HIOKI

INSTRUCTION MANUAL

For 8835(-01), 8826, 8841, 8842
MEMORY HiCORDER
INTERFACE

9557 RS-232C CARD
9558 GP-IB CARD

HIOKI E. E. CORPORATION

Contents

Introduction

Thank you for purchasing the HIOKI "9557 RS-232C CARD / 9558 GP-IB CARD" . To obtain maximum performance from the product, please read this manual first, and keep it handy for future reference.

When using the HIOKI MEMORY HiCORDER can be used with the HIOKI "9557 RS-232C CARD / 9558 GP-IB CARD" except following products, reffer to the communication comands manual (Floppy disk) supplied with the MEMORY HiCORDER.

■ The products consultable this manual:
8826, 8835, 8835-01, 8841, 8842

Safety Notes

This manual contains information and warnings essential for safe operation of the product and for maintaining it in safe operating condition. Before using the product, be sure to carefully read the following safety notes.

⚠ DANGER	**This product is designed to conform to IEC 61010 Safety Standards, and has been thoroughly tested for safety prior to shipment. However, mishandling during use could result in injury or death, as well as damage to the product. Be certain that you understand the instructions and precautions in the manual before use. We disclaim any responsibility for accidents or injuries not resulting directly from product defects.**

Safety symbol

	In the manual, the ⚠ symbol indicates particularly important information that the user should read before using the product.

The following symbols in this manual indicate the relative importance of cautions and warnings.

⚠ DANGER	Indicates that incorrect operation presents an extreme hazard that could result in serious injury or death to the user.
⚠ CAUTION	Indicates that incorrect operation presents a possibility of injury to the user or damage to the product.
NOTE	Indicates advisory items related to performance or correct operation of the product.

Chapter Summary

Chapter 1 GP-IB and RS-232C interfaces
Contains the functions and specifications of both the interfaces.

Chapter 2 Method of operation
Describes the operation procedures of both the interfaces.

Chapter 3 Commands
Describes the details of all the commands th at can be used.

Chapter 4 Example programs
Describes the program to operate GP-IB interface.

Appendix
Contains the information related to the IEEE488.2-1987 standard.

Chapter 1
GP-IB and RS-232C Interfaces

1.1 GP-IB Interface

1.1.1 Outline

The GP-IB (General Purpose Interface Bus) was developed as an interface for general use by programmable instrumentation, and as an interface is rich in expandability and has many distinctive features.

There are various interfaces with specific names apart from the GP-IB, such as the IEEE-488 bus, the IEC bus, and the HP-IB which is an internal standard within the Hewlett-Packard Company. These are basically the same standard, but, because the number of connector pins and the arrangement of the signals and so on differ, much care should be exercised.

In this explanation of management and operation, only the GP-IB related resources of the 8835 and 8826 will be described.

If more detailed knowledge of the GP-IB interface is required, reference should be made to the following literature:
The Institute of Electrical and Electronics Engineers, Inc.: "IEEE Standard Digital Interface for Programmable Instrumentation", IEEE Std 488.1-1987, IEEE Std 488.2-1987 (1987)

1.1.2 Specification

Standards

IEEE Standard 488.1-1987
IEEE Standard 488.2-1987

Interface Functions

Function	Implementation
SH1	SH (Source Handshake) - All Functions
AH1	AH (Acceptor Handshake) - All Functions
T5	Basic Talk Function, Serial Poll Function, Talk Only Function MLA (My Listen Address) Talk Release Function
L4	Basic Listener Function MTA (My Talk Address) Listen Release Function
SR1	SR (Service Request) - All Functions
RL1	RL (Remote/Local) - All Functions
PP0	PP (Parallel Poll) - No Function
DC1	DC (Device Clear) - All Functions
DT0	DT (Device Trigger) - No Function
C0	C (Control) - No Function

GP-IB Signal Lines

Bus Signal Lines		Remarks	
Data bus	DIO 1 (Data Input Output 1) DIO 2 (Data Input Output 2) DIO 3 (Data Input Output 3) DIO 4 (Data Input Output 4) DIO 5 (Data Input Output 5) DIO 6 (Data Input Output 6) DIO 7 (Data Input Output 7) DIO 8 (Data Input Output 8)	Apart from input and output of data, these are used for input and output of interface messages and device messages.	
Transfer bus	DAV (Data Valid)	Signal which indicates data bus information validity.	These perform acceptor and source handshake.
	NRFD (Not Ready For Data)	Input preparation completed signal.	
	NDAC (Not Data Accepted)	Input completed signal.	
Control bus	ATN (Attention)	Signal which indicates that the information on the data bus is an interface message or a device message.	
	IFC (Interface Clear)	Signal which sets the interface bus system to the initial condition.	
	SRQ (Service Request)	Signal which requests a non-synchronous service.	
	REN (Remote Enable)	Signal which performs changeover of remote and local control.	
	EOI (End or Identify)	Indicates the last byte of data.	

Connector Pin Assignment

RC40-24RR (made by HIROSE) or compatible.

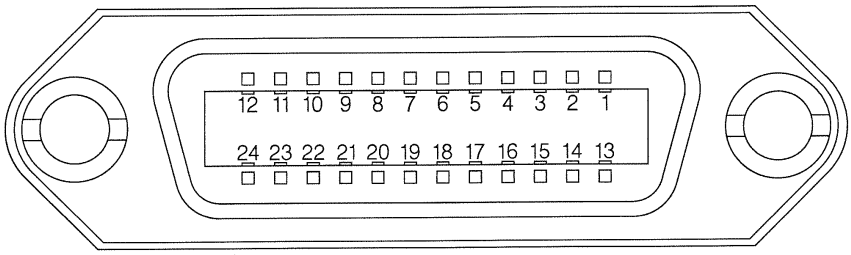


Fig. 1.1 Pin arrangement diagram for the GP-IB interface connector

Pin number	Name of signal line	Pin number	Name of signal line
1	DIO1	13	DIO5
2	DIO2	14	DIO6
3	DIO3	15	DIO7
4	DIO4	16	DIO8
5	EOI	17	REN
6	DAV	18	GND
7	NRFD	19	GND
8	NDAC	20	GND
9	IFC	21	GND
10	SRQ	22	GND
11	ATN	23	GND
12	SHIELD	24	LOGIC GND

1.2 RS-232C Interface

1.2.1 Outline

RS-232C is a serial interface standard defined by the EIA (Electronic Industries Association). It specifies the interface parameters for communication between a DTE (Data Terminal Equipment) and DCE (Data Communications Equipment).
The MEMORY HiCORDER incorporates a partial implementation of the RS-232C specification (only certain signal lines) to allow data exchange and remote control using a personal computer.

1.2.2 Specification

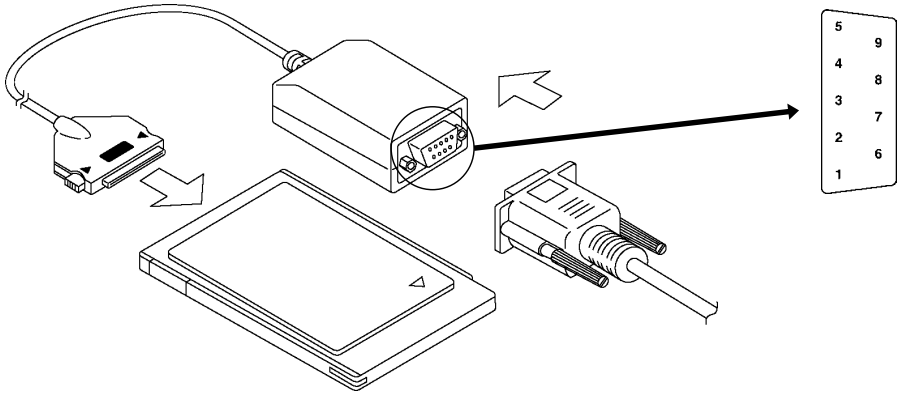
Standard
EIA RS-232C

General Specifications

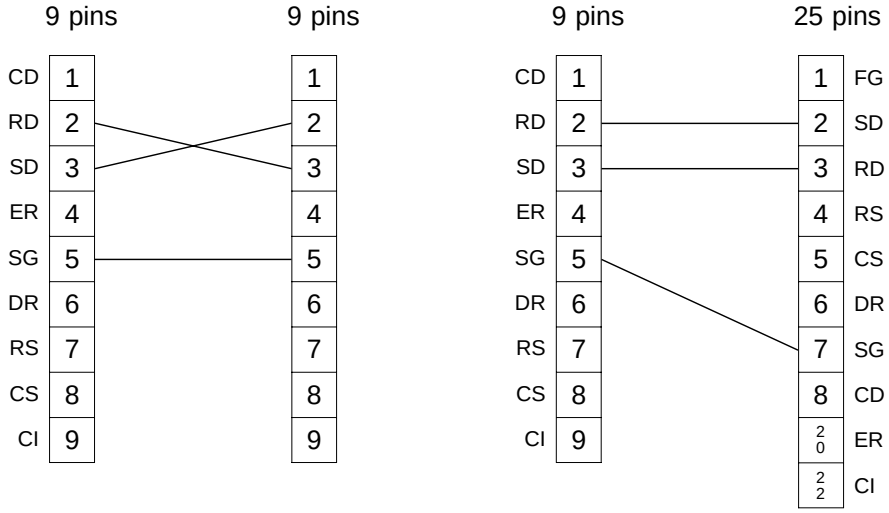
Communication mode	Full-duplex
Synchronization	Start-stop synchronization
Transfer rate	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 (bits/s), (set from the setting screen of the unit)
Start bit	1 bit
Stop bits	1 or 2 bits (set from the setting screen of the unit)
Data length	7 or 8 bits
Parity	None, even, or odd (set from the setting screen of the unit)
Delimiter	LF, CR+LF
Flow control	Xon/Xoff, hardware, none

Connector Pin Assignment

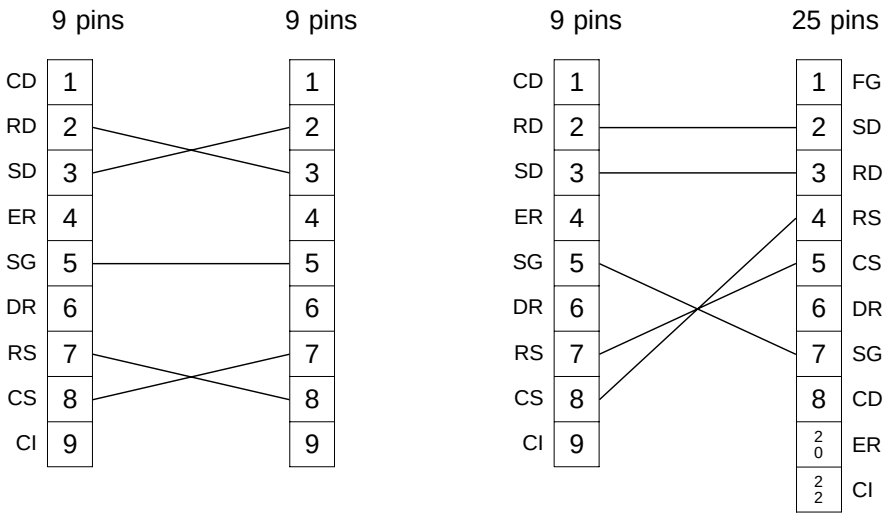
The connector on the PC card is a D-sub 9-pin connector (male).
Make connection to the computer using a null-modem cable with the type of connector that matches the computer.



OFF/Xon-Xoff flow control



Hard flow control



25 pins

Pin number	Circuit designation	CCITT circuit number	EIA symbol	JIS symbol	Common symbol
1	Protective ground	101	AA	-	FG
2	Transmitted data	103	BA	SD	TxD
3	Received data	104	BB	RD	RxD
4	Request to send	105	CA	RS	RTS
5	Clear to send	106	CB	CS	CTS
7	Signal ground	102	AB	SG	GND

9 pins

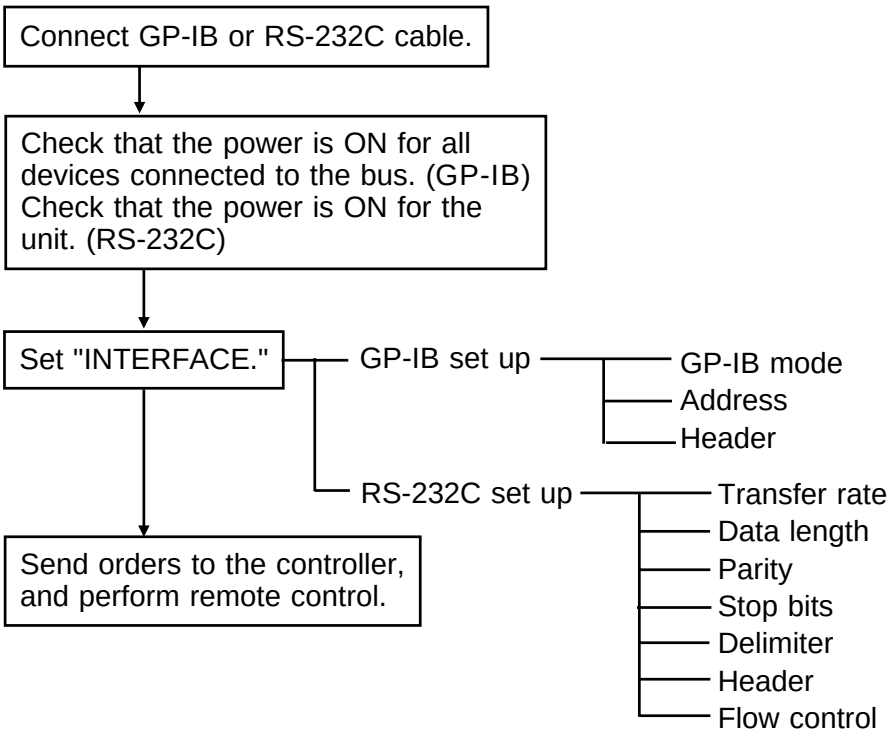
Pin number	Circuit designation	CCITT circuit number	EIA symbol	JIS symbol	Common symbol
2	Received data	104	BB	RD	RxD
3	Transmitted data	103	BA	SD	TxD
5	Signal ground	102	AB	SG	GND
7	Request to send	105	CA	RS	RTS
8	Clear to send	106	CB	CS	CTS

2

Chapter 2
Method of Operation

2.1 Basic Operational Procedure

> **⚠ CAUTION** The GP-IB or RS-232C interface is not isolated from the unit system. Exercise caution, because the ground of the logic inputs and the GP-IB or RS-232C interface ground are connected.



2.2 Cable Connection

> **⚠ CAUTION** When making the connection, the cable connector and PC card should be properly aligned, so that the connector can be pushed in straight. Do not exert strong force on the PC card connector, to prevent the possibility of damage and contact problems.

This section explains procedures for connecting the cable using the 8835 as an example. For other models, refer to the instruction manual included with the unit.

(1) Cable and PC card connection

1. Pass the PC card protector through the connection cable, as shown below.

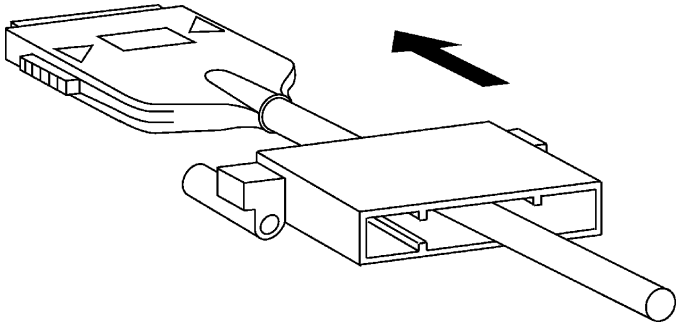


Fig. 2.1 Passing protector through connection cable

2. Plug the PC card end of the connection cable into the PC card. The top side of the cable connector (marked with a ▲) should match the top side of the PC card, as shown below.

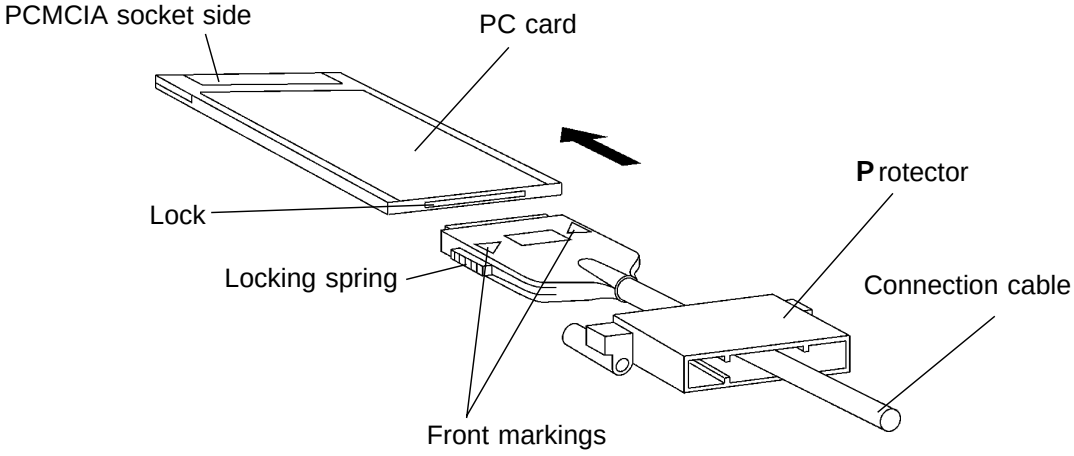


Fig. 2.2 Connection cable and PC card connection

(2) Inserting the PC card

⚠ CAUTION

The following actions may result in damage to the PC card or connector and must be avoided.

- Inserting the card with the wrong orientation or in other ways than described above.
- Inserting the card while attached to the connection cable.
- Moving the unit while the connection cable is connected to the card.
- Pulling the card out by the cable or exerting excessive force on the connector.
- Placing objects on the connection cable connector.

1. Insert the PC card in the PC card slot on the unit. Verify that the ▲ mark on the card points in the correct direction as shown below, and make sure that the card is properly seated in the slot.
   The PC card is keyed to prevent wrong insertion, but exerting excessive force may damage the card or the slot.

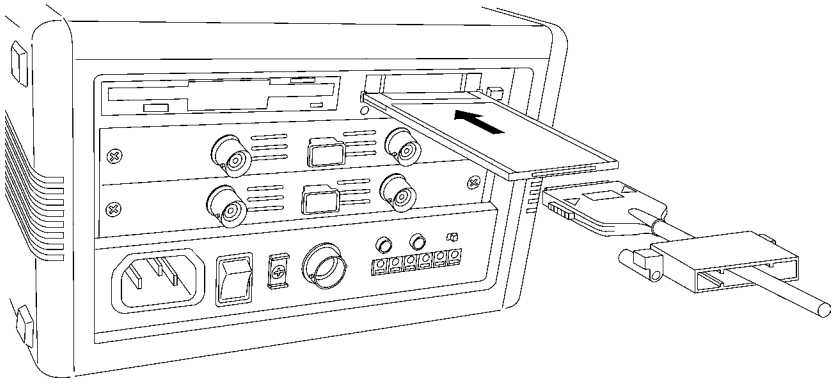


Fig. 2.3 PC card insertion

2. Attach the PC card protector to the unit as shown below.

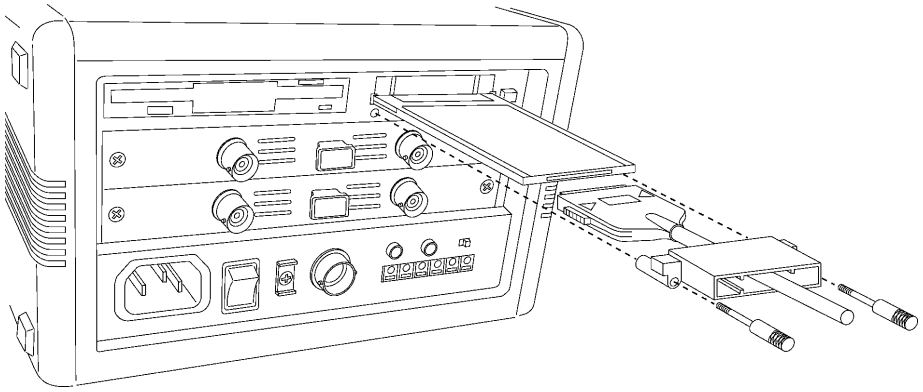


Fig. 2.4 Attaching the protector

(3) Removing the PC card

1. Remove the PC card protector as shown below.

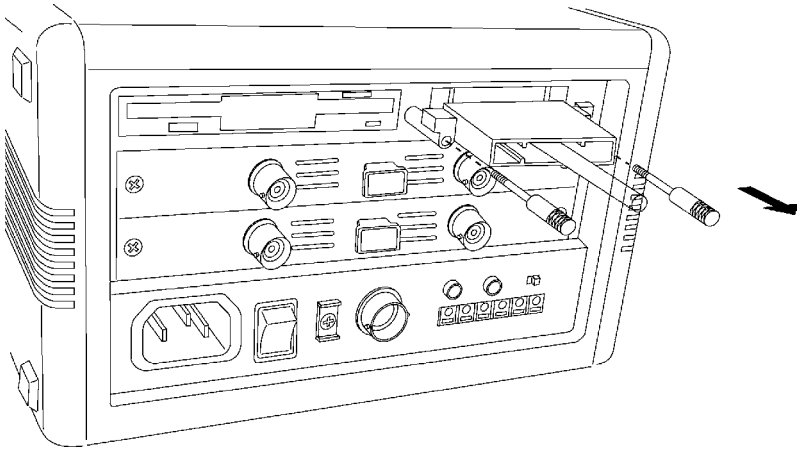


Fig. 2.5 Removing the protector

2. To remove the PC card, press the eject button as shown below.

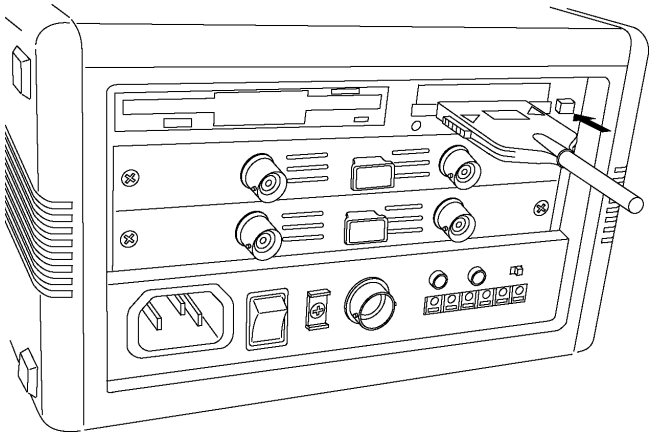


Fig. 2.6 Removing the PC card

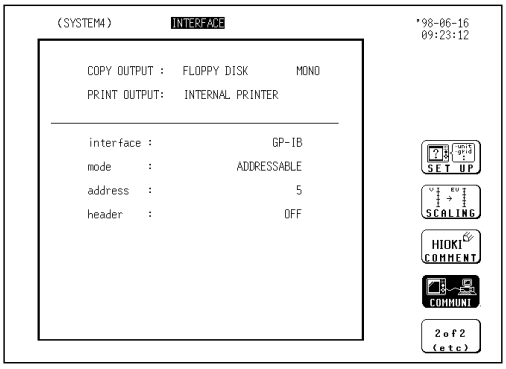
NOTE Do not press the eject button before removing the PC card protector.

2.3 Setup Procedure

2.3.1 GP-IB Setup Procedure

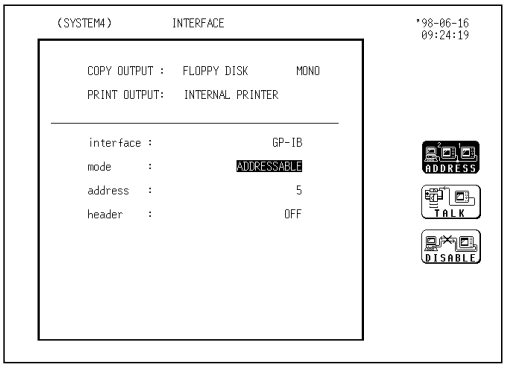
- On the unit, set the GP-IB address for the unit, and select whether or not to use headers mode, and delimiter in messages output by the unit.
- Use the interface setting screen, accessed from the "system" screen.
- This section explains procedures for setting the GP-IB using the 8835 as an example. For other models, refer to the instruction manual included with the unit.

Method



System screen (INTERFACE)

1. Press the **SYSTEM** key to call up the interface setting screen.

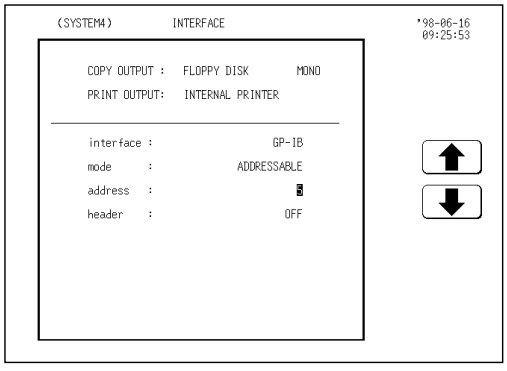


2. Set the GP-IB operation mode for this unit.
   Set the GP-IB address for this unit on the bus.
   [ADDRESSABLE, TALK ONLY, DISABLE]
   Move the flashing cursor to the position shown in the figure on the left, and use the function keys to make the setting.

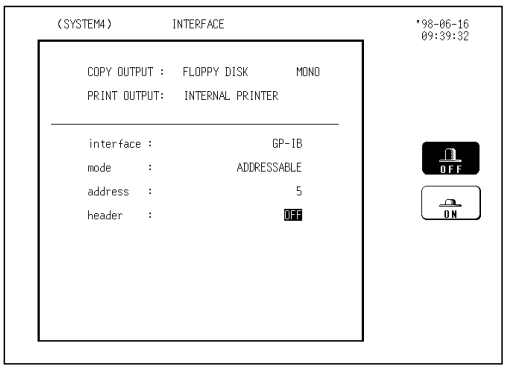
   ADDRESS : (ADDRESSABLE) Assign a device address, so this unit can be used both as talker and listener.

   TALK : (TALK ONLY) Use this unit as talker only (used when transmitting the BMP data).

   DISABLE : (DISABLE) Do not use the GP-IB interface.



3. Set the GP-IB device address.
   Move the flashing cursor to the position shown in the figure on the left, and use the function keys or the jog control to adjust the numerical value. [0 to 30]



4. Enable or disable the headers.
   Select whether or not this unit as talker should output an identifying header at the beginning of each message it sends.
   Move the flashing cursor to the position shown in the figure on left, and use the function keys to make the setting.

   OFF : Header information is not appended.

   ON : Header information is appended.

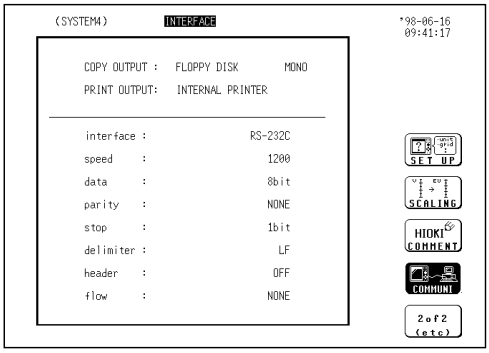
NOTE

- The unit automatically recognizes which type of PC card is inserted, and the appropriate setting items appear on the display. Perform the setting procedure after inserting the GP-IB card.
- Do not change the settings during communications.

2.3.2 RS-232C Setup Procedure

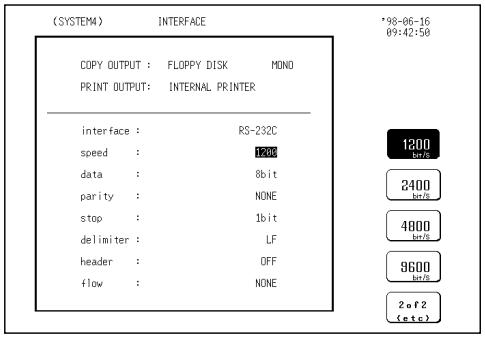
- On the unit, make the settings for the RS-232C transfer rate, data length, parity, stop bits, delimiter and flow control
- Use the interface setting screen, accessed from the "system" screen.
- This section explains procedures for setting the RS-232C using the 8835 as an example. For other models, refer to the instruction manual included with the unit.

Method

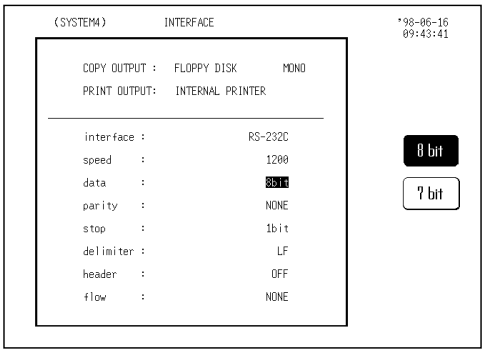


System screen (INTERFACE)

1. Press the **SYSTEM** key to call up the interface setting screen.



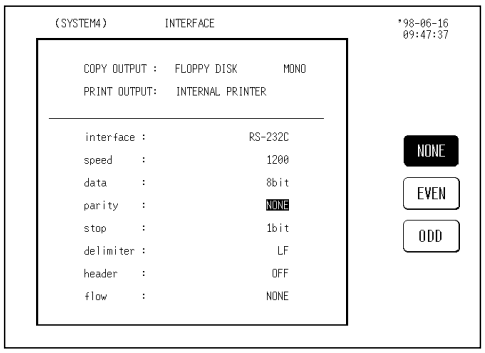
2. Set the transfer rate.
   Move the flashing curson to the position shown in the figure on the left, and use the function keys to make the selection.



3. Set the data length.
   Move the flashing curson to the position shown in the figure on the left, and use the function keys to make the selection.

   8 bit : Sets the data length to 8 bits.

   7 bit : Sets the data length to 7 bits.

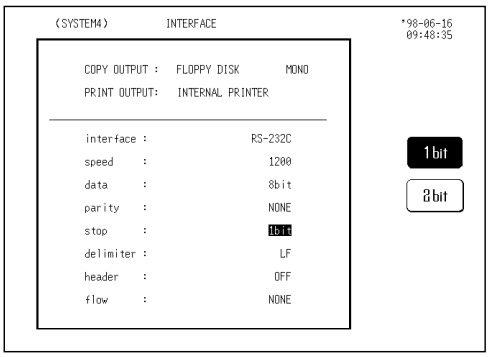


4. Set the parity.
   Move the flashing curson to the position shown in the figure on the left, and use the function keys to make the selection.

   NONE : No parity

   EVEN : Even number parity

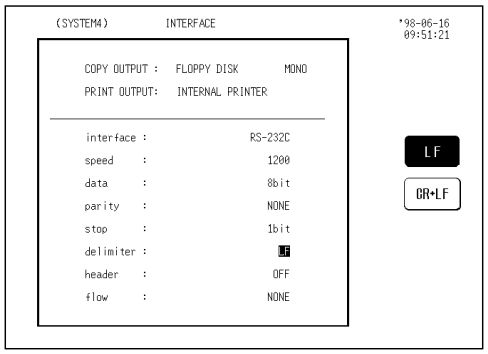
   ODD : Odd number parity



5. Set the stop bits.
   Move the flashing curson to the position shown in the figure on the left, and use the function keys to make the selection.

   [1bit]: Sets the stop bit to 1 bit.

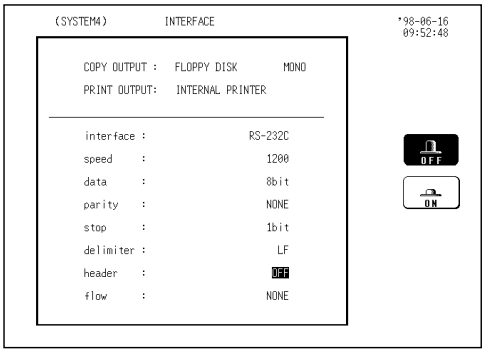
   [2bit]: Sets the stop bit to 2 bits.



6. Set the delimiter.
   Move the flashing curson to the position shown in the figure on the left, and use the function keys to make the selection.

   [LF]: Sets the delimiter to LF.

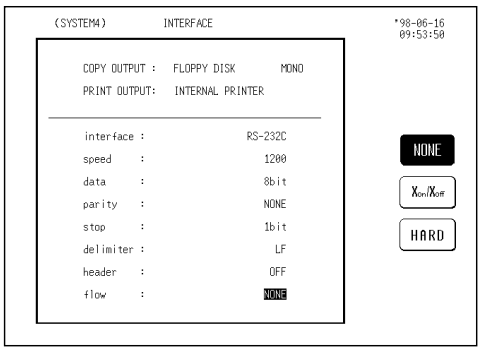
   [CR+LF]: Sets the delimiter to CR+LF.



7. Set the headers.
   Move the flashing curson to the position shown in the figure on the left, and use the function keys to make the selection.

   [OFF]: Header information is not appended.

   [ON]: Header information is appended.



8. Set the flow control.
   Move the flashing curson to the position shown in the figure on the left, and use the function keys to make the selection.

   [NONE] : No flow control

   [Xon/Xoff] : Software handshake

   [HARD] : Hardware handshake

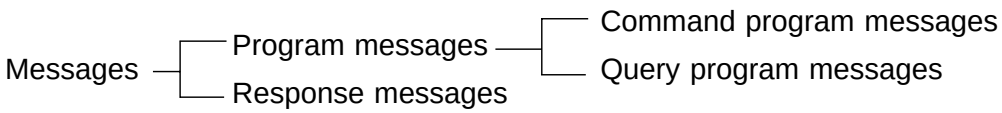
NOTE

- If an overrun error, a framing error or the like occurs, reduce the transfer rate.
- The unit automatically recognizes which type of PC card is inserted, and the appropriate setting items appear on the display. Perform the setting procedure after inserting the RS-232C card.
- Do not change the settings during communications.

2.4 Receive and Send Protocols

(1) Messages

Data received or sent by the GP-IB or RS-232C interface is called a message. The following are the message types:



Of these, program messages are those received by the unit from the controller, while response messages are those sent from the unit to the controller.
Program messages are command messages or query messages.
Command messages are orders for control of the device, such as for making settings or for reset or the like.
Query messages are orders for responses relating to the results of operation, results of measurement, or the state of device settings.
Response messages are sent in response to query program messages. After a query message has been received, a response message is produced the moment that its syntax has been checked.

(2) Command syntax

When no ambiguity would arise, the term "command" is henceforth used to refer to both command and query program messages.
The unit accepts commands without distinction between lower case and upper case letters.

The names of commands are as far as possible mnemonic. Furthermore, all commands have a long form, and an abbreviated short form.
In command references in this manual, the short form is written in upper case letters, and then this is continued in lower case letters so as to constitute the long form. Either of these forms will be accepted during operation, but intermediate forms will not be accepted.
Further, during operation both lower case letters and upper case letters will be accepted without distinction.
The unit generates response messages in the long form (when headers are enabled) and in upper case letters.
(Example)
For "DISPlay", either "DISPLAY" (the long form) or "DISP" (the short form) will be accepted. However, any one of "DISPLA", "DISPL", or "DIS" is wrong and will generate an error.

(3) Command program headers

Commands must have a header, which identifies the command in question. There are three kinds of header: the simple command type, the compound command type, and standard command type.

- Simple command type header

  The first word constitute the header.

  Example :HEADer ON

  Simple command type header / Data

- Compound command type header

  A header made up from a plurality of simple command type headers marked off by colons.

  Example :CONFigure:TDIV 1.E-3

  Simple command type header / Data / Compound command type header

- Standard command type header

  A command beginning with an asterisk and stipulated by IEEE 488.2

  Example *RST

(4) Query program headers

These are for commands used for interrogating the unit about the result of an operation or about a setting.
These can be recognized as queries by a question mark appearing after the program header. The structure of the header is identical to that of a command program header, with "?" always being affixed to the last command. There are queries possible in each of the three previously described types of command form.

Example :HEADER? ON

Query program header / Data

(5) Response messages

Response messages relating to queries are made up from header portions (which also may be absent due to header disablement) and data portions identical to those of program messages, and as a general rule are sent in an identical format to the format of the program message corresponding to their originating query.

(6) Terminators and separators

① Message Terminator

A terminator is used in order to separate the transmission of one message from another, and this terminator is not itself included in the message.

GP-IB interface

LF, EOI, or LF+EOI is used as the message terminator, and LF+EOI is also used as the response message terminator.

RS-232C interface

Set the delimiter for the message terminator (see Section 2.3.2).

② Message Unit Separator

A semicolon ";" is used as a message unit separator when it is desired to set out several messages on a single line.

Example :CONFIGURE:TDIV 1. E-3;:CONFIGURE:SHOT 15

Message unit separator

③ Header separator

With a message which has both a header and data, a space "_" is used as a header separator to separate the header from the data. The space "_" is used by way of explanation, but it does not appear on the actual program.

Example :CONFIGURE:SHOT_15

Header separator

④ Data separator

Commas are used as data separators for separating several data items from one another.

Example :DISPLAY:DRAW_CH1,DARK

Simple command type header

Compound command type header

Data separator

Header separator

(7) The command tree

The rule when writing several messages of compound command form on the same line, when no colon is prefixed to the next header after the semicolon (the message unit separator), is that that header is considered as continuing on from the header before the last colon in the message directly preceding. This corresponds to the general concept of the current directory in the directory structure of UNIX or MS-DOS, and this directly preceding header is called the "current path".

Example 1 :CONF:TDIV 1. E-3;:CONF:SHOT 15

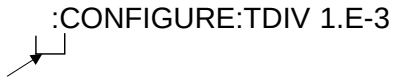
Example 2 :CONF:TDIV 1. E-3;SHOT 15

Both Example 1 and Example 2 are messages setting TIME/DIV to 1 ms and recording length to 15 divisions.

With Example 1, because there is a colon directly after the semicolon, the current position is the "root". Accordingly the reference of the next command is performed from the root.

On the other hand, with Example 2, because with ":CONF:TDIV 1. E-3;" the current path has become ":CONF", it is now possible to omit the ":CONF:" before "SHOT".

To reiterate, the colon at the beginning of a command forces the search for the command to begin from the root. Thus in Example 1:



The first colon indicates that the "CONFIGURE" command is at the root level.

(8) Data format

The unit uses character data, decimal data and character string data as a data format.

- Character data

① The first character must be alphabetic.

② The characters after the first character can only be alphabetic characters, numerals, or underline characters (_).

③ As alphabetic characters, during sending only upper case letters are used, but during receiving both upper case and lower case letters are permitted.

- Decimal data

Decimal data values are represented in what is termed NR format.
There are three types of NR format from NR1 to NR3, and each of these can appear as either a signed number or an unsigned number. Unsigned numbers are taken as positive.

Further, if the accuracy of a numerical value exceeds the range with which the unit can deal, it is rounded off. (5 and above is rounded up; 4 and below is rounded down.)

NR1 format - integer data
Examples: +15, -20, 25
NR2 format - fixed point numbers
Examples: +1.23, -4.56, 7.89
NR3 format - floating point numbers
Examples: +1.0E-3, -2.3E+3

} NRf format

The term "NRf format" includes all these three formats.

When the unit is receiving it accepts NRf format, but when it is sending it utilizes whichever one of the formats NR1 to NR3 is indicated in the particular command.

- Character string data

Character string data is enclosed within quotation marks.
① The data is composed of 8 bit ASCII characters.
② Characters which cannot be handled by the unit are replaced by spaces.
③ When the unit is sending, only the double quotation mark (") is used as a quotation mark, but when receiving both this double quotation mark and also the single quotation mark (') are accepted.

2.5 The Status Byte and the Event Registers

(1) The status byte

Each bit of the status byte is a summary (logical OR) of the event register corresponding to that bit.

Further, for GP-IB, the status byte and each event register has an enable register corresponding to it, and according to the setting of this enable register (which starts off at zero when the power is turned on) it is possible to mask the service requests originating from each event.

For RS-232C, only the values for the status byte, standard event status register, and event status register 0 are valid. The enable register setting has no effect and is disregarded.

Status byte bit settings

bit 7	Unused: 0
bit 6 RQS MSS	Set when a service request is issued. (For RS-232C, unused: 0)
bit 5 ESB	Event summary bit. Shows a summary of the standard event status register.
bit 4 MAV	Message available. Shows that a message is present in the output queue.
bit 3	Unused: 0
bit 2	Unused: 0
bit 1	Unused: 0
bit 0 ESB0	Event summary bit 0 Shows a summary of event status register 0.

The following commands are used for reading the status byte, and for setting the service request enable register and for reading it.

Reading the status byte	*STB?
Setting the service request enable register	*SRE (GP-IB)
Reading the service request enable register	*SRE? (GP-IB)

(2) Standard event status register (SESR)

The summary of this register is set in bit 5 of the status byte.

For GP-IB, each bit is masked by setting the standard event status enable register (which starts off at zero when the power is turned on).

The circumstances when the contents of the standard event status register are cleared are as listed below.

1. When the *CLS command is received.
2. When the contents have been read by an *ESR? query.
3. When the power is turned off and turned on again.

Bit allocations in the standard event status register

bit 7 PON	The power has been turned on again. Since this register was last read, the unit has been powered off and on.
bit 6 URQ	User request: not used.
bit 5 CME	Command error. There is an error in a command that has been received; either an error in syntax, or an error in meaning.
bit 4 EXE	Execution error. An error has occurred while executing a command. Range error; Mode error.
bit 3 DDE	Device dependent error. It has been impossible to execute some command, due to an error other than a command error, a query error, or an execution error.
bit 2 QYE	Query error. The queue is empty, or data loss has occurred (queue overflow).
bit 1	Request for controller right (not used) Unused: 0
bit 0 OPC	Operation finished. Only set for the *OPC command.

The following commands are used to read the standard event status register, and to set or read the standard event status enable register.

Read the standard event status register *ESR?

Set the standard event status enable register *ESE (GP-IB)

Read the standard event status enable register *ESE? (GP-IB)

(3) Event status register 0 (ESR0)

The summary of this register is set in bit 0 of the status byte.

For GP-IB, each bit is masked when the event status enable register 0 (which starts off at zero when the power is turned on) is set.

The circumstances when the contents of event status register 0 are cleared are as listed below.

1. When the *CLS command is received.
2. When the contents have been read by an :ESR0? query.
3. When the power is turned off and turned on again.

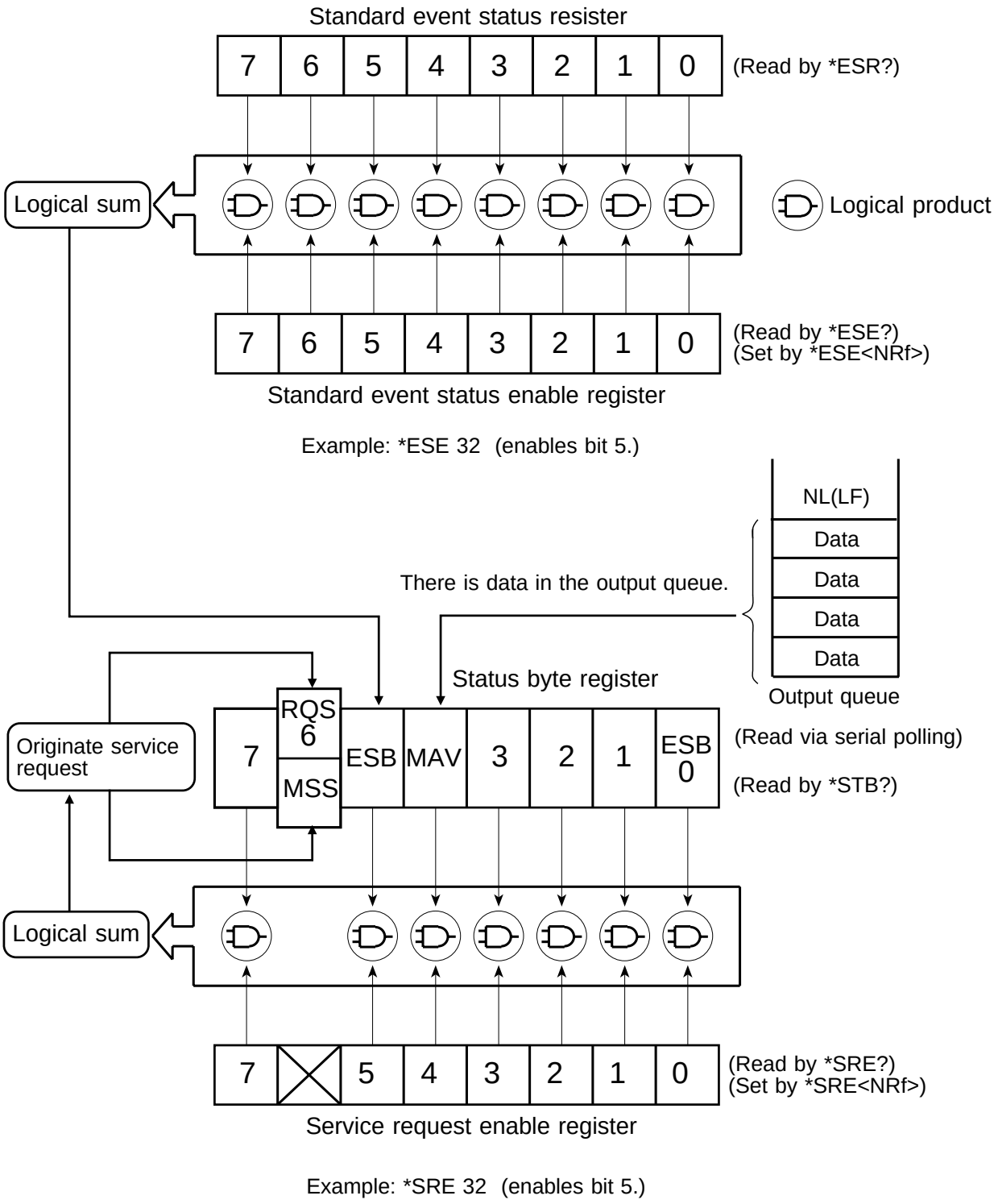
The bits of event status register 0

bit 7	Waveform decision fail (NG).
bit 6	Parameter decision fail (NG).
bit 5	Parameter calculation finished.
bit 4	Waveform processing calculation finished.
bit 3	Printer operation finished (print, or copy output).
bit 2	Trigger wait finished (set when the trigger event occurs).
bit 1	Measurement operation concluded (set by STOP).
bit 0	Error not related to the GP-IB interface; printer error etc.

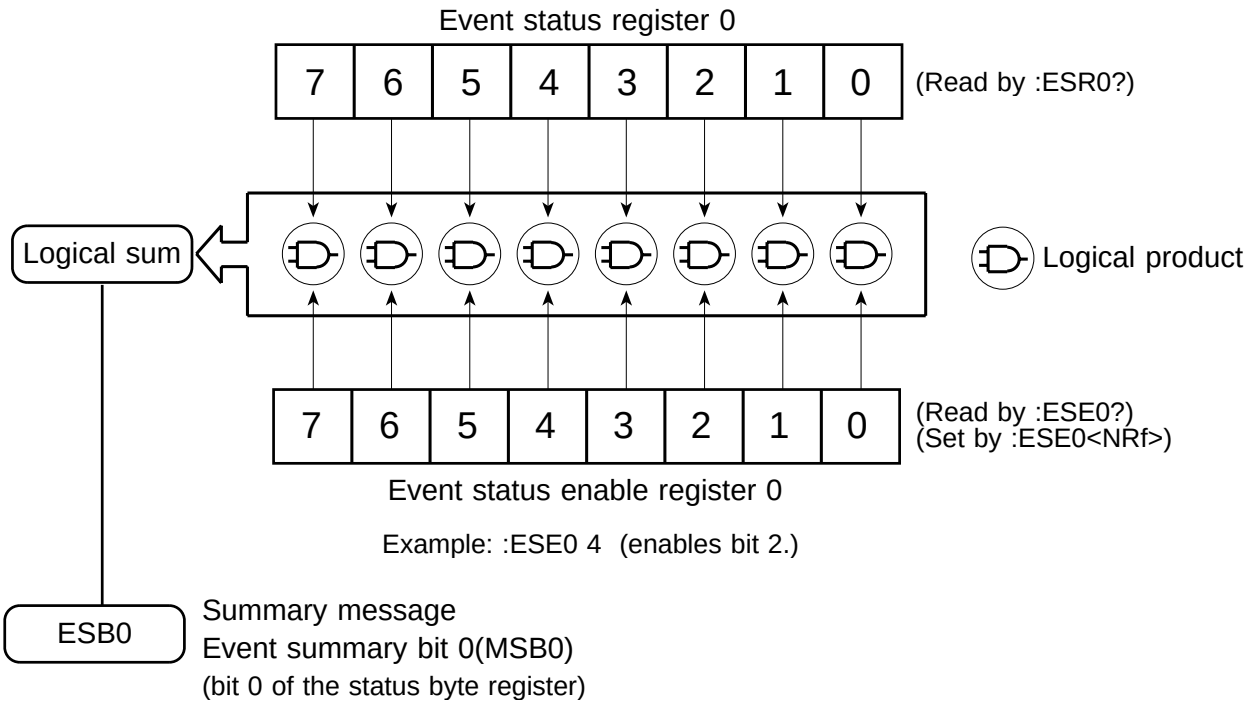
The following commands are used for reading the event status register 0, and for setting the event status enable register 0 and for reading it.

Reading event status register 0	:ESR0?
Setting event status enable register 0	:ESE0 (GP-IB)
Reading event status enable register 0	:ESE0? (GP-IB)

Status byte data structure



Event status register 0 data structure



2.6 The Input Buffer and the Output Queue

(1) Input buffer

The unit has an input buffer of 1024 bytes capacity.
Messages which are received are put into this buffer and executed in order.
However, an ABORT command is executed instantly as soon as it is received.

(2) Output queue

The unit has an output queue of 512 bytes capacity.
Response messages are accumulated in this queue and are read out from the controller.
If the length of a response message has exceeded 512 bytes, a query error occurs.

The circumstances when the output queue is cleared are as listed below:

1. When the controller has read out its entire contents.
2. When a device clear is issued.
3. When the power is turned off and turned on again.
4. Upon receipt of the next message.

2.7 Others

2.7.1 GP-IB

(1) Remote Control

- Local state

  This is the state in which the unit is controlled by its keys. When the power is turned on, the unit always comes up in local state.

- Remote state

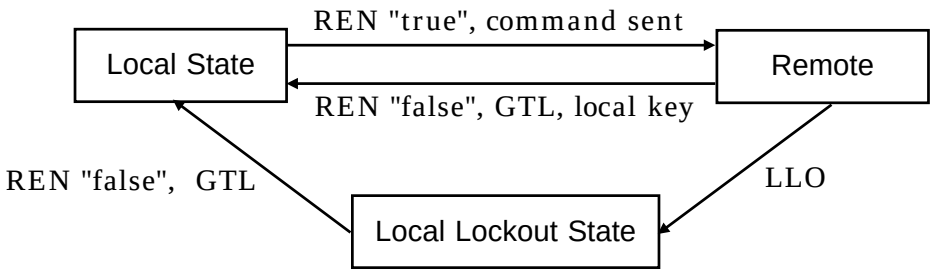
  In this state the unit is controlled from the GP-IB interface (the REN line is "true"), and its keys are disabled. When in the remote state, the unit returns to local state if the local key (the **[LOCAL]** function key) is pressed.

- Local lockout state

  When an LLO (Local Lockout) command (this is a GP-IB universal command) is received, even if the local key is pressed, the unit is prevented from returning to the local state. This state is called the local lockout state.

  In order to return the unit from the local lockout state to the local state, it is necessary either (a) to send a GTL (Go To Local) command (this is a GP-IB universal command), or (b) to turn the power to the unit temporarily off and then on again, or (c) to bring the line REN to "false".

  If a command is sent with REN in the "false" state, then the only way to return to the local state is with the local key.



Program example HP-9816 (Hewlett-Packard)

local lockout	LOCAL LOCKOUT 7
local	LOCAL 7

(2) Device Clear

When the unit receives the device clear command, it clears the input buffer and the output queue (see Section 2.6).

The device clear command is exemplified by the following:
HP 9816 (made by Hewlett-Packard)
CLEAR 7

(3) GP-IB Errors

When a command which has been received contains an error, that one of bits 2 to 5 of the standard event status register which corresponds to the event which has occurred is set.

Further, if a command has given rise to an error (apart from an execution error), commands accumulated in the input buffer and waiting for execution after that command are ignored.

2.7.2 RS-232C

RS-232C Errors

(1) Parity error

The parity bit can be set to even parity, odd parity, or no parity. When even or odd is selected, the "1" count is used to detect transmission errors. If the parity count is different at the receiving end, a parity error is returned.

(2) Framing error

When counting from the start bit, if the stop bit is "L", a framing error is returned.

Possible reason (1): Transmission rate, parity, stop bit or other parameter setting mismatch

Possible reason (2): Noise

(3) Overrun error

The transmission controller uses double buffering for receiving data (shift buffer for each bit and reception buffer read by the CPU).

When there are data in the reception buffer, and the shift register completes reception of the next character before the data are read by the CPU, an overrun error occurs. Because the new data overwrites the previous data in the reception buffer, immediately preceding data are lost.

Possible reason (1): Transmission rate is too high.

Possible reason (2): Some interrupt inhibit intervals are too long.

Possible reason (3): Execution time for higher-priority interrupt is too long, reducing the time available for the receive interrupt.

Flow Control

The RS-232C interface can transfer data at the selected transfer rate, but if the CPU cannot keep up with the data that are being sent, later data will overwrite data that were received earlier. To prevent this, the receiving side must alert the sending side when the reception buffer is about to become full, so that the transfer can be temporarily paused. This is called flow control. Two types of flow control are possible, namely hardware handshaking and software handshaking.

(1) Hardware handshaking

Flow control is performed by setting the signal lines RTS (RS) and CTS (CS) to ON and OFF.

Receiving data

When input buffer content exceeds 3/4, RTS is set to Low.
When input buffer content falls below 1/4, RTS is set to High.

Sending data

When CTS becomes Low, data send is interrupted.
When CTS becomes High, data send is resumed.

(2) Software handshaking

Flow control is performed using the Xon and Xoff code.

Receiving data

When input buffer content exceeds 3/4, D3 (13H) is sent.
When input buffer content falls below 1/4, D1 (11H) is sent.

Sending data

When D3 (13H) is received, data send is interrupted.
When D1 (11H) is received, data send is resumed.

Note: Buffer size is as follows.
Input buffer: 1024 bytes
Output buffer: 512 bytes

Chapter 3
Commands

When using the HIOKI MEMORY HiCORDER can be used with the HIOKI "9557 RS-232C CARD / 9558 GP-IB CARD" except following products, reffer to the communication comands manual (Floppy disk) supplied with the MEMORY HiCORDER.

■ The products consultable this manual: 8826, 8835, 8835-01, 8841, 8842

3.1 Command Summary

3.1.1 Standard Commands Specified by IEEE 488.2

Command	Data (for a query, response data)	Explanation	Ref page	35	26 41 42	20
*IDN?	Maker's name, model number, serial number, software version (not used, zero)	Queries device ID.	59	Y	Y	Y
*OPT?	Whether channel 1 to 4 input units exist (8835) Whether channel 1 to 32 input units exist (8826) Whether channel 1 to 16 input units exist (8841, 8842, 8720) 0: none, 1: analog, 2: voltage/ temperature, 3: strain, 4: FFT, 5: F/V, 6: charge, 7: 4-channel unit	Queries device option provision.	59	Y	Y	Y
*RST		Device initial setting.	60	Y	Y	Y
*TST?	*A* <NR1> (0 = normal, 1 = failure)	Queries the result of the ROM/RAM check.	60	Y	Y	Y
*OPC		Sets the LSB of SESR after all action has been completed.	60	Y	Y	Y
*OPC?	*A* <NR1>	ASCII 1 is the response after all action has been completed.	61	Y	Y	Y
*WAI		Executes the following command after action has been completed.	61	Y	Y	Y
*CLS		Clears the status byte and associated queues.	61	Y	Y	Y

Command	Data (for a query, response data)	Explanation	Ref page	35	26 41 42	20
*ESE *A*	*A:* 0 to 255	Sets SESER. (GP-IB only)	62	Y	Y	Y
*ESE?	*A* <NR1> 0 to 255	Queries SESER				
*ESR?	*A* <NR1> 0 to 255	Queries SESR.	62	Y	Y	Y
*SRE *A*	*A:* 0 to 255	Sets SRER. (GP-IB only)	63	Y	Y	Y
*SRE?	*A* <NR1> 0 to 63, 128 to 191	Queries SRER.				
*STB?	*A* <NR1> 0 to 255	Reads the STB and the MSS bit, without performing serial polling.	63	Y	Y	Y
:ESE0 *A*	*A:* 0 to 255	Sets ESER0. (GP-IB only)	64	Y	Y	Y
:ESE0?	*A* <NR1> 0 to 255	Queries ESER0.				
:ESR0?	*A* <NR1> 0 to 255	Queries ESR0.	64	Y	Y	Y

Note 35: 8835 (-01), 26: 8826, 41: 8841, 42: 8842
20: 8720, Y: Yes, A: Advanced version

3.1.2 Specific Commands

① Execution control etc. (common to all functions)

Command	Data (for a query, response data)	Explanation	Ref page	35	26 41 42	20
:STARt		Same as the START key.	65	Y	Y	Y
:STOP		Same as the STOP key.	65	Y	Y	Y
:ABORT		Forced halt.	65	Y	Y	Y
:PRINt		Same as the PRINT key.	65	Y	Y	N
:HCOPy		Same as the COPY key.	65	Y	Y	N
:FEED *A*	*A*: 1 to 255 (unit mm)	Feeds the paper the specified distance.	66	Y	Y	N
:REPOrt		Same as the FEED key + COPY key.	66	Y	Y	N
:AUTO		Sets the time axis and the voltage axis automatically. (Only the memory recorder function)	66	Y	Y	N
:ERRor?	*A* <NR1> error number	Queries 8835 error number.	66	Y	Y	Y
:HEADer *A$*	*A$*: OFF, ON	Enables and disables headers.	67	Y	Y	Y
:HEADer?	*A$*	Queries headers.				
:FUNCtion *A$*	*A$*: MEM, REC, RMS, R_M, FFT	Changes the function.	67	Y	Y	N
:FUNCtion?	*A$*	Queries the function.				
:CERRor?	A, B, C: number of times A: parity error B: overrun error C: framing error	Queries the communication errors. (RS-232C only)	67	Y	Y	Y
:STATus?	A <NR1> 0 to 127	Queries the status.	66	*	Y	N

Note 35: 8835 (-01), 26: 8826, 41: 8841, 42: 8842, 20: 8720
Y: Yes, A: Advanced version, *: 8835-01 only

② CONFigure command (Setting and querying the time axis range, the recording length, etc.)
:CONFigure

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:TDIV *A*	*A*: TIME/DIV (unit seconds) 100 µs to 5 min/DIV (MEM) (0: external sampling (except 8835)) 10 ms to 1 h/DIV (REC) (8835) 20 ms to 1 h/DIV (REC) (8826, 8841, 8842) 5 s to 1 h/DIV (RMS)	Sets the time axis range.	MEM REC RMS	68	Y	Y	N
:TDIV?	*A* <NR3> (unit seconds)	Queries the time axis range.					
:TDIV *A*, *B*	*A*: TIME/DIV for REC, *B*: TIME/DIV for MEM	Sets the time axis ranges.	R&M	68	A	Y	N
:TDIV?	*A*, *B* <NR3> (unit seconds)	Queries the time axis ranges.					
:SAMPle *A*	*A*: sampling rate (unit seconds) 1 µs to 100 ms	Sets the sampling period.	REC	69	Y	Y	N
:SAMPle?	*A* <NR3>	Queries the sampling period.					
:SAMPle *A$*	*A*: FAST, SLOW	Sets the sampling speed.	REC	69	N	N	Y
:SAMPle?	*A$*	Queries the sampling speed.					
:FREQuency *A*	*A*: 50, 60 (Hz)	Sets the frequency.	RMS	69	Y	Y	N
:FREQuency?	*A* <NR1>	Queries the frequency.					
:SHOT *A*	*A*: recording length (unit DIV) 1 to 20000: 8835 (MEM) 1 to 40000: 8835-01 1 to 160000: 8826, 8841, 8842 1 to CONT: (REC, RMS)	Sets the recording length (during memory segmentation).	MEM REC RMS	70	Y	Y	N
:SHOT?	*A* <NR1> (unit DIV)	Queries the recording length.			Y	Y	N
:SHOT *A*, *B*	*A*: REC recording length *B*: MEM recording length	Sets the recording lengths.	R&M	70	A	Y	N
:SHOT?	*A*, *B* <NR1> (unit DIV)	Queries the recording lengths.					
:RECTime *A*	*A*: Recording time (unit s) 0 (continuous), 1 to 35999999	Sets the recording time.	REC	71	N	N	Y
:RECTime?	*A* <NR1> (unit s)	Queries the recording time.					
:RECSpeed *A*	*A*: Recording speed (unit s) 0.002 to 180 (unit s)	Sets the recording speed.	REC	71	N	N	Y
:RECSpeed?	*A* <NR3> (unit s)	Queries the recording speed.					

:FORMat *A$*	*A$*: SINGle, DUAL, QUAD, XYDot, XYLine (MEM, REC) SINGle, DUAL, QUAD (RMS, R&M) SINGle, DUAL, NYQuist (FFT)	Sets the format.	All	72	Y	N	N
	(8826) *A$*: SINGle, DUAL, QUAD, OCT, HEX, XYSingle, XYQuad (MEM, REC) SINGle, DUAL, QUAD, OCT, HEX (RMS, R&M) SINGle, DUAL, NYQuist (FFT) (8841, 8842) *A$*: SINGle, DUAL, QUAD, OCT, HEX, XYSingle, XYDual (MEM, REC) SINGle, DUAL, QUAD, OCT, HEX (RMS, R&M) SINGle, DUAL, NYQuist (FFT)				N	Y	N
	A$: SINGle, DUAL, QUAD, OCT, XYSingle, XYDual				N	N	Y
:FORMat?	*A$*	Queries the format.			Y	Y	Y
:DOTLine *A$*	*A$*: DOT, LINE (8835: FFT only)	Sets the interpolation function.	All	72	A	Y	Y
:DOTLine?	*A$*	Queries the interpolation function.					
:PRKInd *A$*	*A$*: WAVE, LOGGing	Specifies the printer output style.	All	73	Y	Y	N
:PRKInd?	*A$*	Queries the printer output style					
:SMOOth *A$*	*A$*: OFF, ON	Enables and disables smooth printing.	MEM R&M	73	Y	Y	N
:SMOOth?	*A$*	Queries smooth printing enablement.					
:LOGGing *A*	*A*: 0.01 to 100	Specifies the logging output interval.	All	73	Y	Y	N
:LOGGing?	*A* <NR2>	Queries the logging output interval.					
:ROLL *A$*	*A$*: OFF, ON	Enables and disables roll mode.	MEM	74	Y	Y	N
:ROLL?	*A$*	Queries roll mode enablement.					
:ATPRint *A$ (,B$)*	*A$*: OFF, ON, LAN *B$*: MONO, COLOR	Enables and disables auto print.	MEM	74	Y	Y	N
:ATPRint?	*A$* (,*B$*)	Queries auto print enablement.					

:ATSAve *A$, B$ (, C$)*	*A$*: OFF, FD, PC, LAN (8835) OFF, FD, PC, SCSI, MO, LAN (8826, 8841, 8842) *B$*: Bin, Text *C$*: MEM, REC, R_M (R&M only)	Sets auto save.	All	75	Y	Y	Y
:ATSAve?	*A$, B$, (, C$)*	Queries auto save.					
:ATFIle '*NAME$*'	*NAME$*: file name (8 characters)	Sets the file name for auto save function.	All	75	*	Y	Y
:ATFIle?	'*NAME$*'	Queries the file name for auto save function.					
:DELSave *A$*	*A$*: DEL, NORMal	Sets the delete save function.	All	76	*	Y	Y
:DELSave?	*A$*	Queries the delete save function.					
:THINout *A$*	*A$*: OFF, 1_2 to 1_1000	Sets the degree of thinning before storing.	All	76	N	N	Y
:THINout?	*A$*	Queries the degree of thinning before storing.					
:OVERlay *A$*	*A$*: OFF, ON	Sets waveform overlay.	MEM	76	Y	Y	N
:OVERlay?	*A$*	Queries waveform overlay.					
:AVERage *A*	*A*: 0, 2, 4, 8, 16, 32, 64, 128, 256 (0: OFF)	Sets the count for averaging.	MEM	77	A	Y	N
:AVERage?	*A* <NR1>	Queries the count for averaging.					
:WVCOmp *A$*	*A$*: OFF, OUT, ALLOut	Sets the waveform decision mode.	MEM FFT	77	A	Y	N
:WVCOmp?	*A$*	Queries the waveform decision mode.					
:CMPStop *A$*	*A$*: GO, NG, G-N	Sets the waveform decision stop mode.	MEM FFT	77	A	Y	N
:CMPStop?	*A$*	Queries the waveform decision stop mode.					
:VIRTual *A$*	*A$*: OFF, ON	Enables and disables the additional recording function.	REC RMS R&M	78	Y	Y	Y
:VIRTual?	*A$*	Queries the additional recording function enablement.					
:PRINt *A$*	*A$*: OFF, ON	Sets printer output.	REC RMS R&M	78	Y	Y	N
:PRINt?	*A$*	Queries printer output.					
:EXTSample *A*	*A*: 10 to 1000	Sets data number per 1 DIV for external sampling.	MEM	78	*	Y	N
:EXTSample?	*A* <NR1>	Queries data number per 1 DIV for external sampling.					
:MEMDiv *A$*	*A$*: OFF, SEQ, MULTI (MULTI: MEM only)	Sets memory segmentation.	MEM R&M	79	A	Y	N
:MEMDiv?	*A$*	Queries memory segmentation.					

:USEBlock *A*	*A*: 1 to number of segmentations (255 max.)	Sets the memory block used.	MEM R&M	79	A	Y	N
:USEBlock?	*A* <NR1>	Queries the memory block used.					
:STTBlock *A*	*A*: 1 to number of blocks	Sets the start block (during sequential save).	MEM R&M	80	A	Y	N
:STTBlock?	*A* <NR1>	Queries the start block.					
:ENDBlock *A*	*A*: 1 to number of blocks	Sets the end block (during sequential save).	MEM R&M	80	A	Y	N
:ENDBlock?	*A* <NR1>	Queries the end block.					
:SEQDisp *A$*	*A$*: OFF, ON	Sets the follow-up waveform display (during sequential save).	MEM	80	A	Y	N
:SEQDisp?	*A$*	Queries the follow-up waveform display.					
:MAXBlock *A*	*A*: 3, 7, 15, 31, 63, 127, 255	Sets the number of memory blocks (during multi-block).	MEM	81	A	Y	N
:MAXBlock?	*A* <NR1>	Queries the number of memory blocks.					
:REFBlock *A*	*A*: 0, 1 to number of memory segmentations (0: OFF)(8835(-01) only) *A*: 0 (OFF), 1 (ON) (except 8835(-01))	Sets the reference block.	MEM	81	A	Y	N
:REFBlock?	*A* <NR1>	Queries the reference block.					
:REFBlock *A*, *B$*	*A*: 1 to number of memory segmentations *B$*: ON, OFF	Sets the reference block (during multi-block).	MEM	81	N	Y	N
:REFBlock? *A*	*A* <NR1>, *B$*	Queries the reference block.					
:FFTAVERage *A*	*A*: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096	Sets the count for averaging in the FFT function.	FFT	82	A	Y	N
:FFTAVERage?	*A* <NR1>	Queries the current setting of the count for averaging in the FFT function.					
:FFTAVKind *A$*	*A$*: OFF, T_EXP, F_EXP, T_LIN, F_LIN, F_PEAK	Sets the averaging method.	FFT	82	A	Y	N
:FFTAVKind?	*A$*	Queries the currently set averaging method.					
:FFTMode *A*, *ch1$*, *(,ch2$)*	*A*: 1, 2 *ch1$*, *ch2$*: CH1 to CH32 (8835: CH1 to CH4)	Sets the FFT channel mode.	FFT	83	A	Y	N
:FFTMode?	*A* <NR1>, *ch1$*, *ch2$*	Queries the current FFT channel mode.					
:FFTWind *A$* *(,B)*	*A$*: RECTan, HANNing, EXPOnential *B*: 0 to 99 (%)	Sets the window function.	FFT	83	A	Y	N
:FFTWind?	*A$*, *B* <NR1>	Queries the current window function.					

:FFTFunction *A$, B$*	*A$*: G1, G2 *B$*: STR, LIN, RMS, PSP, ACR, HIS, TRF, CSP, CCR, IMP, COH, OCT	Sets the FFT analysis mode.	FFT	84	A	Y	N
:FFTFunction? *A$*	*A$, B$*	Queries the current FFT analysis mode.					
:FFTRef *A$*	*A$*: NEW, MEM	Designates the source for FFT analysis data.	FFT	85	A	Y	N
:FFTRef?	*A$*	Queries the current FFT analysis data source.					
:FFTSCale *A$, B$*	*A$*: G1, G2 *B$*: AUTO, MANUal	Sets the display scaling method for a graph.	FFT	85	A	Y	N
:FFTSCale? *A$*	*A$, B$*	Queries the current display scaling method for a graph.					
:FFTUp *A$, B*	*A$*: G1, G2 *B*: -9.9999E+29 to +9.9999E+29	Sets the vertical axis upper limit for a graph.	FFT	86	A	Y	N
:FFTUp? *A$*	*A$, B* <NR3>	Queries the current vertical axis upper limit for a graph.					
:FFTLow *A$, B*	*A$*: G1, G2 *B*: -9.9999E+29 to +9.9999E+29	Sets the vertical axis lower limit for a graph.	FFT	86	A	Y	N
:FFTLow? *A$*	*A$, B* <NR3>	Queries the current vertical axis lower limit for a graph.					
:FFTXaxis *A$, B$*	*A$*: G1, G2 *B$*: 1_1oct, 1_3oct (octave analysis) LINhz, LOGhz (otherwise)	Sets the x-axis.	FFT	87	A	Y	N
:FFTXaxis? *A$*	*A$, B$*	Queries the current x-axis setting.					
:FFTYaxis *A$, B$*	*A$*: G1, G2 *B$*: LINMAg, LINREal, LINIMag, LOGMAg, PHASE	Sets the y-axis.	FFT	87	A	Y	N
:FFTYaxis? *A$*	*A$, B$*	Queries the current y-axis setting.					
:FREQ *A*	*A*: 400000, 200000, 80000, 40000, 20000, 8000, 4000, 2000, 800, 400, 200, 80, 40, 20, 8, 4, 1.33, 0.667, 0.333, 0.133, 0	Sets the frequency range.	FFT	89	A	Y	N
:FREQ?	*A* <NR3>	Queries the currently set frequency range.					
:OCTFilter *A$*	*A$*: NORMal, SHARp	Sets the type of octave filter.	FFT	89	A	Y	N
:OCTFilter?	*A$*	Queries the currently set type of octave filter.					
:PEAK *A$*	*A$*: OFF, PEAK, MAX	Sets the peak value display.	FFT	89	A	Y	N
:PEAK?	*A$*	Queries the currently set peak value display.					

:FFTSAmple *A*	*A*: 1000, 2000, 5000, 10000	Sets the number of FFT points.	FFT	90	A	Y	N
:FFTSAmple?	*A* <NR1>	Queries the number of FFT points.					
:RTSAve *A$*	*A$*: ON, OFF	Sets the real time save function.	R&M	90	N	Y	N
:RTSAve?	*A$*	Queries the real time save function.					
:CMPOld *A$*	*A$*: ON, OFF	Sets comparison of separate files.	All	90	N	N	Y
:CMPOld?	*A$*	Queries comparison of separate files.					
:OTSAve *A$*	*A$*: FD, PC, MO, SCSI, LAN	Sets one-touch save setting.	All	91	N	N	Y
:OTSAve?	*A$*	Queries one-touch save setting.					

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version
*: 8835-01 only

MEM: memory recorder function REC: recorder function
RMS: RMS recorder function R&M: recorder and memory function
FFT: FFT function
All: all MEM, REC, RMS, R&M and FFT functions

③ TRIGger command (Setting and querying trigger.)

:TRIGger

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:MODE *A$*	*A$*: SINGle, REPEat, AUTO (MEM, FFT) SINGle, REPEat (REC, RMS) SINGle, REPEat, TIMEr (R&M)	Sets trigger mode.	All	91	Y	Y	Y
:MODE?	*A$*	Queries trigger mode.					
:PRETrig *A*	*A*: 0, 2, 10, ... 90, 95, 100, -95% (MEM, R&M, FFT) 0, 5, 10 DIV (RMS, 8720)	Sets pre-trigger.	MEM RMS R&M FFT	92	Y	Y	Y
:PRETrig?	*A* <NR1> (unit %)	Queries pre-trigger.					
:TIMIng *A$*	*A$*: START, STOP, S_S	Sets trigger timing.	REC	92	Y	Y	N
:TIMIng?	*A$*	Queries trigger timing.					
:SOURce *A$*	*A$*: OR, AND	Sets trigger logical operator to AND or OR.	All	92	Y	Y	Y
:SOURce?	*A$*	Queries trigger logical operator (AND or OR).					
:MANU *A$*	*A$*: OFF, ON	Sets manual trigger.	All	93	Y	Y	N
:MANU?	*A$*	Queries manual trigger.					

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:KIND *ch$, A$*	*A$*: OFF, LEVEl, IN, OUT, DROP,PERIod (MEM, R&M, FFT) OFF, LEVEl, IN, OUT, PERIod (REC) OFF, RMS (RMS)	Sets type of trigger.	All	93	Y	Y	Y
:KIND? *ch$*	*ch$, A$*	Queries type of trigger.					
:LEVEl *ch$, A*	*A*: trigger level (unit V)	Sets the trigger level of the level trigger.	MEM REC R&M FFT	93	Y	Y	Y
:LEVEl? *ch$*	*ch$, A* <NR3>	Queries the trigger level of the level trigger.					
:SLOPe *ch$, A$*	*A$*: UP, DOWN	Sets the trigger direction (slope) (level trigger, period trigger).	MEM REC R&M FFT	94	Y	Y	Y
:SLOPe? *ch$*	*ch$, A$*	Queries the trigger direction (slope).					
:FILTer *ch$, A*	*A*: 0 (OFF), 0.1, 0.2, 0.5, 1.0, 1.5, 2.0, 2.5, 5.0, 10.0 (DIV) (MEM, R&M, FFT) 0 (OFF), 1 (ON) (REC)	Sets trigger filter.	MEM REC R&M FFT	94	Y	Y	Y
:FILTer? *ch$*	*ch$, A* <NR2>	Queries trigger filter.					
:UPPEr *ch$, A*	*A*: upper limit level (unit V)	Sets upper limit level of window-in/-out trigger.	MEM REC R&M FFT	95	Y	Y	Y
:UPPEr? *ch$*	*ch$, A* <NR3>	Queries upper limit level of window-in/-out trigger.					
:LOWEr *ch$, A*	*A*: lower limit level (unit V)	Sets lower limit level of window-in/-out trigger.	MEM REC R&M FFT	95	Y	Y	Y
:LOWEr? *ch$*	*ch$, A* <NR3>	Queries lower limit level of window-in/-out trigger.					
:VFREq *ch$, A*	*A*: 50/60 (Hz)	Sets measurement frequency of voltage drop trigger.	MEM R&M FFT	96	Y	Y	N
:VFREq? *ch$*	*ch$, A* <NR1>	Queries measurement frequency of voltage drop trigger.					
:VLEVel *ch$, A*	*A*: drop level (V)	Sets drop level of voltage drop trigger.	MEM R&M FFT	96	Y	Y	N
:VLEVel? *ch$*		Queries drop level of voltage drop trigger.					
:PUPPer *ch$, A*	*A*: upper limit level (s)	Sets upper period limit of period trigger.	MEM REC R&M FFT	97	Y	Y	N
:PUPPer? *ch$*	*ch$, A* <NR3>	Queries upper period limit of period trigger.					
:PLOWer *ch$, A*	*A*: lower limit level (s)	Sets lower period limit of period trigger.	MEM REC R&M FFT	97	Y	Y	N
:PLOWer? *ch$*	*ch$, A* <NR3>	Queries lower period limit of period trigger.					

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:PLEVel *ch$, A*	*A*: trigger level (V)	Sets the trigger level of period trigger.	MEM REC R&M FFT	97	Y	Y	N
:PLEVel? *ch$*	*ch$, A* <NR3>	Queries the trigger level of period trigger.					
:RLEVel *ch$, A*	*A*: trigger level (V)	Sets the trigger level of RMS level trigger.	RMS	98	Y	Y	N
:RLEVel? *ch$*	*ch$, A* <NR3>	Queries the trigger level of RMS level trigger.					
:RSLOpe *ch$, A*	*A$*: UP, DOWN	Sets the direction (slope) of RMS level trigger.	RMS	98	Y	Y	N
:RSLOpe? *ch$*	*ch$, A$*	Queries the direction (slope) of RMS level trigger.					
:LOGAnd *ch$, A$*	*A$*: OFF, OR, AND	Sets AND/OR for the logic trigger pattern.	MEM REC RMS R&M	99	Y	Y	N
:LOGAnd? *ch$*	*ch$, A$*	Queries AND/OR for the logic trigger pattern.					
:LFILter *ch$, A*	*A*: 0 (OFF), 0.1, 0.2, 0.5, 1.0, 1.5, 2.0, 2.5, 5.0, 10.0 (DIV) (MEM) 0 (OFF), 1 (ON) (REC, RMS)	Sets logic trigger filter.	MEM REC RMS R&M	99	Y	Y	N
:LFILter? *ch$*	*ch$, A* <NR2>	Queries logic trigger filter.					
:LOGPat *ch$, 'A$'*	*A$*: xxxx trigger pattern (x, 0, 1)	Sets the pattern for a logic trigger.	MEM REC RMS R&M	100	Y	Y	N
:LOGPat? *ch$*	*ch$, "A$"*	Queries the pattern for a logic trigger.					
:TIMEr *A$*	*A$*: OFF, ON	Sets timer trigger.	MEM REC RMS FFT	100	Y	Y	N
:TIMEr?	*A$*	Queries timer trigger.					
:TMSTArt *month, day, hour, min*	*month*: 1 to 12 *day*: 1 to 31 *hour*: 0 to 23 *min*: 0 to 59	Sets start time of timer trigger.	All	101	Y	Y	N
:TMSTArt?	*month, day, hour, min* all <NR1>	Queries start time of timer trigger.					
:TMSTOp *month, day, hour, min*	Same as :TMSTArt	Sets stop time of timer trigger.	All	101	Y	Y	N
:TMSTOp?	Same as :TMSTArt?	Queries stop time of timer trigger.					
:TMINTvl *day, hour, min, sec*	*day*: 0 to 99 *hour*: 0 to 23 *min*: 0 to 59 *sec*: 0 to 59	Sets time interval for timer trigger.	All	102	Y	Y	N
:TMINTvl?	*day, hour, min, sec* all <NR1>	Queries time interval for timer trigger.					

Command	Data (for a query, response data)	Explanation	Func-tion	Ref page	35	26 41 42	20
:DETECTTime *hour, min, sec*	*hour*: 0 to 23 *min*: 0 to 59 *sec*: 0 to 59	Sets the time point for trigger detection.	All	102	Y	Y	Y
:DETECTTime?	*hour, min, sec* all <NR1>	Queries the currently set time point for trigger detection.					
:DETECTDate *year, month, day*	*year*: 0 to 99 *month*: 1 to 12 *day*: 1 to 31	Sets the date for trigger detection.	All	103	Y	Y	Y
:DETECTDate?	*year, month, day* all <NR1>	Queries the currently set date for trigger detection.					
:STOPTime *hour, min, sec*	*hour*: 0 to 23 *min*: 0 to 59 *sec*: 0 to 59	Sets the termination time of operation.	REC R&M	103	Y	Y	Y
:STOPTime?	*hour, min, sec* all <NR1>	Queries the termination time of operation.					
:STOPDate *year, month, day*	*year*: 0 to 99 *month*: 1 to 12 *day*: 1 to 31	Sets the date of termination.	REC R&M	104	Y	Y	Y
:STOPDate?	*year, month, day* all <NR1>	Queries the date of termination.					
:EXTErnal *A$*	*A$*: OFF, ON	Enables and disables external trigger.	All	104	Y	Y	Y
:EXTErnal?	*A$*	Queries external trigger enablement.					

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version

④ UNIT command (Setting and querying input channel)

:UNIT

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:RANGe *ch$, A*	*A*: voltage axis range(V, με, ℃)	Sets input channel voltage axis range.	All	105	Y	Y	Y
:RANGe? *ch$*	*ch$, A* <NR3>	Queries input channel voltage axis range.					
:COUPling *ch$, A$*	*A$*: GND, DC, AC	Sets input channel coupling.	All	105	Y	Y	Y
:COUPling? *ch$*	*ch$, A$*	Queries input channel coupling.					
:POSItion *ch$, A*	*A*: position value (unit %)	Sets the origin position for an input channel.	All	106	Y	Y	Y
:POSItion? *ch$*	*ch$, A* <NR1>	Queries the origin position for an input channel.					
:FILTer *ch$, A*	*A*: 0 (OFF), 5, 500, 5000, 100000 0 (OFF), 5, 500 (when measuring temperature with the 8937) 0 (OFF), 10, 30, 300, 3000 (8939)	Sets input channel filter.	All	106	Y	Y	Y
:FILTer? *ch$*	*ch$, A$*	Queries input channel filter.					
:SENSor *ch$, A$*	*A$*: K, E, J, T, N, R, S, B, OFF (voltage)	Sets the type of the voltage/ temperature unit sensor.	All	107	Y	Y	Y
:SENSor? *ch$*	*ch$, A$*	Queries the type of the voltage/ temperature unit sensor.					
:RJC *ch$, A$*	*A$*: INT, EXT	Sets reference contact compensation of the voltage/ temperature unit.	All	107	Y	Y	Y
:RJC? *ch$*	*ch$, A$*	Queries reference contact compensation of the voltage/ temperature unit.					
:DRIFt *ch$, A$*	*A$*: OFF, ON	Sets drift compensation of the voltage/temperature unit.	All	108	Y	Y	Y
:DRIFt? *ch$*	*ch$, A$*	Queries drift compensation of the voltage/temperature unit.					
:DFILter *ch$, A$*	*A$*: OFF, ON	Sets digital filter of the voltage/temperature unit.	All	108	Y	Y	Y
:DFILter? *ch$*	*ch$, A$*	Queries digital filter of the voltage/temperature unit.					
:AAFilter *ch$, A$*	*A$*: OFF, ON	Turns on or off the FFT anti-aliasing filter.	All	109	Y	Y	Y
:AAFilter? *ch$*	*ch$, A$*	Queries the current on or off state of the FFT anti-aliasing filter.					
:ADJUST		Carries out zero adjustment.	All	109	Y	Y	Y

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:BALAnce		Carries out auto-balancing for all of the strain unit channels.	All	109	Y	Y	Y
:CHBAlance *ch$*		Carries out auto-balancing for the selected channel (strain unit).	All	109	Y	Y	Y
:OFSCancel *ch$, A$*	*A$*: OFF, ON	Executes the baseline offset.	All	110	*	Y	Y
:OFSCancel? *ch$*	*ch$, A$*	Queries the baseline offset.					
:CHKClamp		Performs the clamp check in the F/V unit.	All	110	*	Y	Y
:FVMOde *ch$, A$*	*A$*: FREQ, COUNT, DUTY, VOLT, CURRent	Sets the measurement mode of the F/V unit.	All	110	Y	Y	Y
:FVMOde? *ch$*	*ch$, A$*	Queries the measurement mode of the F/V unit.					
:FRANge *ch$, A$*	*A$*:	Sets the frequency range of the F/V unit.	All	111	Y	Y	Y
:FRANge? *ch$*	*ch$, A$*	Queries the frequency range of the F/V unit.					
:FVLEvel *ch$, A*	*A*: -10 to 10	Sets the threshold level of the F/V unit.	All	111	Y	Y	Y
:FVLEvel? *ch$*	*ch$, A* <NR3>	Queries the threshold level of the F/V unit.					
:FVHOld *ch$, A$*	*A$*: ON, 10MS, 1S	Sets the hold of the F/V unit.	All	112	Y	Y	Y
:FVHOld? *ch$*	*ch$, A$*	Queries the hold of the F/V unit.					
:PULLup *ch$, A$*	*A$*: OFF, ON	Sets the input switch of the F/V unit.	All	112	Y	Y	Y
:PULLup? *ch$*	*ch$, A$*	Queries the input switch of the F/V unit.					
:CMODe *ch$, A$*	*A$*: VOLT, CHARge, PREamp	Sets the measurement mode of the charge unit.	All	112	Y	Y	Y
:CMODe? *ch$*	*ch$, A$*	Queries the measurement mode of the charge unit.					
:CSENs *ch$, A*	*A*: 0.1 to 10	Sets the sensor sensitivity of the charge unit.	All	113	Y	Y	Y
:CSENs? *ch$*	*ch$, A* <NR3>	Queries the sensor sensitivity of the charge unit.					

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version
*: 8835-01 only

⑤ DISPlay command (Setting and querying changeover of the screen mode, waveform display, etc.)

:DISPlay

Command	Data (for a query, response data)	Explanation	Func-tion	Ref page	35	26 41 42	20
:CHANge *A$*	*A$*: STATus, CHANnel, DISPlay, SYSTem, FILE	Changes over the display screen.	All	113	Y	N	N
	A$: SYSTem, STATus, TRIGger, CHANnel, DISPlay, FILE				N	Y	Y
:CHANge?	*A$*	Queries the display screen.			Y	Y	Y
:PAGE *A*	*A*: 1 to 6 (system screen) 1 to 5 (status screen) 1, 2 (channel screen)	Changes over the page of the screen.	All	114	Y	N	N
	A: 1 to 6 (system screen) 1 to 4 (status screen) 1, 2 (channel screen)				N	Y	N
	A: 1 to 4 (system screen) 1, 2 (status screen) 1 to 4 (channel screen)				N	N	Y
:PAGE?	*A* <NR1>	Queries the page of the screen.			Y	Y	Y
:DRAWing *ch$, A$*	*A$*: OFF, C1 to C12	Sets waveform display color.	All	114	Y	Y	Y
:DRAWing? *ch$*	*ch$, A$*	Queries waveform display color.					
:GRAPh *ch$, A*	*A*: 1, 2, 3, 4 (for DUAL format, 1, 2) 1 to 8 (OCT, HEX format: 8841, 8842, 8720)	Sets waveform display graph (when the format is other than SINGLE).	MEM REC RMS R&M	115	Y	Y	Y
:GRAPh? *ch$*	*ch$, A* <NR1>	Queries waveform display graph					
:LOGDraw *ch$, N, A$*	*A$*: OFF, C1 to C12 *N*: 1, 2, 3, 4	Sets logic waveform display color.	MEM REC RMS R&M	115	Y	Y	Y
:LOGDraw? *ch$, N*	*ch$, N, A$*	Queries logic waveform display color.					
:LOGPosi *ch$, A*	*A*: 1, 2, 3, 4, 5, 6, 7, 8	Sets the position of logic waveform display.	MEM REC RMS R&M	116	Y	Y	Y
:LOGPosi? *ch$*	*ch$, A* <NR1>	Queries the position of logic waveform display.					
:XMAG *A$*	*A$*: ×10 to ×1_2000 (MEM) ×1 to ×1_50 (REC, RMS)	Sets the magnification/ compression factor on the time axis.	MEM REC RMS R&M	116	Y	N	N
	A$: ×10 to ×10000 (MEM) ×1 to ×1_500 (REC, RMS)				N	Y	N
	A$: ×4 to ×1_500				N	N	Y

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:XMAG?	*A$*	Queries the magnification/ compression factor on the time axis.	MEM REC RMS R&M	116	Y	Y	Y
:YMAG *ch$, A$*	*A$*: ×1_2, ×1, ×2, ×5, ×10	Sets the magnification/ compression factor on the voltage axis.	MEM REC RMS R&M	117	Y	N	N
	A$*: ×1_2, ×1, ×2, ×5, ×10 (SINGLE, XY SINGLE format) *A$: ×1_4, ×1_2, ×1, ×2.5, ×5 (other than the above)				N	Y	Y
:YMAG? *ch$*	*ch$, A$*	Queries the magnification/ compression factor on the voltage axis.			Y	Y	Y
:ZOOM *A$*	*A$*: OFF, ON	Enables and disables the zoom function.	MEM	117	Y	Y	Y
:ZOOM?	*A$*	Queries the zoom function enablement.					
:ZOOMMag *A$*	Same as *A$*: XMAG	Sets the zoom magnification.	MEM	118	Y	Y	Y
:ZOOMMag?	*A$*	Queries the zoom magnification.					
:XYDRawing *A, B$*	*A*: 1 to 4 *B$*: OFF, C1 to C12	Enables and disables the XY waveform display.	MEM REC	118	N	Y	Y
:XYDRawing? *A*	*A* <NR1>, *B$*	Queries the XY waveform display enablement.					
:XAXIs *ch$*	*ch$:* CH1 to CH4	In X-Y format, sets the X-axis.	MEM REC	119	Y	N	N
:XAXIs?	*ch$*	In X-Y format, queries the X-axis.					
:XAXIs *A, ch$*	*A*: 1 to 4	In X-Y format, sets the X-axis.	MEM REC	119	N	Y	Y
:XAXIs? *A*	*ch$*	In X-Y format, queries the X-axis.					
:YAXIs *A, ch$*	*A*: 1 to 4	In X-Y format, sets the Y-axis.	MEM REC	119	N	Y	Y
:YAXIs? *A*	*A* <NR1>, *ch$*	In X-Y format, queries the Y-axis.					
:WAVE *A$*	*A$*: ACUR (A-cursor), TRIG (trigger point), POINT (the point set with :MEMory:POINt)	Executes waveform display.	MEM R&M	120	Y	Y	Y
:VARIable *ch$, A$*	*A$*: ON, OFF	Sets the variable function.	All	120	Y	Y	Y
:VARIable? *ch$*	*ch$, A$*	Queries the variable function.					
:VARIUPLOw *ch$, B, C*	*B, C*: -9.9999E+29 to +9.9999E+29	Sets the upper and lower limit values of the variable.	All	120	Y	Y	Y
:VARIUPLOw? *ch$*	*ch$, B* <NR3>, *C* <NR3>	Queries the upper and lower limit values of the variable.					

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:VARIRng *ch$*, *A*, *B*	*A*, *B*: -9.9999E+29 to +9.9999E+29	Sets values for variable range and position.	All	121	*	Y	Y
:VARIRng? *ch$*	*ch$*: *A*, *B* <NR3>	Queries values for variable range and position.					
:XYCLr *A$*	*A$*: OFF, ON	Sets the display clear function.	REC	121	Y	Y	Y
:XYCLr?	*A$*	Queries the display clear function.					
:SIZE *A$*	*A$*: NORMal, WIDE	Sets the screen size.	MEM REC RMS R&M	121	N	Y	Y
:SIZE?	*A$*	Queries the screen size.					
:RMDIsplay *A$*	*A$*: REC, MEM	Sets the CRT display waveform in the R&M function.	R&M	122	A	Y	N
:RMDIsplay?	*A$*	Queries the CRT display waveform in the R&M function.					
:SYNC *A$*	*A$*: OFF, ON	Sets synchronization function.	All	122	N	N	Y
:SYNC?	*A$*	Queries synchronization function.					
:VIEWSel *A*	*A*: 1, 2	Sets operational screen.	All	122	N	N	Y
:VIEWSel?	*A* <NR1>	Queries operational screen.					
:VIEWPart *A*	*A*: 0, 1, 2	Executes screen partition.	All	123	N	N	Y
:VIEWPart?	*A* <NR1>	Queries screen partition.					

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version
*: 8835-01 only

⑥ CURSor command (Cursor setting and reading)

:CURSor

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:MODE *A$*	*A$*: OFF, TIME, VOLT, TRACe OFF, Xcur, Ycur, TRACe (in X-Y format) OFF, TRACe (FFT)	Sets the A and B cursor type.	All	123	Y	Y	Y
:MODE?	*A$*	Queries the A and B cursor type.					
:ABCUrsor *A$*	*A$*: A, ORA, ORB, A_B	Chooses among the A, B and A&B cursors.	All	124	Y	Y	Y
:ABCUrsor?	*A$*	Queries among the A, B and A&B cursors.					

Command	Data (for a query, response data)	Explanation	Func-tion	Ref page	35	26 41 42	20
:ACHAnnel *ch$*	*ch$*: CH1 to CH4 (8835) CH1 to CH8, ALL (8835-01) CH1 to CH32, ALLH, ALLL, ALL (8826) CH1 to CH16, ALL (8841, 8842) CH1 to CH16 (8720) X1 to X4 (in X-Y format: other than 8835)	Sets the A cursor channel.	MEM REC RMS R&M	124	Y	Y	Y
:ACHAnnel?	*ch$*	Queries the A cursor channel.					
:BCHAnnel *ch$*	*ch$*: CH1 to CH4 (8835) CH1 to CH32 (8826) CH1 to CH16 (8841, 8842, 8720) X1 to X4 (in X-Y format: other than 8835)	Sets the B cursor channel.	MEM REC RMS R&M	124	Y	Y	Y
:BCHAnnel?	*ch$*	Queries the B cursor channel.					
:APOSition *A*	(vertical cursor, trace cursor) *A*: 0 to amount of stored data 0 to 400 (X-Y format) 0 to 480 (X-Y format wide screen: 8826) 0 to 320 (X-Y format: 8720) 0 to 9999: FFT (STR, ACR, CCR, IMP) 0 to 400: FFT (HIS, OCT) 0 to 4000: FFT (others) (horizontal cursor) *A*: 0 to 400 0 to 480 (wide screen: 8826) 0 to 639 (8841, 8842) 0 to 320 (X-Y format: 8720)	Sets the position of the A cursor.	All	125	Y	Y	Y
:APOSition?	*A* <NR1>	Queries the position of the A cursor.					
:BPOSition *A*	Same as :APOSition	Sets the position of the B cursor.	All	125	Y	Y	Y
:BPOSition?	*A* <NR1>	Queries the position of the B cursor.					
:DTREad? *A$*	*B$* *A$*: A, B, B_A *B$*: readout value (t)	Queries the cursor readout value (t).	MEM REC RMS R&M	125	Y	Y	Y
:DVREad? *A$*	*B$ (, C$)* *A$*: A, B, B_A *B$, C$*: readout value (V, ℃, μ ε)	Queries the cursor readout value (V).	MEM REC RMS R&M	126	Y	Y	Y

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:ABCHAnnel *A$*	*A$*: G1, G2	Sets the graph for the A and B cursors.	FFT	126	A	Y	N
:ABCHAnnel?	*A$*	Queries the graph setting for the A and B cursors.					
:DFREad? *A$*	*B$, C$* *A$*: A, B, B_A *B$*: readout position for x-axis data *C$*: readout position for y-axis data	Queries the current cursor readout position.	FFT	127	A	Y	N

Note 35: 8835 (-01), 26: 8826, 41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version

⑦ MEMory command (Setting and querying input and output, etc., from the memory)
:MEMory

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:POINt *ch$, A*	*A*: 0 to 2000000 (8835) 0 to 4000000 (8835-01) 0 to 16000000 (8826, 8841, 8842) 0 to 1000000 (8720)	Sets point in memory for input and output.	MEM REC R&M	127	Y	Y	Y
:POINt?	*ch$, A* <NR1>	Queries point in memory for input and output.					
:MAXPoint?	*A* <NR1>: 0 (not stored) 100 to 2000000 (8835) 100 to 4000000 (8835-01) 100 to 16000000 (8826, 8841, 8842) 100 to 1000000 (8720) (÷ 100 = number of divisions)	Queries the amount of data stored.	MEM REC R&M	128	Y	Y	Y
:PREPare		Prepares the memory for receipt of waveform data.	MEM REC R&M	128	Y	Y	Y

Command	Data (for a query, response data)	Explanation	Func-tion	Ref page	35	26 41 42	20
:ADATa *B, C,...*	*B, C,...*: -2048 to 2047 (-737 to 3358 when measuring temperature with the 8937)	Input data to memory (ASCII).	MEM R&M	129	Y	Y	N
:ADATa? *A*	*A*: 1 to 80 (number of output units) *B, C,...*<NR1>: -2048 to 2047 (-737 to 3358 when measuring temperature with the 8937)	Output data from memory (ASCII).					
:VDATa *B, C,...*	*B, C,...*: voltage values (units V, μ ε, ℃)	Input data to memory (voltage values).	MEM R&M	130	Y	Y	N
:VDATa? *A*	*A*: 1 to 40 (amount of data) *B, C,...*<NR3>: voltage values (units V, μ ε, ℃)	Output stored data (voltage values).					
:GETReal		Captures real time data.	All	130	Y	Y	Y
:AREAl? *ch$*	*A* <NR1>: -2048 to 2047 (-737 to 3358 when measuring temperature with the 8937)	Output stored data. Real time data output (ASCII)	All	131	Y	Y	Y
:VREAl? *ch$*	*A* <NR3>: voltage value (units V, μ ε, ℃)	Real time data output (voltage value)	All	131	Y	Y	Y
:LDATa *B, C,...*	*B, C,...*: 0 to 15	Input logic data from memory.	MEM R&M	132	Y	Y	N
:LDATa? *A*	*A*: 1 to 100 (amount of output data) Response data <NR1>: 0 to 15	Output logic data from memory.					
:BDATa? *A*	*A*: 1 to 200 (amount of output data) Response data, binary, integer data (GP-IB only)	Performs binary transfer for stored data (GP-IB only).	MEM R&M	133	Y	Y	N
:LREAl? *ch$*	*A* <NR1>: 0 to 15	Logic real time data output	All	133	Y	Y	Y
:BREAl? *ch$*	Response data, binary, integer data	Real time data output (binary)	All	134	Y	Y	Y
:FFTPOint *A$, B*	*A$*: G1, G2 *B*: 0 to 9999 (STR, ACR, CCR, IMP) 0 to 4000 (LIN, RMS, PSP, TRF, COH, CSP) 0 to 400 (HIS, OCT)	Sets the output point for FFT data.	FFT	134	A	Y	N
:FFTPOint?	*A$, B* <NR1>	Queries the current output point for FFT data.					
:FFTData?	*A* unit, *B* unit *A*: X-axis data <NR3> *B*: Y-axis data <NR3>	Output FFT data.	FFT	135	A	Y	N
:RTLOad *A* (,*B*)	*A*: data start point *B*: data end point	Reads real-time store data.	R&M	135	N	Y	N

Command	Data (for a query, response data)	Explanation	Func-tion	Ref page	35	26 41 42	20
:RECAData *B1, B2, C1, C2,...*	*B1, B2, C1, C2,...*: -2048 to 2047	Storage data output (ASCII).	REC	136	N	N	Y
:RECAData? *A*	*A*: 1 to 40 (number of the output sample) *B1, B2, C1, C2,...*: <NR1>: -2048 to 2047	Queries the current output for storage data.					
:RECVData *B1, B2, C1, C2,...*	*B1, B2, C1, C2,...*: voltage value	Storage data output (voltage value).	REC	137	N	N	Y
:RECVData? *A*	*A*: 1 to 20 (number of the output sample) *B1, B2, C1, C2,...*: <NR3> voltage value	Queries the current output for storage data.					
:RECBData? *A*	*A*: 1 to 100 (number of the output sample) response data: binary, integer data	Transfers storage data in binary (only GP-IB).	REC	137	N	N	Y
:RECLData *B1, B2, C1, C2,...*	*B1, B2, C1, C2,...*: 0 to 15	Storage data output (logic).	REC	138	N	N	Y
:RECLData? *A*	*A*: 1 to 50 (number of the output sample) *B1, B2, C1, C2,...*: <NR1>: 0 to 15	Queries the current output for storage data.					

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version

⑧ SYSTem command (Setting and querying the system screen)
:SYSTem

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:USECH *A*	*A*: 1, 2, 4 (8835) 1, 2, 4, 8 (8835-01) 4, 8, 16, 32 (8826) 2, 4, 8, 16 (8841, 8842)	Sets number of channels used.	MEM	139	Y	Y	N
:USECH?	*A* <NR1>	Queries number of units used.					
:STARt *A$*	*A$*: ON, OFF	Enables and disables start key backup.	All	139	Y	Y	Y
:STARt?	*A$*	Queries start key backup enablement.					
:GRID *A$*	*A$*: OFF, STD, FINE, STD_Dark, FINE_Dark (TIME, TIME_Dark: 8826 only)	Sets the grid type.	All	140	Y	Y	Y
:GRID?	*A$*	Queries the grid type.					
:CHMArk *A$*	*A$*: OFF, NUMBer, COMMent	Enables and disables channel markers.	All	140	Y	Y	N
:CHMArk?	*A$*	Queries enablement of channel markers.					
:TMAXis *A$*	*A$*: TIME, TIME (60), SCALE, DATE	Sets the time axis display.	All	141	Y	Y	N
:TMAXis?	*A$*	Queries the time axis display.					
:LIST *A$*	*A$*: OFF, LIST, GAUGE, L_G	Sets list and gauge functions.	All	141	Y	Y	N
:LIST?	*A$*	Queries list and gauge functions.					
:PRIDensity *A*	*A*: 1 to 5	Sets the printer density.	All	141	Y	Y	N
:PRIDensity?	*A*	Queries the printer density.					
:CRTOff *A$*	*A$*: 0 (OFF), 1 to 30 (minutes)	Enables and disables the backlight saver.	All	142	Y	Y	Y
:CRTOff?	*A$*	Queries enablement of the backlight saver.					
:LCDDisp *A$*	*A$*: C1 to C9	Sets the screen color.	All	142	Y	Y	Y
:LCDDisp?	*A$*	Queries the screen color.					
:SETColor *A, B, C, D*	*A*: 0 to 26 (8835 (-01)) 0 to 30 (8826) 0 to 33 (8841, 8842, 8720) *B, C, D*: 0 to 7	Sets the customer color.	All	143	Y	Y	Y
:SETColor? *A*	*A, B, C, D* <NR1>	Queries the customer color.					
:BEEPer *A$*	*A$*: ON, OFF (8835) ON1, ON2, OFF	Enables and disables the beep sound.	All	144	Y	Y	Y
:BEEPer?	*A$*	Queries beep sound enablement.					

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:LANGuage *A$*	*A$*: JAPAnese, ENGLish	Sets the language.	All	144	Y	Y	Y
:LANGuage?	*A$*	Queries the language.					
:PRIUplow *A$*	*A$*: OFF, ON	Sets printing of the upper and lower limits.	All	144	N	Y	N
:PRIUplow?	*A$*	Queries printing of the upper and lower limits.					
:ZEROcom *A$*	*A$*: OFF, ON	Sets the zero position comment.	All	145	N	Y	N
:ZEROcom?	*A$*	Queries the zero position comment.					
:COUNter *A$ (,"NAME$", B)*	*A$*: OFF, DATE, NAME *NAME$*: counter name *B*: counter value	Sets the counter print.	All	145	N	Y	N
:COUNter?	*A$, (NAME$, B)*	Queries the counter print.					
:COPY *A$ (,B$)*	*A$*: IN_PRinter, EX_PRinter, FD, PC, COM (8835) IN_PRinter, EX_PRinter, FD, PC, COM, SCSI, MO (8826, 8841, 8842) *B$*: ESCP, RASTer	Sets the output destination by the COPY key.	All	146	Y	Y	N
:COPY?	*A$*	Queries the output destination by the COPY key.					
:BMPColor *A$*	*A$*: COLOR, GRAY, MONO, MONO_R	Sets the bit map file color.	All	146	Y	Y	N
:BMPColor?	*A$*	Queries the bit map file color.					
:BMPFile '*NAME$*'	*NAME$*: file name (8 characters)	Sets filenames of stored bitmaps.	All	147	*	Y	Y
:BMPFile?	*NAME$*	Queries filenames of stored bitmaps.					
:PRINt *A$ (,B$)*	*A$*: IN_PRinter, EX_PRinter, LAN *B$*: ESCP, RASTer	Sets the output destination by the PRINT key.	All	147	Y	Y	N
:PRINt?	*A$*	Queries the output destination by the PRINT key.					
:PRIColor *A$*	*A$*: COLOR, MONO	Sets the print color (external printer).	All	147	Y	Y	N
:PRIColor?	*A$*	Queries the print color.					
:SCSI *A$, B*	*A$*: 8826, SCSI *B*: 0 to 7	Sets the SCSI interface ID number.	All	148	N	Y	Y
:SCSI? *A$*	*A$, B* <NR1>	Queries the SCSI interface ID number.					

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:DATE *A, B, C*	*A*: 0 to 99 *B*: 1 to 12 *C*: 1 to 31	Sets the calendar.	All	148	Y	Y	Y
:DATE?	*A, B, C* (all <NR1>)	Queries the calendar.	All	148	Y	Y	Y
:TIME *A, B*	*A*: 0 to 23 *B*: 0 to 59	Sets the time.	All	149	Y	Y	Y
:TIME?	*A, B, C* (all <NR1>)	Queries the current time.					
:DATAClear		Clear data.	All	149	Y	Y	Y
:WAVEDensity *A$, B$*	*A$*: C1 to C12 *B$*: DARK, MIDDark, NORMal, LIGHt	Sets the printer density of each waveform color.	All	149	N	Y	N
:WAVEDensity? *A$*	*A$, B$*	Queries the printer density of each waveform color.					
:EXTterm *A$*	*A$*: PRINt, SMPL	Sets the external terminals.	All	150	*	N	N
:EXTterm?	*A$*	Queries the external terminals.					

Note 35: 8835 (-01), 26: 8826, 41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version
*: 8835-01 only

⑨ SCALing command (Setting and querying scaling)
:SCALing

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:KIND *A$*	*A$*: POINT, RATIO	Sets the type of scaling.	All	150	Y	Y	Y
:KIND?	*A$*	Queries the type of scaling.					
:SET *ch$, A$*	*A$*: OFF, SCI, ENG	Enables and disables scaling.	All	150	Y	Y	Y
:SET? *ch$*	*ch$, A$*	Queries scaling enablement.					
:VOLT *ch$, A*	*A*: -9.999E+9 to +9.999E+9	Sets the scaling conversion value (RATIO).	All	151	Y	Y	Y
:VOLT? *ch$*	*ch$, A* <NR3>	Queries the scaling conversion value.					
:OFFSet *ch$, A*	*A*: -9.999E+9 to +9.999E+9	Sets scaling offset (RATIO).	All	151	Y	Y	Y
:OFFSet? *ch$*	*ch$, A* <NR3>	Queries scaling offset.					
:UNIT *ch$, 'A$'*	*A$*: scaling unit (7 characters)	Sets scaling unit.	All	152	Y	Y	Y
:UNIT? *ch$*	*ch$,* "*A$*"	Queries scaling unit.					
:VOUPLOw *ch$, B, C*	*B, C*: -9.999E+29 to +9.999E+29	Sets the scaling VOLT UP, LOW (POINT).	All	152	Y	Y	Y
:VOUPLOw? *ch$*	*ch$, B* <NR3>, *C*<NR3>	Queries VOLT UP, LOW.					
:SCUPLOw *ch$, B, C*	*B, C*: -9.999E+29 to +9.999E+29	Sets the scaling SC UP, LOW (POINT).	All	153	Y	Y	Y
:SCUPLOw? *ch$*	*ch$, B* <NR3>, *C*<NR3>	Queries the scaling SC UP, LOW.					

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version

⑩ COMMent command (Setting and querying comments)
:COMMent

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:TITLe *A$, 'B$'*	*A$*: OFF, SETTing, COMMent, S_C *B$*: comment string (up to 40 characters)	Sets a title comment.	All	153	Y	Y	Y
:TITLe?	*A$, "B$"*	Queries a title comment.					
:EACHch *(ch$,) A$*	*ch$*: logic only (omitted for analog) *A$*: OFF, SETTing, COMMent, S_C (analog) OFF, ON (logic)	Enables or disables a channel comment.	All	154	Y	Y	Y
:EACHch? *(ch$,)*	*(ch$,) A$*	Queries channel comment enablement.					
:CH *ch$, (NO$,) 'A$'*	*ch$*: CH1 to CH4, CHA to CHD (8835) CH1 to CH32, CHA to CHH (8826) CH1 to CH16, CHA to CHD (8841, 8842, 8720) *NO$*: NO1 to NO4 (logic only, omitted for analog) *A$*: comment string (up to 40 characters)	Sets a comment for a particular channel.	All	154	Y	Y	Y
:CH? *ch$ (,NO$)*	*ch$, (NO$,) 'A$'*	Queries comment for a particular channel.					

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version

⑪ CALCulate command (Calculation setting and querying)
:CALCulate

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:MEASure *A$*	*A$*: ON, OFF, EXEC (execute)	Sets waveform parameter calculation.	MEM	155	Y	Y	Y
:MEASure?	*A$*	Queries waveform parameter calculation.					
:MEASPrint *A$*	*A$*: OFF, ON	Sets printing calculation results.	MEM	155	Y	Y	N
:MEASPrint?	*A$*	Queries printing calculation results.					
:MEASFsave *A$*	*A$*: OFF, FD, PC (8835, 8835-01) OFF, FD, PC, SCSI, MO (8826, 8841, 8842, 8720)	Sets storing a calculation result.	MEM	156	Y	Y	Y
:MEASFsave?		Queries storing a calculation result.					
:MEASSet *NO$, A$, ch$* (*ch1$, ch2$* (XYAREA))	*NO$*: NO1 to NO4 *A$*: OFF, MAX, MIN, MAXT MINT, PP, AVE, RMS, AREA, PERI, FREQ, RISE, FALL, XYAREA *ch$*: ALL, CH1 to CH4 (8835) ALL, CH1 to CH16 (8841, 8842, 8720) ALL, CH1 to CH32 (8826)	Sets waveform parameter calculation.	MEM	156	Y	Y	Y
:MEASSet? *NO$*	*A$, ch$* (*A$, ch1$, ch2$* (XYAREA))	Queries waveform parameter calculation.					
:ANSWer? *NO$, ch$*	*A$*: OFF, MAX, MIN, MAXT MINT, PP, AVE, RMS, AREA, PERI, FREQ, RISE, FALL, XYAREA *NO$*: NO1 to NO4 *A$, B* <NR3>: calculation result	Queries a calculation result.	MEM	157	Y	Y	Y
:COMP *NO$, A$*	*NO$*: NO1 to NO4 *A$*: ON, OFF	Enables and disables decision for waveform parameter calculation.	MEM	157	A	Y	Y
:COMP? *NO$*	*A$*	Queries enablement of decision for waveform parameter calculation.					
:COMPArea *NO$, upper, lower*	*NO$*: NO1 to NO4 *upper, lower*: -9.9999E+29 to +9.9999E+29	Sets upper and lower limits for decision for waveform parameter calculation.	MEM	158	A	Y	Y
:COMPArea? *NO$*	*upper* <NR3>, *lower* <NR3>	Queries upper and lower limits for decision for waveform parameter calculation.					

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:WVCALc *A$*	*A$*: ON, OFF, EXEC (execute)	Sets waveform processing calculation.	MEM	158	A	Y	N
:WVCALc?	*A$*	Queries waveform processing calculation.					
:Z *Z$*, "*A$*"	*Z$*: Z1 to Z16 *A$*: calculation equation	Sets the waveform processing calculation equation.	MEM	159	A	Y	N
:Z? *Z$*	*A$*	Queries the waveform processing calculation equation.					
:FACTor *A$*, *B*	*A$*: A to P *B*: -9.9999E+29 to +9.9999E+29	Sets coefficients a to p.	MEM	159	A	Y	N
:FACTor? *A$*	*B* <NR3>	Queries coefficients a to p.					
:ZDIsplay *Z$*, *ch$*, *A$*	*ch$*: NONE, CH1 to CH32 *Z$*: Z1 to Z16 *A$*: AUTO, MANUal	Sets the display channel for the calculated result.	MEM	160	A	Y	N
:ZDIsplay? *Z$*	*ch$*, *A$*	Queries the display channel for the calculated result.					
:MOVE *Z$*, *A*	*Z$*: Z1 to Z16 *A*: 0 to 4000	Sets the moving averaging.	MEM	160	A	Y	N
:MOVE? *Z$*	*A* <NR1>	Queries the moving averaging.					
:SLIDe *Z$*, *A*	*Z$*: Z1 to Z16 *A*: -4000 to 4000	Sets the parallel movement.	MEM	161	A	Y	N
:SLIDe? *Z$*	*A* <NR1>	Queries the parallel movement.					
:COMPStop A$	A$: GO, NG, G_N	Sets the stop mode.	MEM	158	*	Y	N
:COMPStop?	*A$*	Queries the stop mode.					
:COMPJudge? *NO$, ch$*	*A$:* GO, NG, * *NO$:* NO1 to NO4	Queries the result of the judgement.	MEM	159	*	Y	N

Note 35: 8835 (-01), 26: 8826, 41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version
* : 8835-01 only

⑫ FDISK command (Setting and querying file operation)

:FDISK

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:MEDIA *A$*	*A$*: FD, PC (8835) FD, PC, SCSI, MO (8826, 8841, 8842, 8720)	Sets the media type.	All	161	Y	Y	Y
:MEDIA?	*A$*	Queries the media type.					
:SAVE *'NAME1$. NAME2$', A$, B$ (, C$)*	*NAME1$*: file name (up to 8 characters) *NAME2$*: file extension (up to 3 characters) *A$*: type of file Bin: binary data Text: text data Set: settings Area: waveform decision area *A$*: type of file (During memory segmentation or in R&M) BAll: binary data (All blocks are saved.) BOne: binary data (One block is saved.) TAll: text data (All blocks are saved.) TOne: text data (One block is saved.) *B$*: channels to save ALL, CH1 to CH4, LOGIC (8835) ALL, CH1 to CH32, LOGIC (8826) ALL, CH1 to CH16, LOGIC (8841, 8842, 8720) *C$*: spacing OFF, 1_2 to 1_1000	Saves a file.	All	162	Y	Y	Y
:LOAD *NO (, A$)*	*NO*: file number *A$*: NEW, ADD	Load a file.	All	163	Y	Y	Y
:LOAD *'NAME1$. NAME2$' (, A$)*	*NAME1$*: file name (up to 8 characters) *NAME2$*: file extension (up to 3 characters) *A$*: NEW, ADD	Load a file.	All	163	Y	Y	Y

Command	Data (for a query, response data)	Explanation	Function	Ref page	35	26 41 42	20
:INFOr? *NO*	*NO*, "*NAME$*", "*DATE$*","*TIME$*", *A*, *B$*, *C$*, *D*, "*TDATE$*", "TTIME$" *NAME$*: file name *DATE$*: year/month/day of save *TIME$*: hour:min:sec of save *A*: file size (bytes) *B$*: function *C$*: measurement contents *D*: recording length *TDATE$*: year/month/day of trigger *TTIME$*: trigger time	Queries information about a file.	All	163	Y	Y	Y
:DELEte *NO*	*NO*: file number	Deletes a file or directory.	All	164	Y	Y	Y
:DELEte *'NAME1$. NAME2$'*	*NAME1$*: file name (up to 8 characters) *NAME2$*: file extension (up to 3 characters)	Deletes a file or directory.	All	164	Y	Y	Y
:FORMat *(A$)*	*A$*: 2DD, 2HD, 2HC	Formats media.	All	164	Y	Y	Y
:MKDIR *'A$'*	*A$*: directory name	Creates a directory.	All	164	Y	Y	Y
:CHDIR *NO*	*NO*: file number	Changes the current directory.	All	165	Y	Y	Y
:FILE?	*A* <NR1>: number of files	Queries the number of files.	All	165	Y	Y	Y
:NINFor? *NO*	*NO*, "*NAME$*", *A$* *NO*: file number *NAME$*: file name *A$*: directory of a file	Queries filename.	All	165	Y	Y	Y
:DIR?	*A$*: directory name	Queries the current directory.	All	165	Y	Y	Y
:FREE?	*A$*: allowable number of bytes	Queries the allowable number of bytes.	All	166	Y	Y	Y

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version

⑬ GRAPh Command (Commands relating to graphics editor)

:GRAPh

Command	Data (for a query, response data)	Explanation	Func-tion	Ref page	35	26 41 42	20
:EDIT *A$*	*A$*: OFF, ON	Enables and disables the editor.	MEM FFT	166	A	Y	N
:EDIT?	*A$*	Queries editor enablement.					
:PAINT *X, Y*	*X*: x-coordinate *Y*: y-coordinate	Begins solid fill from the point specified by (X, Y).	MEM FFT	166	A	Y	N
:PARAllel *high, low, right, left*	*high, low, right, left*: 0 to 10 (div)	Carries out a parallel movement of the drawing.	MEM FFT	167	A	Y	N
:LINE *X1, Y1, X2, Y2*	*X1, X2*: x-coordinates *Y1, Y2*: y-coordinates	Draws a line from (*X1, Y1*) to (*X2, Y2*).	MEM FFT	167	A	Y	N
:ERASe *X1, Y1, X2, Y2*	*X1, X2*: x-coordinates *Y1, Y2*: y-coordinates	Erases the line from (*X1, Y1*) to (*X2, Y2*).	MEM FFT	167	A	Y	N
:STORage		Loads a waveform into the editor.	MEM FFT	168	A	Y	N
:REVErse		Reverses the video of the drawing.	MEM FFT	168	A	Y	N
:ALLClear		Clears the entire drawing.	MEM FFT	168	A	Y	N
:CLEAr *X1, Y1, X2, Y2*	*X1, X2*: x-coordinates *Y1, Y2*: y-coordinates	Clears the rectangle with the points (*X1, Y1*) and (*X2, Y2*) at diagonally opposite corners.	MEM FFT	168	A	Y	N
:UNDO		Reverses the effect of the immediately previous editor command.	MEM FFT	168	A	Y	N
:SAVE		Saves the decision area created with the editor.	MEM FFT	168	A	Y	N

Note 35: 8835 (-01), 26: 8826,
41: 8841, 42: 8842
20: 8720
Y: Yes
A: Advanced version

3.2 Detailed Explanation of the Commands

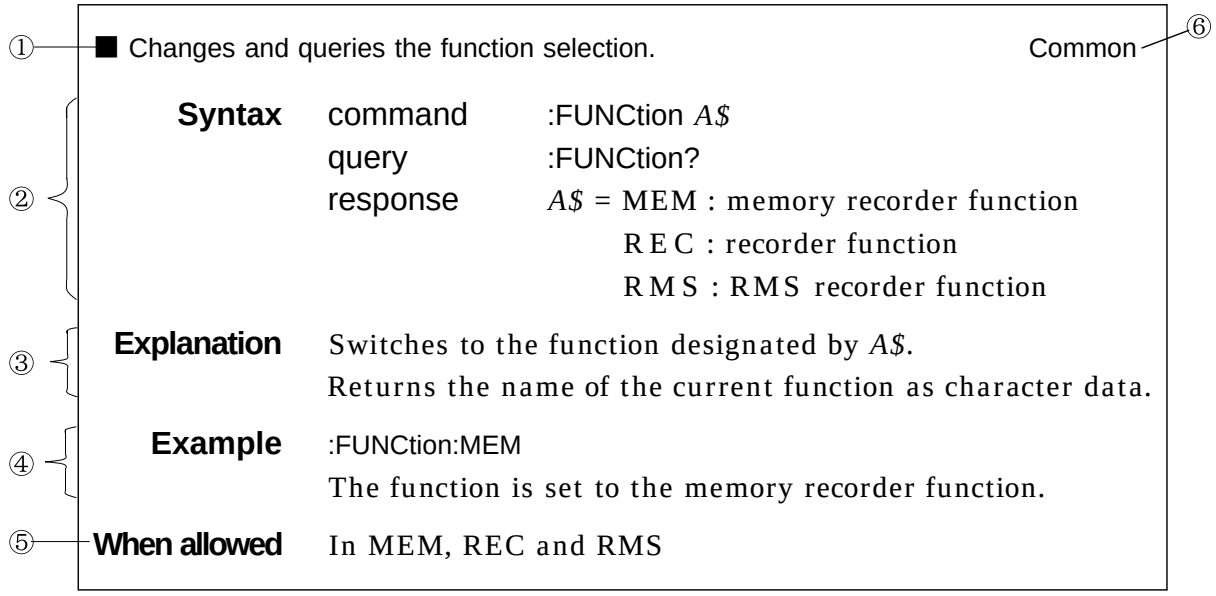
NOTE When using the HIOKI MEMORY HiCORDER can be used with the HIOKI "9557 RS-232C CARD / 9558 GP-IB CARD" except following products, reffer to the communication comands manual (Flopply disk) supplied with the MEMORY HiCORDER.

■ The products consultable this manual: 8826, 8835, 8835-01, 8841, 8842

3.2.1 Explanation

The following sections describe the format and functions of individual commands.
The following is an example of how the descriptions are organized.

Example



① Command function

② Command syntax
command gives the syntax of a command program message,
query the syntax of a query program message, and
response the format of the response message.
The parameters, referred to as data, are shown as follows:

A, B, C,...	Numerical data (e.g. 1.5, 10E-3)
A$, B$,...	Character data (e.g. A, B1, GND, OFF)
"*A*", "*A$*",...	Character string data (e.g. "1.5", "mA")

(Single quotation marks (') can be used instead of double quotation marks (").)

The format of numerical data follows the formats <NR1>, <NR2>, and <NR3>.

Example

A <NR1>	Numerical parameter in NR1 format
B <NR2>	Numerical parameter in NR2 format
C <NR3>	Numerical parameter in NR3 format

NOTE If no format is mentioned, <NR1> format is accepted.

NR1 format	integer data
NR2 format	fixed point numbers
NR3 format	floating point numbers

The term "NRf format" includes all these three formats.

When the unit is receiving a command or query program message, it accepts format, but when it is sending it utilizes whichever one of the formats <NR1> to <NR3> is indicated in the particular command.
Response messages may or may not have headers prefixed.

③ Explanation of the command function.

④ Example of command use.

⑤ This lists the functions in which the command may be used.

MEM	memory recorder function
REC	recorder function
RMS	RMS recorder function

⑥ Models

Common	Common command that can be used for all the models
8835	Command that can be used for the 8835
8835-01	Command that can be used for the 8835-01
8826	Command that can be used for the 8826
8841	Command that can be used for the 8841
8842	Command that can be used for the 8842
8720	Command that can be used for the 8720

Execution of commands

- Commands are input into the input buffer and are executed in order.
- However the :ABORT command is executed immediately, even if commands are waiting in the input buffer - more precisely, at the instant its terminator is received.
- Commands other than those which can be handled by the unit in its current state are not executed but generate execution errors. This happens, for example, when in memory recorder function it is attempted to execute a recorder mode setting.
- Further, almost all commands cannot be executed during measurement operation.

8841 and 8842

Commands related to FFT, and recorder and memory functions are supported from Version 2.00.

Regarding the command for specifying the channel number

Unless specifically mentioned, the ch$ character string specifying the channels becomes as follows.

ch$ = CH1 to CH4 (8835)
CH1 to CH8 (8835-01)
CH1 to CH16 (8841, 8842, 8720)
CH1 to CH32 (8826)

3.2.2 Standard Commands Stipulated by IEEE 488.2

1. System data commands

*IDN? Common

■ Queries device ID.

Syntax query *IDN?
response HIOKI, 8835, 0, V1. 00
① ② ③ ④
①: First field Manufacturer's name
②: Second field Model name
③: Third field Serial number (not used: 0)
④: Fourth field Software version

*OPT? Common

■ Queries device option provision.

Syntax query *OPT?
response *ch1* <NR1>, *ch2* <NR1>,..., *ch4* <NR1> (8835)
ch1 <NR1>, *ch2* <NR1>,..., *ch8* <NR1> (8835-01)
ch1 <NR1>, *ch2* <NR1>,..., *ch32* <NR1> (8826)
ch1 <NR1>, *ch2* <NR1>,..., *ch16* <NR1> (8841, 8842, 8720)
ch1: Whether or not channel 1 input unit present
ch2: Whether or not channel 2 input unit present
.
.
.
ch32: Whether or not channel 32 input unit present
0: not present
1: analog unit
2: voltage/temperature unit
3: strain unit
4: FFT unit
5: F/V unit
6: charge unit
7: 4-channel unit

Explanation Whether or not input unit present is returned as an NR1 numerical value.

2. Internal operation commands

*RST — Common

■ Device initial setting.

Syntax command *RST

Explanation Initializes the unit (same as system reset).

Note It does not clear GP-IB and RS-232C related items.
(the event registers, the enable registers, the input buffer and the output queue)

*TST? — Common

■ Queries the result of the ROM/RAM check.

Syntax query *TST?
response *A* <NR1>
A = 0, 1
0: normal
1: failure

Explanation The result of the ROM/RAM check of the unit is returned as an NR1 numerical value.

Note If the unit communicates with LAN, it can not receive the response.

3. Synchronous commands

*OPC — Common

■ After all action has been completed during execution, sets the LSB (bit 0) of SESR (the standard event status register).

Syntax command *OPC

Explanation When the command preceding the *OPC command completes execution, the LSB of SESR is set.

Example :FUNC MEM; *OPC;:CONF:TDIV +500.0E-6
A$* *B$
(After the execution of the commands *A$* and *B$* is completed, the LSB of SESR is set.)

*OPC? Common

■ After execution is completed, replies with ASCII [1].

Syntax query *OPC?
response 1

Explanation When the command preceding the *OPC command completes execution, the response of ASCII [1] is made.

*WAI Common

■ After the execution of the command is completed, subsequently performs the following command.

Syntax command *WAI

Example *A$*;*B$*;*WAI;*C$*
The command *C$* following *WAI is not executed until the execution of the commands *A$* and *B$* is completed.

4. Status and event control commands

*CLS Common

■ Clears the status byte and associated queues (except for the output queue).

Syntax command *CLS

Explanation This instruction clears the event register associated with each bit of the status byte register. It also clears the status byte register.

Note Because it does not clear the output queue, it has no effect upon bit 4 (MAV) of the status byte.

*ESE — Common

■ Writes the standard event status enable register (SESER). (GP-IB only)

Syntax command *ESE *A*
A = 0 to 255

Explanation Sets the mask pattern of SESER to a value in the range 0 to 255. Outside this range, an execution error occurs. (The initial value (when the power is turned on) is 0.)

Example *ESE 36
Bit 5 and bit 2 of SESER are set.

*ESE? — Common

■ Reads the standard event status register (SESER). (GP-IB only)

Syntax query *ESE?
response *A* <NR1>
A = 0 to 255

Explanation The contents of SESER as set by the *ESE command are returned as an integral value in the range 0 to 255.

*ESR? command — Common

■ Reads out and clears the contents of the standard event status register (SESR).

Syntax query *ESR?
response *A* <NR1>

Explanation The contents of SESR are returned as an NR1 numerical value.

*SRE — Common

■ Writes the service request enable register (SRER). (GP-IB only)

Syntax command *SRE *A*
A = 0 to 255

Explanation Sets the mark pattern of SRER to a value in the range 0 to 255. Outside this range, an execution error occurs. However, the value of bit 6 is disregarded.

Example *SRE 33
Bits 5 and 0 of SRER are set.

*SRE? — Common

■ Reads the service request enable register (SRER). (GP-IB only)

Syntax query *SRE?
response *A* <NR1>
A = 0 to 63, 128 to 191

Explanation The contents of SRER as set by the *SRE command are returned as an NR1 numerical value in the range 0 to 63, 128 to 191. Bit 6 is always 0.

*STB? — Common

■ Reads the status byte and MSS bit, without performing serial polling.

Syntax query *STB?
response *A* <NR1>
A = 0 to 255

Explanation This is the same as reading out the status byte with serial polling.

Note Bit 6 is not RQS, but is MSS.

:ESE0 — Common

■ Writes event status enable register 0 (ESER0). (GP-IB only)

Syntax command :ESE0 *A*
A = 0 to 255

Explanation Sets the mask pattern of ESER0 to a value in the range of 0 to 255. Outside this range, an execution error occurs. The initial value (when the power is turned on) is 0.

Example :ESE0 36
This sets bit 5 and bit 2 of ESER0.

:ESE0? Common

■ Reads event status enable register 0 (ESER0).

Syntax

query :ESE0?
response *A* <NR1>
A = 0 to 255

Explanation The contents of ESER0 are returned as an NR1 numerical value.

:ESR0? Common

■ Reads event status register 0 (ESR0).

Syntax

query :ESR0?
response *A* <NR1>
A = 0 to 255

Explanation The contents of ESR0 are returned as an NR1 numerical value, and ESR0 is cleared.

3.2.3 Specific Commands

1. Execution control commands

■ Performs starting. Common

Syntax command :STARt

Explanation Same as the START key of the unit.
Starts waveform sampling operation.

When allowed In all functions.

■ Performs stopping. Common

Syntax command :STOP

Explanation Same as the STOP key of the unit.
Terminates at the instant that waveform sampling operation is completed.

When allowed In all functions.

■ Aborts processing. Common

Syntax command :ABORT

Explanation Same as the STOP key of the unit. Forced halt. Terminates even if waveform sampling operation is not yet completed. Also stops printer operation.

When allowed In all functions.

■ Performs printing. Except 8720

Syntax command :PRINt

Explanation Same as the PRINT key of the unit.

When allowed In all functions.

■ Screen copy function. Except 8720

Syntax command :HCOPy

Explanation Same as the COPY key of the unit. Produces a hard copy of the screen.

When allowed In all functions.

■ Feeds printer paper. Except 8720

Syntax command :FEED *A*
A = 1 to 255

Explanation Feeds the paper by a distance from 1 to 255 in millimeters determined by the numerical value.

When allowed In all functions.

■ Performs report printing. Except 8720

Syntax command :REPOrt

Explanation Same as the FEED key + COPY key of the unit. Performs report printing.

When allowed In all functions.

■ Performs automatic range setting. Except 8720

Syntax command :AUTO

Explanation Same as the AUTO key of the unit. Sets the time axis range and the voltage axis range automatically, and measures.

When allowed In MEM.

■ Queries the unit error number. Common

Syntax query :ERRor?
response *A* <NR1>
A = error no.

Explanation The number of error or warning that has occurred on the unit is returned in <NR1> as a numerical value. (For errors or warnings, refer to the instruction manual included with the unit.)

When allowed In all functions.

■ Queries the status. 8835-01, 8826, 8841, 8842

Syntax query :STATus?
response *A*

A <NR1>						
bit6	bit5	bit4	bit3	bit2	bit1	bit0

A = bit0: starting
bit1: in storage
bit2: waiting for the trigger
bit3: waiting for the pre-trigger
bit4: making the waveform
bit5: saving
bit6: printing

Explanation Returns the current status of the unit.

When allowed In all functions.

■ Enables and disables headers, and queries header enablement.

Common

Syntax

command :HEADer *A$*
query :HEADer?
response *A$*
A$ = OFF, ON

Explanation Sets header enablement. When headers are enabled, responses to queries are prefixed by headers; when headers are disabled, responses are not so prefixed. Returns whether or not headers are prefixed to responses to queries. The initial toggle state for headers (when the power is turned on) is OFF

Example Response to :HEADer?:
① When headers are disabled: OFF
② When headers are enabled: :HEADER ON

When allowed In all functions.

■ Changes and queries the function selection. Common

Syntax

command :FUNCtion *A$*
query :FUNCtion?
response *A$*
A$ = MEM: memory recorder function
REC: recorder function
RMS: RMS recorder function
R_M: recorder and memory function
FFT: FFT function

Explanation Switches to the function designated by *A$*.
Returns the name of the current function as character data.

Example :FUNCtion MEM
The function is set to the memory recorder function.

When allowed In all functions.

■ Queries the communication errors. (RS-232C only) Common

Syntax

command :CERRor?
response *A*, *B*, *C* <NR1>
A: parity error
B: overrun error
C: framing error

Explanation The number of times of communication errors are returned in <NR1> as a numerical value.

2. CONFigure command (Sets and queries time axis range, recording length, etc.)

:CONFigure

■ Sets and queries the time axis range. Except 8720

Syntax

command	:CONFigure:TDIV *A*
query	:CONFigure:TDIV?
response	*A* <NR3>, 0: External sampling (except 8835)

Explanation Sets the time axis range to a numerical value (unit seconds).
Returns the currently set value of the time axis range as an NR3 numerical value.
(If an attempt is made to set the time axis range to a non-permitted value, and there is a range above that value, that range will be selected.)

Example
```
:CONFigure:TDIV +500.0E-6
```
Sets the time axis range to 500 µs.

When allowed In MEM, REC and RMS.

■ Sets and queries the time axis ranges (recorder and memory function). 8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:TDIV *A*, *B*
query	:CONFigure:TDIV?
response	*A*, *B* <NR3>, 0: External sampling (except 8835) *A* = time axis range for REC *B* = time axis range for MEM

Explanation Sets the time axis ranges, for both recorder and memory recorder modes, to numerical values (unit seconds).
Returns the currently set values of the time axis ranges, for both REC and MEM, as NR3 numerical values.
(If an attempt is made to set either of these time axis ranges to a non-permitted value, and there is a range above that value, that range will be selected.)

Example
```
:CONFigure:TDIV +500.E-3, +100.E-6
```
Sets the time axis range for recorder mode to 500 ms, and the time axis range for memoery recorder mode to 100 µs.

When allowed In R&M.

■ Sets and queries the sampling period. Except 8720

Syntax

command	:CONFigure:SAMPle *A*
query	:CONFigure:SAMPle?
response	*A* <NR3> (unit seconds)

Explanation Sets the sampling period.
The available sampling period depends on the time axis range.

Example **:CONFigure:SAMPle +1.00E-6**
Sets the sampling period to 1 µs.

When allowed In REC.

■ Sets and queries the sampling speed. 8720

Syntax

command	:CONFigure:SAMPle *A$*
query	:CONFigure:SAMPle?
response	*A$*

A = Sampling speed
FAST, SLOW

Explanation Sets the sampling speed.
Returns the current setting of the sampling speed.

Example **:CONFigure:SAMPLE FAST**
Samples at high speed.

When allowed In REC.

■ Sets and queries the frequency. Except 8720

Syntax

command	:CONFigure:FREQuency *A*
query	:CONFigure:FREQuency?
response	*A* <NR1>

A = 50, 60

Explanation Sets the frequency of the input signal.

When allowed In RMS.

■ Sets and queries the recording length. Except 8720

Syntax

command	:CONFigure:SHOT *A*
query	:CONFigure:SHOT?
response	*A* <NR1>

Explanation Sets the numerical value of the recording length (unit divisions).
Sets the recording length during memory segmentation.
Returns the currently set value of the recording length as an NR1 numerical value.
0 indicates CONT.

Example
```
:CONFigure:SHOT 15
```
Sets the recording length to 15 divisions.

When allowed In MEM, REC and RMS.

■ Sets and queries the recording length (recorder and memory function).
8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:SHOT *A, B*
query	:CONFigure:SHOT?
response	*A, B* <NR1>

A = recording length for REC (0: continuous)
B = recording length for MEM

Explanation Sets the numerical value of the recording lengths (unit divisions).
Returns the currently set values of the recording lengths as NR1 numerical values.

Example
```
:CONFigure:SHOT 0,25
```
Sets the recording length for recorder mode to continuous, and the recording length for memoery recorder mode to 25 divisions.

When allowed In R&M.

■ Sets and queries the recording time. 8720

Syntax

command :CONFigure:RECTime *A*
query :CONFigure:RECTime?
response *A* <NR1>
A = Recording time (unit seconds)
0: Continuous, 1 to 35999999

Explanation

Sets the data recording time in units of seconds.
Recording length settings that exceed 10000 DIV cannot be set.
Sets to 0 for continuous recording.
Returns the currently set value of the recording time.

Example

```
:CONFigure:RECTime 5025
```

Sets the recording time to 1 hour 23 minutes 45 seconds.

When allowed

In REC.

■ Sets and queries the recording speed. 8720

Syntax

command :CONFigure:RECSpeed *A*
query :CONFigure:RECSpeed?
response *A* <NR3>
A = Recording speed (unit seconds)
0.002 to 180

Explanation

Sets the data reading speed.
Sets the measuring time interval between single pieces of data.
Measured time is set to approximately 1 DIV for data recorded at a speed of 100x.
Returns the currently set value of the recording time.

Example

```
:CONFigure:RECSpeed 0.1
```

Sets the recording speed to 100 ms.

When allowed

In REC.

■ Sets and queries the format. Common

Syntax

command :CONFigure:FORMat *A$*
query :CONFigure:FORMat?
response *A$*

(8835)
A$ = SINGle, DUAL, QUAD, XYDot, XYLine : MEM, REC
SINGle, DUAL, QUAD : RMS, R&M
SINGle, DUAL, NYQuist : FFT

(8826)
A$ = SINGle, DUAL, QUAD, OCT, HEX, XYSingle,
XYQuad : MEM, REC
SINGle, DUAL, QUAD, OCT, HEX : RMS, R&M
SINGle, DUAL, NYQuist : FFT

(8841, 8842)
A$ = SINGle, DUAL, QUAD, OCT, HEX, XYSingle,
XYDual : MEM, REC
SINGle, DUAL, QUAD, OCT, HEX : RMS, R&M
SINGle, DUAL, NYQuist : FFT

A$ = SINGle, DUAL, QUAD, OCT
XYSingle, XYDual

(8720)

Explanation Sets the format.
Returns the current format as character data.

Example **:CONFigure:FORMat SINGle**
Sets the format to SINGle.

When allowed In all functions.

■ Sets and queries the interpolation function.

8835 (-01) A, 8826, 8841, 8842, 8720

Syntax

command :CONFigure:DOTLine *A$*
query :CONFigure:DOTLine?
response *A$*

A$ = DOT, LINE

Explanation Sets the interpolation function (DOT or LINE).
Returns the currently set interpolation as character data.

Example **:CONFigure:DOTLine LINE**
Sets the interpolation function to LINE.

When allowed In MEM and REC (when the XY screen is selected), (8835: in FFT only).

■ Sets and queries the printer output style. Except 8720

Syntax

command	:CONFigure:PRKInd *A$*
query	:CONFigure:PRKInd?
response	*A$*

A$ = WAVE, LOGGing

Explanation Sets the printer output style.
Returns the current setting of the printer output style as character data.

Example `:CONFigure:PRKInd WAVE`
Sets the printer output style to be waveform.

When allowed In all functions.

■ Enables and disables, and queries the smooth printing function. Except 8720

Syntax

command	:CONFigure:SMOOth *A$*
query	:CONFigure:SMOOth?
response	*A$*

A$ = OFF, ON

Explanation Enables and disables the smooth printing function.
Returns the current enablement state of the smooth printing function as character data.

Example `:CONFigure:SMOOth ON`
Sets the smooth printing function to ON.

When allowed In MEM and R&M.

■ Sets and queries the logging output interval. Except 8720

Syntax

command	:CONFigure:LOGGing *A*
query	:CONFigure:LOGGing?
response	*A* <NR2>

A = 0.01 to 100

Explanation Sets the logging output interval.
Returns the current setting of the logging output interval as an NR2 numerical value.
In the recorder and memory function, sets the logging output interval for the current display function.

Example `:CONFigure:LOGGing 100`
Sets the logging output interval to 100 samples.

When allowed In all functions.

■ Enables and disables, and queries the roll mode function. Except 8720

Syntax

command	:CONFigure:ROLL *A$*
query	:CONFigure:ROLL?
response	*A$*

A$ = OFF, ON

Explanation Enables and disables the roll mode function.
Returns the current enablement state of the roll mode function as character data.

Example

```
:CONFigure:ROLL ON
```

Sets the roll mode function to ON.

When allowed In MEM.

■ Sets and queries the auto print function. Except 8720

Syntax

command	:CONFigure:ATPRint *A$* (,*B$*)
query	:CONFigure:ATPRint?
response	*A$* (,*B$*)

A$ = OFF, ON, LAN
B$ = MONO, COLOR

Explanation Sets the auto print function.
Returns the current enablement state of the auto print function as character data. For A$=LAN, color should also be set.

Example

```
:CONFigure:ATPRint ON
```

Sets the auto print function to ON (built-in printer).

When allowed In MEM and FFT.

■ Sets and queries the auto save function (output target). Common

Syntax

command :CONFigure:ATSAve *A$, B$ (, C$)*
query :CONFigure:ATSAve?
response *A$, B$ (, C$)*

A$ = OFF, FD, PC, LAN (8835)
OFF, FD, PC, SCSI, MO, LAN (8826, 8841, 8842, 8720)
OFF: Auto save is disabled. (*B$* and *C$* are omitted.)
FD: Stores on floppy disk automatically
PC: Stores on PC card automatically
SCSI: Stores on SCSI device automatically
MO: Stores on MO disk automatically
LAN: Stores on PC connected to LAN.

B$ = store format
Bin: binary data
Text: text data

C$ = saved function (only in R&M)
REC: Stores only the REC waveform
MEM: Stores only the MEM waveform
R_M: Stores both the REC and MEM waveforms

Explanation

Sets the auto save function (output target).
Omit *B$* and *C$* only when *A$* = OFF.
In the R&M function, sets the saved function as well.
Returns the current setting of the auto save function as character data.

Example

```
:CONFigure:ATSAve FD, Bin
```
Stores on floppy disk automatically as binary data.
```
:CONFigure:ATSAve PC, Text, R_M
```
In the R&M function, stores on PC card automatically as text data (stores both the REC and MEM waveforms).

When allowed

In all functions.

■ Sets and queries the file name for auto save function.

8835-01,8826,8841,8842,8720

Syntax

command :CONFigure:ATFIle 'NAME*$*'
query :CONFigure:ATFIle?
response 'NAME*$*'

NAME*$* = File name (within 8 characters)

Explanation

Sets the file name for auto save function(within 8 characters).
Returns the current file name in auto saving as character data.

Example

```
:CONFigure:ATFIle 'AUTO'
```
Sets the file name to 'AUTO' in auto saving.

When allowed

In all functions.

■ Sets and queries the delete save function.

8835-01, 8826, 8841, 8842, 8720

Syntax

command	:CONFigure:DELSave *A$*
query	:CONFigure:DELSave?
response	*ch$, A$*

A$ = DEL (Deleted and saved), NORMal (Normal save)

Explanation Sets the delete save function during automatic saving.
Returns the current setting of the delete save function as a character string.

Example **:CONFigure:DELSave DEL**
In automatic saving, the old file is deleted before saving.

When allowed In all functions.

■ Sets and queries the degree of thinning for the auto save function.

8841, 8842, 8720

Syntax

command	:CONFigure:THINout *A$*
query	:CONFigure:THINout?
response	*A$*

A$ = degree of thinning
OFF, 1_2 to 1_1000

Explanation Sets the degree of thinning that is applied when data is stored in text format by the auto save function.
This setting can also be made when auto save is OFF or data is stored in binary format, but the setting will not be reflected on the screen.
Returns the current setting of the degree of thinning as character data.

Example **:CONFigure:THINout 1_2**
Sets the degree of thinning to 1/2.

When allowed In all functions.

■ Sets and queries the waveform overlay function. Except 8720

Syntax

command	:CONFigure:OVERlay *A$*
query	:CONFigure:OVERlay?
response	*A$*

A$ = OFF, ON

Explanation Enables and disables screen waveform overlay.
Returns the current enablement state of the waveform overlay as character data.

Example **:CONFigure:OVERlay ON**
Sets the screen waveform overlay to ON.

When allowed In MEM.

■ Sets and queries the count for averaging.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:AVERage *A*
query :CONFigure:AVERage?
response *A* <NR1>
A = 0: OFF
2, 4, 8, 16, 32, 64, 128, 256

Explanation Sets the count for averaging.
Returns the current setting of the count for averaging as NR1 numerical value.

Example **:CONFigure:AVERage 32**
Sets the count for averaging to 32.

When allowed In MEM.

■ Sets and queries the waveform decision mode.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:WVCOmp *A$*
query :CONFigure:WVCOmp?
response *A$*
A$ = OFF, OUT, ALLOut

Explanation Sets the waveform decision mode.
Returns the current waveform decision mode as character data.

Example **:CONFigure:WVCOmp OUT**
Sets the waveform decision mode to OUT.

When allowed In MEM and FFT.

■ Sets and queries the waveform decision stop mode.

8835 (-01) A, 8826 8841, 8842

Syntax

command :CONFigure:CMPStop *A$*
query :CONFigure:CMPStop?
response *A$*
A$ = GO, NG, G-N

Explanation Sets the stop mode during waveform decision.
Returns the current stop mode as character data.

Example **:CONFigure:CMPStop GO**
Sets the stop mode during waveform decision to GO.

When allowed In MEM and FFT.

■ Sets and queries the additional recording function. Common

Syntax

command	:CONFigure:VIRTual *A$*
query	:CONFigure:VIRTual?
response	*A$*

A$ = OFF, ON

Explanation Sets the additional recording function.
Returns the current setting of the additional recording function as character data.

Example **:CONFigure:VIRTual ON**
Sets the additional recording function to ON.

When allowed In REC, RMS and R&M.

■ Sets and queries printer output. Except 8720

Syntax

command	:CONFigure:PRINt *A$*
query	:CONFigure:PRINt?
response	*A$*

A$ = OFF, ON

Explanation Sets the printer output.
Returns the currently set state of the printer output as character data.

Example **:CONFigure:PRINt ON**
Sets the printer output to ON.

When allowed In REC, RMS and R&M.

■ Sets and queries data number per 1 DIV for external sampling.
8835-01,8826,8841,8842

Syntax

command	:CONFigure:EXTSample *A*
query	:CONFigure:EXTSample?
response	*A* <NR1>

A = 10 to 1000

Explanation Sets the data number per 1 DIV for external sampling.
Returns the data number of 1 DIV for the current external sample.

Example **:CONFigure:EXTSample 100**
Sets the data number of the external sample for each 1 DIV to 100.

When allowed In MEM.

■ Sets and queries memory segmentation.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:MEMDiv *A$*
query :CONFigure:MEMDiv?
response *A$*
(MEM)
A$ = OFF
SEQ : sequential save
MULTI : multi-block
(R&M)
A$ = OFF, SEQ

Explanation Sets the method of memory segmentation recording.
Returns the current setting for method of memory segmentation recording as character data.

Example `:CONFigure:MEMDiv SEQ`
Sets the method of memory segmentation recording to sequential save.

When allowed In MEM and R&M.

■ Sets and queries the memory block used.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:USEBlock *A*
query :CONFigure:USEBlock?
response *A* <NR1>
A = 1 to number of segmentations

Explanation During memory segmentation, sets the memory block used ("using block").
Returns the currently used memory block as an NR1 numerical value.

Example `:CONFigure:USEBlock 15`

Sets the block used to 15.

When allowed In MEM and R&M, when the memory segmentation function is in use.

■ Sets and queries the start block. 8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:STTBlock *A*
query	:CONFigure:STTBlock?
response	*A* <NR1>

Explanation Sets the start block.
Returns the current start block as an NR1 numerical value.

Example
```
:CONFigure:STTBlock 5
```
Sets the start block to 5.

When allowed In MEM and R&M, when the sequential save function is in use.

■ Sets and queries the end block. 8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:ENDBlock *A*
query	:CONFigure:ENDBlock?
response	*A* <NR1>

Explanation Sets the end block.
Returns the current end block as an NR1 numerical value.

Example
```
:CONFigure:ENDBlock 120
```
Sets the end block to 120.

When allowed In MEM and R&M, when the sequential save function is in use.

■ Sets and queries the follow-up waveform display.
8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:SEQDisp *A$*
query	:CONFigure:SEQDisp?
response	*A$*

A$ = OFF, ON

Explanation Sets whether or not the data are displayed on the screen after they are saved to the blocks.
Returns the current setting of the follow-up waveform display as character data.

Example
```
:CONFigure:SEQDisp ON
```
Displays the data on the screen after they are saved to the blocks.

When allowed In MEM, when the sequential save function is in use.

■ Sets and queries the number of memory blocks.

8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:MAXBlock *A*
query	:CONFigure:MAXBlock?
response	*A* <NR1>
	A = 3, 7, 15, 31, 63, 127, 255

Explanation Sets the number of memory blocks for the multi-block function.
Returns the current number of memory blocks as an NR1 numerical value.

Example
```
:CONFigure:MAXBlock 15
```
Sets the number of memory blocks to 15.

Note Set the recording length during sequential save using the :CONFigure:SHOT command (see "Sets and queries the recording length").

When allowed In MEM, when the multi-block function is in use.

■ Sets and queries the reference block. 8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:REFBlock *A*
query	:CONFigure:REFBlock?
response	*A* <NR1>
	A = 0 : OFF, 1 to number of memory segmentations (8835(-01))
	A = 0 : OFF, 1 : ON (except 8835(-01))

Explanation Sets the reference block during multi-block.
Sets the reference block. (except 8835(-01))
Returns the current reference block as an NR1 numerical value.

Example
```
:CONFigure:REFBlock 15
```
Sets the reference block to 15.

When allowed When the multi-block function is in use. (except 8835(-01))
When the sequential save function is in use. (8835(-01))

■ Sets and queries the reference block. 8826,8841,8842

Syntax

command	:CONFigure:REFBlock *A,B$*
query	:CONFigure:REFBlock? *A*
response	*A* <NR1>, *B$*
	A = 1 to number of segmentations
	B$ = ON, OFF

Explanation Sets reference ON, OFF for each block when memory is allocated.
Returns the reference ON, OFF for each current block.

Example
```
:CONFigure:REFBlock 1,ON
```
Sets the reference for the first block to ON.

When allowed In MEM.

■ Sets and queries the count for averaging in the FFT function.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTAVERage *A*
query :CONFigure:FFTAVERage?
response *A* <NR1>
A = 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096

Explanation

Sets the count for averaging in the FFT function.
Returns the current setting of the count for averaging in the FFT function as NR1 numerical values.

Example

```
:CONFigure:FFTAVERage 2048
```

Sets the count for averaging to 2048.

When allowed

In FFT.

■ Sets and queries the type of averaging in the FFT function.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTAVKind *A$*
query :CONFigure:FFTAVKind?
response *A$*
A$ = OFF
T_LIN: simple time axis averaging
T_EXP: exponential time axis averaging
F_LIN: simple frequency axis averaging
F_EXP: exponential frequency axis averaging
F_PEAK: frequency axis peak hold

Explanation

Sets the averaging method designated by *A$*.
Returns the currently set averaging method as character data.

Example

```
:CONFigure:FFTAVKind T_EXP
```

Sets time axis exponential averaging.

When allowed

In FFT.

■ Sets and queries the FFT channel mode.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTMode *A, ch1$* (*,ch2$*)
query :CONFigure:FFTMode?
response *A* <NR1>*, ch1$, ch2$*
A = 1: one-channel FFT mode
2: two-channel FFT mode
ch1$ = CH1 to CH32: analysis channel W1
ch2$ = CH1 to CH32: analysis channel W2
(8835: CH1 to CH4, 8841, 8842: CH1 to CH16)

Explanation

Sets the FFT channel mode. I.e., designates the object channel or channels for FFT channel mode and the number thereof. In the one-channel FFT mode (only) the specification of channel 2 can be omitted, and if it is provided it is ignored. Transfer function, coherence function, cross power spectrum, cross correlation function and impulse response are only effective in the two-channel FFT mode.
Returns the current FFT channel mode as a numerical value in NR1 format, and the analysis channel as character data.

Example

```
:CONFigure:FFTMode 2, CH3, CH5
```

The channel mode is set to the two-channel FFT mode, and the object channels for FFT mode are set to be channel 3 and channel 5.

When allowed

In FFT.

■ Sets and queries the FFT window function.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTWind *A$* (*,B*)
query :CONFigure:FFTWind?
response *A$, B* <NR1>
A$ = RECTan: rectangular window
HANNing: Hanning window
EXPOnential: exponential function window
B = 0 to 99 (units %): coefficient for the exponential function

Explanation

Sets the window function as indicated by *A$*. If the exponential window function is designated by *A$,* its exponential function coefficient can be set by using *B*.
Returns the current window function as character data, and the current exponential function coefficient as a numerical value in NR1 format.

Example

```
:CONFigure:FFTWind HANN
```

The window function is set to Hanning window.

When allowed

In FFT.

■ Sets and queries the FFT analysis mode.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTFunction *A$, B$*
query :CONFigure:FFTFunction? *A$*
response *A$, B$*
A$ = G1, G2: graph number
B$ = STR: stored waveform
LIN: linear spectrum
RMS: RMS spectrum
PSP: power spectrum
ACR: auto-correlation function
HIS: histogram
TRF: transfer function (*)
CSP: cross power spectrum (*)
CCR: cross correlation function (*)
IMP: impulse response (*)
COH: coherence function (*)
OCT: octave analysis
(*) can only be used when the two-channel FFT mode is set.

Explanation

Sets the FFT analysis mode.
The FFT analysis mode can be set to transfer function, coherence function, cross power spectrum, cross correlation function, or impulse response only in the two-channel FFT mode (FFTMODE 2, *ch1$*, *ch2$*). In this case, the corresponding function is calculated from channel 1 and channel 2. The result of the calculation is displayed on the graph designated by *A$*. G2 can be designated even if the display format is SINGLE, but this does not affect the display.
Returns the current FFT analysis mode as character data.

Example

```
:CONFigure:FORMat DUAL
:CONFigure:FFTMode 2, CH1, CH3
:CONFigure:FFTFUNCTION G1, IMP
:CONFigure:FFTFUNCTION G2, TRF
```

The impulse response calculated from channel 1 and channel 3 is displayed on G1, and the transfer function calculated from these channels is displayed on G2.

When allowed

In FFT.

■ Sets and queries the FFT data source.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTRef *A$*
query :CONFigure:FFTRef?
response *A$*
A$ = NEW: new data
MEM: data stored in the memory

Explanation Designates the source for FFT data as specified by *A$*.
Returns the current FFT data source as character data.

Example

```
:CONFigure:FFTRef NEW
```

New data is used as FFT data.

When allowed In FFT.

■ Sets and queries the FFT display scaling method.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTSCale *A$, B$*
query :CONFigure:FFTSCale?
response *A$, B$*
A$ = G1, G2
B$ = AUTO, MANUal

Explanation Sets the display scaling method for the graph number designated by *A$*.
Returns the current display scaling method for the graph number designated by *A$* as character data.

Example

```
:CONFigure:FFTSCale G1,AUTO
```

The scaling method for graph number 1 is set to automatic.

When allowed In FFT.

■ Sets and queries the FFT display scale vertical axis upper limit.

8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:FFTUp *A$, B*
query	:CONFigure:FFTUp? *A$*
response	*A$, B* <NR3>
	A$ = G1, G2
	B = -9.9999E+29 to +9.9999E+29

Explanation Sets the FFT display scale vertical axis upper limit for the graph number designated by *A$* to the value designated by *B*.
Returns the current FFT display scale vertical axis upper limit for the graph number designated by *A$* as a numerical value in NR3 format.

Example
```
:CONFigure:FFTUp G2,100
```
The FFT display scale vertical axis upper limit for graph 2 is set to 100.

When allowed In FFT.

■ Sets and queries the FFT display scale vertical axis lower limit.

8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:FFTLow *A$, B*
query	:CONFigure:FFTLow? *A$*
response	*A$, B* <NR3>
	A$ = G1, G2
	B = -9.9999E+29 to +9.9999E+29

Explanation Sets the FFT display scale vertical axis lower limit for the graph number designated by *A$* to the value designated by *B*.
Returns the current FFT display scale vertical axis lower limit for the graph number designated by *A$* as a numerical value in NR3 format.

Example
```
:CONFigure:FFTLow G2,100
```
The FFT display scale vertical axis lower limit for display graph 2 is set to 100.

When allowed In FFT.

■ Sets and queries the FFT x-axis. 8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTXaxis *A$, B$*
query :CONFigure:FFTXaxis? *A$*
response *A$, B$*
A$ = G1, G2
B$ = 1_1oct, 1_3oct: during octave analysis
LINhz, LOGhz: otherwise

Explanation

Sets the x-axis of the graph number designated by *A$*. When the analysis mode is octave analysis, 1_1oct or 1_3oct can be set; otherwise, LINhz or LOGhz can be set. Some settings are not available for some analysis modes. If a setting is not available, an execution error is generated (see the table on the next page.)
Returns the current x-axis setting as character data.

Example

```
:CONFigure:FFTXaxis G1, LINHZ
```

The setting for the x-axis of graph 1 is set to LINHZ.

When allowed

In FFT.

■ Sets and queries the FFT y-axis. 8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FFTYaxis *A$, B$*
query :CONFigure:FFTYaxis? *A$*
response *A$, B$*
A$ = G1, G2
B$ = LINMAg: linear magnitude
LINREal: linear real axis magnitude
LINIMag: linear imaginary axis magnitude
LOGMAg: logarithmic magnitude
PHASE

Explanation

Sets the y-axis of the graph number designated by *A$*. Some settings are not available for some analysis modes. If a setting is not available, an execution error is generated (see the table on the next page.)
Returns the current y-axis setting as character data.

Example

```
:CONFigure:FFTYaxis G1,LINMAG
```

The setting for the y-axis of graph 1 is set to LINMAG.

When allowed

In FFT.

Display settings available on the x-axis

Analysis mode	X-axis				
	Linear-Hz	Log-Hz	1/1 octave	1/3 octave	Fixed scale
STR					TIME
LIN	●	●			
RMS	●	●			
PSP	●	●			
ACR					TIME
HIS					VOLT
TRF	●	●			
CSP	●	●			
CCR					TIME
IMP					TIME
COH	●	●			
OCT			●	●	

Display settings available on the y-axis

Analysis mode	Y-axis					
	Linear-real	Linear-imaginary	Linear-magnitude	Log-magnitude	Phase	Fixed scale
STR						LINEAR
LIN	●	●	●	●	●	
RMS	●	●	●	●	●	
PSP			●	●		
ACR						LINEAR
HIS						LINEAR
TRF	●	●	●	●	●	
CSP	●	●	●	●	●	
CCR						LINEAR
IMP						LINEAR
COH						LINEAR
OCT			●	●		

■ Sets and queries the FFT frequency range.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:FREQ *A*
query :CONFigure:FREQ?
response *A* <NR3>
A = 400000, 200000, 80000, 40000, 20000, 8000, 4000, 2000, 800, 400, 200, 80, 40, 20, 8, 4, 1.33, 0.667, 0.333, 0.133, 0 (external sampling)

Explanation

Sets the frequency range. If an attempt is made to set an unacceptable value, i.e. a value which is not one of the above, then the frequency range is set to the next higher one of the above values.
Returns the currently set frequency range as a numerical value in NR3 format.

Example

```
:CONFigure:FREQ 80
```

The frequency range is set to 80 Hz.

When allowed

In FFT.

■ Sets and queries octave filter type. 8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:OCTFilter *A$*
query :CONFigure:OCTFilter?
response *A$*
A$ = NORMal, SHARp

Explanation

Sets the type of octave filter.
Returns the currently set type of octave filter as character data.

Example

```
:CONFigure:OCTFilter NORMal
```

Sets the octave filter type to NORMAL.

When allowed

In FFT.

■ Sets and queries peak value display. 8835 (-01) A, 8826, 8841, 8842

Syntax

command :CONFigure:PEAK *A$*
query :CONFigure:PEAK?
response *A$*
A$ = OFF, PEAK, MAX

Explanation

Sets the peak value display.
Returns the currently set peak value display as character data.

Example

```
:CONFigure:PEAK PEAK
```

Sets the peak value display to PEAK.

When allowed

In FFT.

■ Sets and queries the number of FFT points.

8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CONFigure:FFTSAmple *A*
query	:CONFigure:FFTSAmple?
response	*A* <NR1>

A = 1000, 2000, 5000, 10000

Explanation Sets the number of FFT points.
Returns the currently set number of FFT points as a numerical value in NR1 format.

Example **:CONFigure:FFTSAmple 2000**
Sets the number of FFT points to 2000.

When allowed In FFT.

■ Sets and queries the real time save function. 8826,8841,8842

Syntax

command	:CONFigure:RTSAve *A$*
query	:CONFigure:RTSAve?
response	*A$*

A$ = ON, OFF

Explanation Sets the real time save function.
Returns the current enablement state of the real time save function.

Example **:CONFigure:RTSAve ON**
Sets the real time function to ON.

When allowed In R&M. (Real time save function version only)

■ Sets and queries comparison of separate files. 8720

Syntax

command	:CONFigure:CMPOld *A$*
query	:CONFigure:CMPOld?
response	*A$*

A$ = ON,OFF

Explanation Sets comparison of separate files.
Returns ON, OFF settings of currently compared files. The order in which the separate files are read by transmission is as follows:
1. :DISPlay:VIEWPart 1 partitions monitor screen
2. :DISPlay:VIEWSel 1 selects left side of monitor screen
3. :FDISK:LOAD reads the file

Example **:CONFigure:CMPOld ON**
Sets separate file comparison to ON.

When allowed In all functions.

■ Sets and queries one-touch save function. 8720

Syntax

command	:CONFigure:OTSAve A$
query	:CONFigure:OTSAve?
response	*A$*

A$ = FD,PC,MO,SCSI,LAN

Explanation Sets media for the one-touch save function.
Returns the current media setting for the one-touch save function.

Example **:CONFigure:OTSAve FD**
Sets a floppy disk for one-touch save.

When allowed In all functions.

3. TRIGger command (Sets and queries trigger.)

:TRIGger

■ Sets and queries trigger mode. Common

Syntax

command	:TRIGger:MODE *A$*
query	:TRIGger:MODE?
response	*A$*

A$ = SINGle, REPEat, AUTO : MEM, FFT
SINGle, REPEat : REC, RMS
SINGle, REPEat, TIMEr : R&M

Explanation Sets the trigger mode.
Returns the current trigger mode as character data.

Example **:TRIGger:MODE REPEat**
Sets the trigger mode to repeat.

When allowed In all functions.

■ Sets and queries pre-trigger. Common

Syntax

command :TRIGger:PRETrig *A*
query :TRIGger:PRETrig?
response *A* <NR1>
A = 0, 2, 5, 10, 20,..., 90, 95, 100, -95 (unit %) : MEM, R&M, FFT
A = 0, 5, 10 (DIV) : RMS, 8720

Explanation

Sets pre-trigger value to a numerical value.
If an attempt is made to set a value which cannot be set on the 8835, setting is performed to the next higher permitted value.
The currently set pre-trigger value is returned as an NR1 numerical value.

Example

```
:TRIGger:PRETrig 10
```

Pre-trigger value is set to 10%.

When allowed

In MEM, RMS, R&M and FFT.

■ Sets and queries trigger timing. Except 8720

Syntax

command :TRIGger:TIMIng *A$*
query :TRIGger:TIMIng?
response *A$*
A$ = START, STOP, S_S

Explanation

Sets the trigger timing.
The current trigger timing setting is returned as character data.

Example

```
:TRIGger:TIMIng START
```

Sets the trigger timing to START.

When allowed

In REC.

■ Sets and queries trigger logical operator (AND/OR). Common

Syntax

command :TRIGger:SOURce *A$*
query :TRIGger:SOURce?
response *A$*
A$ = OR, AND

Explanation

Sets the logical operator determining whether the internal, logic, external and timer triggers are ANDed or ORed.
Returns the current setting of the trigger logical operator (AND/OR) as character data.

Example

```
:TRIGger:SOURce OR
```

Sets the trigger source to OR.

When allowed

In all functions.

■ Sets and queries manual trigger. Except 8720

Syntax

command	:TRIGger:MANU *A$*
query	:TRIGger:MANU?
response	*A$*

A$ = OFF, ON

Explanation Enables and disables manual trigger.

Example **:TRIGger:MANU ON**
Sets the manual trigger to ON.

When allowed In all functions.

■ Sets and queries the kind of trigger. Common

Syntax

command	:TRIGger:KIND *ch$, A$*
query	:TRIGger:KIND? *ch$*
response	*ch$, A$*

A$ = OFF
LEVEl : level trigger (MEM, REC, R&M, FFT)
IN : window-in trigger (MEM, REC, R&M, FFT)
OUT : window-out trigger (MEM, REC, R&M, FFT)
DROP : voltage drop trigger (MEM, R&M, FFT)
PERIod : period trigger (MEM, REC, R&M, FFT)
RMS : RMS level trigger (RMS)

Explanation Sets the type of trigger for the channel designated by *ch$*.
Returns as character data the type of the current trigger for the channel designated by *ch$*.

Example **:TRIGger:KIND CH1, LEVEl**
Sets channel 1 to level trigger.

When allowed In all functions.

■ Sets and queries trigger level of the level trigger. Common

Syntax

command	:TRIGger:LEVEl *ch$, A*
query	:TRIGger:LEVEl? *ch$*
response	*ch$, A* <NR3>

A = voltage value (V)

Explanation Sets the trigger level of the level trigger of the channel designated by *ch$*.
Returns as an NR3 numerical value the current trigger level of the channel designated by *ch$*.

Example **:TRIGger:LEVEl CH1, 50E-3**
Sets the trigger level of channel 1 to 50 mV.

When allowed In MEM, REC, R&M and FFT.

■ Sets and queries trigger direction (slope). Common

Syntax

command :TRIGger:SLOPe *ch$, A$*
query :TRIGger:SLOPe? *ch$*
response *ch$, A$*
A$ = UP: rising
DOWN: falling

Explanation Sets the trigger direction of the level trigger or the period trigger of the channel designated by *ch$*.
Returns as a character value the current trigger direction of the channel designated by *ch$*.

Example

```
:TRIGger:SLOPe CH1, UP
```

Sets the trigger direction of channel 1 to rising.

When allowed In MEM, REC, R&M and FFT.

■ Sets and queries the filter width. Common

Syntax

command :TRIGger:FILTer *ch$, A*
query :TRIGger:FLITer? *ch$*
response *ch$, A* <NR2>
A = 0 (OFF), 0.1, 0.2, 0.5, 1.0, 1.5, 2.0, 2.5, 5.0, 10.0 (DIV)
(MEM, R&M, FFT)
0 (OFF), 1 (ON) (REC)

Explanation Sets the filter width for a trigger of the channel designated by *ch$* to 1 to 10 divisions.
For the recorder function, only ON/OFF can be selected.
Returns the current filter width as an NR2 numerical value.

Example

```
:TRIGger:FILTer CH1, 0.1
```

Sets the filter width of channel 1 to 0.1 (DIV).

When allowed In MEM, REC, R&M and FFT.

■ Sets and queries upper limit level for a window-in/-out trigger.

Common

Syntax

command	:TRIGger:UPPEr *ch$, A*
query	:TRIGger:UPPEr? *ch$*
response	*ch$, A* <NR3>
	A = voltage value (V)

Explanation Sets the upper limit level of the window trigger of the channel designated by *ch$* as a voltage value.
Returns the current upper limit level of the window trigger as an NR3 numerical value.

Example
```
:TRIGger:UPPEr CH1, +1.0E-3
```
Sets the upper limit level of the window trigger of channel 1 to +1.0 mV.

When allowed In MEM, REC, R&M and FFT.

■ Sets and queries lower limit level for a window-in/-out trigger.

Common

Syntax

command	:TRIGger:LOWEr *ch$, A*
query	:TRIGger:LOWEr? *ch$*
response	c*h$, A* <NR3>
	A = voltage value (V)

Explanation Sets the lower limit level of the window trigger of the channel designated by *ch$* as a voltage value.
Returns the current lower limit level of the window trigger as an NR3 numerical value.

Example
```
:TRIGger:LOWEr CH1, -1.0E-3
```
Sets the lower limit level of the window trigger of channel 1 to -1.0 mV.

When allowed In MEM, REC, R&M and FFT.

■ Sets and queries measurement frequency for a voltage drop trigger. Except 8720

Syntax

command	:TRIGger:VFREq *ch$, A*
query	:TRIGger:VFREq? *ch$*
response	c*h$, A* <NR1>
	A = 50, 60 (Hz)

Explanation Sets the measurement frequency of the voltage drop trigger of the channel designated by *ch$* as a frequency.
Returns the current measurement frequency of the voltage drop trigger as an NR1 numerical value.

Example
```
:TRIGger:VFREq CH1, 50
```
Sets the measurement frequency of the voltage drop trigger of channel 1 to 50 Hz.

When allowed In MEM, R&M and FFT.

■ Sets and queries drop level for a voltage drop trigger. Except 8720

Syntax

command	:TRIGger:VLEVel *ch$, A*
query	:TRIGger:VLEVel? *ch$*
response	c*h$, A* <NR3>
	A = voltage value (V)

Explanation Sets the drop level of the voltage drop trigger of the channel designated by *ch$* as a voltage value.
Returns the current drop level of the voltage drop trigger as an NR3 numerical value.

Example
```
:TRIGger:VLEVel CH1, 1.0E2
```
Sets the drop level of the voltage drop trigger of channel 1 to 100 V.

When allowed In MEM, R&M and FFT.

■ Sets and queries upper period limit for a period trigger. Except 8720

Syntax

command	:TRIGger:PUPPer *ch$, A*
query	:TRIGger:PUPPer? *ch$*
response	c*h$, A* <NR3>

A = upper period limit (s)

Explanation Sets the upper period limit of the period trigger of the channel designated by *ch$* as a period.
Returns the current upper period limit of the period trigger as an NR3 numerical value.

Example
```
:TRIGger:PUPPer CH1, 2.2E-5
```
Sets the upper period limit of the period trigger of channel 1 to 22 µs.

When allowed In MEM, REC, R&M and FFT.

■ Sets and queries lower period limit for a period trigger. Except 8720

Syntax

command	:TRIGger:PLOWer *ch$, A*
query	:TRIGger:PLOWer? *ch$*
response	c*h$, A* <NR3>

A = lower period limit (s)

Explanation Sets the lower period limit of the period trigger of the channel designated by *ch$* as a period.
Returns the current lower period limit of the period trigger as an NR3 numerical value.

Example
```
:TRIGger:PLOWer CH1, 2.0E-5
```
Sets the lower period limit of the period trigger of channel 1 to 20 µs.

When allowed In MEM, REC, R&M and FFT.

■ Sets and queries trigger level for a period trigger. Except 8720

Syntax

command	:TRIGger:PLEVel *ch$, A*
query	:TRIGger:PLEVel? *ch$*
response	*ch$, A* <NR3>

A = voltage value (V)

Explanation Sets the trigger level of the period trigger of the channel designated by *ch$* as a voltage value.
Returns the current trigger level of the period trigger as an NR3 numerical value.

Example
```
:TRIGger:PLEVel CH1, 1.0E0
```
Sets the trigger level of the period trigger of channel 1 to 1 V.

When allowed In MEM, REC, R&M and FFT.

■ Sets and queries trigger level for an RMS level trigger. Except 8720

Syntax

command	:TRIGger:RLEVel *ch$, A*
query	:TRIGger:RLEVel? *ch$*
response	*ch$, A* <NR3>

A = voltage value (V)

Explanation Sets the trigger level of the RMS level trigger of the channel designated by *ch$* as a voltage value.
Returns the current trigger level of the RMS level trigger as an NR3 numerical value.

Example
```
:TRIGger:RLEVel CH1, 1.0E0
```
Sets the trigger level of the RMS level trigger of channel 1 to 1 V.

When allowed In RMS.

■ Sets and queries trigger direction (slope) for an RMS level trigger. Except 8720

Syntax

command	:TRIGger:RSLOpe *ch$, A$*
query	:TRIGger:RSLOpe? *ch$*
response	*ch$, A$*

A$ = UP: rising
DOWN: falling

Explanation Sets the trigger direction of the RMS level trigger of the channel designated by *ch$*.
Returns the current trigger direction of the RMS level trigger as character data.

Example
```
:TRIGger:RSLOpe CH1, UP
```
Sets the trigger direction of the RMS level trigger of channel 1 to rising.

When allowed In RMS.

■ Sets and queries the logical operator (AND/OR) for the trigger pattern of a logic trigger. Except 8720

Syntax

command :TRIGger:LOGAnd *ch$, A$*
query :TRIGger:LOGAnd? *ch$*
response *ch$, A$*
ch$ = CHA to CHD (8835 (-01), 8841, 8842),
CHA to CHH (8826)
A$ = OFF, OR, AND

Explanation

Sets the AND/OR logical operator for the trigger pattern of a logic trigger of the channel designated by *ch$*.
Returns the present AND/OR setting for the trigger pattern of a logic trigger as a character string.

Example

:TRIGger:LOGAnd CHA, OR
Sets the AND/OR logical operator for the trigger pattern of a logic trigger of channel A to OR.

When allowed

In MEM, REC, RMS and R&M.

■ Sets and queries the filter width for a logic trigger. Except 8720

Syntax

command :TRIGger:LFILter *ch$, A*
query :TRIGger:LFILter? *ch$*
response *ch$, A* <NR2>
ch$ = CHA to CHD (8835 (-01), 8841, 8842),
CHA to CHH (8826)
A = 0 (OFF), 0.1, 0.2, 0.5, 1.0, 1.5, 2.0, 2.5, 5.0, 10.0 (DIV) :
MEM, R&M
0 (OFF), 1 (ON) : REC, RMS

Explanation

Sets the filter width for a logic trigger of the channel designated by *ch$*.
Returns the current setting for the filter width for a logic trigger as an NR2 numerical value.

Example

:TRIGger:FILter CHA, 0.1
Sets the filter width for a logic trigger of channel A to 0.1 (DIV).

When allowed

In MEM, REC, RMS and R&M.

■ Sets and queries the trigger pattern for a logic trigger. Except 8720

Syntax

command	:TRIGger:LOGPat *ch$*, '*A$*'
query	:TRIGger:LOGPat? *ch$*
response	*ch$*, '*A$*'

ch$ = CHA to CHD (8835 (-01), 8841, 8842),
CHA to CHH (8826)
A$ = XXXX : trigger pattern (X, 0, 1)

Explanation Sets the trigger pattern for the logic trigger of the channel designated by *ch$*.
Returns the current setting for the trigger pattern for the logic trigger as that specified by the given character data.

Example
```
:TRIGger:LOGPat CHA, '011X'
```
Sets the trigger pattern for the logic trigger of channel A to '011X'.

When allowed In MEM, REC, RMS and R&M.

■ Sets and queries the timer trigger. Except 8720

Syntax

command	:TRIGger:TIMEr *A$*
query	:TRIGger:TIMEr?
response	*A$*

A$ = OFF, ON

Explanation Sets the timer trigger.
Returns the current timer trigger setting as character data.

Example
```
:TRIGger:TIMEr ON
```
Sets the timer trigger.

When allowed In MEM, REC, RMS and FFT.

■ Sets and queries the start instant for the timer trigger. Except 8720

Syntax

command :TRIGger:TMSTArt *A, B, C, D*
query :TRIGger:TMSTArt?
response *A, B, C, D*
A = *month*: 1 to 12
B = *day*: 1 to 31
C = *hour*: 0 to 23
D = *min*: 0 to 59
month, day, hour, min all <NR1>

Explanation

Sets the start instant for the timer trigger.
Returns the current setting for the timer trigger start instant as NR1 numerical values.

Example

:TRIGger:TMSTArt 7, 22, 11, 22
Sets the start instant for the timer trigger to 11:22 on July 22nd.

When allowed

In all functions.

■ Sets and queries the stop instant for the timer trigger. Except 8720

Syntax

command :TRIGger:TMSTOp *A, B, C, D*
query :TRIGger:TMSTOp?
response *A, B, C, D*
A = *month*: 1 to 12
B = *day*: 1 to 31
C = *hour*: 0 to 23
D = *min*: 0 to 59
month, day, hour, min all <NR1>

Explanation

Sets the stop instant for the timer trigger.
Returns the current setting for the timer trigger stop instant as NR1 numerical values.

Example

:TRIGger:TMSTOp 7, 22, 11, 45
Sets the stop instant for the timer trigger to 11:45 on July 22nd.

When allowed

In all functions.

■ Sets and queries the time interval for the timer trigger. Except 8720

Syntax

command	:TRIGger:TMINTvl *A, B, C, D*
query	:TRIGger:TMINTvl?
response	*A, B, C, D*

A = day: 0 to 99
B = hour: 0 to 23
C = min: 0 to 59
D = sec: 0 to 59
day, hour, min, sec all <NR1>

Explanation Sets the time interval for the timer trigger.
Returns the current setting for the timer trigger time interval as NR1 numerical values.

Example **:TRIGger:TMINTvl 1, 20, 30**
Sets the time interval for the timer trigger to one hour, twenty minutes, and thirty seconds.

When allowed In all functions.

■ Sets and queries the time point for trigger detection. Common

Syntax

command	:TRIGger:DETECTTime *A, B, C*
query	:TRIGger:DETECTTime?
response	*A, B, C*

A = hour: 0 to 23
B = min: 0 to 59
C = sec: 0 to 59
hour, min, sec all <NR1>

Explanation Sets the time point for trigger detection.
Returns the setting for the time point for trigger detection as a numerical value in NR1 format.

Example **:TRIGger:DETECTTime?**
The currently set time point for trigger detection is queried.

When allowed In all functions.

■ Sets and queries the date for trigger detection. Common

Syntax

command :TRIGger:DETECTDate *A, B, C*
query :TRIGger:DETECTDate?
response *A, B, C*
A = *year*: 0 to 99
B = *month*: 1 to 12
C = *day*: 1 to 31
year, *month*, *day* all <NR1>

Explanation

Sets the date for trigger detection.
Returns the setting for the date for trigger detection as a numerical value in NR1 format.

Example

:TRIGger:DETECTDate?
The currently set date for trigger detection is queried.

When allowed

In all functions.

■ Sets and queries the time for start operating termination. Common

Syntax

command :TRIGger:STOPTime *A, B, C*
query :TRIGger:STOPTime?
response *A, B, C*
A = *hour*: 0 to 23
B = *min*: 0 to 59
C = *sec*: 0 to 59
hour, *min*, *sec* all <NR1>

Explanation

Sets the time for start operating termination.
Returns the currently set time for start operating termination as a numerical value in NR1 format.

Example

:TRIGger:STOPTime?
The currently set time for start operating termination is queried.

When allowed

In REC and R&M.

■ Sets and queries the date for start operating termination. Common

Syntax

command :TRIGger:STOPDate *A, B, C*
query :TRIGger:STOPDate?
response *A, B, C*
A = *year*: 0 to 99
B = *month*: 1 to 12
C = *day*: 1 to 31
year, *month*, *day* all <NR1>

Explanation Sets the date for start operating termination.
Returns the currently set date for start operating termination as a numerical value in NR1 format.

Example **:TRIGger:STOPDate?**
The currently set date for start operating termination is queried.

When allowed In REC and R&M.

■ Sets and queries external trigger. Common

Syntax

command :TRIGger:EXTErnal *A$*
query :TRIGger:EXTErnal?
response *A$*
A$ = OFF, ON

Explanation Enables and disables external trigger.
Returns the current external trigger enablement state as character data.

Example **:TRIGger:EXTErnal OFF**
Sets the external trigger to OFF.

When allowed In all functions.

4. UNIT command (Sets and queries input channel.)

:UNIT

■ Sets and queries the measurement range of an input channel. Common

Syntax

command :UNIT:RANGe *ch$, A*
query :UNIT:RANGe? *ch$*
response *ch$, A* <NR3>
A = voltage (V, μ ε, ℃)

Explanation Sets the measurement range for the channel designated by *ch$* to a numerical value.
Returns the current measurement range for the channel designated by *ch$* as an NR3 numerical value.

Example

```
:UNIT:RANGe CH1, +10.E-3
```

Sets the voltage axis range for channel 1 to 10 mV.

When allowed In all functions.
When using the frequency range in the F/V unit, use the :UNIT:FRANge command.

■ Sets and queries input coupling for an input channel. Common

Syntax

command :UNIT:COUPling *ch$, A$*
query :UNIT:COUPling? *ch$*
response *ch$, A$*
A$ = GND, DC, AC (except when measuring temperature with the 8937 VOLTAGE/TEMPERATURE UNIT)

Explanation Sets the input coupling for the channel designated by *ch$*.
Returns the current input coupling for the channel designated by *ch$* as character data.

Example

```
:UNIT:COUPling CH1, DC
```

Sets the input coupling for channel 1 to DC.

When allowed In all functions.

■ Sets and queries input channel origin position. Common

Syntax

command :UNIT:POSItion *ch$, A*
query :UNIT:POSItion? *ch$*
response *ch$, A* <NR1>
A = origin position (%)

Explanation Sets the origin position for the channel designated by *ch$* in the range to a numerical value.
Returns the current origin position for the channel designated by *ch$* as an NR1 numerical value (unit percent).

Example

```
:UNIT:POSItion CH1, 50
```

Sets the origin position for channel 1 to 50%

When allowed In all functions.

■ Sets and queries the filter for an input channel. Common

Syntax

command :UNIT:FILTer *ch$, A*
query :UNIT:FILTer? *ch$*
response *ch$, A$*
A = 0 (OFF), 5, 500, 5000, 100000 (Hz)
0 (OFF), 10, 30, 300, 3000 (Hz) : 8939
(0 (OFF), 5, 500 (Hz) when temperature is measured with the 8937 or the digital filter is ON)
A$ = 0 (OFF), 5, 500, 5 k, 100 k (Hz)
0 (OFF), 10, 30, 300, 3 k (Hz) : 8939

Explanation Sets the filter for the channel designated by *ch$*.
Returns the current filter setting for the channel designated by *ch$* as character data

Example

```
:UNIT:FILTer CH1, 500
```

Sets the filter for channel 1 to 500 Hz.

When allowed In all functions.

■ Sets and queries the type of the voltage/temperature unit sensor.

Common

Syntax

command	:UNIT:SENSor *ch$, A$*
query	:UNIT:SENSor? *ch$*
response	*ch$, A$*
	A$ = K, E, J, T, N, R, S, B, OFF (voltage)

Explanation Sets the type of the voltage/temperature unit sensor on the channel designated by *ch$*.
Returns the type of the voltage/temperature unit sensor on the channel designated by *ch$* as character data.

Example
```
:UNIT:SENSor CH1, K
```
The type of the voltage/temperature unit sensor on channel 1 is set to "K".

When allowed In all functions, when the 8937 is installed.

■ Sets and queries reference junction compensation of the voltage/temperature unit.

Common

Syntax

command	:UNIT:RJC *ch$, A$*
query	:UNIT:RJC? *ch$*
response	*ch$, A$*
	A$ = INT, EXT

Explanation Sets the reference junction compensation of the voltage/temperature unit on the channel designated by *ch$*.
Returns the reference junction compensation of the voltage/temperature unit on the channel designated by *ch$* as character data.

Example
```
:UNIT:RJC CH1, INT
```
The reference junction compensation of the voltage/temperature unit on channel 1 is set to "INT".

When allowed In all functions, when the 8937 is installed.

■ Sets and queries drift compensation of the voltage/temperature unit.

Common

Syntax

command	:UNIT:DRIFt *ch$, A$*
query	:UNIT:DRIFt? *ch$*
response	*ch$, A$*
	A$ = OFF, ON

Explanation Sets the drift compensation of the voltage/temperature unit on the channel designated by *ch$*.
Returns the setting of the drift compensation of the voltage/temperature unit on the channel designated by *ch$* as character data.

Example `:UNIT:`DRIFt `CH1, ON`
The drift compensation of the voltage/temperature unit on channel 1 is set to ON.

When allowed In all functions, when the 8937 is installed.

■ Sets and queries digital filter of the voltage/temperature unit.

Common

Syntax

command	:UNIT:DFILter *ch$, A$*
query	:UNIT:DFILter? *ch$*
response	*ch$, A$*
	A$ = OFF, ON

Explanation Enables and disables the digital filter of the voltage/temperature unit on the channel designated by *ch$*.
Returns the enablement state of the digital filter of the voltage/temperature unit on the channel designated by *ch$* as character data.

Example `:UNIT:DFILter CH1, ON`
The digital filter of the voltage/temperature unit on channel 1 is set to ON.

When allowed In all functions, when the 8937 is installed.

■ Sets and queries the anti-aliasing filter of the FFT unit. Common

Syntax

command :UNIT:AAFilter *ch$, A$*
query :UNIT:AAFilter? *ch$*
response *ch$, A$*
A$ = OFF, ON

Explanation Enables and disables the anti-aliasing filter of the FFT unit on the channel designated by *ch$*.
Returns the enablement state of the anti-aliasing filter of the FFT unit on the channel designated by *ch$* as character data.

Example

```
:UNIT:AAFilter CH1, ON
```

The anti-aliasing filter of the FFT unit on channel 1 is set to ON.

When allowed In all functions.

■ Carries out zero adjustment for the input units. Common

Syntax command :UNIT:ADJUST

Explanation Carries out zero adjustment for the input units (except when temperature is selected in the 8937).

When allowed In all functions.

■ Carries out auto-balancing for all of the strain unit channels. Common

Syntax command :UNIT:BALAnce

Explanation Carries out auto-balancing for all of the strain unit channels.

When allowed In all functions.

■ Carries out auto-balancing for each strain unit channel. Common

Syntax command :UNIT:CHBAlance *ch$*

Explanation Carries out auto-balancing for the selected channel.

Example

```
:UNIT:CHBAlance CH1
```

Carries out auto-balancing for channel 1.

When allowed In all functions.

■ Executes and queries the baseline offset.

8835-01, 8826, 8841, 8842, 8720

Syntax

command :UNIT:OFSCancel *ch$, A$*
query :UNIT:OFSCancel? *ch$*
response *ch$, A$*
ch$ = channel, ALL
A$ = OFF, ON

Explanation Executes the baseline offset for the channel designated by *ch$*.
If ALL is designated for *ch$*, executes the baseline offset for all the channels.
Returns the enablement state of the baseline offset for the channel designated by *ch$*.

Example

```
:UNIT:OFSCancel CH1, ON
```

Executes the baseline offset for channel 1.

When allowed In all functions.
The baseline offset cannot be executed in the strain unit, the universal unit (temperature range) and the F/V unit (frequency, count and pulse duty ratio).

■ Performs the clamp check in the F/V unit.

8835-01, 8826, 8841, 8842, 8720

Syntax

command :UNIT:CHKClamp

Explanation Performs the clamp check when using the clamp or the differential probe in the F/V unit. Always performs the clamp check before measurement when changing the clamp or the differential probe.

Example

```
:UNIT:CHKClamp
```

Performs the clamp check.

When allowed In the current measurement mode in the F/V unit.

■ Sets and queries the measurement mode of the F/V unit. Common

Syntax

command :UNIT:FVMOde *ch$, A$*
query :UNIT:FVMOde? *ch$*
response *ch$, A$*
A$ = FREQ (frequency), COUNT (integration), DUTY (pulse duty ratio), VOLT (voltage), CURRent

Explanation Sets the measurement mode of the F/V unit on the channel designated by *ch$*.
Returns the setting of the measurement mode of the F/V unit on the channel designated by *ch$*.

Example

```
:UNIT:FVMOde CH1, FREQ
```

Measures frequency on channel 1.

When allowed In all functions.

■ Sets and queries the frequency range of the F/V unit. Common

Syntax

command :UNIT:FRANge *ch$, A$*
query :UNIT:FRANge? *ch$*
response *ch$, A$*
A$ = 0.1HZ, 0.2HZ, 1HZ, 2HZ, 10HZ, 20HZ, 100HZ, 200HZ, 1KHZ, 2KHZ, 10KHZ, 10RPM, 20RPM, 100RPM, 200RPM, 1KRPM, P50HZ, P60HZ (8835 (-01))
0.05HZ, 0.1HZ, 0.5HZ, 1HZ, 5HZ, 10HZ, 50HZ, 100HZ, 500HZ, 1KHZ, 5KHZ, 5RPM, 10RPM, 50RPM, 100RPM, 500RPM, P50HZ, P60HZ (8826, 8841, 8842)

Explanation

Sets the frequency range when the F/V unit on the channel designated by *ch$* is in the frequency measurement mode.
Returns the setting of the frequency range of the F/V unit on the channel designated by *ch$*.

Example

```
:UNIT:FRANge CH1, 0.1HZ
```

The frequency range of the F/V unit on channel 1 is set to 0.1 Hz.

When allowed

In all functions.
In the frequency measurement mode in the F/V unit.

■ Sets and queries the threshold level of the F/V unit. Common

Syntax

command :UNIT:FVLEvel *ch$, A*
query :UNIT:FVLEvel? *ch$*
response *ch$, A* <NR3>
A = -10 to 10 (unit: V)

Explanation

Sets the threshold level of the F/V unit.
Returns the setting of the threshold level of the F/V unit as an NR3 numerical value.

Example

```
:UNIT:FVLEvel CH1, 2.4
```

The threshold level on channel 1 is set to 2.4 V.

When allowed

In all functions.
In the frequency, count and pulse duty ratio measurement modes in the F/V unit.

■ Sets and queries the hold of the F/V unit. Common

Syntax

command :UNIT:FVHOld *ch$, A$*
query :UNIT:FVHOld? *ch$*
response *ch$, A$*
A$ = ON, 10MS, 1S

Explanation Sets the hold of the F/V unit.
Returns the setting of the hold of the F/V unit as a character string.

Example

```
:UNIT:FVHOld CH1, 10MS
```

The hold of the F/V unit on channel 1 is set to 10 ms.

When allowed In all functions.
In the frequency measurement mode in the F/V unit.

■ Sets and queries the input switch of the F/V unit. Common

Syntax

command :UNIT:PULLup *ch$, A$*
query :UNIT:PULLup? *ch$*
response *ch$, A$*
A$ = OFF, ON

Explanation Sets the input switch of the F/V unit.
Returns the setting of the input switch of the F/V unit as a character string.

Example

```
:UNIT:PULLup CH1, ON
```

The input switch of the F/V unit on channel 1 is set to ON.

When allowed In all functions.

■ Sets and queries the measurement mode of the charge unit. Common

Syntax

command :UNIT:CMODe *ch$, A$*
query :UNIT:CMODe? *ch$*
response *ch$, A$*
A$ = VOLT (voltage), CHARge, PREamp (preamplifier)

Explanation Sets the measurement mode of the charge unit.
Returns the setting of the measurement mode of the charge unit as a character string.

Example

```
:UNIT:CMODe CH1, CHARGE
```

Measures the charge on channel 1.

When allowed In all functions.

■ Sets and queries the sensor sensitivity of the charge unit. Common

Syntax

command	:UNIT:CSENs *ch$, A*
query	:UNIT:CSENs? *ch$*
response	*ch$, A* <NR3>

A = 0.1 to 10 (sensor sensitivity)

Explanation Sets the sensor sensitivity of the charge unit.
Returns the setting of the sensor sensitivity of the charge unit as an NR3 numerical value.

Example
```
:UNIT:CSENs CH1, 3.5
```
The sensor sensitivity of the charge unit on channel 1 is set to 3.5.

When allowed In all functions.

5. DISPlay command (Sets and queries changeover of the screen mode and waveform display.)

:DISPlay

■ Sets and queries the screen mode. Common

Syntax

command	:DISPlay:CHANge *A$*
query	:DISPlay:CHANge?
response	*A$*

A$ = STATus, CHANnel, DISPlay, SYSTem, FILE
(8835 (-01))
SYSTem, STATus, TRIGger, CHANnel, DISPlay, FILE (8826, 8841, 8842)

Explanation Changes the screen mode.
Returns the current screen mode as character data.

Example
```
:DISPlay:CHANge DISPlay
```
Switches to the display mode.

When allowed In all functions.

■ Sets and queries changeover of the page of the screen. Common

Syntax

command :DISPlay:PAGE *A*
query :DISPlay:PAGE?
response *A* <NR1>
A: 1 to 6 (system screen) (8835 (-01))
1 to 5 (status screen)
1, 2 (channel screen)
A: 1 to 6 (system screen) (8826, 8841, 8842)
1 to 4 (status screen)
1, 2 (channel screen)
A: 1 to 4 (system screen) (8720)
1, 2 (status screen)
1 to 4 (channel screen)

Explanation Sets the page of the screen according to the NR1 numerical value.
Returns the current page of the screen as a NR1 numerical value.

When allowed In all functions.

■ Sets and queries waveform display color. Common

Syntax

command :DISPlay:DRAWing *ch$, A$*
query :DISPlay:DRAWing? *ch$*
response *ch$, A$*
A$ = OFF, C1 to C12

Explanation Sets the waveform display color for the channel designated by *ch$*.
Returns the waveform display color for the channel designated by *ch$* as character data.

Example

```
:DISPlay:DRAWing CH1, C1
```

Displays the channel 1 waveform in display color 1.

When allowed In all functions.

■ Sets and queries waveform display graph (when the display format is other than SINGLE). Common

Syntax

command :DISPlay:GRAPh *ch$, A*
query :DISPlay:GRAPh? *ch$*
response *ch$, A* <NR1>
A$ = 1, 2, 3, 4 (for DUAL format, 1, 2)
1 to 8 (OCT, HEX format: 8841, 8842, 8720)

Explanation

Sets the waveform display graph on the screen.
On the screen, returns the current waveform display graph for the channel designated by *ch$* as a numerical value in NR1 format.

Example

```
:DISPlay:GRAPh CH1, 1
```

Displays the channel 1 waveform in display graph 1.

When allowed

In MEM, REC, RMS and R&M.

■ Sets and queries logic waveform display color. Common

Syntax

command :DISPlay:LOGDraw *ch$, N, A$*
query :DISPlay:LOGDraw? *ch$, N*
response *ch$, N, A$*
ch$ = CHA to CHD (8835 (-01), 8841, 8842),
CHA to CHH (8826)
A$ = OFF, C1 to C12
N = 1, 2, 3, 4

Explanation

Sets the waveform display color for the logic channel designated by *ch$, N*.
Returns the waveform display color for the logic channel designated by *ch$, N* as character data.

Example

```
:DISPlay:LOGDraw CHA, 1, C1
```

Displays the waveform 1 of channel A in display color 1.

When allowed

In MEM, REC, RMS and R&M.

■ Sets and queries operation of logic waveform display. Common

Syntax

command :DISPlay:LOGPosi *ch$, A*
query :DISPlay:LOGPosi? *ch$*
response *ch$, A* <NR1>
ch$ = CHA to CHD (8835 (-01), 8841, 8842),
CHA to CHH (8826)
A = 1 to 8

Explanation Sets the position of logic waveform display.
Returns the position of the current logic waveform display as a numerical value in NR1 format.

Example `:DISPlay:LOGPosi CHA, 1`
Sets the position of logic waveform display for channel A to 1.

When allowed In MEM, REC, RMS and R&M.

■ Sets and queries magnification/compression factor on the time axis. Common

Syntax

command :DISPlay:XMAG *A$*
query :DISPlay:XMAG?
response *A$*
MEM: (8835 (-01))
A$ = X10, X5, X2, X1, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, X1_200, X1_500, X1_1000, X1_2000
REC, RMS:
A$ = X1, X1_2, X1_5, X1_10, X1_20, X1_50
MEM: (8826, 8841, 8842)
A$ = X10, X5, X2, X1, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, X1_200, X1_500, X1_1000, X1_2000, X1_5000, X1_10000
REC, RMS:
A$ = X1, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, X1_200, X1_500
A$ = X4, X2, X1, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, X1_200, X1_500 (8720)

Explanation Sets the magnification/compression factor on the time axis according to character data.
In the recorder and memory function, sets the magnification/ compression factor for the currently displayed waveform.
Returns the current magnification/compression factor on the time axis as character data.

Example `:DISPlay:XMAG X1_10`
Sets the compression ratio along the time axis to be 1/10.

When allowed In MEM, REC, RMS and R&M.

■ Sets and queries magnification/compression factor on the voltage axis. Common

Syntax

command :DISPlay:YMAG *ch$, A$*
query :DISPlay:YMAG? *ch$*
response *ch$, A$*
A$ = X1_2, X1, X2, X5, X10
(SINGLE, XY SINGLE format)
X1_4, 1_2, X1, X2.5, X5
(8826, 8841, 8842: other than the above)

Explanation

Sets the magnification/compression factor on the voltage axis for the channel designated by *ch$* according to the character data.
Returns the current magnification/compression factor on the voltage axis for the channel designated by *ch$* as character data.

Example

```
:DISPlay:YMAG CH1, X2
```

Sets the magnification ratio along the voltage axis for channel 1 to be X2.

When allowed

In MEM, REC, RMS and R&M.

■ Enables and disables, and queries the zoom function. Common

Syntax

command :DISPlay:ZOOM *A$*
query :DISPlay:ZOOM?
response *A$*
A$ = OFF, ON

Explanation

Enables and disables the zoom function.
Returns the current enablement state of the zoom function as character data.

Example

```
:DISPlay:ZOOM ON
```

Enables the zoom function.

When allowed

In MEM.

■ Sets and queries magnification/compression factor on the time axis, when the zoom function is used. Common

Syntax

command :DISPlay:ZOOMMag *A$*
query :DISPlay:ZOOMMag?
response *A$*

A$ = X10, X5, X2, X1, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, X1_200, X1_500, X1_1000, X1_2000(*), X1_5000(*) (*:8826 only)

Explanation

Sets the magnification/compression factor on the time axis for the lower screen, when the zoom function is used.
Returns as character data the current magnification/ compression factor on the time axis for the lower screen in the zoom function.
The display magnification of the lower display can only be set at a value that exceeds the magnification of the upper display. (E.g., if the upper magnification is X2, the lower can only be set to X5 or X10.)

Example

```
:DISPlay:ZOOMMag X1_100
```

Sets to be 1/100 the compression ratio along the time axis for the lower screen in the zoom function.

When allowed

In MEM.

■ Enales and disables the XY waveform display. 8826, 8841, 8842, 8720

Syntax

command :DISPlay:XYDRawing *A, B$*
query :DISPlay:XYDRawing? *A*
response *A* <NR1>, *B$*

A = 1 to 4
B$ = OFF, C1 to C12

Explanation

Sets the waveform display color for the graph designated by *A*.
Returns the current waveform display color for the graph designated by *A* as character data.

Example

```
:DISPlay:XYDRawing 1, C1
```

Displays the waveform of graph 1 in display color 1.

When allowed

In MEM and REC in XY format.

■ Sets and queries the X-axis, in the X-Y format. 8835 (-01)

Syntax

command	:DISPlay:XAXIs *ch$*
query	:DISPlay:XAXIs?
response	*ch$*

Explanation Sets the X-axis channel in the X-Y format.
Returns the current X-axis channel in the X-Y format as character data.

Example
```
:DISPlay:XAXIs CH2
```
Sets channel 2 to the X-axis.

When allowed In MEM and REC in XY format.

■ Sets and queries the X-axis in the XY format. 8826, 8841, 8842, 8720

Syntax

command	:DISPlay:XAXIs *A, ch$*
query	:DISPlay:XAXIs? *A*
response	*A* <NR1>, *ch$*
	A = 1 to 4

Explanation Sets the X-axis channel in the XY format for the display graph designated by *A*.
Returns the current X-axis channel in the XY format for the display graph designated by *A* as a numerical value in NR1 format.

Example
```
:DISPay:XAXIs 1, CH1
```
Sets the X-axis of graph 1 to channel 1.

When allowed In MEM and REC in XY format.

■ Sets and queries the Y-axis in the XY format. 8826, 8841, 8842, 8720

Syntax

command	:DISPlay:YAXIs *A, ch$*
query	:DISPlay:YAXIs? *A*
response	*A* <NR1>, *ch$*
	A = 1 to 4

Explanation Sets the Y-axis channel in the XY format for the display graph designated by *A*.
Returns the current Y-axis channel in the XY format for the display graph designated by *A* as a numerical value in NR1 format.

Example
```
:DISPay:YAXIs 1, CH2
```
Sets the Y-axis of graph 1 to channel 2.

When allowed In MEM and REC in XY format.

■ Performs waveform display. Common

Syntax

command :DISPlay:WAVE *A$*
response *A$*
A$ = ACUR (the A cursor: line cursor (vertical), trace cursor)
TRIG (the trigger point)
POINT (the point set by :MEMory:POINt)

Explanation Displays the waveform on the screen from the position indicated by *A$* (in the recorder and memory function, when the memory recorder waveform is displayed).

Example

```
:DISPlay:WAVE ACUR
```

Displays the waveform from the position of A cursor.

When allowed In MEM and R&M, when the display format is other than XY. (8720: in REC only)

■ Enables and disables, and queries the variable function. Common

Syntax

command :DISPlay:VARIable *ch$, A$*
query :DISPlay:VARIable? *ch$*
response *ch$, A$*
A$ = ON, OFF

Explanation Enables or disables the variable function for the channel designated by *ch$*. Returns the current enablement state of the variable function for the channel designated by *ch$* as character data.

When allowed In all functions.

■ Sets and queries the upper and lower limits of the variable function. Common

Syntax

command :DISPlay:VARIUPLOw *ch$, B, C*
query :DISPlay:VARIUPLOw? *ch$*
response *ch$, B, C* <NR3>
B = *C* = -9.9999E+29 to +9.9999E+29
B: lower limit value <NR3>, C: upper limit value <NR3>

Explanation Sets the upper and lower limits of the waveform on the display screen. Returns the current upper and lower limits of the waveform on the display screen as an NR3 numerical value.

When allowed In all functions.

■ Sets and queries values for variable range and position.

8835-01,8841,8842,8720

Syntax

command	:DISPlay:VARIRng *ch$,A,B*
query	:DISPlay:VARIRng? *ch$*
response	*ch$,A*<NR3>,B<NR3>

A,B = -9.9999E+29～+9.9999E+29

Explanation Sets the values of the variable range and position for each channel.
Returns the current values of the variable range and position.

Example **:DISPlay:VARIRng ch1,2,30**
Sets variable values for Channel 1, 2 (units) per 1 DIV for the range and 30% for the position.

When allowed In all functions.

■ Sets and queries the display clear function. Common

Syntax

command	:DISPlay:XYCLr *A$*
query	:DISPlay:XYCLr?
response	*A$*

A$ = OFF, ON

Explanation Sets the display clear function.
Returns the current setting of the display clear function as character data.

Example **:DISPlay:XYCLr ON**
Sets the display clear function to ON.

When allowed In REC in XY format.

■ Sets and queries the screen size. 8826, 8720

Syntax

command	:DISPlay:SIZE *A$*
query	:DISPlay:SIZE?
response	*A$*

A$ = NORMal, WIDE

Explanation Sets the screen size (normal or wide).
Returns the current screen size as character data.

Example **:DISPlay:SIZE WIDE**
Sets the screen size to wide.

When allowed In MEM, REC, RMS and R&M.

■ Sets and queries the CRT display waveform for the recorder and memory function. 8835 (-01) A, 8826, 8841, 8842

Syntax

command :DISPlay:RMDIsplay *A$*
query :DISPlay:RMDIsplay?
response *A$*
A$ = REC, MEM

Explanation Sets the waveform shown on the screen, in the recorder and memory function, according to the character data.
Returns the waveform shown on the screen, in the recorder and memory function, as character data.

Example

```
:DISPlay:RMDIsplay MEM
```

Sets the waveform shown in the recorder and memory function to the memory recorder waveform.

When allowed In MEM, REC, RMS and R&M.

■ Sets and queries synchronization function. 8720

Syntax

command :DISPlay:SYNC *A$*
query :DISPlay:SYNC?
response *A$*
A$ = OFF,ON

Explanation Operates same as the synchronization key on the main unit.
Enabled only after the monitor screen has been partitioned.
Returns the current setting of the synchronization function.

Example

```
:DISPlay:SYNC ON
```

Use the synchronization function.

When allowed Monitor screen partitioned.

■ Sets and queries operational screen. 8720

Syntax

command :DISPlay:VIEWSel *A*
query :DISPlay:VIEWSel?
response *A* <NR1>
A = 1,2 (screen selection)

Explanation Selects the screen for operation, equivalent to selecting the screen with the screen setting key on the main unit. Enabled only when the screen has been partitioned.
Returns the current screen selection.

Example

```
:DISPlay:VIEWSel 1
```

Selects the left (upper screen).

When allowed Monitor screen partitioned.

■ Executes and queries screen partition. 8720

Syntax

command :DISPlay:VIEWPart *A*
query :DISPlay:VIEWPart?
response *A* <NR1>
A=0 (normal), 1 (left and right), 2（upper and lower）

Explanation Executes screen partitioning, equivalent to operation of the screen partitioning key on the main unit.

Example

```
:DISPlay:VIEWPart 1
```

Divides the screen into left and right.

When allowed —

6. CURSor command (Cursor setting and reading)

:CURSor

■ Sets and queries the A and B cursor type. Common

Syntax

command :CURSor:MODE *A$*
query :CURSor:MODE?
response *A$*
A$ = OFF, TIME, VOLT, TRACe
OFF, TRACe (FFT)
OFF, Xcur, Ycur, TRACe (in X-Y format)
TIME, Xcur: vertical cursor
VOLT, Ycur: horizontal cursor
TRACe: trace cursor

Explanation Sets the A and B cursor type (vertical cursor, horizontal cursor, trace cursor). Returns the current A and B cursor type as character data.

Example

```
:CURSor:MODE TIME
```

Sets vertical cursors.

When allowed In all functions.

■ Selects among, and queries, A, B and A & B cursors. Common

Syntax

command :CURSor:ABCUrsor *A$*
query :CURSor:ABCUrsor?
response *A$*
A$ = A, ORA, ORB, A_B

Explanation Selects among A, B and A & B cursors.
Returns whether currently the A cursor, B cursor or both A & B cursors are in use, as character data.

Example

```
:CURSor:ABCUrsor A
```

Sets A cursor.

When allowed In all functions.

■ Sets and queries the channel for the A cursor. Common

Syntax

command :CURSor:ACHAnnel *ch$*
query :CURSor:ACHAnnel?
response *ch$*
ch$ = CH1 to CH4 (8835),
CH1 to CH8, ALL(MEM,REC,R&M,RMS) (8835-01),
CH1 to CH32, ALLH, ALLL(REC,RMS), ALL(MEM) (8826),
CH1 to CH16, ALL(MEM,REC,RMS) (8841, 8842),
CH1 to CH16 (8720)
X1 to X4 (in X-Y format: 8826, 8841, 8842, 8720)

Explanation Sets the channel for the A cursor.
(ALL, ALLL, and ALLH can be set during use of the trace cursor and A cursor.)
Returns the current A cursor channel as character data.

Example

```
:CURSor:ACHAnnel CH1
```

Sets the channel for the A cursor to channel 1.

When allowed During use of the trace cursor or the horizontal cursor (except in FFT).
(On the 8826 in X-Y format, the vertical cursor as well)

■ Sets and queries the channel for the B cursor. Common

Syntax

command :CURSor:BCHAnnel *ch$*
query :CURSor:BCHAnnel?
response *ch$*
ch$ = CH1 to CH4 (8835), CH1 to CH8 (8835-01),
CH1 to CH32 (8826), CH1 to CH16 (8841, 8842, 8720)
X1 to X4 (in X-Y format: 8826, 8841, 8842, 8720)

Explanation Sets the channel for the B cursor.
Returns the current B cursor channel as character data.

Example

```
:CURSor:BCHAnnel CH1
```

Sets the channel for the B cursor to channel 1.

When allowed During use of the trace cursor or the horizontal cursor (except in FFT).
(On the 8826 in X-Y format, the vertical cursor as well)

■ Sets and queries the position of the A cursor. Common

Syntax

command :CURSor:APOSition *A*
query :CURSor:APOSition?
response *A* <NR1>
(line cursor (vertical), trace cursor)
A = 0 to (number of stored data values)
(100 × recording length)
0 to 400 (X-Y format)
0 to 480 (X-Y format wide screen: 8826)
0 to 320 (X-Y format wide screen: 8720)
0 to 9999 : FFT (STR, ACR, CCR, IMP)
0 to 400 : FFT (HIS, OCT)
0 to 4000 : FFT (others)
(line cursor (horizontal))
A = 0 to 400
0 to 480 (wide screen: 8826)
0 to 639 (8841, 8842, 8720)
0 to 320 (X-Y format wide screen: 8720)

Explanation Sets the A cursor position.
Returns the current A cursor position as a numerical value in NR1 format.

Example

```
:CURSor:APOSition 1000
```

Move the A cursor position to 1000 points (10 DIV).

When allowed In all functions.

■ Sets and queries the position of the B cursor. Common

Syntax

command :CURSor:BPOSition *A*
query :CURSor:BPOSition?
response *A* <NR1>

Explanation Same as in APOSition.

■ Queries the cursor readout value (t). Common

Syntax

query :CURSor:DTREad? *A$*
response *B$*
A$ = A, B, B_A
B$ = the readout value (t)

Explanation Returns the cursor readout value (t) as character data.

Example

query **:CURSor:DTREad? A**
response **:CURSor:DTREad 5ms**

Queries the A cursor readout value.

When allowed During use of the trace cursor or the vertical cursor (except in FFT).

■ Queries the cursor readout value (V). Common

Syntax

query :CURSor:DVREad? *A$ (,ch$)*
response *B$, (C$)*
A$ = A, B, B_A
ch$*= CH1 to CH4 (8835), CH1 to CH8 (8835-01), CH1 to CH32 (8826), CH1 to CH16 (8841, 8842, 8720) (only *A$ is *A*)
B$ = the readout value (V, ℃, μ ε)
(In the recorder and RMS recorder functions)
B$: maximum value
C$: minimum value
(when the time axis is 5 s/DIV in the RMS recorder function, *B$* only)
Trace in X-Y format
B$: voltage on the X-axis
C$: voltage on the Y-axis

Explanation Returns the cursor readout value (V, ℃, μ ε) as character data.

Example

query **:CURSor:DVREad? A**
response **:CURSor:DVREad 385**

Queries the A cursor readout value.

When allowed During use of the trace cursor or the horizontal cursor (except in FFT).

■ Sets and queries the graph for the A and B cursors.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CURSor:ABCHAnnel *A$*
query :CURSor:ABCHAnnel?
response *A$*
A$ = G1, G2

Explanation Sets the graph for the A and B cursors when the display format is DUAL. If the display format is SINGle or NYQuist, the cursor is displayed on graph 1, whatever setting is made with this command.
Returns the current graph setting for the A and B cursors as character data.

Example

```
:CONFigure:FORMat DUAL
:CURSor:ABCHAnnel G1
:CURSor:MODE TRACEe
```

The A and B cursors are displayed on graph 1.

When allowed In FFT.

■ Queries the cursor readout position for FFT data.

8835 (-01) A, 8826, 8841, 8842

Syntax

query :CURSor:DFREad? *A$*

response *B$, C$*

A$ = A, B, B_A

B$ = readout position for x-axis data

C$ = readout position for y-axis data

Explanation Returns the current cursor readout position for FFT data as character data.

Example

```
:CURSor:DFREad? A
```

The A cursor readout position is returned as character data.

When allowed In FFT.

7. MEMory command (Sets and queries input and output, etc., from the memory.)

:MEMory

■ Sets and queries the point in memory for input/output. Common

Syntax

command :MEMory:POINt *ch$, A*

query :MEMory:POINt?

response *ch$, A* <NR1>

ch$ = CH1 to CH4, CHA to CHD (8835)
CH1 to CH8, CHA to CHD (8835-01)
CH1 to CH32, CHA to CHH (8826)
CH1 to CH16, CHA to CHD (8841, 8842, 8720)

A = 0 to 2000000 (8835)
0 to 4000000 (8835-01)
0 to 16000000 (8826, 8841, 8842)
0 to 1000000 (8720)

Explanation Sets the input/output point in memory.
Returns the current input/output point in memory as an NR1 numerical value.

Example

```
:MEMory:POINt CH1, 100
```

Sets the input/output point for channel 1 to the 100th location from the start of memory.

When allowed In MEM and R&M (8720: in REC only).

■ Queries the number of data samples stored. Common

Syntax

query :MEMory:MAXPoint?

response *A* <NR1>

A = 0 : no data stored

1 to 2000000 (divided by 100 gives the number of divisions) (8835)

1 to 4000000 (divided by 100 gives the number of divisions) (8835-01)

1 to 16000000 (divided by 100 gives the number of divisions) (8826, 8841, 8842)

1 to 1000000 (divided by 100 gives the number of divisions) (8720)

Explanation Returns the number of data samples stored in the memory as a numerical value in NR1 format.

Example

:MEMory:MAXPoint?

response :MEMory:MAXPoint 1000 (when headers are on)

The number of data samples stored in the memory is 1000 (10 divisions).

When allowed In MEM and R&M. (8720: in REC only)

■ Prepares the memory. Common

Syntax

command only :MEMory:PREPare

Explanation If there is no waveform data in the unit, ensures that the memory is in a state ready and able to receive transmitted data.

Example

:MEMory:PREPare

Prepares the memory for receipt of waveform data.

Note If data is currently stored in memory, a waveform is erased.

When allowed In MEM and R&M (8720: in REC only).

■ Inputs data to memory, and outputs stored data (in ASCII). Except 8720

Syntax

command :MEMory:ADATa *B, C,...*
query :MEMory:ADATa? *A*
response *B, C,...* all <NR1>
B, C,... = -2048 to 2047 (-737 to 3358 when measuring temperature with the 8937)
A = 1 to 80 (number of data values to be output)

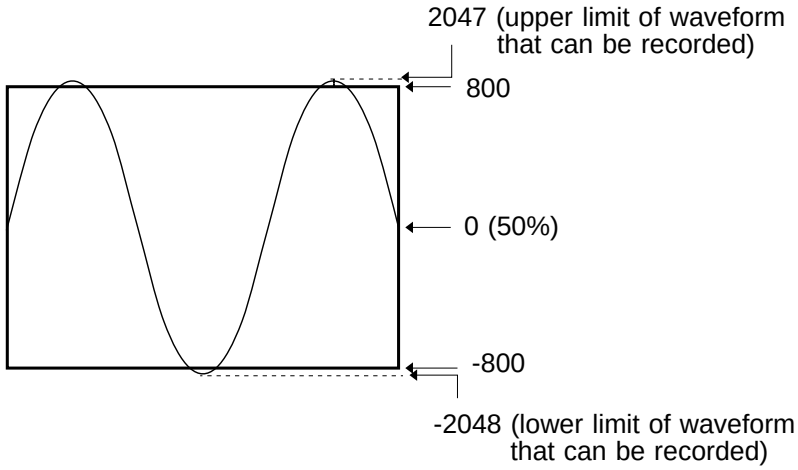
Explanation

Puts the data of the data portion into the memory at the channel and point set by the :MEMory:POINt command. If there are several data values, they are input in order from the point set by the :MEMory:POINt command.
The input/output point is incremented by the number of data values.
The number of data values specified by A are output from the memory channel and point set by the :MEMory:POINt command. The input/output point is incremented by the number of data values.
This cannot be executed during measurement operation.

∗Relationship between data values in memory and measured voltages

The following figure illustrates the relationship between the data values (-2048 to 2047)* input and output using the :MEMory:ADATa command and the measured voltage values.

*: (-737 to 3358 (0 ℃: 0) when measuring temperature with the 8937)



Measured voltage value = data value×voltage range/160 (8835)
Measured voltage value = data value×voltage range/80 (8826, 8841, 8842, 8720)

Example:
When voltage range = 1 (V/DIV), data value = 768
Measurement voltage = 768×1/160 = 4.8 (V)

Example

```
:MEMory:POINt CH1, 0
:MEMory:ADATa? 10
```

Sets the input/output point to channel 1 and data value zero in memory, then outputs 10 stored data values.

When allowed

In MEM and R&M, provided that stored data is present, and provided that the input/output point is lower than the amount of data stored.

■ Input voltage data to memory, and output voltage data from memory.

Except 8720

Syntax

command :MEMory:VDATa *B, C,...*
query :MEMory:VDATa? *A*
response *B, C,...* all (NR3>
B, C,... = voltage values (unit V, μ ε , ℃)
A = 1 to 40 (amount of data)

Explanation

Puts the data values (voltage values) in the data portion into the memory at the channel and point set by the :MEMory:POINt command.
If there are several data values, they are input in order from the point set by the :MEMory:POINt command. The input/output point is incremented by the number of data values.
The number of stored data values specified by *A* are output as voltage values from the memory channel and point set by the :MEMory:POINt command. The input/output point is incremented by the number of data values.
＊When scaling, the scaled values are input and output.
When calculating the waveform, calculated results are input and output.
This cannot be executed during measurement operation.

Example

```
:MEMory:POINt CH1, 0
:MEMory:VDATa? 10
```

Sets the input/output point to channel 1 and data value zero in memory, then outputs 10 stored data values as voltage values.

When allowed

In MEM and R&M, provided that stored data is present, and provided that the input/output point is lower than the amount of data stored. Stored data is obtained when there is no waveform data (see "Prepares the memory.").

■ Captures real time data. Common

Syntax

command :MEMory:GETReal

Explanation

Captures the values which are currently input on the channel for all the channel.

When allowed

Providing that measurement operation is not taking place.

■ Outputs real time data (in ASCII). Common

Syntax
query :MEMory:AREAl? *ch$*
response *A* <NR1>
A = -2048 to 2047

Explanation Returns the value input on the channel designated by *ch$*.

Example
query **:MEMory:AREAl? CH1**
response **:MEMory:AREAl 2000** (HEADER ON)

Note When the :MEMory:GETReal command is not executed before this command, the returned value is not fixed.

When allowed Providing that measurement operation is not taking place.

■ Outputs real time data (voltage values). Common

Syntax
query :MEMory:VREAl? *ch$*
response *A* <NR3>
A = a voltage value (unit V, μ ε, ℃)

Explanation Returns as a voltage value the value input on the channel designated by *ch$*.

Example
query **:MEMory:VREAl? CH1**
response **:MEMory:VREAl 4.7E-2** (HEADER ON)

Note When the :MEMory:GETReal command is not executed before this command, the returned value is not fixed.

When allowed Providing that measurement operation is not taking place.

■ Input logic data to memory, and output logic data from memory.

Except 8720

Syntax

command :MEMory:LDATa *B, C,...*
query :MEMory:LDATa? *A*
response *B, C,...* <NR1>
B, C,... = 0 to 15
A = 1 to 100 (number of data values to be output)

Explanation

Puts the data values (logic values) in the data portion into the memory at the channel and point set by the :MEMory:POINt command. If there are several data values, they are input in order from the point set by the :MEMory:POINt command. The input/output point is incremented by the number of data values.
The number of stored data values specified by A are output as logic values from the memory channel and point set by the :MEMory:POINt command. The input/output point is incremented by the number of data values.
This cannot be executed during measurement operation.

The following is the correspondence between the channels set by the :MEMory:POINt command and the logic channel groups:

CHA---CHA1 to CHA4 (8835 (-01), 8841, 8842)
CHB---CHB1 to CHB4
CHC---CHC1 to CHC4
CHD---CHD1 to CHD4

CHA---CHA1 to CHA4 (8826)
CHB---CHB1 to CHB4
:
CHH---CHH1 to CHH4

The eight logic channels in each group are encoded as binary bits in the NR1 data value, as shown in the following example.

Bit	7	6	5	4	3	2	1	0
Data	0	0	0	0	A4	A3	A2	A1

(CHA)

Example

:MEMory:POINt CH1, 0
query **:MEMory:LDATa? 1**
response **:MEMory:LDATa 10**
Channels A1 to A4 are as follows;

Bit	7	6	5	4	3	2	1	0
Data	0	0	0	0	1 A4	0 A3	1 A2	0 A1

LOW : 0
HIGH : 1

When allowed

In MEM and R&M, provided that stored data is present, and provided that the input/output point is lower than the amount of data stored.

■ Binary transfer of stored data. Except 8720

Syntax

query :MEMory:BDATa? *A*

response #0 • • • •

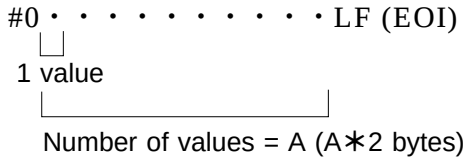
A = 1 to 200 (number of data values to be output)

Explanation

Outputs the data stored by a :MEMory:POINt specification in binary format. The input/output point is incremented by the number of data values.

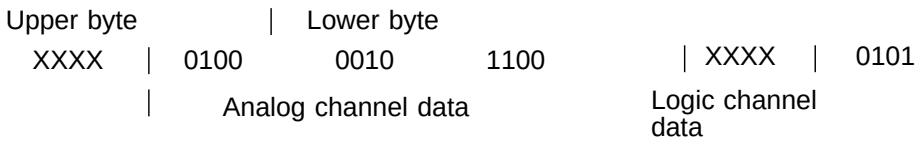
The format of the output data is as follows:

- Initially: "#0" (Indicates binary format.)
- After the "#0", the number of data values specified by A (each value is two bytes (one byte: logic data)), is transmitted.
- The data is followed by LF (0AH) + EOI.



The data consists of the unaltered binary codes of the data stored in memory. The bits are transmitted most significant bit first.

Example:



The data obtained is the same as that for ADATa? and LDATa?; for details refer to these commands.
It is not possible to input data in binary format.

Example

```
:MEMory:POINt CH1, 0
:MEMory:BDATa? 10
```

This sets the input/output point to channel 1, and stored data value to address 0 in memory, then outputs 10 data values in binary format.

When allowed

In MEM and R&M, provided that stored data is present, and provided that the input/output point is lower than the amount of data stored.

■ Outputs real time data (logic). Common

Syntax

query :MEMory:LREAl? *ch$*

response *A* <NR1>

A = 0 to 15

Explanation

See "Input logic data to memory, and output logic data from memory."

Note

When the :MEMory:GETReal command is not executed before this command, the returned value is not fixed.

When allowed

Providing that measurement operation is not taking place.

■ Outputs real time data (binary). Except 8720

Syntax

query	:MEMory:BREAl? *ch$*
response	Binary data (2 bytes)

Explanation See "Binary transfer of stored data."

Note When the :MEMory:GETReal command is not executed before this command, the returned value is not fixed.

When allowed Providing that measurement operation is not taking place.

■ Sets and queries the output point for FFT data.

8835 (-01) A, 8826, 8841, 8842

Syntax

command	:MEMory:FFTPOint *A$, B*
query	:MEMory:FFTPOint?
response	*A$, B*<NR1>

A$ = G1, G2

B = 0 to 9999 : in analysis mode STR, ACR, CCR, or IMP (maximum value: number of FFT points - 1)

0 to 4000 : in analysis mode LIN, RMS, PSP, TRF, COH, or CSP (maximum value: number of FFT points × 0.4)

0 to 400 : HIS or OCT

Explanation Sets the output point for FFT data on the graph number designated by *A$*. Returns the current output point as an NR1 format.

Example

```
:MEMory:FFTPOint G1,100
```

Sets the output point for FFT data on the graph 1 to 100.

When allowed In FFT.

■ Queries the FFT data at the output point.

8835 (-01) A, 8826, 8841, 8842

Syntax

query only :MEMory:FFTData?

response *A* unit, *B* unit

A = x-axis data (in <NR3> numerical format)

B = y-axis data (in <NR3> numerical format)

Explanation Returns the x-axis and y-axis FFT data at the output point specified by the instruction :MEMORY:FFTPOint in <NR3> numerical format.
When this command is executed, only one output point is calculated, and then the specified output point is increased by one. By executing this command repeatedly, a continuous set of data can be obtained.

Example

```
:MEMory:FFTPOint G1,100
:MEMory:FFTData?
```

Returns the x-axis and y-axis FFT data at points of 100 on graph 1.

When allowed In FFT.

■ Reads real-time store data. 8826,8841,8842

Syntax

command :MEMory:RTLOad *A(,B)*

A = data start point

B = data end point

Explanation Reads the memory waveform of data stored in real time. Previously read real time store data is required.
When B (data end point) has not been set, then data is read from A (data start point) to 2000DIV.

Example

```
:MEMory:RTLOad 100,1100
```

When real time store data is read by the main unit, memory waveform data is read from the 100th point to point 1100.

When allowed After real time data has been stored.

■ Storage data output (ASCII). 8720

Syntax

command	:MEMory:RECAData *B1,B2,C1,C2,…*
query	:MEMory:RECAData? *A*
response	*B1,B2,C1,C2,…* all <NR1>
	B1,B2,C1,C2,… = -2048～2047
	A = 1～40（number of the output sample）

Explanation

Inputs data into storage memory from the channel and point specified by MEMory:POINt. When there are several pieces of data, then data are input sequentially from the point specified by MEMory:POINt, and the input/output points increment the data number.
Outputs the storage data from the channel and point specified by MEMory:POINt for the number of the sample specified at A.
In the recorder function, one sample consists of 2 pieces of data which hold maximum and the minimum values. For the response B1, B2, C1, C2, for example, B1, B2 comprise the first piece of data where B1 is the maximum value and B2 is the minimum value. C1, C2 is the second piece of data where C1 is the maximum value and C2 is the minimum value. During startup this cannot be executed.
* For details about the relationship between storage data and the voltage value, refer to :MEMory:ADATa.

Example

```
:MEMory:POINt CH1,0
:MEMory:RECAData? 10
```

Sets the I/O point to channel 1 and storage data to 0, and outputs 10 samples (data number: 20) of storage data.

When allowed

When there is storage data.

■ Storage data output (voltage value). 8720

Syntax

command	:MEMory:RECVData *B1,B2,C1,C2,…*
query	:MEMory:RECVData? *A*
response	*B1,B2,C1,C2,* … all <NR3>
	A = 1～20（number of the output sample）

Explanation

Inputs data (voltage value) into storage memory from the channel and point specified by MEMory:POINt. When there are several pieces of data, then data are input sequentially from the point specified by MEMory:POINt, and the input/output points increment the data number.
Outputs the storage data from the channel and point specified by MEMory:POINt for the number of the sample specified at A.
In the recorder function, one sample consists of 2 pieces of data which hold maximum and the minimum values. For the response B1, B2, C1, C2, for example, B1, B2 comprise the first piece of data where B1 is the maximum value and B2 is the minimum value. C1, C2 is the second piece of data where C1 is the maximum value and C2 is the minimum value. During startup this cannot be executed.
* When scaling, outputs the scaling value.

Example

```
:MEMory:POINt CH1,0
:MEMory:RECVData? 10
```

Sets the I/O point to channel 1 and storage data to 0, and outputs 10 samples (data number: 20) of storage data.

When allowed When there is storage data.

■ Transfers storage data in binary. 8720

Syntax

command	:MEMory:RECBData? *A*
query	#0‥‥‥
response	*A* = 1～100（number of the output sample）

Explanation

Outputs in binary format an output sample (specified at A) of the storage data from the channel and point specified by MEMory:POINt. The input/output points increment the data number. For details about the output point, refer to MEMory:BDATa.
In the recorder function, one sample consists of 2 pieces of data which hold maximum and the minimum values. The sequence of the data is identical to that of :RECAData, the first sample maximum value, first sample minimum value, second sample maximum value, second sample minimum value, and so on.

Example

```
:MEMory:POINt CH1,0
:MEMory:RECBData? 10
```

Sets the I/O point to channel 1 and storage data to 0, and outputs 10 samples (data number: 20) of storage data.

When allowed When there is storage data.

■ Storage data output (logic). 8720

Syntax

command	:MEMory:RECLData *B1,B2,C1,C2,…*
query	:MEMory:RECLData? *A*
response	*B1,B2,C1,C2,…* all <NR1>
	B1,B2,C1,C2,…=0～15
	A = 1～50（number of the output sample）

Explanation

Inputs data into storage memory from the channel and point specified by MEMory:POINt. When there are several pieces of data, then data are input sequentially from the point specified by MEMory:POINt, and the input/output points increment the data number.
Outputs the storage data from the channel and point specified by MEMory:POINt for the number of the sample specified at A.
In the recorder function, one sample consists of 2 pieces of data which hold maximum and the minimum values. For the response B1, B2, C1, C2, for example, B1, B2 comprise the first piece of data where B1 is the maximum value and B2 is the minimum value. C1, C2 is the second piece of data where C1 is the maximum value and C2 is the minimum value. During startup this cannot be executed.
* For details about logic data, refer to :MEMory:LDATa.

Example

```
:MEMory:POINt CH1,0
:MEMory:RECLData? 10
```

Sets the I/O point to channel 1 and storage data to 0, and outputs 10 samples (data number: 20) of storage data.

When allowed

When there is storage data.

8. SYSTem command (Sets and queries the system screen.)

:SYSTem

■ Sets and queries the number of channels used. Exept 8720

Syntax

command :SYSTem:USECH *A*
query :SYSTem:USECH?
response *A* <NR1>
A = 1, 2, 4 (8835)
1, 2, 4, 8 (8835-01)
4, 8, 16, 32 (8826)
2, 4, 8, 16 (8841, 8842)

Explanation

Sets the number of channels used to a numerical value.
Returns the current number of channels used as an NR1 numerical value.

Example

```
:SYSTem:USECH 4
```

Sets the number of channel used to 4.

When allowed

In MEM.

■ Enables and disables, and queries the start key backup function.
Common

Syntax

command :SYSTem:STARt *A$*
query :SYSTem:STARt?
response *A$*
A$ = OFF, ON

Explanation

Enables and disables the start key backup function.
Returns the current enablement state of the start key backup function as character data.

Example

```
:SYSTem:STARt ON
```

Sets the start key backup function to ON.

When allowed

In all functions.

■ Sets and queries the grid type. Common

Syntax

command :SYSTem:GRID *A$*
query :SYSTem:GRID?
response *A$*
A$ = OFF, STD, FINE, STD_Dark, FINE_Dark, TIME, TIME_Dark
OFF: No grid
STD: Standard grid
FINE: Fine grid
STD_Dark: Standard (dark) grid
FINE_Dark: Fine (dark) grid
TIME: Time axis grid (8826)
TIME_Dark: Dark time axis grid (8826)
A$ = OFF, STD (8720)

Explanation

Sets the type of grid displayed.
Returns the current grid setting as character data.

Example

```
:SYSTem:GRID STD
```

Sets the standard grid.

When allowed

In all functions.

■ Enables and disables, and queries the channel marker. Except 8720

Syntax

command :SYSTem:CHMArk *A$*
query :SYSTem:CHMArk?
response *A$*
A$ = OFF, NUMBer, COMMent

Explanation

Makes the corresponding channel marker setting.
Returns the current channel marker setting as character data.

Example

```
:SYSTem:CHMArk ON
```

Sets the channel marker to ON.

When allowed

In all functions.

■ Sets and queries the time axis display. Except 8720

Syntax

command	:SYSTem:TMAXis *A$*
query	:SYSTem:TMAXis?
response	*A$*

A$ = TIME, TIME (60), SCALE, DATE

Explanation Sets the time axis display as character data.
Returns the current time axis display setting as character data.

Example

```
:SYSTem:TMAXis TIME
```

Sets the time axis display to TIME.

When allowed In all functions.

■ Sets and queries the list function and the gauge function. Except 8720

Syntax

command	:SYSTem:LIST *A$*
query	:SYSTem:LIST?
response	*A$*

A$ = OFF, LIST, GAUGE, L_G (LIST&GAUGE)

Explanation Sets the list function and the gauge function according to character data.
Returns the current settings for the list function and the gauge function as character data.

Example

```
:SYSTem:LIST LIST
```

Sets the list function.

When allowed In all functions.

■ Sets and queries the printer density. Except 8720

Syntax

command	:SYSTem:PRIDensity *A*
query	:SYSTem:PRIDensity?
response	*A* <NR1>

A = 1 to 5 (1: light, 5: dark)

Explanation Sets the printer density according to character data.
Returns the current printer density setting as a numerical value in NR1 format.

Example

```
:SYSTem:PRIDensity 3
```

Sets the printer density to 3.

When allowed In all functions.

■ Enables and disables, and queries the backlight saver function.

Common

Syntax

command	:SYSTem:CRTOff *A*
query	:SYSTem:CRTOff?
response	*A* <NR1>
	A = 0 (OFF), 1 to 30 (minutes)

Explanation Enables or disables the backlight saver function.
Returns the current enablement state of the backlight saver function as a numerical value in NR1 format.

Example

```
:SYSTem:CRTOff 1
```

Sets the backlight saver function to one minute.

When allowed In all functions.

■ Sets and queries the screen color. Common

Syntax

command	:SYSTem:LCDDisp *A$*
query	:SYSTem:LCDDisp?
response	*A$*
	A$ = C1 to C9 (C9: customer color)

Explanation Sets the screen color according to character data.
Returns the current screen color setting as character data.

Example

```
:SYSTem:LCDDisp C1
```

Sets the screen color to 1.

When allowed In all functions.

■ Sets and queries the customer color. Common

Syntax

command :SYSTem:SETColor *A, B, C, D*
query :SYSTem:SETColor? *A*
response *A, B, C, D* <NR1>
A = color number
0 to 26 (8835 (-01))
0 to 30 (8826)
0 to 33 (8841, 8842, 8720)

Color assignment tables

Color	Color assignment name	
0	Waveform background	Waveform background
1	Inside warning	Warning character
2 3 4 5 6	Background color	Fixed character Setting character Background Frame GUI
7 8 9 10 11 12 13 14 15 16 17 18 19	Waveform	Grid Waveform color 1 Waveform color 2 Waveform color 3 Waveform color 4 Waveform color 5 Waveform color 6 Waveform color 7 Waveform color 8 Waveform color 9 Waveform color 10 Waveform color 11 Waveform color 12
20 21 22 23	Inside window	Fixed character Setting character Background Frame
24 25 26	Inside display	A/B cursors Cursor value Calculation result
27 28 29 30	Inside window 2 (8826 only)	Fixed character Setting character Background Window shadow

8835 (-01), 8826, 8720

Color	Color assignment name
0	Waveform background
1	Grid
2	A/B cursors
3	Cursor value
4	Calculation result
5	CH. SET character
6	Waveform color 1
7	Waveform color 2
8	Waveform color 3
9	Waveform color 4
10	Waveform color 5
11	Waveform color 6
12	Waveform color 7
13	Waveform color 8
14	Waveform color 9
15	Waveform color 10
16	Waveform color 11
17	Waveform color 12
18	Background 1
19	Fixed character
20	Setting character
21	Setting background
22	Title character
23	Title frame
24	Window frame (upper left)
25	Window frame (lower right)
26	GUI (OFF)
27	Character (OFF)
28	GUI (ON)
29	Character (ON)
30	File list background
31	File character
32	Directory character
33	Message frame

8841, 8842

B (red), C (green), D (blue) = 0 to 7

Explanation Sets the color of the color number specified by A to one of 8 shades of red, green, and blue.
Returns the settings for red, green, and blue of the color number specified by A as a numerical value in NR1 format.

Example **:SYSTem:SETColor 7, 3, 4, 5**
Sets the grid (8835, 8826) color to red 3, green 4, and blue 5.

When allowed In all functions.

■ Enables and disables, and queries the sound of beeper. Common

Syntax

command	:SYSTem:BEEPer *A$*
query	:SYSTem:BEEPer?
response	*A$*

A$ = OFF, ON (8835)
OFF, ON1, ON2 (8835-01, 8826, 8841, 8842, 8720)

Explanation Enables and disables the beeper sound.
Returns the current enablement state of the beeper sound as character data.

Example **:SYSTem:BEEPer ON**
Sets the beeper sound to ON.

When allowed In all functions.

■ Sets and queries the language. Common

Syntax

command	:SYSTem:LANGuage *A$*
query	:SYSTem:LANGuage?
response	*A$*

A$ = JAPAnese, ENGlish

Explanation Sets the language.
Returns the current language setting as character data.

Example **:SYSTem:LANGuage JAPAnese**
Sets the language to Japanese.

When allowed In all functions.

■ Sets and queries printing of the upper and lower limits.
8826, 8841, 8842

Syntax

command	:SYSTem:PRIUplow *A$*
query	:SYSTem:PRIUplow?
response	*A$*

A$ = OFF, ON

Explanation Enables and disables printing of the upper and lower limits.
Returns the current enablement state of printing of the upper and lower limits as character data.

Example **:SYSTem:PRIUplow ON**
Prints the upper and lower limits.

When allowed In all functions.

■ Sets and queries the zero position comment. 8826, 8841, 8842

Syntax

command	:SYSTem:ZEROcom *A$*
query	:SYSTem:ZEROcom?
response	*A$*

A$ = OFF, ON

Explanation Enables and disables the zero position comment.
Returns the current enablement state of the zero position comment as character data.

Example

```
:SYSTem:ZEROcom ON
```

Prints comments in the zero position.

When allowed In all functions.

■ Sets and queries the counter print. 8826, 8841, 8842

Syntax

command	:SYSTem:COUNter *A$ (,"NAME$", B)*
query	:SYSTem:COUNter?
response	*A$* = OFF, DATE, NAME
	NAME$ = counter name (ten characters)
	B = counter value (0 to 9999)

Explanation Sets the counter print.
Returns the current counter print setting as character data.
NAME$* is effective only when *A$ is set to NAME.

Example

```
:SYSTem:COUNter OFF
```

The counter print is disabled.

```
:SYSTem:COUNter DATE, 100
```

Prints the counter and date. ('98-2-20-100)

```
:SYSTem:COUNter NAME, "counter name", 100
```

Prints the counter name and counter. (counter name-100)

When allowed In all functions.

■ Sets and queries the output destination by the COPY key. Except 8720

Syntax

command :SYSTem:COPY *A$ (,B$)*
query :SYSTem:COPY?
response *A$, B$*
A$ = IN_PRinter, EX_PRinter, FD, PC, COM (8835 (-01))
IN_PRinter, EX_PRinter, FD, PC, COM, SCSI, MO (8826, 8841, 8842)
B$ = ESCP, RASTer

Explanation

Sets the output destination by the COPY key.
Returns the current setting of the output destination by the COPY key as character data.
IN_PRinter: Prints on the internal printer.
EX_PRinter: Prints on the external printer.
FD: Stores the screen data on floppy disk.
PC: Stores the screen data on PC card.
COM: Sends the screen data to interface.
SCSI: Stores the screen data on SCSI device.
MO: Stores the screen data on MO disk.
When the output destination is set to the external printer, select the control code.
ESCP: Uses ESC/P as a control code.
RASTer: Uses ESC/P raster as a control code.

Example

```
:SYSTem:COPY IN_PRinter
```

Prints on the internal printer.

When allowed

In all functions.

■ Sets and queries the bit map file color. Except 8720

Syntax

command :SYSTem:BMPColor *A$*
query :SYSTem:BMPColor?
response *A$*
A$ = COLOR, GRAY, MONO, MONO_R

Explanation

Sets the hard copy color.
Returns the hard copy color setting as character data.

Note

When printing on the external printer or LAN, only COLOR or MONO can be selected.

Example

```
:SYSTem:BMPColor COLOR
```

Sets the hard copy color to color.

When allowed

When the output destination is set to other than the internal printer in the previous command.

■ Sets and queries filenames of stored bitmaps.

8835-01,8826,8841,8842,8720

Syntax

command	:SYSTem:BMPFile *'NAME$'*
query	:SYSTem:BMPFile?
response	*'NAME$'*

NAME$ = File name (8 characters)

Explanation Sets the file name of a stored bitmap. (Filename: not longer than 8 characters)
Returns file name setting as text for currently stored bitmap.

Example

```
:SYSTem:BMPFile 'BMP'
```

Sets filename BMP for storing a bitmap file.

When allowed In all functions.

■ Sets and queries the output destination by the PRINT key. Except 8720

Syntax

command	:SYSTem:PRINt *A$ (,B$)*
query	:SYSTem:PRINt?
response	*A$, B$*

A$ = IN_PRinter, EX_PRinter, LAN
B$ = ESCP, RASTer

Explanation Sets the output destination by the PRINT key.
When the output destination is set to the external printer, select the control code.
Returns the current output destination by the PRINT key as character data.

Example

```
:SYSTem:PRINt EX_PRinter
```

Prints on the external printer.

When allowed In all functions.

■ Sets and queries the print color. Except 8720

Syntax

command	:SYSTem:PRIColor *A$*
query	:SYSTem:PRIColor?
response	*A$*

A$ = COLOR, MONO

Explanation Sets the print color of the external printer or LAN.
Returns the current print color of the external printer as character data.

Example

```
:SYSTem:PRIColor COLOR
```

Prints in color.

When allowed When the output destination is set to the external printer in the previous command.

■ Sets and queries the SCSI interface ID number.

8826, 8841, 8842, 8720

Syntax

command	:SYSTem:SCSI *A$, B*
query	:SYSTem:SCSI? *A$*
response	*A$, B* <NR1>

A$ = 8826, 8841, 8842 (unit ID), SCSI (target ID)
B = 0 to 7

Explanation Set the ID number of the unit or SCSI device on the SCSI bus.
Returns as an NR1 numerical value the setting for the ID number of the unit or SCSI device on the SCSI bus.

Note The ID number of the internal MO drive (optional) is set to 4. Therefore, if the internal MO drive has been mounted, the ID number 4 cannot be specified.
Do not set the unit ID and target ID to the same number.

Example
```
:SYSTem:SCSI 8826, 7
```
Sets the SCSI ID of the 8826 to 7.

When allowed In all functions.

■ Sets the calendar date, and queries the current calendar date.

Common

Syntax

command	:SYSTem:DATE *A, B, C*
query	:SYSTem:DATE?
response	*A, B, C* all <NR1>

A = 0 to 99 (year)
B = 1 to 12 (month)
C = 1 to 31 (day)

Explanation Sets the date on the internal calendar.
Returns the current date.

Example
```
:SYSTem:DATE 97, 7, 22
```
Sets the internal calendar to July 22nd, 1997.

When allowed In all functions.

■ Sets the time, and queries the current time. Common

Syntax

command :SYSTem:TIME *A, B*
query :SYSTem:TIME?
response *A, B, C* all <NR1>
A = 0 to 23 (hour)
B = 0 to 59 (min)
C = 0 to 59 (second)

Explanation

Sets the time.
Returns the current time.

Example

:SYSTem:TIME 10, 0
Sets the internal clock to 10:00.

When allowed

In all functions.

■ Clearing waveform data. Common

Syntax

command :SYSTem:DATAClear

Explanation

Clear the waveform data.

When allowed

In all functions.

■Sets and queries the printer density of each waveform display color.
8826

Syntax

command :SYSTem:WAVEDensity *A$, B$*
query :SYSTem:WAVEDensity? *A$*
response *A$, B$*
A$ = C1 to C12
B$ = DARK
MIDDark (semi-dark)
NORMal
LIGHt

Explanation

Sets the printer density (line type) of each waveform color (1 to 12).
Returns the printer density of each waveform color.

Example

:SYSTem:WAVEDensity C1, DARK
The printer density of waveform color 1 is set to DARK.
The waveform of the channel set to waveform color 1 is printed dark.

When allowed

In all functions.

■ Sets and queries the external terminals. 8835-01

Syntax

command	:SYSTem:EXTterm *A$*
query	:SYSTem:EXTterm?
response	*A$*

A$ = PRINt,SMPL

Explanation Selects whether the external terminal is to be used for external printing or external sampling.
Returns the current setting of the external terminals.

Example **:SYSTem:EXTterm PRINT**
Selects external terminal for use as printing terminal.

When allowed In all functions.

9. SCALing command (Sets and queries scaling.)

:SCALing

■ Sets and queries the scaling function. Common

Syntax

command	:SCALing:KIND *A$*
query	:SCALing:KIND?
response	*A$*

A$ = POINT, RATIO

Explanation Sets the scaling type according to a character string.
Returns the current scaling type setting as a character string.

Example **:SCALing:KIND POINT**
The 2-point scaling is used.

When allowed In all functions.

■ Enables and disables, and queries the scaling function. Common

Syntax

command	:SCALing:SET *ch$, A$*
query	:SCALing:SET? *ch$*
response	*ch$, A$*

A$ = OFF, SCI, ENG

Explanation Enables or disables the scaling function for the channel designated by *ch$*.
Returns the current state of enablement of the scaling function for the channel designated by *ch$* as a character string.

Example **:SCALing:SET CH1, ENG**
Sets the scaling function for channel 1 to ENG.

When allowed In all functions.

■ Sets and queries the scaling conversion value. Common

Syntax

command	:SCALing:VOLT *ch$, A*
query	:SCALing:VOLT? *ch$*
response	*ch$, A* <NR3>
	A = -9.999E+9 to +9.999E+9

Explanation Sets the scaling conversion value for the channel designated by *ch$*. Returns the current scaling conversion value setting for the channel designated by *ch$* as an NR3 numerical value.

Example
```
:SCALing:VOLT CH1, +2. 0E-3
```
Sets the scaling conversion value (eu/V) for channel 1 to +2. 0E-3.

When allowed In all functions, when the conversion scaling is set.

■ Sets and queries the scaling offset. Common

Syntax

command	:SCALing:OFFSet *ch$, A*
query	:SCALing:OFFSet? *ch$*
response	*ch$, A* <NR3>
	A = -9.999E+9 to +9.999E+9

Explanation Sets the scaling offset for the channel designated by *ch$*. Returns the current scaling offset for the channel designated by *ch$* as an NR3 numerical value.

Example
```
:SCALing:OFFSet CH1, +1. 0E-3
```
Sets the scaling offset (eu offset) for channel 1 to +1. 0E-3.

When allowed In all functions, when the conversion scaling is set.

■ Sets and queries the scaling unit. Common

Syntax

command	:SCALing:UNIT *ch$*, '*A$*'
query	:SCALing:UNIT? *ch$*
response	*ch$*, '*A$*'

A$ = scaling unit (up to 7 characters)

Explanation Sets the scaling unit for the channel designated by *ch$* (up to 7 characters allowed).
Entry of the special characters is as follows:
(Characters other than the following are replaced by spaces.)

^2 (^{2})	^3 (^{3})	~c ($^\circ$)	~e (ε)	~u (**μ**)	~o (**Ω**)

Returns the current scaling unit for the channel designated by *ch$* as character data.
Double quotation marks (") can be used instead of single quotation marks (').

Example
```
:SCALing:UNIT CH1, 'mA'
```
Sets the scaling unit for channel 1 to milliamps.

When allowed In all functions.

■ Sets and queries the scaling VOLT UP and LOW. Common

Syntax

command	:SCALing:VOUPLOw *ch$, B, C*
query	:SCALing:VOUPLOw? *ch$*
response	*ch$, B, C* <NR3>,

B, C = -9.9999E+29 to +9.9999E+29

Explanation Sets the scaling VOLT UP and VOLT LOW values for the channel designated by *ch$*.
Returns the current scaling VOLT UP and VOLT LOW values for the channel designated by *ch$* as an NR3 numerical value.

Example
```
:SCALing:VOUPLOw ch1, +2.0E-1, 0
```
Sets the values of the two points preceding conversion.

When allowed In all functions, when the 2-point scaling is set.

■ Sets and queries the scaling SCALE UP and LOW. Common

Syntax

command :SCALing:SCUPLOw *ch$, B, C*
query :SCALing:SCUPLOw? *ch$*
response *ch$, B, C* <NR3>
B, C = -9.9999E+29 to +9.9999E+29

Explanation Sets the scaling SC UP and SC LOW values for the channel designated by *ch$*.
Returns the current scaling SC UP and SC LOW values for the channel designated by *ch$* as an NR3 numerical value.

Example

```
:SCALing:SCUPLOw ch1, 1.0E+1, 0
```

Sets the converted values of the two points.

When allowed In all functions, when the 2-point scaling is set.

10. COMMent command (Sets and queries comments.)

:COMMent

■ Enables and disables, and queries title comments, and inputs comment characters. Common

Syntax

command :COMMent:TITLe *A$*, '*B$*'
query :COMMent:TITLe?
response *A$*, '*B$*'
A$ = OFF, SETTing, COMMent, S_C (setting &comment)
B$ = comment characters (up to 40 characters)

Explanation Enables and disables comments, and inputs a string of comment characters.
Entry of the special characters is as follows:
(Characters other than the following are replaced by spaces.)

^2 (^{2})	^3 (^{3})	~c (°)	~e (ε)	~u (**μ**)	~o (**Ω**)

Comments may be omitted.
Returns the current enablement state of title comments, and the characters of the comment if any, as character data.
Double quotation marks (") can be used instead of single quotation marks(').

Example

```
:COMMent:TITLe COMMent, 'HIOKI 8835'
```

Inputs "HIOKI 8835" as a title comment.

When allowed In all functions.

■ Enables and disables, and queries, comments for all channels.

Common

Syntax

command :COMMent:EACHch *(ch$,) A$* (*ch$* is omitted for analog.)
query :COMMent:EACHch?
response *A$*
A$ = OFF, SETTing, COMMent, S_C (analog)
OFF, ON (logic)
ch$ = CHA to CHD (8835 (-01), 8841, 8842, 8720),
CHA to CHH (8826)

Explanation

Enables and disables comments for all channels.
Returns the current enablement state of comments for all channels as character data.
Double quotation marks (") can be used instead of single quotation marks(').

Example

```
:COMMent:EACHch COMMent
```

Prints the comments for analog channels on the recording paper.

When allowed In all functions.

■ Setting and queries comment characters for each channel. Common

Syntax

command :COMMent:CH *ch$, (NO$,) 'A$'*
query :COMMent:CH? *ch$ (,NO$)*
response *ch$, (NO$,) 'A$'*
ch$ = CH1 to CH4, CHA to CHD (8835)
CH1 to CH8, CHA to CHD (8835-01)
CH1 to CH32, CHA to CHH (8826)
CH1 to CH16, CHA to CHD (8841, 8842, 8720)
NO$ = NO1 to NO4 (logic only, omitted for analog)
A$ = comment characters (up to 40 characters)

Explanation

Sets a string of comment characters for the channel specified by *ch$*
Entry of the special characters is as follows:
(Characters other than the following are replaced by spaces.)

^2 (^{2})	^3 (^{3})	~c ($^\circ$)	~e (ε)	~u (**μ**)	~o (**Ω**)

Comments may be omitted.
Returns a string of comment characters for the channel specified by *ch$* as character data.
Double quotation marks (") can be used instead of single quotation marks (').

Example

```
:COMMent:CH CH1, 'ch1 = TEST'
```

Sets the comment display for channel 1 to "ch1 = TEST".

When allowed In all functions.

11. CALCulate command (Calculation setting and querying)

:CALCulate

■ Sets and queries waveform parameter calculation. Common

Syntax

command	:CALCulate:MEASure *A$*
query	:CALCulate:MEASure?
response	*A$*

A$ = OFF, ON, EXEC (execute)

Explanation Sets the waveform parameter calculation.
Returns the current setting of the waveform parameter calculation as character data.
Only valid when execution (EXEC) is enabled.

Example

```
:CALCulate:MEASure ON
```

Sets the waveform parameter calculation to ON.

When allowed In MEM.

■ Sets and queries printing calculation results. Except 8720

Syntax

command	:CALCulate:MEASPrint *A$*
query	:CALCulate:MEASPrint?
response	*A$*

A$ = OFF, ON

Explanation Sets printing waveform parameter calculation results.
Returns the setting for printing waveform parameter calculation results as character data.

Example

```
:CALCulate:MEASPrint ON
```

Sets printing waveform parameter calculation results to ON.

When allowed In MEM.

■ Sets and queries storing a calculation result. Common

Syntax

command :CALCulate:MEASFsave *A$*
query :CALCulate:MEASFsave?
response *A$*
A$ = OFF, FD, PC (8835 (-01))
OFF, FD, PC, SCSI, MO (8826, 8841, 8842, 8720)

Explanation Sets the store destination of a waveform parameter calculation result. Returns the current store destination of a waveform parameter calculation result as character data.

Example

```
:CALCulate:MEASFsave FD
```

Saves a calculation result on a floppy disk.

When allowed In MEM.

■ Sets and queries waveform parameter calculations. Common

Syntax

command :CALCulate:MEASSet *NO$, A$, ch$*
:CALCulate:MEASSet *NO$, A$, ch1$, ch2$* (XYAREA: 8826, 8841, 8842)
query :CALCulate:MEASSet? *NO$*
response *A$, ch$*
A$, ch1$, ch2$ (XYAREA: 8826, 8841, 8842)
NO$ = NO1 to NO4
A$ = OFF
AVE : average value (except 8720)
RMS : effective value (except 8720)
PP : peak value
MAX : maximum value
MAXT : time to maximum value
MIN : minimum value
MINT : time to minimum value
PERI : period (except 8720)
FREQ : frequency (except 8720)
RISE : rise time (except 8720)
FALL : fall time (except 8720)
STD : standard deviation (except 8720)
AREA : area value (except 8720)
XYAREA : X-Y area value (except 8720)
ch$, ch1$, ch2$ = CH1 to CH4, ALL (8835)
CH1 to CH8, ALL (8835-01)
CH1 to CH32, ALL (8826)
CH1 to CH16, ALL (8841, 8842, 8720)
(when *A$* is set to other than OFF)

Explanation Sets the channel and the calculation item of the waveform parameter calculation designated by *NO$*.
Returns the channel and the calculation item of the waveform parameter calculation designated by *NO$*.

Example **:CALCulate:MEASSet NO1, MAX, CH1**
Sets the calculation to be of the maximum value on channel 1 for the calculation NO1.

When allowed In MEM.

■ Queries result of waveform parameter calculation. Common

Syntax
query :CALCulate:ANSWer? *NO$, ch$*
:CALCulate:ANSWer? *NO$* (XYAREA: 8826, 8841, 8842)
response *A$, B* <NR 3>
NO$ = NO1 to NO4
A$ = OFF, AVE, RMS, PP, MAX, MAXT, MIN, MINT, AREA, PERI, FREQ, RISE, FALL, STD, XYAREA, NONE
B = calculation result (*A$* = except NONE)

Explanation Returns the calculation result for the waveform parameter calculation item and result specified by *NO$* and *ch$*.
When *A$* is "NONE", there is no calculation result.

Example
query **CALCulate:ANSWer? NO1, CH1**
response **CALCulate:ANSWer MIN, -1.2345E-2** (HEADER ON)
Queries the calculation result of NO1 for the channel 1.

When allowed In MEM.

■ Enables and disables, and queries decision for waveform parameter calculation. 8835 (-01) A, 8826, 8841, 8842, 8720

Syntax
command :CALCulate:COMP *NO$, A$*
query :CALCulate:COMP? *NO$*
response *NO$, A$*
NO$ = NO1 to NO4
A$ = OFF, ON

Explanation Enables and disables the decision for the waveform parameter calculation.
Returns, as character data, the current enablement state of the decision for the waveform parameter calculation.

Example **:CALCulate:COMP NO1, ON**
Sets the decision of the calculation result of NO1 to ON.

When allowed In MEM.

■ Sets and queries upper and lower limits for decision for waveform parameter calculation. 8835 (-01) A, 8826, 8841, 8842, 8720

Syntax

command :CALCulate:COMPArea *NO$, upper, lower*
query :CALCulate:COMPArea? *NO$*
response *NO$, upper* <NR3>, *lower* <NR3>
NO$ = NO1 to NO4
upper, lower = -9.9999E+29 to +9.9999E+29

Explanation

Sets the upper limit and the lower limit for the decision for the waveform parameter calculation designated by *NO$*.
Returns the settings of the upper limit and the lower limit for the decision for the waveform parameter calculation designated by *NO$* as NR3 numerical values.

Example

```
:CALCulate:COMPArea NO1, +1.000E+0, -1.000E+0
```

Sets the decision value for the waveform parameter calculation NO1 to be in the range -1.000E+0 < NO1 < +1.000E+0

When allowed In MEM.

■ Sets and queries waveform processing calculation. 8835 (-01) A, 8826, 8841, 8842

Syntax

command :CALCulate:WVCALc *A$*
query :CALCulate:WVCALc?
response *A$*
A$ = OFF, ON, EXEC (execute)

Explanation

Sets the waveform processing calculation.
Returns the current setting of the waveform processing calculation as character data.
Only valid when execution (EXEC) is enabled.

Example

```
:CALCulate:WVCALc ON
```

Sets the waveform processing calculation to ON.

When allowed In MEM.

■ Sets and queries the stop mode. 8835-01, 8826, 8841, 8842

Syntax

command :CALCulate:COMPStop *A$*
query :CALCulate:COMPStop?
response *A$*
A$ = GO, NG, G_N

Explanation

Sets the stop condition for the judgement.
Returns the current setting of the stop condition as character data.

Example

```
:CALCulate:COMPStop NG
```

Sets the stop condition for the judgement to NG.

When allowed In MEM.

■ Queries the result of the judgement. 8835-01, 8826, 8841, 8842

Syntax

query :CALCulate:COMPJudge? *NO$, ch$*
response *A$*
A$ = GO, NG, *
NO$: NO1 to NO4

Explanation Returns the result of the waveform parameter calculation designated by *NO$* and *ch$* as character data.

When allowed In MEM.

■ Sets and queries the waveform processing calculation equation.
8835 (-01) A, 8826, 8841, 8842

Syntax

command :CALCulate:Z *Z$*, "*A$*"
query :CALCulate:Z? *Z$*
response *Z$*, "*A$*"
Z$ = Z1 to Z16
A$ = calculation equation (up to 80 characters, alphabets in small letter, operator in capital letter)

Explanation Sets the waveform processing calculation equation.
Single quotation marks(') can be used instead of double quotation marks (").
Returns the setting of the waveform processing calculation equation as character data.

Example

```
:CALCulate:Z Z1 'a+b+ABS(CH1)'
```

Sets up the calculation equation for Z1 to be Z1 = a+b+ABC(CH1)

When allowed In MEM.

■ Sets and queries coefficients a to p. 8835 (-01) A, 8826, 8841, 8842

Syntax

command :CALCulate:FACTor *A$*, *B*
query :CALCulate:FACTor? *A$*
response *A$*, *B* <NR3>
A$ = A to P
B = -9.9999E+29 to +9.9999E+29

Explanation Sets the one of the coefficients which is designated by *A$*.
Returns as an NR3 numerical value the current setting of that one of the coefficients which is designated by *A$*.

Example

```
:CALCulate:FACTor A, +1.234E+1
```

Sets the coefficient a to be equal to +1.234E+1

When allowed In MEM.

■ Sets and queries the display channel for the calculated result.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CALCulate:ZDIsplay *Z$, ch$, A$*
query :CALCulate:ZDIsplay? *Z$*
response *Z$, ch$ (,A$)*
Z$ = Z1 to Z16
ch$ = CH1 to CH32, NONE
A$* = MANUal, AUTO (when *ch$ is set to CH1 to CH32)

Explanation

Displays the calculated result of the calculation equation for *Z$* on the channel designated by *ch$*.
When *A$* is MANUal, displays within upper and lower limits on the variable screen. (When scaling, displays in its unit.)
Returns the currently set display channel of the calculated result of the calculation equation for *Z$* .

Example

```
:CALCulate:ZDIsplay Z1, ch1, MANUal
```

Displays the calculated result of the waveform processing calculation equation for Z1 on channel 1. Displays the range between upper and lower limits for the channel 1 on the variable screen.

When allowed

In MEM.

■ Sets the moving averaging.

8835 (-01) A, 8826, 8841, 8842

Syntax

command :CALCulate:MOVE *Z$*, *A*
query :CALCulate:MOVE? *Z$*
response *Z$*, *A* <NR1>
Z$ = Z1 to Z16
A = 0 to 4000 <NR1>

Explanation

Sets the moving averaging for the calculation designated by *Z$*.
Returns as an <NR1> numerical value the current setting of the value of the moving averaging for the calculation designated by *Z$*.

Example

```
:CALCulate:MOVE Z1, 200
```

Sets the moving averaging of Z1 equation to 200.

When allowed

In MEM.

■ Sets the parallel movement. 8835 (-01) A, 8826, 8841, 8842

Syntax

command	:CALCulate:SLIDe *Z$*, *A*
query	:CALCulate:SLIDe? *Z$*
response	*Z$*, *A* <NR1>

Z$ = Z1 to Z16
A = -4000 to 4000 <NR1>

Explanation Sets the parallel movement for the calculation designated by *Z$*.
Returns as an <NR1> numerical value the current setting of the value of the parallel movement for the calculation designated by *Z$*.

Example **:CALCulate:SLIDe Z1, 200**
Sets the parallel movement of Z1 equation to 200.

When allowed In MEM.

12. FDISK command (Setting and querying relating to the file)

:FDISK

■ Sets and queries the media type. Common

Syntax

command	:FDISK:MEDIA *A$*
query	:FDISK:MEDIA?
response	*A$*

A$ = FD, PC (8835 (-01))
FD, PC, SCSI, MO (8826, 8841, 8842, 8720)

Explanation Sets the media type.
Returns the current media type as character data

Example **:FDISK:MEDIA FD**
Floppy disk media are used.

When allowed Providing that measurement operation is not taking place.

■ Saves a file. Common

Syntax command :FDISK:SAVE '*NAME1$. NAME2$*', *A$, B$ (,C$)*
:FDISK:SAVE '*NAME1$. NAME2$*', *A$* (when *A$* = Set, Area or in the FFT function)

NAME1$ = file name (8 characters)
NAME2$ = extension (3 characters)
A$ = type of file
Bin: binary data
Text: text data
Set: settings
Area: waveform decision area
A$ = type of file (During memory segmentation or in the R&M function)
BAll: binary data (All blocks (all waveforms) are saved.)
BOne: binary data (One block (the displayed waveform) is saved.)
TAll: text data (All blocks (all waveforms) are saved.)
TOne: text data (One block (the displayed waveform) is saved.)
* In the R&M function
BAll, TAll: Both the MEM and REC waveforms are saved simultaneously.
BOne, TOne: Only the waveform in the display function is saved.
B$ = saved channels
ALL, CH1 to CH4, LOGIC (8835)
ALL, CH1 to CH8, LOGIC (8835-01)
ALL, CH1 to CH32, LOGIC (8826)
ALL, CH1 to CH16, LOGIC (8841, 8842, 8720)
C$ = degree of thinning (text only)
OFF, 1_2 to 1_1000

Explanation Saves the information specified by *A$*. If an attempt is made to save to a filename that already exists, an execution error is generated.
Double quotation marks (") can be used instead of single quotation marks (').

Example **:FDISK:SAVE 'TEST. DAT', Bin, ALL**
Saves all channels of measurement data under the file name 'TEST. DAT'.

When allowed Providing that measurement operation is not taking place.

■ Loads a file. Common

Syntax

command :FDISK:LOAD *NO (,A$)* (File number)
:FDISK:LOAD '*NAME1$. NAME2$*' *(,A$)* (File name)
NO = file number
A$ = NEW, ADD
NAME1$ = file name (8 characters)
NAME2$ = extension (3 characters)

Explanation

Loads the data in the file numbered *NO*. Or loads the data of the specified file name.
When loading the waveform data, "new load (NEW)" or "overwrite load (ADD)" can be set. (Default is NEW if omitted.)

Example

```
:FDISK:LOAD 1, NEW
```
Loads the waveform data of the file numbered 1.
```
:FDISK:LOAD 'TEST.MEM', NEW
```
Loads a file called TEST.MEM.

When allowed

Providing that measurement operation is not taking place.

■ Queries information about a file or directory. Common

Syntax

query :FDISK:INFOr? *NO*
response *NO* <NR1>, "*NAME$*", "*DATE$*", "*TIME$*", *A* <NR1>, *B$*, *C$*, *D* <NR1>, "*TDATE$*", "*TTIME$*"
NO : file or directory number
NAME$: file name
DATE$: date of save
TIME$: time of save
A : size of file (bytes)
B$: function (MEM, REC, RMS)
C$: measurement contents (WAVE, SET)
D : recording length
TDATE$: year/month/day of trigger
TTIME$: trigger time

Explanation

Returns information about the file numbered *NO*.
If the file cannot be read, returns: *NO*, *"NAME$"*, *"DATE$"*, *"TIME$"*, A, 0
Double quotation marks (") can be used instead of single quotation marks (').

When allowed

Providing that measurement operation is not taking place.

■ Deletes a file or directory. Common

Syntax command :FDISK:DELEte *NO* (Number)
:FDISK:DELEte '*NAME1$*. *NAME2$*' (Name)
NO = file or directory number
NAME1$ = file or directory name (8 characters)
NAME2$ = extension (3 characters)

Explanation Deletes the file or directory numbered *NO*. Or deletes the file or directory of the specified name.

Example
```
:FDISK:DELEte 1
```
Deletes the file (directory) numbered 1.

When allowed Providing that measurement operation is not taking place.

■ Formats media. Common

Syntax command :FDISK:FORMat *(A$)*
A$ = 2HD (1.2 MB)
2HC (1.44 MB)
2DD (720 KB)

Explanation Formats media.
If a floppy disk is selected, select the format type.

Example
```
:FDISK:FORMat 2HD
```
Formats in 2HD (1.2 M-byte) format.

When allowed Providing that measurement operation is not taking place.

■ Creates a directory. Common

Syntax command :FDISK:MKDIR '*NAME$*'
NAME$ = subdirectory name

Explanation Creates a subdirectory in the current directory on the media.
Double quotation marks (") can be used instead of single quotation marks (').

Example
```
:FDISK:MEDIA FD
:FDISK:MKDIR 'TEST'
```
Creates a subdirectory called TEST on the floppy disk.

When allowed Providing that measurement operation is not taking place.

■ Changes the current directory. Common

Syntax
command :FDISK:CHDIR *NO*
NO = file number (directory)

Explanation Changes the current directory to the directory numbered *NO* on the media.

When allowed Providing that measurement operation is not taking place.

■ Queries the number of files. Common

Syntax
query :FDISK:FILE?
response *A* <NR1>
A = number of files

Explanation Returns the total number of files which are currently saved as an NR1 numerical value.

When allowed Providing that measurement operation is not taking place.

■ Queries the filename. Common

Syntax
query :FDISK:NINFor? *NO*
response *NO*, "*NAME$*", *A$*
NO = file number
NAME$ = name of the file
A$ = FILE (file)
DIR (directory)

Explanation Returns the filename numbered *NO* as character data.

Example
query **:FDISK:NINFor? 1**
response **:FDISK:NINFor 1, "TEST. DAT", FILE**

When allowed Providing that measurement operation is not taking place.

■ Queries the current directory. Common

Syntax
query :FDISK:DIR?
response *A$*
A$ = directory name

Explanation Returns the current directory name (with the pass) on the media as character data.

When allowed Providing that measurement operation is not taking place.

■ Queries the allowable number of bytes. Common

Syntax
query :FDISK:FREE?
response *A* <NR1>
A = allowable number of bytes

Explanation Returns the allowable number of bytes for the floppy disk as an NR1 numerical value.

When allowed Providing that measurement operation is not taking place.

13. GRAPh Command (Commands relating to graphics editor)

:GRAPh

■ Enables and disables, and queries the enablement of the graphics editor. 8835 (-01) A, 8826, 8841, 8842

Syntax
command :GRAPh:EDIT *A$*
query :GRAPh:EDIT?
response *A$*
A$ = OFF, ON

Explanation Enables and disables the graphic editor mode.
Returns whether or not the graphic editor mode is enabled as character data.

Example `:GRAPh:EDIT ON`
Sets the graphic editor mode to ON.

When allowed In MEM in SINGLE, XY SINGLE format and in FFT in SINGLE, Nyquist format.

■ Paints the drawing. 8835 (-01) A, 8826, 8841, 8842

Syntax
command :GRAPh:PAINT *X*, *Y*
X = x-coordinate
Y = y-coordinate

Explanation Begins solid fill from the point specified by (X, Y).
Refer to the :GRAPh:LINE command for details of X and Y.

When allowed In MEM and FFT, when in the editor mode.

■ Parallel movement 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh: PARAllel *high, low, right, left*

high, low, right, left = 0 to 10.00 (div)
(8835: 0 to 5.00 div)

Explanation Carries out a parallel movement of the drawing.
The *high* and *low* parameters and the *right* and *left* parameters are set in units of 0.05 steps.

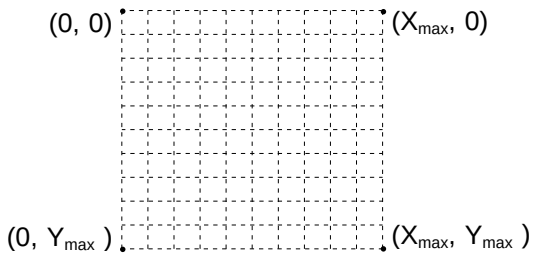
When allowed In MEM and FFT, when in the editor mode.

■ Draws a line. 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh: LINE *X1, Y1, X2, Y2*

X1, X2 = x-coordinates
Y1, Y2 = y-coordinates

Explanation Draws a line from (*X1, Y1*) to (*X2, Y2*).



Display format	X_{max}	Y_{max}
8835		
MEM (SINGLE)	500	400
MEM (XY)	400	400
FFT	400	400
8826		
MEM (SINGLE)	500	479
MEM (XY SINGLE)	480	479
FFT	400	400
8841, 8842		
MEM (SINGLE)	375	639
MEM (XY SINGLE)	320	320
FFT	400	400

Example `:GRAPH:LINE 10,20,100,200`

Draws a line from (10, 20) to (100, 200).

When allowed In MEM and FFT, when in the editor mode.

■ Erases the line. 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh:ERASe *X1, Y1, X2, Y2*

X1, X2 = x-coordinates
Y1, Y2 = y-coordinates

Explanation Erases the line from (*X1, Y1*) to (*X2, Y2*).
Refer to the :GRAPh:LINE command for details of X and Y.

When allowed In MEM and FFT, when in the editor mode.

■ Loads a waveform into the editor. 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh:STORage

Explanation Loads a waveform into the editor.

When allowed In MEM and FFT, when in the editor mode.

■ Reverses the video of the drawing. 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh:REVErse

Explanation Reverses the video of the drawing.

When allowed In MEM and FFT, when in the editor mode.

■ Clears all drawing. 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh:ALLClear

Explanation Clears the entire drawing.

When allowed In MEM and FFT, when in the editor mode.

■ Clears drawing. 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh: CLEAr *X1, Y1, X2, Y2*
X1, X2 = x-coordinates
Y1, Y2 = y-coordinates

Explanation Clears the rectangle with the points (*X1, Y1*) and (*X2, Y2*) at diagonally opposite corners.
Refer to the :GRAPh:LINE command for details of X and Y.

When allowed In MEM and FFT, when in the editor mode.

■ Undoes the drawing. 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh:UNDO

Explanation Reverses the effect of the immediately previous editor command.

When allowed In MEM and FFT, when in the editor mode.

■ Saves the drawing (decision area). 8835 (-01) A, 8826, 8841, 8842

Syntax command :GRAPh:SAVE

Explanation Saves the decision area created with the editor in the internal memory**.**

When allowed In MEM and FFT, when in the editor mode.

Chapter 4
Example Programs

4.1 Visual Basic Example Programs

4.1.1 GP-IB Example Programs

NOTE The following example programs are for the National Instruments GP-IB board (including the PC card).

Example 1 Using a setting command

Send the command in the format specified, when the conditions for the command to be acceptable are met.

```
Private Sub Sample1_Click()
'*****************************************************************************
'  GPIB SAMPLE PROGRAM NO.1
'*****************************************************************************

    Adr = 5     'GP-IB Address = 5

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":FUNCTION MEM")
    Call ibwrt(rec%, ":CONFIGURE:TDIV +1.E-3")
    Call ibwrt(rec%, ":CONFIGURE:SHOT 20")

    Call ibwrt(rec%, ":TRIGGER:SOURCE OR")
    Call ibwrt(rec%, ":TRIGGER:KIND CH1,LEVEL")
    Call ibwrt(rec%, ":TRIGGER:PRETRIG 5")
    Call ibwrt(rec%, ":TRIGGER:LEVEL CH1,2")
    Call ibwrt(rec%, ":TRIGGER:SLOPE CH1,UP")
    Call ibwrt(rec%, ":TRIGGER:KIND CH2,OFF")
    Call ibwrt(rec%, ":TRIGGER:KIND CH3,OFF")
    Call ibwrt(rec%, ":TRIGGER:KIND CH4,OFF")

    Call ibwrt(rec%, ":START")

    Call ibonl(rec%, 0)
End Sub
```

Example 2 Using a query

- Send the query in the format specified, when the conditions for the query to be acceptable are met.
  Next switch the unit to be the talker, and receive the output data.
- The response data from the query is returned in the format specified for the corresponding command.

```
Private Sub Sample2_Click()
'************************************************************************
'  GPIB SAMPLE PROGRAM NO.2
'************************************************************************
    Adr = 5     'GP-IB Address = 5

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":HEADER OFF")
    Call ibwrt(rec%, ":FUNCTION?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
    Text1.Text = Left$(rd$, ibcnt% - 1)
    Call ibwrt(rec%, ":SYSTEM:TIME?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
    Text1.Text = Text1.Text & Chr$(13) & Chr$(10) & Left$(rd$, ibcnt% - 1)

    Call ibonl(rec%, 0)
End Sub
```

Example 3 Using service requests

- Using the *SRE and *ESE commands, this program sets the service request response enable, and sets the jump address in the controller for a service request interrupt. It then enables the service request interrupt.
- The response data from the query is returned in the format specified for the corresponding command.

```
Private Sub Sample3_Click()
'*************************************************************************
'  GPIB SAMPLE PROGRAM NO.3
'*************************************************************************
    Adr = 5

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, "HEADER OFF")
    Call ibwrt(rec%, "*SRE 32")
    Call ibwrt(rec%, "*ESE 16")
    Call ibwrt(rec%, "*CLS")
    Call ibwrt(rec%, ":FUNCTION MEM")

    For i = 1 To 16
        gra$ = ":DISPLAY:GRAPH CH1," + LTrim(Str$(i))
        Call ibwrt(rec%, gra$)
        Call ibwait(rec%, RQS Or TIMO)
        If (ibsta% And RQS) Then
            Call err_check (rec)
            Exit Sub
        End If
    Next

    Call ibonl(rec%, 0)
End Sub

Sub err_check(rec As Integer)
    Call ibrsp(rec%, spr%)
    Call ibwrt(rec%, "*ESR?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
    b = Val(rd$)
    If (b And &H4) <> 0 Then Text1.Text = Text1.Text & Chr$(13) & Chr$(10) _
     & "Query Error!"
    If (b And &H8) <> 0 Then Text1.Text = Text1.Text & Chr$(13) & Chr$(10) _
     & "Machine Error!"
    If (b And &H10) <> 0 Then Text1.Text = Text1.Text & Chr$(13) & Chr$(10) _
     & "Execute Error!"
    If (b And &H20) <> 0 Then Text1.Text = Text1.Text & Chr$(13) & Chr$(10) _
     & "Command Error!"

    Call ibonl(rec%, 0)

End Sub
```

Example 4 Outputting stored data

- Using the :MEMORY:MAXPOINT? query, this program checks whether data can be output from memory. If this query returns zero, no data is stored, and it cannot therefore be output.
- Next, the program specifies the channel and point for output, using the :MEMORY:POINT command. As data is input or output, the point is incremented automatically. If capturing data consecutively, it is sufficient to specify the point once only.
- To capture data in ASCII format use the :MEMORY:ADATA? query, and to capture data as voltage values use the :MEMORY:VDATA? query. The number of data samples which may be output in one set is 1 to 80 using :ADATA? and 1 to 40 using the :VDATA? query.
- Outputting data in bigger sets reduces the overall processing time.
- Read data (2000 samples) for channel 1 when stored with a 20-division recording length.

```
Private Sub Sample4_Click()
'*************************************************************************
'  GPIB SAMPLE PROGRAM NO.4
'*************************************************************************
    Adr = 5
    Dim d(2000)

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)        '
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":FUNCTION MEM")
    Call ibwrt(rec%, ":CONFIGURE:SHOT 20")
    Call ibwrt(rec%, ":TRIGGER:MODE SINGLE")
    Call ibwrt(rec%, ":START;:STOP;*OPC?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
    Call ibwrt(rec%, ":HEADER OFF")
    Call ibwrt(rec%, ":MEMORY:MAXPOINT?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
    mx% = Val(rd$)
    If (mx% <> 2000) Then
        Call ibonl(rec%, 0)
        Exit Sub
    End If
    Call ibwrt(rec%, ":MEMORY:POINT CH1,0")
    For i = 0 To 2000
        Call ibwrt(rec%, ":MEMORY:VDATA? 1")

        rd$ = Space$(30)
        Call ibrd(rec%, rd$)
        d(i) = Val(rd$)
    Next
    For i = 0 To 2000
        Text1.Text = d(i)
    Next

    Call ibonl(rec%, 0)
End Sub
```

Example 5 Inputting storage data

- This program prepares storage memory, using the :MEMORY:PREPARE command.
- Next, the program specifies the channel and point for input, using the :MEMORY:POINT command, and then uses the :MEMORY:ADATA command to input data.

```
Private Sub Sample5_Click()
'*************************************************************************
'  GPIB SAMPLE PROGRAM NO.5
'*************************************************************************
    Adr = 5

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":FUNCTION MEM")
    Call ibwrt(rec%, ":CONFIGURE:SHOT 20")
    Call ibwrt(rec%, ":MEMORY:PREPARE")
    Call ibwait(rec%, TIMO)
    Call ibwrt(rec%, ":MEMORY:POINT CH1,0")

    For i = 0 To 2000
        SND$ = ":MEMORY:ADATA " + LTrim(Str$(Int(500 * Sin(3.14 * i / 500))))
        Call ibwrt(rec%, SND$)
    Next
    Call ibonl(rec%, 0)
End Sub
```

Example 6 Start measurement operation mode, and if no trigger is detected execute a STOP.

- Bit 5 (ESB) of the status register is obtained by serial polling.
- Forcible termination by :ABORT results if no trigger is detected when bit 2 (trigger wait completed) is checked by :ESR0?.
- Termination occurs upon verification of bit 1 (start processing completed) by :ESR0?.

```
Private Sub Sample6_Click()
'************************************************************************
'  GPIB SAMPLE PROGRAM NO.6
'************************************************************************
    Adr = 5

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":HEADER OFF")
    Call ibwrt(rec%, "*CLS;*ESE 1")
    Call ibwrt(rec%, ":FUNCTION MEM")
    Call ibwrt(rec%, ":CONFIGURE:TDIV 1.E-3")
    Call ibwrt(rec%, ":CONFIGURE:SHOT 20")
    Call ibwrt(rec%, ":TRIGGER:SOURCE OR")
    Call ibwrt(rec%, ":TRIGGER:KIND CH1,LEVEL;KIND CH2,LEVEL")
    Call ibwrt(rec%, ":TRIGGER:KIND CH3,OFF;KIND CH4,OFF")
    Call ibwrt(rec%, ":TRIGGER:LEVEL CH1,0;SLOPE CH1,UP")
    Call ibwrt(rec%, ":TRIGGER:LEVEL CH2,0;SLOPE CH2,UP")
    Call ibwrt(rec%, ":TRIGGER:MODE SINGLE")
    Call ibwrt(rec%, ":START;*OPC")

    Do
        Call ibrsp(rec%, spr%)
    Loop While ((spr% And &H20) = 0)
    Call ibwait(rec%, TIMO)
    Call ibwrt(rec%, ":ESR0?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
    esr0% = Val(rd$)
    If (esr0% And &H4) = 0 Then
        Call ibwrt(rec%, ":ABORT")
        Call ibonl(rec%, 0)
        MsgBox "NOT TRIGGER"
        Exit Sub
    End If

    Do While ((esr0% And &H2) = 0)
        Call ibwrt(rec%, ":ESR0?")
        rd$ = Space$(30)
        Call ibrd(rec%, rd$)
        esr0% = Val(rd$)
    Loop

    MsgBox "STORAGE END"
    Call ibonl(rec%, 0)
End Sub
```

Example 7 Checking the presence of input unit, and displaying the input ranges on the screen.

```
Private Sub Sample7_Click()
'*************************************************************************
'  GPIB SAMPLE PROGRAM NO.7
'*************************************************************************
    Adr = 5

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":HEADER OFF")
    Call ibwrt(rec%, "*OPT?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
        ch1% = Val(Mid$(rd$, 1, 1))
        ch2% = Val(Mid$(rd$, 3, 1))
        ch3% = Val(Mid$(rd$, 5, 1))
        ch4% = Val(Mid$(rd$, 7, 1))
    Call ibwrt(rec%,":MEMORY:GETREAL")
    If (ch1% <> 0) Then
        Call ibwrt(rec%, ":MEMORY:AREAL? CH1")
        rd$ = Space$(30)
        Call ibrd(rec%, rd$)
        ch1_data$ = "CH1 = " & rd$
    Else
        ch1_data$ = "CH1 = NON"
    End If

    If (ch2% <> 0) Then
        Call ibwrt(rec%, ":MEMORY:AREAL? CH2")
        rd$ = Space$(30)
        Call ibrd(rec%, rd$)
        ch2_data$ = "CH2 = " & rd$
    Else
        ch2_data$ = "CH2 = NON"
    End If

    If (ch3% <> 0) Then
        Call ibwrt(rec%, ":MEMORY:AREAL? CH3")
        rd$ = Space$(30)
        Call ibrd(rec%, rd$)
        ch3_data$ = "CH3 = " & rd$
    Else
        ch3_data$ = "CH3 = NON"
    End If

    If (ch4% <> 0) Then
        Call ibwrt(rec%, ":MEMORY:AREAL? CH4")
        rd$ = Space$(30)
        Call ibrd(rec%, rd$)
        ch4_data$ = "CH4 = " & rd$
    Else
        ch4_data$ = "CH4 = NON"
    End If

    ret$ = Chr$(13) & Chr$(10)
    Text1.Text = ch1_data$ & ret$ & ch2_data$ & ret$ & ch3_data$ & ret$ & ch4_data$

    Call ibonl(rec%, 0)
End Sub
```

Example 8 Outputting stored data (binary data)

- Using the :MEMORY:MAXPOINT? query, this program checks whether data can be output from memory. If this query returns zero, no data is stored, and it cannot therefore be output.
- Next, the program specifies the channel and point for output, using the :MEMORY:POINT command. As data is input or output, the point is incremented automatically. If capturing data consecutively, it is sufficient to specify the point once only.
- After converting the binary data to voltage value obtained by :MEMORY:BDATA?, it is output on the screen. The number of data samples which may be output in one set is 1 to 200 using :BDATA?.

```
Private Sub Sample8_Click()
'*************************************************************************
'  GPIB SAMPLE PROGRAM NO.8
'*************************************************************************
    Adr = 5
    Dim dat(512) As Byte

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":HEADER OFF")
    Call ibwrt(rec%, ":MEMORY:MAXPOINT?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
    sp% = Val(rd$)
    If (sp% = 0) Then
        Call ibonl(rec%, 0)
        Exit Sub
    End If
    Call ibwrt(rec%, ":UNIT:RANGE? CH1")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
    rd$ = Left(rd$, ibcnt - 1)
    range# = Val(rd$)
    Call ibwrt(rec%, ":MEMORY:POINT CH1,0")

    Call ibwrt(rec%, ":MEMORY:BDATA? 200")
    Call ibrd32(rec%, dat(0), 403)
    For i = 1 To 200
        a% = (dat(2 * i) And &H7) - (dat(2 * i) And &H8)
        b% = dat(2 * i + 1)
        c% = a% * 256 + b%
        d% = c% * range% / 160
        Text1.Text = Str$(d#)
    Next

    Call ibonl(rec%, 0)
End Sub
```

Example 9 Saving stored data onto drive A (sequential file)

```
Private Sub Sample9_Click()
'*************************************************************************
'  GPIB SAMPLE PROGRAM NO.9
'*************************************************************************
    Adr = 5

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":HEADER OFF")
    Call ibwrt(rec%, ":MEMORY:MAXPOINT?")
    rd$ = Space$(30)
    Call ibrd(rec%, rd$)
        sp% = Val(rd$)
    If (sp% = 0) Then
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    na$ = "a:\sample.dat"
    Open na$ For Output As #1
    Call ibwrt(rec%, ":MEMORY:POINT CH1,0")
    Print #1, 10
    For i = 0 To 10
        Call ibwrt(rec%, ":MEMORY:ADATA? 1")
        rd$ = Space$(30)
        Call ibrd(rec%, rd$)
        Print #1, Val(rd$)
    Next

    Close #1
    Call ibonl(rec%, 0)
End Sub
```

Example 10 Reading the data saved in Example 9, and loading it into the unit.

```
Private Sub Sample10_Click()
'*************************************************************************
'  GPIB SAMPLE PROGRAM NO.10
'*************************************************************************
    Adr = 5

    Call ibdev(0, Adr, 0, T1s, 1, 0, rec%)
    If (rec% < 0) Then
        MsgBox "Could not open device", 64, "ERROR"
        Call ibonl(rec%, 0)
        Exit Sub
    End If

    Call ibwrt(rec%, ":HEADER OFF")
    Call ibwrt(rec%, ":MEMORY:PREPARE")
    Call ibwait(rec%, TIMO)

    na$ = "a:\sample.dat"
    Open na$ For Output As #1
    Call ibwrt(rec%, ":MEMORY:POINT CH1,0")
    Line Input #1, mx
    For i = 0 To mx
        INPUT #1, dt
        Call ibwrt(rec%, ":MEMORY:ADATA " + LTrim(Str$(dt%)))

    Next

    Close #1
    Call ibonl(rec%, 0)
End Sub
```

4.1.2 RS-232C Example Programs

Example 1 Using a setting command

Send the command in the format specified, when the conditions for the command to be acceptable are met.

```
Private Sub Sample1_Click()
'*****************************************************************************
'  RS232C SAMPLE PROGRAM NO.1
'*****************************************************************************

    deli$ = Chr$(13) & Chr$(10)

    Comm1.PortOpen = True

    Comm1.Output = ":FUNCTION MEM" & deli$
    Comm1.Output = ":CONFIGURE:TDIV 1.E-3" & deli$
    Comm1.Output = ":CONFIGURE:SHOT 20" & deli$

    Comm1.Output = ":TRIGGER:SOURCE OR" & deli$
    Comm1.Output = ":TRIGGER:KIND CH1,LEVEL" & deli$
    Comm1.Output = ":TRIGGER:PRETRIG 5" & deli$
    Comm1.Output = ":TRIGGER:LEVEL CH1,2" & deli$
    Comm1.Output = ":TRIGGER:SLOPE CH1,UP" & deli$
    Comm1.Output = ":TRIGGER:KIND CH2,OFF" & deli$
    Comm1.Output = ":TRIGGER:KIND CH3,OFF" & deli$
    Comm1.Output = ":TRIGGER:KIND CH4,OFF" & deli$

    Comm1.Output = ":START" & deli$

    Comm1.PortOpen = False
End Sub
```

Example 2 Using a query

- Send the query in the format specified, when the conditions for the query to be acceptable are met.
- The response data from the query is returned in the format specified for the corresponding command.

```
Private Sub Sample2_Click()
'*****************************************************************************
'  RS232C SAMPLE PROGRAM NO.2
'*****************************************************************************

    deli$ = Chr$(13) & Chr$(10)

    Comm1.PortOpen = True

    Comm1.Output = ":HEADER OFF" & deli$
    Comm1.Output = ":FUNCTION?" & deli$
    Do
        ans$ = ans$ & Comm1.Input
    Loop Until InStr(1, ans$, Chr$(10))

    Comm1.Output = ":SYSTEM:TIME?" & deli$
    Do
        tm$ = tm$ & Comm1.Input
    Loop Until InStr(1, tm$, Chr$(10))

    Text1.Text = "FUNCTION = " & ans$
    Text1.Text = Text1.Text & "TIME = " & tm$
    Comm1.PortOpen = False
End Sub
```

Example 3 Outputting stored data

- Using the :MEMORY:MAXPOINT? query, this program checks whether data can be output from memory. If this query returns zero, no data is stored, and it cannot therefore be output.
- Next, the program specifies the channel and point for output, using the :MEMORY:POINT command. As data is input or output, the point is incremented automatically. If capturing data consecutively, it is sufficient to specify the point once only.
- To capture data in ASCII format use the :MEMORY:ADATA? query, and to capture data as voltage values use the :MEMORY:VDATA? query. The number of data samples which may be output in one set is 1 to 80 using :ADATA? and 1 to 40 using the :VDATA? query.
- Outputting data in bigger sets reduces the overall processing time.
- Read data (2000 samples) for channel 1 when stored with a 20-division recording length.

```
Private Sub Sample3_Click()
'*****************************************************************************
'  RS232C SAMPLE PROGRAM NO.3
'*****************************************************************************

    deli$ = Chr$(13) & Chr$(10)
    Dim d(2000)

    Comm1.PortOpen = True

    Comm1.Output = ":FUNCTION MEM" & deli$
    Comm1.Output = ":CONFIGURE:SHOT 20" & deli$
    Comm1.Output = ":TRIGGER:MODE SINGLE" & deli$
    Comm1.Output = ":START;*OPC?" & deli$
    Do
        o$ = o$ & Comm1.Input
    Loop Until InStr(1, o$, Chr$(10))
    Comm1.Output = ":HEADER OFF" & deli$
    Comm1.Output = ":MEMORY:MAXPOINT?" & deli$
    Do
        mx$ = mx$ & Comm1.Input
    Loop Until InStr(1, mx$, Chr$(10))
    If (Val(mx$) <> 2000) Then
        Comm1.PortOpen = False
        Exit Sub
    End If
    Comm1.Output = ":MEMORY:POINT CH1,0" & deli$
    For i = 0 To 2000
        Comm1.Output = ":MEMORY:VDATA? 1" & deli$
        Do
            vd$ = vd$ & Comm1.Input
        Loop Until InStr(1, vd$, Chr$(10))
        d(i) = Val(vd$)
    Next
    For i = 0 To 2000
        Text1.Text = d(i)
    Next

    Comm1.PortOpen = False
End Sub
```

Example 4 Inputting storage data.

- This program prepares storage memory, using the :MEMORY:PREPARE command.
- Next, the program specifies the channel and point for input, using the :MEMORY:POINT command, and then uses the :MEMORY:ADATA command to input data.

```
Private Sub Sample4_Click()
'*************************************************************************
'  RS232C SAMPLE PROGRAM NO.4
'*************************************************************************

   deli$ = Chr$(13) & Chr$(10)

    Comm1.PortOpen = True

    Comm1.Output = ":FUNCTION MEM" & deli$
    Comm1.Output = ":CONFIGURE:SHOT 20" & deli$
    Comm1.Output = ":MEMORY:PREPARE;*OPC?" & deli$
    Do
         o$ = Comm1.Input
    Loop Until InStr(1, o$, Chr$(10))
    Comm1.Output = ":MEMORY:POINT CH1,0" & deli$
    For i = 0 To 2000
        Comm1.Output = ":MEMORY:ADATA " & Ltrim(Str$(Int(500 * Sin(3.14 * i / 500)))) &
deli$
    Next

    Comm1.PortOpen = False
End Sub
```

Example 5 Checking the presence of input unit, and displaying the input ranges on the screen.

```
Private Sub Sample5_Click()
'*******************************************************************************
'  RS232C SAMPLE PROGRAM NO.5
'*******************************************************************************

    deli$ = Chr$(13) & Chr$(10)

    Comm1.PortOpen = True

    Comm1.Output = ":HEADER OFF" & deli$
    Comm1.Output = "*OPT?" & deli$
    Do
        op$ = op$ & Comm1.Input
    Loop Until InStr(1, op$, Chr$(10))
    ch1% = Val(Mid$(op$, 1, 1))
    ch2% = Val(Mid$(op$, 3, 1))
    ch3% = Val(Mid$(op$, 5, 1))
    ch4% = Val(Mid$(op$, 7, 1))
    Comm1.Output = ":MEMORY:GETREAL"& deli$

    If (ch1% <> 0) Then
        Comm1.Output = ":MEMORY:AREL? CH1" & deli$
        ar$ = ""
        Do
            ar$ = ar$ & Comm1.Input
        Loop Until InStr(1, ar$, Chr$(10))
        ch1_data$ = "CH1 = " & ar$
    Else
        ch1_data$ = "CH1 = NON"
    End If

    If (ch2% <> 0) Then
        Comm1.Output = ":MEMORY:AREL? CH2" & deli$
        ar$ = ""
        Do
            ar$ = ar$ & Comm1.Input
        Loop Until InStr(1, ar$, Chr$(10))
        ch2_data$ = "CH2 = " & ar$
    Else
        ch2_data$ = "CH2 = NON"
    End If

    If (ch3% <> 0) Then
        Comm1.Output = ":MEMORY:AREL? CH3" & deli$
        ar$ = ""
        Do
            ar$ = ar$ & Comm1.Input
        Loop Until InStr(1, ar$, Chr$(10))
        ch3_data$ = "CH3 = " & ar$
    Else
        ch3_data$ = "CH3 = NON"
    End If

    If (ch4% <> 0) Then
        Comm1.Output = ":MEMORY:AREL? CH4" & deli$
        ar$ = ""
        Do
            ar$ = ar$ & Comm1.Input
        Loop Until InStr(1, ar$, Chr$(10))
        ch4_data$ = "CH4 = " & ar$
    Else
        ch4_data$ = "CH4 = NON"
```

```
    End If

    Text1.Text = ch1_data$ & deli$ & ch2_data$ & deli$ & ch3_data$ & deli$ & ch4_data$ &
deli$

    Comm1.PortOpen = False
End Sub
```

Example 6 Saving stored data onto drive A

```
Private Sub Sample6_Click()
'*************************************************************************
'  RS232C SAMPLE PROGRAM NO.6
'*************************************************************************

    deli$ = Chr$(13) & Chr$(10)
    na$ = "a:\sample.dat"

    Comm1.PortOpen = True

    Comm1.Output = ":HEADER OFF" & deli$
    Comm1.Output = ":MEMORY:MAXPOINT?" & deli$
    Do
        mx$ = mx$ & Comm1.Input
    Loop Until InStr(1, mx$, Chr$(10))
    If (Val(mx$) = 0) Then
        Comm1.PortOpen = False
        Exit Sub
    End If

    Open na$ For Output As #1
    Comm1.Output = ":MEMORY:POINT CH1,0" & deli$
    Print #1, 10
    For i = 0 To 10
        Comm1.Output = ":MEMORY:ADATA? 1" & deli$
        ad$ = ""
        Do
            ad$ = ad$ & Comm1.Input
        Loop Until InStr(1, ad$, Chr$(10))
        Print #1, Val(ad$)
    Next

    Close #1
    Comm1.PortOpen = False
End Sub
```

Example 7 Reading the data saved in Example 6, and loading it into the unit.

```
Private Sub Sample7_Click()
'*****************************************************************************
'  RS232C SAMPLE PROGRAM NO.7
'*****************************************************************************

    deli$ = Chr$(13) & Chr$(10)
    na$ = "a:\sample.dat"

    Comm1.PortOpen = True

    Comm1.Output = ":HEADER OFF" & deli$
    Comm1.Output = ":MEMORY:PREPARE" & deli$
    Do
         o$ = Comm1.Input
    Loop Until InStr(1, o$, Chr$(10))

    Open na$ For Output As #1
    Comm1.Output = ":MEMORY:POINT CH1,0" & deli$
    Input #1, mx
    For i = 0 To mx
        Input #1, dt
        Comm1.Output = ":MEMORY:ADATA " & Str$(dt) & deli$
    Next

    Close #1
    Comm1.PortOpen = False
End Sub
```

Appendix

Appendix 1 IEEE 488.2-1987

The following information relates to the compliance with the IEEE 488.2 standard.

(1) IEEE 488.1 interface functions

These are detailed in "Interface functions" in Section 1.1.2.

(2) Operations with a device address other than 0 through 30

It is not possible to set to other than 0 through 30.

(3) Timing of changed device address recognition

A change of address is recognized immediately after powering on.

(4) Device settings at powering on, including all commands which further restrict the initial setting

The status information is cleared. However, the points specified by the ":MEMORY:POINT" command are all reinitialized, and all other items are preserved.

(5) List of message exchange options

(a) Input buffer capacity and operation

The unit has an input buffer of 1024 bytes capacity. If the data accumulated in this buffer exceeds 1024 bytes the buffer full, and until a space again becomes available in the buffer, the IEEE 488.1 bus goes into the waiting state.

(b) Queries to which multiple response message units are returned

There are no queries to return multiple response messages.

(c) Queries producing responses as syntax checking is performed

All queries produce responses when syntax checking is performed.

(d) Whenever any queries produce responses when read

There are no queries which produce response messages at the instant they are read in by the controller.

(e) Whether any commands are coupled

There are no relevant commands.

(6) Summary of functional elements for use when constructing device specific commands, and whether compound commands or program headers can be used

Program message, program message terminator, program message unit, program message unit separator, command message unit, query message unit, command program header, query program header, program data, character program data, decimal program data, chapter string program data, and compound commands program headers.

(7) Buffer capacity limitations for block data

Block data is not used.

(8) Summary of program data elements used in expressions, and deepest nesting level allowable in sub-expressions, including syntax restrictions imposed by the device

Sub-expressions are not used. Character data and decimal data are the only program data elements used.

(9) Response syntax for queries

Response syntax is detailed in Section 3.2.2, "Standard Commands Stipulated by IEEE 488.2", and Section 3.2.3, "Specific Commands."

(10) Transmission congestion relating to device-to-device messages which do not conform to the general principles for basic response messages

There are no device to device messages.

(11) Response capacity for block data

Block data does not appear in responses.

(12) Summary of standard commands and queries used

This appears in Section 3.1, "Command Summary."

(13) Device state after a calibration query has been completed without any problem

The "*CAL?" query is not used.

(14) When using the "*DDT" command, the maximum length of block used in a trigger macro definition

The "*DDT" command is not used.

(15) When a macro command is being executed, the maximum length of macro label, the maximum length of block for defining a macro, and how echoing is managed when expanding a macro

Macros are not used.

(16) For queries related to identification, explanation of the response to the "*IDN?" query

This is detailed in Section 3.2.2, "Standard Commands Stipulated by IEEE 488.2."

(17) Capacity of the user data storage area reserved for when the "*PUD" command and the "*PUD?" query are being executed

The "*PUD" command and the "*PUD?" query are not used. Further, there is no user data storage area.

(18) Resources when the "*RDT" command and the "*RDT?" query are being used

The "*RDT" command and the "*RDT?" query are not used.

(19) Conditions which are influenced when "*RST", "*LRN?", "*RCL", and "*SAV" are used

"*LRN?", "*RCL", and "*SAV" are not used. The "*RST" command returns the unit to its initial state.

(20) Scope of the self-testing executed as a result of the "*TST?" query

Checks the internal ROM and RAMs.

(21) Additional organization of the status data used in a device status report

This is detailed in Section 2.5, "The Status Byte and the Event Registers."

(22) Whether commands are overlap or sequential type

All the commands are sequential commands except ":ABORT" command. An ":ABORT" command is executed instantly as soon as it is transmitted.

(23) Criterion relating to the functions required at the instant that the termination message is produced, as a response to each command

Termination occurs when the command has been parsed.

Appendix 2 Troubleshooting the GP-IB Faults

Check the items in the following table in the event of operating problems with the GP-IB interface.

Symptom	Likely causes and remedies
The GP-IB does not operate at all.	Is the cable properly connected? See Section 2.2, "Cable Connection."
	Is the GP-IB address of the unit correctly set? Does it clash the address of other equipment on the same bus? See Section 2.3.1, "GP-IB Setup Procedure."
	Are all the devices that are connected powered on?
The unit keys stop working after using GP-IB communications.	Press the [LCL] soft key to end the remote operating state.
	Has an LLO (local lock-out) command been sent to the unit? Send a GTL command to return to the local state.
An attempt to read data using the CALL RECEIVE statement causes the GP-IB bus to hang.	Each and every CALL RECEIVE statement must be preceded by a query.
	Is the query transmitted incorrect?
Although a command was transmitted, the unit did not operate.	Use the "*ESR?" query to check the standard event status register for anomalies. See Section 2.5, "The Status Byte and the Event Registers."
Even though a number of queries were sent, only one response was received.	Has an error occurred?
	The response should be read immediately after each query. To read several responses in one operation, the corresponding queries must be combined into a single line using the message separator.
A service request is sometimes not issued.	Are service request enable register and the event status enable registers set correctly?
	At the end of the SRQ handling routine, use a "*CLS" command to clear all of the event registers. If a bit in the event registers is not cleared, the same event occurring again will not generate a service request.

Appendix 3 Troubleshooting the RS-232C Faults

Check the items in the following table in the event of operating problems with the RS-232C interface.

Symptom	Likely causes and remedies
The RS-232C does not operate at all.	Is the cable properly connected? See Section 2.2, "Cable Connection."
	Is the GP-IB address of the unit correctly set? Does it clash the address of other equipment on the same bus? See Section 2.3.2, "RS-232C Setup Procedure."
	Are all the devices that are connected powered on?
An attempt to read data using the CALL RECEIVE statement causes the RS-232C bus to hang.	Each and every CALL RECEIVE statement must be preceded by a query.
	Is the query transmitted incorrect?
Although a command was transmitted, the unit did not operate.	Use the "*ESR?" query to check the standard event status register for anomalies. See Section 2.5, "The Status Byte and the Event Registers."
Even though a number of queries were sent, only one response was received.	Has an error occurred?
	The response should be read immediately after each query. To read several responses in one operation, the corresponding queries must be combined into a single line using the message separator.
An overrun error or a framing error occurs.	Is the transfer rate too high? Reduce the transfer rate.

A

Index

HIOKI

DECLARATION OF CONFORMITY

Manufacturer's Name:	HIOKI E.E. CORPORATION
Manufacturer's Address:	81 Koizumi, Ueda, Nagano 386-1192, Japan
Product Name:	RS-232C CARD
Model Number:	9557
Product Name:	GP-IB CARD
Model Number:	9558

The above mentioned products conform to the following product specifications:

EMC: EN55022:1994+A1:1995+A2:1997 Class A
EN61000-6-2:2001
EN61000-4-2:1995[contact:level2(±4kV), air:level3(±8kV)]
EN61000-4-3:1996[level3(10V/m)]
EN61000-4-4:1995[power lines:level3(±2kV), signal lines:level3(±1kV)]
EN61000-4-5:1995[power lines:class4]
EN61000-4-6:1996[level3(10V)]
EN61000-4-8:1993[level4 continuous field]

Supplementary Information:

The products herewith comply with the requirements of the EMC Directive 89/336/EEC, but is not applicable to the Low Voltage Directive 73/23/EEC.

HIOKI E.E. CORPORATION

7 March 2005

T. Yoshiike

Tatsuyoshi Yoshiike

President

9557A999-03

HIOKI 9557 RS-232C CARD 9558 GP-IB CARD
Instruction Manual

Publication date: May 2005 Revised edition 8

Edited and published by HIOKI E.E. CORPORATION
Technical Support Section

All inquiries to International Sales and Marketing Department
81 Koizumi, Ueda, Nagano, 386-1192, Japan
TEL: +81-268-28-0562 / FAX: +81-268-28-0568
E-mail: os-com@hioki.co.jp
URL http://www.hioki.co.jp/

Printed in Japan 9557A981-08

HIOKI

HIOKI E. E. CORPORATION

HEAD OFFICE
81 Koizumi, Ueda, Nagano 386-1192, Japan
TEL +81-268-28-0562 / FAX +81-268-28-0568
E-mail: os-com@hioki.co.jp / URL http://www.hioki.co.jp/

HIOKI USA CORPORATION
6 Corporate Drive, Cranbury, NJ 08512, USA
TEL +1-609-409-9109 / FAX +1-609-409-9108

9557A981-08 05-05H

Printed on recycled paper