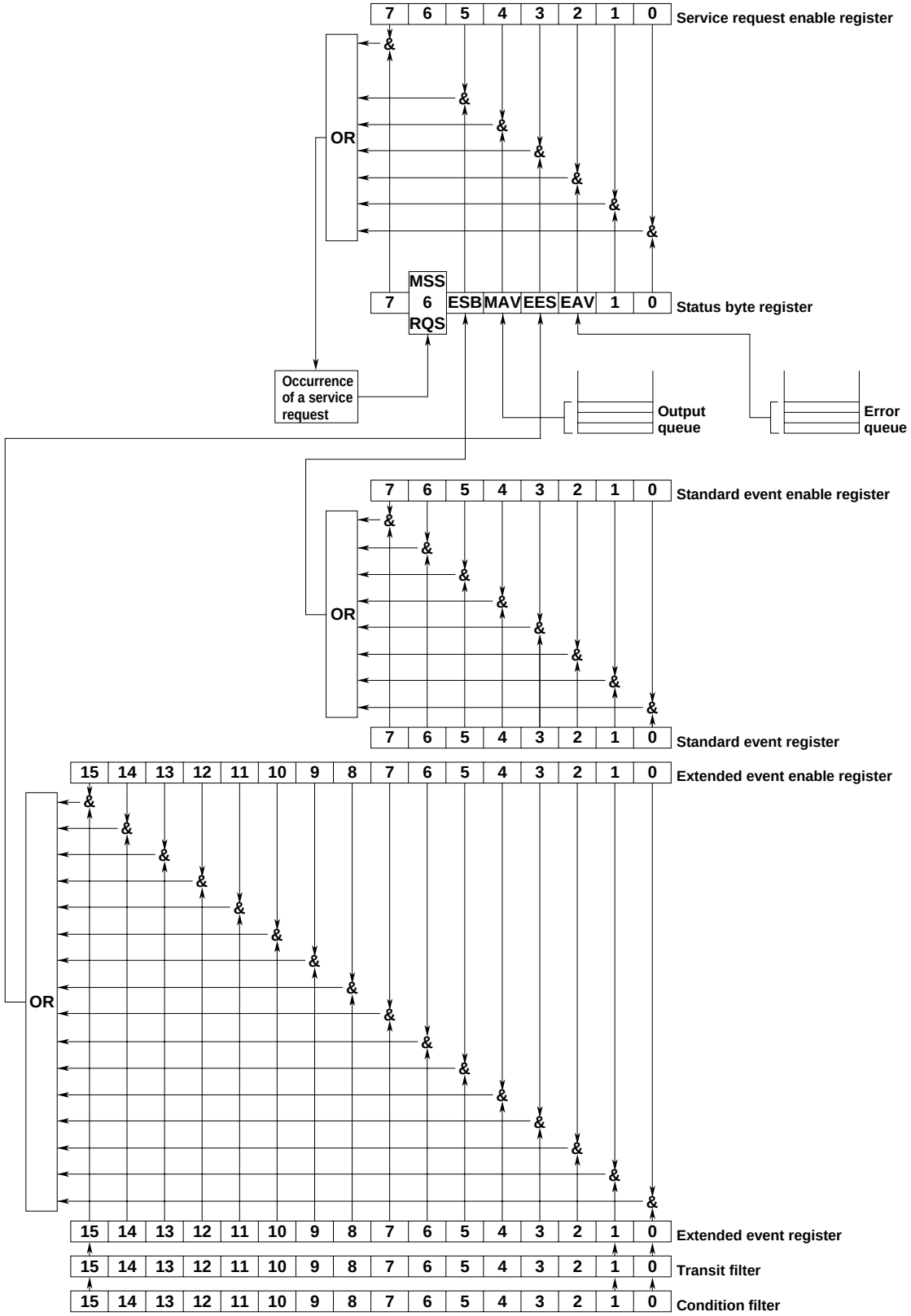
Example
```
*WAI
```

Description
- For the description regarding how to synchronize the program using the *WAI command, see page 3-7.
- The "COMMunicate:OPSE" command is used to specify the overlap commands.

Chapter 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 488.2-1987.



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 *ESR? 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 occurs causing the logical AND of any one bit in the standard event register and the corresponding bit of the enable register to become “1,” bit 5 (ESB) is 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; received string data that have spelling errors or that are not in the selection.

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: received a command that has a parameter that is outside the range or a command that deals with an option that is not installed.

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.

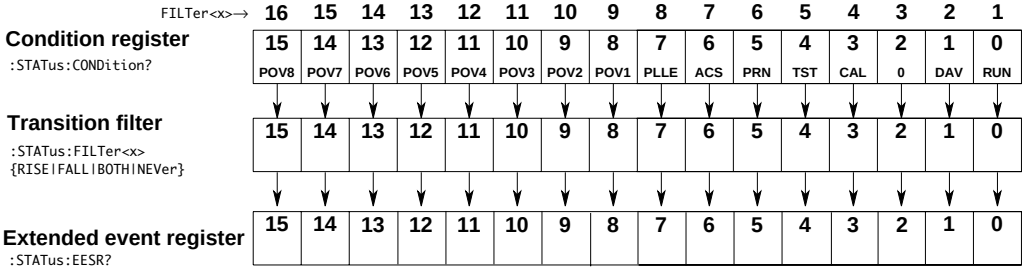
Clearing the Standard Event Register

ïThe 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 1	DAV(numeric Data Available)	Set to “1” when the numerical data are updated. The update is complete when DAV is set.
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, or external SCSI device is being accessed.
Bit 7	PLLE(PLL source input Error)	Set to “1” during harmonic measurement mode, when there is no input at the PLL source and synchronization cannot be achieved.
Bit 8	POV 1(ch1 input Peak Over)	Set to “1” when channel 1 input detects a signal that exceeds the range.
Bit 9	POV 2(ch2 input Peak Over)	Set to “1” when channel 2 input detects a signal that exceeds the range.
Bit 10	POV 3(ch3 input Peak Over)	Set to “1” when channel 3 input detects a signal that exceeds the range.
Bit 11	POV 4(ch4 input Peak Over)	Set to “1” when channel 4 input detects a signal that exceeds the range.
Bit 12	POV 5(ch5 input Peak Over)	Set to “1” when channel 5 input detects a signal that exceeds the range.
Bit 13	POV 6(ch6 input Peak Over)	Set to “1” when channel 6 input detects a signal that exceeds the range.
Bit 14	POV 7(ch7 input Peak Over)	Set to “1” when channel 7 input detects a signal that exceeds the range.
Bit 15	POV 8(ch8 input Peak Over)	Set to “1” when channel 8 input detects a signal that exceeds the range.

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

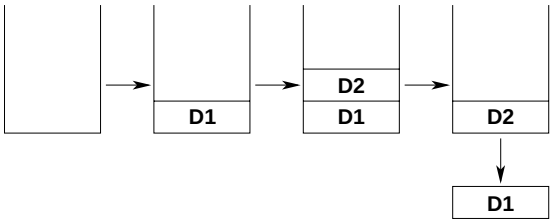
Overview of the Output Queue

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 computer equipped with AT-GPIB/TNT IEEE-488.2 board from National Instruments.
Language: Quick BASIC

Setting up the PZ4000

- **Address 1**
  All the sample programs given in this chapter use address 1 for the PZ4000 so be sure to assign the instrument to address 1 as described on page 1-5.

- **Data Acquisition "STOP"**
  The sample programs in this chapter are written with the premise that the data acquisition on the instrument is in the "STOP" condition. If the data acquisition is in progress, press the "START/STOP" key so that "Stopped" is displayed in the lower left section of the screen. Then, execute the program.

6.2 Example of Normal Measurement Data Output

```
'**************************************************************************
'*                                                                        *
'*  PZ4000 Sample Program1 for GP-IB interface                            *
'*                            Microsoft QuickBASIC 4.0/4.5 Version        *
'*                                                                        *
'**************************************************************************
'*                                                                        *
'*  In the normal measurement mode, set the measurement conditions        *
'*  and start the measurement.                                            *
'*  The following numerical data (ASCII format) are read and displayed    *
'*  on every update.                                                      *
'*      voltage (Urms), current (Irms), active power (P),                 *
'*      apparent power (S), reactive power (Q), power factor (λ),         *
'*      phase difference (ϕ), voltage frequency (fU),                     *
'*      current frequency (fI)                                             *
'*                                                                        *
'**************************************************************************
'
REM $INCLUDE: 'qbdecl4.bas'
'
OPTION BASE 1      ' Minimum value of array subscript = 1
DIM D$(40)   ' Array of numerical data strings
'
DEVICE$ = "DEV1": CALL IBFIND(DEVICE$, PZ%)
CALL IBSIC(PZ%)
BORD$ = "GPIB0": CALL IBFIND(BORD$, BD%)
CALL IBSIC(BD%)
V% = 1: CALL IBSRE(BD%, V%)      ' Remote setting
'
' Set the measurement condition and range
CMD$ = "SETUP:MODE NORMAL"       ' Normal measurement mode
CALL IBWRT(PZ%, CMD$)
CMD$ = "VOLTAGE:RANGE 200"       ' Voltage range = 200Vpk
CALL IBWRT(PZ%, CMD$)
CMD$ = "CURRENT:RANGE 4"  ' Current range = 4Apk
CALL IBWRT(PZ%, CMD$)
CMD$ = "FILTER:LINE OFF"  ' Line filter = OFF
CALL IBWRT(PZ%, CMD$)
CMD$ = "TIMEBASE:OBSERVE 100MS"  ' Observation time = 100msec
CALL IBWRT(PZ%, CMD$)
'
' Set the numerical data output items (ASCII format, preset to pattern 1, number of
output data = 40)
CMD$ = "NUMERIC:FORMAT ASCII;NORMAL:PRESET 1;NUMBER 40"
CALL IBWRT(PZ%, CMD$)
'
' Set the transition filter used to detect the completion of the numerical data updating
CMD$ = "STATUS:FILTER2 RISE"     ' Rising edge of bit1 (DAV)
CALL IBWRT(PZ%, CMD$)
' Clear the extended event register(Read and trash the response)
CMD$ = "STATUS:EESR?"
CALL IBWRT(PZ%, CMD$)
RG$ = SPACE$(8)
CALL IBRD(PZ%, RG$)
'
' Measurement start
CMD$ = "START"
CALL IBWRT(PZ%, CMD$)
'
' Read and display the numerical data (Repeated 10 times in this program)
FOR I% = 1 TO 10
      ' Wait for the completion of the numerical data updating
      CMD$ = "COMMUNICATE:WAIT 2"
      CALL IBWRT(PZ%, CMD$)
      ' Clear the extended event register (Read and trash the response)
      CMD$ = "STATUS:EESR?"
      CALL IBWRT(PZ%, CMD$)
      RG$ = SPACE$(8)
      CALL IBRD(PZ%, RG$)
      '
      ' Read out numerical data
      CMD$ = "NUMERIC:NORMAL:VALUE?"
      CALL IBWRT(PZ%, CMD$)
      RES$ = SPACE$(1000)
      CALL IBRD(PZ%, RES$)
      '
      ' Extract items that are separated by commas (,) from the received numerical data
      C$ = LEFT$(RES$, IBCNT%)                    ' IBCNT% = Number of received bytes
      FOR J% = 1 TO 40
            L% = LEN(C$)
            B% = INSTR(C$, ",")
            IF B% = 0 THEN B% = L% + 1
            D$(J%) = LEFT$(C$, (B% - 1))
            C$ = MID$(C$, (B% + 1))
      NEXT J%
```

```
      '
      ' Display the numerical data
      PRINT I%,"Element1","Element2","Element3","Element4"
      PRINT "Urms [V]", D$(1), D$(11), D$(21), D$(31)
      PRINT "Irms [A]", D$(2), D$(12), D$(22), D$(32)
      PRINT "P    [W]", D$(3), D$(13), D$(23), D$(33)
      PRINT "S   [VA]", D$(4), D$(14), D$(24), D$(34)
      PRINT "Q  [var]", D$(5), D$(15), D$(25), D$(35)
      PRINT "Lambda[]", D$(6), D$(16), D$(26), D$(36)
      PRINT "Phi    []", D$(7), D$(17), D$(27), D$(37)
      PRINT "fU  [Hz]", D$(8), D$(18), D$(28), D$(38)
      PRINT "fI  [Hz]", D$(9), D$(19), D$(29), D$(39)
      PRINT
NEXT I%
'
' Measurement stop
CMD$ = "STOP"
CALL IBWRT(PZ%, CMD$)
'
V% = 0: CALL IBSRE(BD%, V%)     ' Clear remote mode
'
END


'************************************************************************
'*                                                                      *
'*  PZ4000 Sample Program1 for serial(RS-232) interface                 *
'*                              Microsoft QuickBASIC 4.0/4.5 Version    *
'*                                                                      *
'*         Rate:9600 Parity:None CHR:8 STOPBIT:1 XON/XON Term:CR+LF     *
'*                                                                      *
'************************************************************************
'*                                                                      *
'*  In the normal measurement mode, set the measurement conditions      *
'*  and start the measurement.                                          *
'*  The following numerical data (ASCII format) are read and displayed *
'*  on every update.                                                    *
'*      voltage (Urms), current (Irms), active power (P),               *
'*      apparent power (S), reactive power (Q), power factor (λ),       *
'*      phase difference (ϕ), voltage frequency (fU),                   *
'*      current frequency(fI)                                           *
'*                                                                      *
'************************************************************************
'
OPEN "COM1:9600,N,8,1,ASC,CS0,DS0,LF" FOR RANDOM AS #1
'
OPTION BASE 1      ' Minimum value of array subscript=1
DIM D$(40)   ' Array of numerical data strings
'
PRINT #1, "COMMUNICATE:REMOTE ON"      ' Remote setting
'
' Set the measurement condition and range
PRINT #1, "SETUP:MODE NORMAL"    ' Normal measurement mode
PRINT #1, "VOLTAGE:RANGE 200"    ' Voltage range = 200 Vpk
PRINT #1, "CURRENT:RANGE 4"      ' Current range = 4 Apk
PRINT #1, "FILTER:LINE OFF"      ' Line filter=OFF
PRINT #1, "TIMEBASE:OBSERVE 100MS"     ' Observation time = 100 msec
'
' Set the numerical data output items (ASCII format, preset to pattern 1, number of
output data=40)
PRINT #1, "NUMERIC:FORMAT ASCII;NORMAL:PRESET 1;NUMBER 40"
'
' Set the transition filter used to detect the completion of the numerical data updating
PRINT #1, "STATUS:FILTER2 RISE"        ' Rising edge of bit1 (DAV)
' Clear the extended event register (Read and trash the response)
PRINT #1, "STATUS:EESR?"
LINE INPUT #1, RG$
'
' Measurement start
PRINT #1, "START"
'
' Read and display the numerical data (It is repeated 10 times in this program)
FOR I% = 1 TO 10
      ' Wait for the completion of the numerical data updating
      PRINT #1, "COMMUNICATE:WAIT 2"
      ' Clear the extended event register(Read and trash the response)
      PRINT #1, "STATUS:EESR?"
      LINE INPUT #1, RG$
      '
      ' Read out numerical data
      PRINT #1, "NUMERIC:NORMAL:VALUE?"
      '
      ' Receive the items of numerical data that are separated by commas (,)
      FOR J% = 1 TO 40
            INPUT #1, D$(J%)
      NEXT J%
```

```
        '
        ' Display the numerical data
        PRINT I%,"Element1","Element2","Element3","Element4"
        PRINT "Urms [V]", D$(1), D$(11), D$(21), D$(31)
        PRINT "Irms [A]", D$(2), D$(12), D$(22), D$(32)
        PRINT "P    [W]", D$(3), D$(13), D$(23), D$(33)
        PRINT "S   [VA]", D$(4), D$(14), D$(24), D$(34)
        PRINT "Q  [var]", D$(5), D$(15), D$(25), D$(35)
        PRINT "Lambda[]", D$(6), D$(16), D$(26), D$(36)
        PRINT "Phi   []", D$(7), D$(17), D$(27), D$(37)
        PRINT "fU  [Hz]", D$(8), D$(18), D$(28), D$(38)
        PRINT "fI  [Hz]", D$(9), D$(19), D$(29), D$(39)
        PRINT
NEXT I%
'
' Measurement stop
PRINT #1, "STOP"
'
PRINT #1, "COMMUNICATE:REMOTE OFF"     ' Clear remote mode
'
CLOSE #1
'
END


Output example
1             Element1      Element2      Element3      Element4
Urms [V]      102.44E+00    103.67E+00    104.32E+00    103.68E+00
Irms [A]      1.1224E+00    0.8108E+00    1.1202E+00    1.1052E+00
P    [W]      86.11E+00     55.58E+00     87.54E+00     85.82E+00
S   [VA]      114.98E+00    84.06E+00     116.87E+00    114.59E+00
Q  [var]      76.19E+00     63.06E+00     77.42E+00     75.93E+00
Lambda[]      0.7489E+00    0.6612E+00    0.7491E+00    0.7490E+00
Phi   []      41.50E+00     311.39E+00    41.49E+00     41.50E+00
fU  [Hz]      50.008E+00    50.008E+00    50.009E+00    50.009E+00
fI  [Hz]      49.975E+00    50.018E+00    49.985E+00    49.978E+00
:             :             :             :             :
:             :             :             :             :
```

6.3 Example of Harmonic Measurement Data Output

```
'************************************************************************
'*                                                                      *
'*  PZ4000 Sample Program2 for GP-IB interface                          *
'*                              Microsoft QuickBASIC 4.0/4.5 Version    *
'*                                                                      *
'************************************************************************
'*                                                                      *
'*  In the harmonic measurement mode, set the measurement conditions    *
'*  and perform one measurement.                                        *
'*  The following numerical data regarding the current of element 1     *
'*  are read and displayed.                                             *
'*      PLL source frequency (the current frequency of element 1 in     *
'*      this program), total harmonic distortion (Ithd1), total rms     *
'*      value (I1(Total)), DC component (I1(dc)),                       *
'*      fundamental signal(I1(1)), analyzed values from 2nd to 100th    *
'*      order (I1(2) to I1(100))                                        *
'*                                                                      *
'************************************************************************
'
REM $INCLUDE: 'qbdecl4.bas'
'
DIM D$(100)  ' Array of numerical data strings
'
DEVICE$ = "DEV1": CALL IBFIND(DEVICE$, PZ%)
CALL IBSIC(PZ%)
BORD$ = "GPIB0": CALL IBFIND(BORD$, BD%)
CALL IBSIC(BD%)
V% = 1: CALL IBSRE(BD%, V%)      ' Remote setting
'
' Set the measurement conditions
CMD$ = "SETUP:MODE HARMONICS"    ' Harmonic measurement mode
CALL IBWRT(PZ%, CMD$)
CMD$ = "SETUP:PLLSOURCE 2"       ' PLL source =CH2(I1)
CALL IBWRT(PZ%, CMD$)
CMD$ = "MEASURE:HARMONICS:ORDER 0,100" ' Harmonic orders analyzed =0 to 100
CALL IBWRT(PZ%, CMD$)
'
' Set the numerical data output items
CMD$ = "NUMERIC:FORMAT ASCII"    ' ASCII format
CALL IBWRT(PZ%, CMD$)
CMD$ = "NUMERIC:HARMONICS:PRESET 4"    ' Settings to output fI, Ithd1
CALL IBWRT(PZ%, CMD$)
CMD$ = "NUMERIC:LIST:ITEM I,1;ORDER 100;SELECT ALL" ' Numerical data of I1 from Total to
100th order
CALL IBWRT(PZ%, CMD$)
'
' Set the transition filter used to detect the completion of the numerical data updating
CMD$ = "STATUS:FILTER2 RISE"     ' Rising edge of bit1 (DAV)
CALL IBWRT(PZ%, CMD$)
' Clear the extended event register(Read and trash the response)
CMD$ = "STATUS:EESR?"
CALL IBWRT(PZ%, CMD$)
RG$ = SPACE$(8)
CALL IBRD(PZ%, RG$)
'
' Single measurement start
CMD$ = "SSTART"
CALL IBWRT(PZ%, CMD$)
'
' Wait for the completion of the numerical data updating
CMD$ = "COMMUNICATE:WAIT 2"
CALL IBWRT(PZ%, CMD$)
' Clear the extended event register(Read and trash the response)
CMD$ = "STATUS:EESR?"
CALL IBWRT(PZ%, CMD$)
RG$ = SPACE$(8)
CALL IBRD(PZ%, RG$)
'
' Read out the PLL source frequency (fI1)
CMD$ = "NUMERIC:HARMONICS:VALUE? 20"   ' Patter 4 ITEM20=fI1
CALL IBWRT(PZ%, CMD$)
RES$ = SPACE$(20)
CALL IBRD(PZ%, RES$)
PLL$ = LEFT$(RES$, (IBCNT% - 1))
'
' Read out the total harmonic distortion (Ithd1)
CMD$ = "NUMERIC:HARMONICS:VALUE? 27"   ' Pattern 4 ITEM27=Ithd1
CALL IBWRT(PZ%, CMD$)
RES$ = SPACE$(20)
CALL IBRD(PZ%, RES$)
THD$ = LEFT$(RES$, (IBCNT% - 1))
```

```
'
' Read out the harmonic numerical list data (I1(Total) to I1(100))
CMD$ = "NUMERIC:LIST:VALUE?"     ' All 102 data
CALL IBWRT(PZ%, CMD$)
RES$ = SPACE$(1200)
CALL IBRD(PZ%, RES$)
C$ = LEFT$(RES$, IBCNT%)
' Extract items that are separated by commas (,) from the received numerical data
B% = INSTR(C$, ",")' Total
TOTAL$ = LEFT$(C$, (B% - 1))
C$ = MID$(C$, (B% + 1))
FOR I% = 0 TO 100   ' 0(dc) to 100
      L% = LEN(C$)
      B% = INSTR(C$, ",")
      IF B% = 0 THEN B% = L% + 1
      D$(I%) = LEFT$(C$, (B% - 1))
      C$ = MID$(C$, (B% + 1))
NEXT I%
'
' Display the numerical data
PRINT "Freq[Hz]", PLL$    ' PLL source frequency
PRINT "Ithd [%]", THD$    ' Total harmonic distorion
PRINT "Total[A]", TOTAL$  ' Total rms value
PRINT "dc   [A]", D$(0)   ' DC component
FOR I% = 1 TO 100 STEP 2
      PRINT "Or." + STR$(I%), D$(I%),  ' Odd order components
      PRINT "Or." + STR$(I% + 1), D$(I% + 1)  ' Even order compenents
NEXT I%
PRINT
'
V% = 0: CALL IBSRE(BD%, V%)      ' Clear remote mode
'
END


Output example
Freq[Hz]      50.251E+00
Ithd [%]      70.31E+00
Total[A]      1.9485E+00
dc   [A]      1.3834E+00
Or. 1         0.0772E+00    Or. 2         1.0735E+00
Or. 3         0.0821E+00    Or. 4         0.1141E+00
Or. 5         0.0742E+00    Or. 6         0.4238E+00
Or. 7         0.0726E+00    Or. 8         0.1510E+00
Or. 9         0.0892E+00    Or. 10        0.3400E+00
:             :             :             :
:             :             :             :
Or. 99        0.0601E+00    Or. 100       0.0131E+0
```

6.4 Output Example of Waveform Data in ASCII Format

```
'************************************************************************
'*                                                                      *
'*  PZ4000 Sample Program3 for GP-IB interface                          *
'*                              Microsoft QuickBASIC 4.0/4.5 Version    *
'*                                                                      *
'************************************************************************
'*                                                                      *
'*  Read the CH1(U1) waveform data from PZ4000 in ASCII format          *
'*                                                                      *
'************************************************************************
'
REM $INCLUDE: 'qbdecl4.bas'
'
DEVICE$ = "DEV1": CALL IBFIND(DEVICE$, PZ%)
CALL IBSIC(PZ%)
BORD$ = "GPIB0": CALL IBFIND(BORD$, BD%)
CALL IBSIC(BD%)
V% = 1: CALL IBSRE(BD%, V%)      ' Set to remote
'
' Set conditions for reading the waveform
CMD$ = "WAVEFORM:TRACE 1;FORMAT ASCII" ' Target waveform=CH1, ASCII format
CALL IBWRT(PZ%, CMD$)
'
' Query the total number of data points that can be read
CMD$ = "COMMUNICATE:HEADER OFF"
CALL IBWRT(PZ%, CMD$)
CMD$ = "WAVEFORM:LENGTH?"
CALL IBWRT(PZ%, CMD$)
LN$ = SPACE$(10)
CALL IBRD(PZ%, LN$)
B% = INSTR(LN$, CHR$(10))
L& = VAL(LEFT$(LN$, B% - 1))
'
' Read in the waveform data 10 data points at a time
IF L& = 0 THEN GOTO WAVEEXIT
WAV$ = SPACE$(200)
CN& = 0
FOR I& = 0 TO (L& - 2) STEP 10
      CMD$ = "WAVEFORM:START" + STR$(I&) + ";END" + STR$(I& + 9) + ";SEND?"
      CALL IBWRT(PZ%, CMD$)
      CALL IBRD(PZ%, WAV$)
      K% = 1
      FOR J% = 0 TO 9
            IF J% < 9 THEN S% = INSTR(K%, WAV$, ",") ELSE S% = INSTR(K%, WAV$,
CHR$(10))
            CN& = CN& + 1
            PRINT CN&, MID$(WAV$, K%, (S% - K%))
            K% = S% + 1
      NEXT J%
NEXT I&
'
WAVEEXIT:
V% = 0: CALL IBSRE(BD%, V%)      ' Clear remote mode
'
END
```

```
'************************************************************************
'*                                                                      *
'*  PZ4000 Sample Program3 for serial(RS-232) interface                 *
'*                              Microsoft QuickBASIC 4.0/4.5 Version    *
'*                                                                      *
'*         Rate:9600 Parity:None CHR:8 STOPBIT:1 XON/XON Term:CR+LF     *
'*                                                                      *
'************************************************************************
'*                                                                      *
'*  Read the CH1(U1) waveform data from PZ4000 in ASCII format          *
'*                                                                      *
'************************************************************************
'
OPEN "COM1:9600,N,8,1,ASC,CS0,DS0,LF" FOR RANDOM AS #1
'
PRINT #1, "COMMUNICATE:REMOTE ON"      ' Set to remote
'
' Set conditions for reading the waveform
PRINT #1, "WAVEFORM:TRACE 1;FORMAT ASCII"     ' Target waveform=CH1, ASCII format
'
' Query the total number of data points that can be read
PRINT #1, "COMMUNICATE:HEADER OFF"
PRINT #1, "WAVEFORM:LENGTH?"
LINE INPUT #1, LN$
L& = VAL(LN$)
'
' Read in the waveform data
IF L& = 0 THEN GOTO WAVEEXIT
PRINT #1, "WAVEFORM:START 0" + ";END" + STR$(L& - 1) + ";SEND?"
FOR I& = 1 TO L&
      INPUT #1, WAV$
      PRINT I&, WAV$
NEXT I&
'
WAVEEXIT:
PRINT #1, "COMMUNICATE:REMOTE OFF"     ' Clear remote mode
'
CLOSE #1
'
END


Output example
 1            1.8311E+00
 2            2.0752E+00
 3            1.8311E+00
 4            2.0752E+00
 5            1.9531E+00
 6            2.1973E+00
 7            2.3193E+00
 8            2.3193E+00
 9            2.3193E+00
 10           2.0752E+00
 :            :
 :            :
 100000       -2.0752E+00
```

6.5 Output Example of Waveform Data in Binary Format

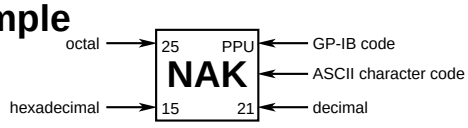
```
'************************************************************************
'*                                                                      *
'*  PZ4000 Sample Program4 for GP-IB interface                          *
'*                              Microsoft QuickBASIC 4.0/4.5 Version    *
'*                                                                      *
'************************************************************************
'*                                                                      *
'*  Read the CH1(U1) waveform data from PZ4000 in binary (WORD) format *
'*                                                                      *
'************************************************************************
'
REM $INCLUDE: 'qbdecl4.bas'
'
DEVICE$ = "DEV1": CALL IBFIND(DEVICE$, PZ%)
CALL IBSIC(PZ%)
BORD$ = "GPIB0": CALL IBFIND(BORD$, BD%)
CALL IBSIC(BD%)
V% = 1: CALL IBSRE(BD%, V%)      ' Set to remote
'
' Set conditions for reading the waveform
CMD$ = "WAVEFORM:TRACE 1;FORMAT BINARY;BYTEORDER LSBFIRST" ' Target waveform=CH1, WORD
format
CALL IBWRT(PZ%, CMD$)
'
' Query the range value (needed to convert binary data to physical values)
CMD$ = "COMMUNICATE:HEADER OFF"
CALL IBWRT(PZ%, CMD$)
CMD$ = "WAVEFORM:RANGE?"
CALL IBWRT(PZ%, CMD$)
RNG$ = SPACE$(20)
CALL IBRD(PZ%, RNG$)
B% = INSTR(RNG$, CHR$(10))
R! = VAL(LEFT$(RNG$, B% - 1))
'
' Query the total number of data points that can be read
CMD$ = "WAVEFORM:LENGTH?"
CALL IBWRT(PZ%, CMD$)
LN$ = SPACE$(10)
CALL IBRD(PZ%, LN$)
B% = INSTR(LN$, CHR$(10))
L& = VAL(LEFT$(LN$, B% - 1))
'
' Read in the waveform data 100 data points at a time
IF L& = 0 THEN GOTO WAVEEXIT
WAV$ = SPACE$(220)
CN& = 0
FOR I& = 0 TO (L& - 2) STEP 100
      CMD$ = "WAVEFORM:START" + STR$(I&) + ";END" + STR$(I& + 99) + ";SEND?"
      CALL IBWRT(PZ%, CMD$)
      CALL IBRD(PZ%, WAV$)
      FOR J% = 0 TO 99
            CN& = CN& + 1
            PRINT CN&, (CVI(MID$(WAV$, J% * 2 + 11, 2)) - 2048) / 2048! * R!
      NEXT J%
NEXT I&
'
WAVEEXIT:
V% = 0: CALL IBSRE(BD%, V%)      ' Clear remote mode
'
END


Output example
1             1.831055
2             2.075195
3             1.831055
4             2.075195
5             1.953125
6             2.197266
7             2.319336
8             2.319336
9             2.319336
10            2.075195
:             :
:             :
100000        -2.075195
```

Appendix 1 ASCII Character Code

ASCII character codes are given

	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 253710-01E.

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
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 Communications-driver	Servicing is required.	—

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 Serial(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.

App

Appendix

Appendix 3 Overview of IEEE 488.2-1987

The GP-IB interface provided with PZ4000 conforms to IEEE 488.2-1987. 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 PZ4000 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
Refer to Chapter 4, “Command List”.

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

No communications between devices.

11 Size of data block of response data

1 to 16000004 (4000001×4) bytes

12 List of supported common commands

Refer to Section 4.26 “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

Refer to Section 4.26 “Common Command Group”.

*LRN?, *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 253710-01E.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z