HIOKI

INSTRUCTION MANUAL

8852

MEMORY HiCORDER

GP-IB INTERFACE

HIOKI E. E. CORPORATION

Contents

Chapter Summary

Chapter 1 gives an overview of the GP-IB interface.

Chapter 2 contains the GP-IB specifications.

Chapter 3 describes the operation procedures.

Chapter 4 describes the GP-IB command list and the details of the commands. The list gives the reference pages, so use it for the command index.

Chapter 5 describes the program to operate GP-IB interface.

Chapter 6 describes the method of setting plotter output.

Chapter 7 contains the standard related to the GP-IB.

Chapter 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 8852 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)

Chapter 2
GP-IB Specification

2.1 Standards

IEEE Standard 488.1-1987
IEEE Standard 488.2-1987

2.2 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

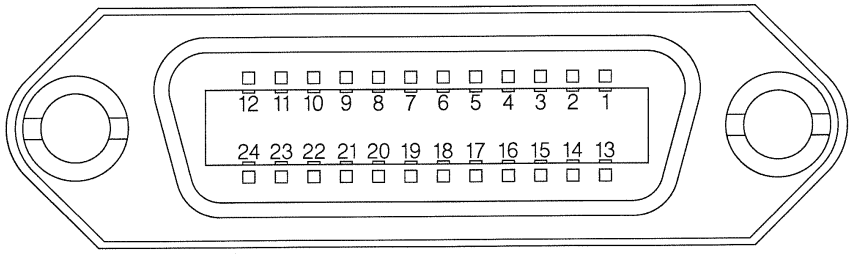
2.3 GP-IB Signal Lines

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.	

2.4 Connector Pin Assignment

On the 8852 57LE-20240 (made by DDK) or compatible.
On the cable 57-10240 (made by DDK) or compatible.

Pin arrangement diagram for the GP-IB interface connector on the 8852

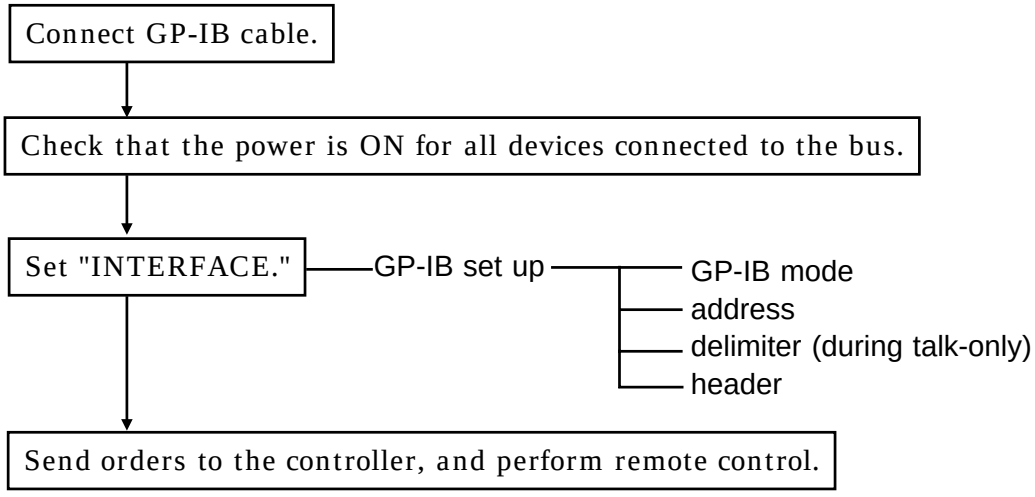


Pin number	Name of signal line	Pin number	Name of signal line
1	DIO 1	13	DIO 5
2	DIO 2	14	DIO 6
3	DIO 3	15	DIO 7
4	DIO 4	16	DIO 8
5	EOI	17	REN
6	DAV	18	GND
7	NRFD	19	GND
8	NDAC	20	GND
9	IFC	21	GND
10	SRQ	22	GND
11	ATN	23	GND
12	SHIELD	24	LOGIC GND

Chapter 3
Method of Operation

3.1 Basic Operational Procedure

⚠ CAUTION	The GP-IB interface is not isolated from the 8852 system. Exercise caution, because the ground of the analog and logic inputs, and the GP-IB interface ground are connected.



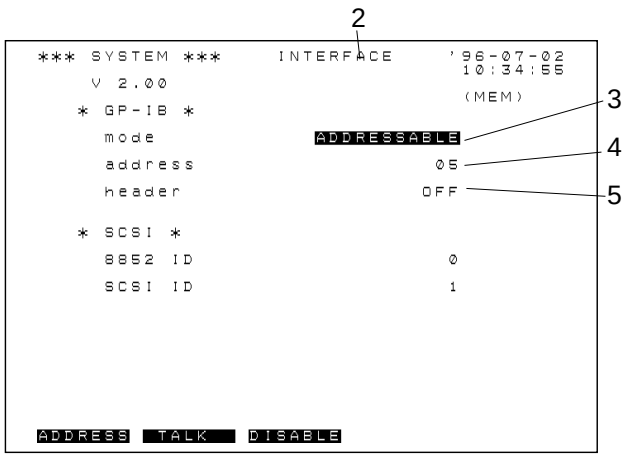
3.2 Setup Procedure

- On the 8852, set the GP-IB address for the unit, and select whether or not to use headers mode, and delimiter in messages output by the 8852.
- Use the interface setting screen, accessed from the "system" screen.

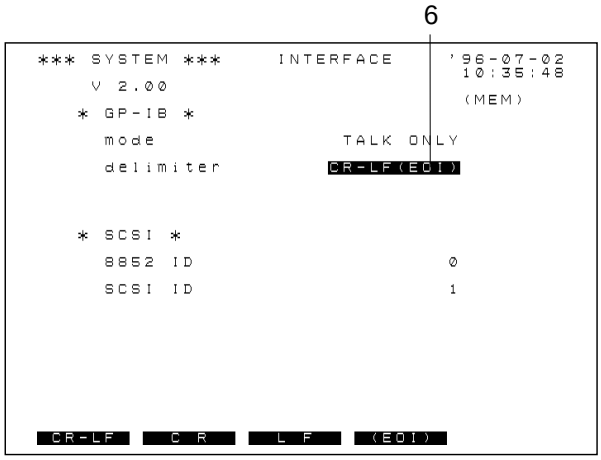
Procedure

1. Press the SYSTEM key to display the system screen.
2. Press the **INTER** soft key, and the interface setting screen appears.

   Using the CURSOR keys, move the flashing cursor in order to the items shown in the figure below by the reference numbers, and make the settings using the soft keys.



System screen (INTERFACE)



In talk-only mode

3. Set the GP-IB operation mode for this unit.

   Set the GP-IB address for this unit on the bus. [ADDRESSABLE, TALK ONLY, DISABLE]

   Soft key indication

   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. (Only when using a plotter.)

   DISABLE : Do not use the GP-IB interface.

4. Set the GP-IB device address.

   Use the ↓ and ↑ soft keys, or the rotary knob to adjust the numerical value. [0 to 30]

5. 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. [OFF, ON]

6. Select the GP-IB delimiter for talk-only mode.

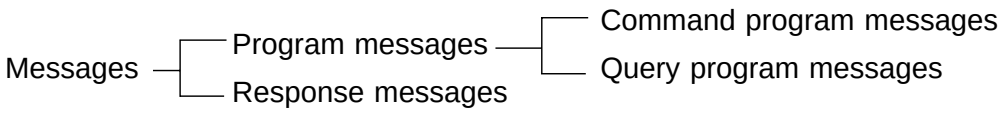
   Select the appropriate delimiter sequence for the plotter being used.

   [CR+LF(EOI), CR(EOI), LF(EOI), (EOI)]

3.3 Receive and Send Protocols

(1) Messages

Data received or sent by the GP-IB 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 8852 accepts commands without distinction between lower case and upper case letters.

It generates response messages in the long form (when headers are enabled) and in upper case letters.

The names of commands for the 8852 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.

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. On the 8852, LF, EOI, or LF+EOI is used as the message terminator, and LF+EOI is also used as the response message terminator.

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".

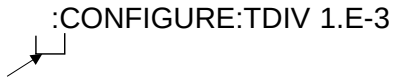
Example 1 :CONF:TDIV 1. E-3;:CONF:SHOT 15

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 8852 uses character data, decimal data and character string data.

- 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 8852 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 8852 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 7 bit ASCII characters.
② Characters which cannot be handled by the 8852 are replaced by spaces.
③ When the 8852 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.

3.4 Remote Control

● Local state

This is the state in which the 8852 is controlled by its keys. When the power is turned on, the 8852 always comes up in local state.

● Remote state

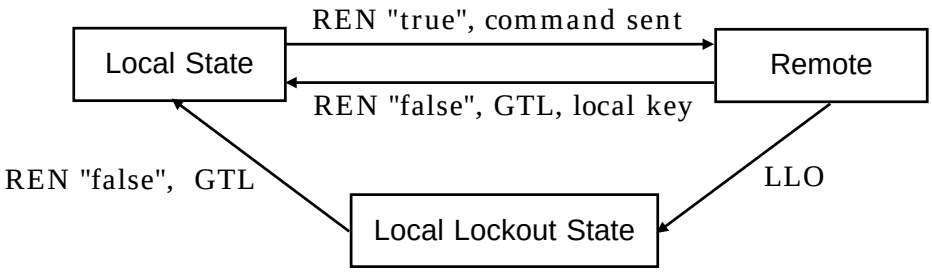
In this state the 8852 is controlled from the GP-IB interface (the REN line is "true"), and its keys are disabled. When in the remote state, the 8852 returns to local state if the local key (the **[LCL]** soft key) is pressed.

● 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 8852 is prevented from returning to the local state. This state is called the local lockout state.

In order to return the 8852 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 8852 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

3.5 Device Clear

When the 8852 receives the device clear command, it clears the input buffer and the output queue.

The device clear command is exemplified by the following:

HP 9816 (made by Hewlett-Packard)

```
CLEAR 7
```

3.6 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, 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.

Status byte bit settings

bit 7	Unused: 0
bit 6 RQS MSS	Set when a service request is issued.
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
Reading the service request enable register	*SRE?

(2) Standard event status register (SESR)

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

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

Read the standard event status enable register *ESE?

(3) Event status register 0 (ESR0)

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

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.

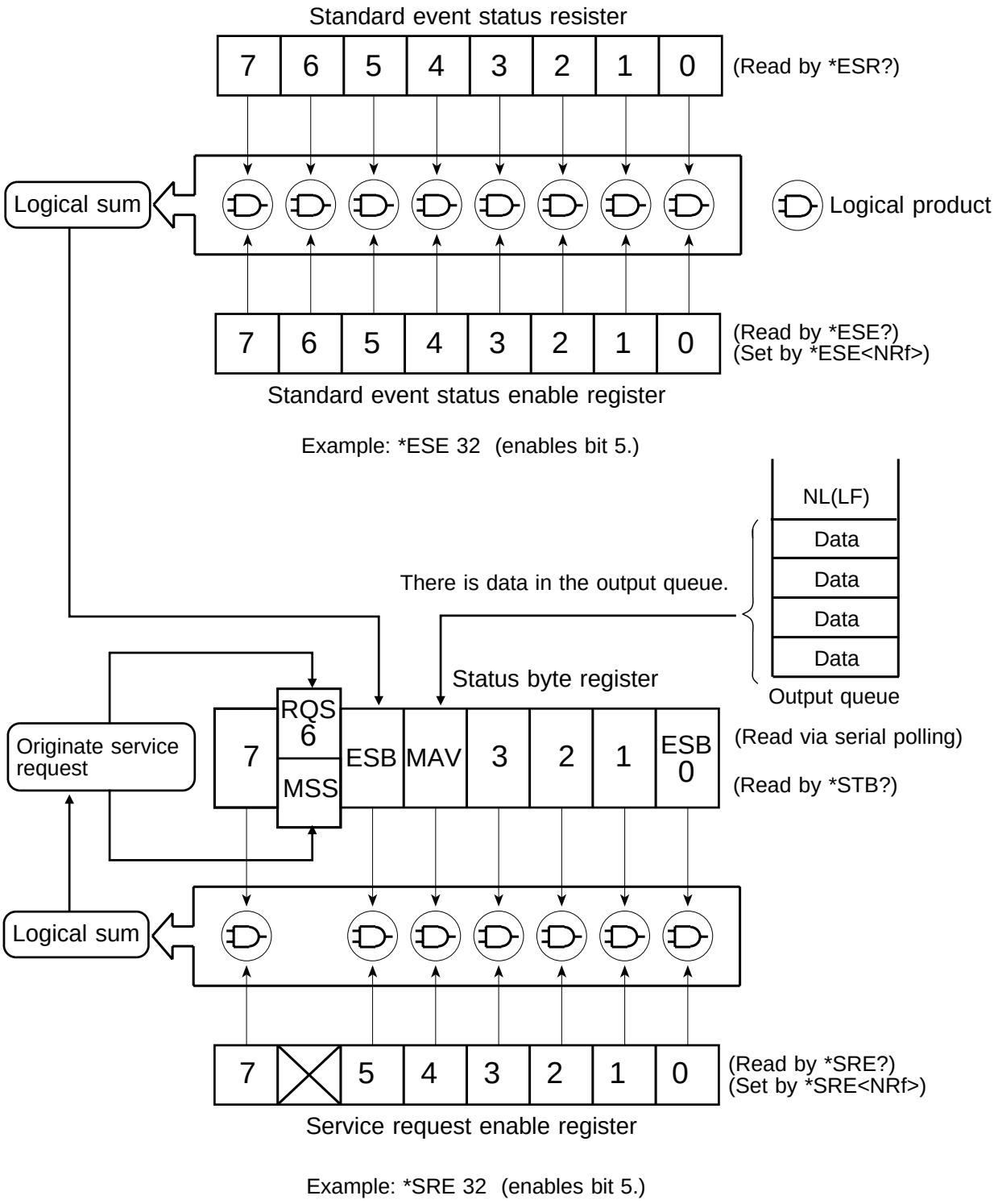
The bits of event status register 0

bit 7	Waveform decision fail (NG).
bit 6	Unused.
bit 5	Waveform 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.

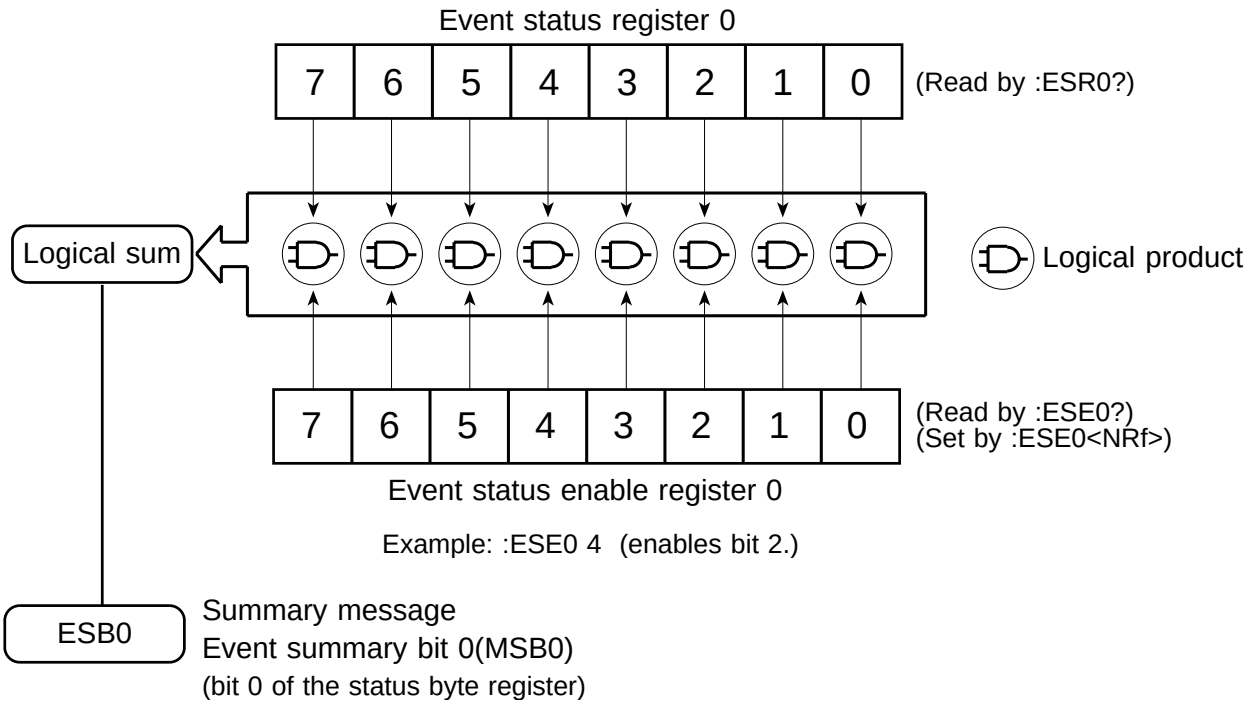
The following commands are used for reading the event status register 0, and for setting the event status enable register 0 and for reading it.

Reading event status register 0	:ESR0?
Setting event status enable register 0	:ESE0
Reading event status enable register 0	:ESE0?

Status byte data structure



Event status register 0 data structure



3.7 The Input Buffer and the Output Queue

(1) Input buffer

The 8852 has an input buffer of 512 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 8852 has an output queue of 256 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 256 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.

3.8 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.

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.

Chapter 4
GP-IB Commands

4.1 Command Summary

(1) Standard commands specified by IEEE 488.2

Command	Data (for a query, response data)	Explanation	Ref page
*IDN?	Maker's name, model number, serial number, software version (not used, zero)	Queries device ID.	48
*RST		Device initial setting.	48
*TST?	*A* <NR1> (0 = normal, 1 = failure)	Queries the result of the self-test.	48
*OPC		Sets the LSB of SESR after all action has been completed.	49
*OPC?	*A* <NR1>	Queries whether all action has been completed. ASCII [1] is the response.	49
*WAI		Wait until action fully completed.	49
*CLS		Clears the status byte and associated queues.	50
*ESE *A*	*A* = 0 to 255	Sets SESER.	50
*ESE?	*A* <NR1> 0 to 255	Queries SESER.	
*ESR?	*A* <NR1>	Queries SESR.	50
*SRE *A*	*A* = 0 to 255	Sets SRER.	51
*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.	51
:ESE0 *A* #	*A* = 0 to 255	Writes ESER0.	52
:ESE0? #	*A* <NR1> 0 to 255	Reads ESER0.	
:ESR0? #	*A* <NR1> 0 to 255	Reads ESR0.	52

#: specific to the 8852.

(2) Commands specific to the 8852

① Execution control etc. (common to all functions)

Command	Data (for a query, response data)	Explanation	Ref page
:STARt		Same as the START key.	53
:STOP		Same as the STOP key.	53
:ABORT		Forced halt.	53
:PRINt		Same as the PRINT key.	53
:HCOPy		Same as the COPY key.	54
:FEED *A*	*A* = 1 to 255 (unit mm)	Feeds the paper the specified distance.	54
:AUTO		Sets the time axis and the voltage axis automatically. (Only the memory recorder function)	54
:ERRor?	*A* <NR1> error number	Queries 8852 error number.	54
:HEADer *A$*	*A$* = OFF, ON	Enables and disables headers.	55
:HEADer?	*A$*	Queries header enablement.	
:FUNCtion *A$*	*A$* = MEM, REC, XYC, FFT	Changes the function.	55
:FUNCtion?	*A$*	Queries the function.	

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

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CONFigure				
:TDIV *A*	*A* = time axis range (unit seconds)	Sets the time axis range.	MEM REC	56
:TDIV?	*A* <NR3> (unit seconds)	Queries the time axis range.		
:SHOT *A*	*A* = recording length (unit DIV)	Sets the recording length.	MEM REC	56
:SHOT?	*A* <NR1> (unit DIV)	Queries the recording length.		
:FORMat *A$*	*A$* = SINGle, DUAL, QUAD, XY (MEM) SINGle, DUAL, QUAD (REC) SINGle, DUAL (FFT)	Sets the format.	MEM REC FFT	57
:FORMat?	*A$*	Queries the format.		
:DOTLine *A$*	*A$* = DOT, LINE	Sets the interpolation function.	MEM XYC FFT	57
:DOTLine?	*A$*	Queries the interpolation function.		

MEM	memory recorder function	REC	recorder function
XYC	X-Y recorder function	FFT	FFT function
All	all MEM, REC, XYC, and FFT function.		

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CONFigure				
:OVWRite *A$*	*A$* = OFF, ON	Enables and disables waveform superimposition.	MEM	57
:OVWRite?	*A$*	Queries waveform superimposition enablement.		
:ATPRint *A$*	*A$* = OFF, ON	Enables and disables auto print.	MEM FFT	58
:ATPRint?	*A$*	Queries auto print enablement.		
:ATSAve *A$*	*A$* = OFF, FD, SCSI (HD)	Enables and disables auto save. FD: Floppy disk SCSI (HD): SCSI device	MEM FFT	58
:ATSAve?	*A$*	Queries auto save enablement.		
:SMOOth *A$*	*A$* = OFF, ON	Enables and disables smooth printing.	MEM	59
:SMOOth?	*A$*	Queries smooth printing enablement.		
:ROLL *A$*	*A$* = OFF, ON	Enables and disables roll mode.	MEM	59
:ROLL?	*A$*	Queries roll mode enablement.		
:AVERage *A*	*A* = 0, 2, 4, 8, 16, 32, 64, 128, 256 (0; OFF)	Sets the count for averaging.	MEM FFT	59
:AVERage?	*A* <NR1>	Queries the current setting of the count for averaging.		
:MEMDiv *A$*	*A$* = OFF, SEQ, MULTI	Sets the memory segmentation function.	MEM	60
:MEMDiv?	*A$*	Queries the memory segmentation function.		
:MAXBlock *A*	*A* = 2 to 255	Sets the memory block number (in multi-block function).	MEM	60
:MAXBlock?	*A* <NR1>	Queries the memory block number.		
:STTBlock *A*	*A* = 1 to number of blocks	Sets the start block (in sequential save function).	MEM	61
:STTBlock?	*A* <NR1>	Queries the start block.		
:ENDBlock *A*	*A* = 1 to number of blocks	Sets the end block (in sequential save function).	MEM	61
:ENDBlock?	*A* <NR1>	Queries the end block.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CONFigure	(*G$* = G1, G2)			
:USEBlock *A*	*A* = 1 to number of memory segmentations	Sets the number of the memory block used (in sequential save and multi-block function).	MEM	61
:USEBlock?	*A* <NR1>	Queries the number of the memory block used.		
:REFBlock *A*	*A* = 0, 1 to number of memory segmentations (0; OFF)	Sets the reference block (in multi-block function).	MEM	62
:REFBlock?	*A* <NR1>	Queries the reference block.		
:SEQUdisp *A$*	*A$* = OFF, ON	Sets the waveform display (in sequential save function).	MEM	62
:SEQUdisp?	*A$*	Queries the waveform display.		
:WVCOmp *A$*	*A$* = OFF, OUT, ALLOut	Sets the waveform decision mode.	MEM FFT	63
:WVCOmp?	*A$*	Queries the waveform decision mode.		
:CMPStop *A$*	*A$* = GO, NG, G_N	Sets the waveform decision stop mode.	MEM FFT	63
:CMPStop?	*A$*	Queries the waveform decision stop mode.		
:PRINt *A$*	*A$* = OFF, ON	Sets printer output.	REC	63
:PRINt?	*A$*	Queries printer output.		
:MAXFreq *A*	*A* = frequency range (unit Hz)	Sets the frequency range for FFT analysis.	FFT	64
:MAXFreq?	*A* <NR3>	Queries the frequency range for FFT analysis.		
:FFTWind *A$*	*A$* = RECTan, HANNing	Sets FFT window.	FFT	64
:FFTWind?	*A$*	Queries FFT window.		
:FFTRef *A$*	*A$* = NEW, MEM	Designates the source for FFT analysis data.	FFT	64
:FFTRef?	*A$*	Queries the current FFT analysis data source.		
:FFTMode *G$*, *A$*	*A$* = STORage, PSPMDB, PSPMAG, LINMAG, LINIMG, LINREAL, PHASE, HISTogram	Sets the FFT analysis mode.	FFT	65
:FFTMode? *G$*	*G$*, *A$*	Queries the current FFT analysis mode.		
:FFTXaxis *G$*, *A$*	*A$* = LINhz, LOGhz	Sets the FFT x-axis.	FFT	66
:FFTXaxis? *G$*	*G$*, *A$* or (volt), (time)	Queries the present FFT x-axis setting.		

MEM memory recorder function
REC recorder function
XYC X-Y recorder function
FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CONFigure	(*G$* = G1, G2)			
:FFTSCale *G$, A$*	*A$* = AUTO, MANUal	Sets the FFT display scaling method for a graph.	FFT	66
:FFTSCale? *G$*	*G$, A$*	Queries the current FFT display scaling method for a graph.		
:FFTUp *G$, A*	*A* = -9.999E+9 to +9.999E+9	Sets vertical axis upper end value for FFT display.	FFT	67
:FFTUp? *G$*	*G$, A* <NR3>	Queries the current vertical axis upper end value for FFT display.		
:FFTLow *G$, A*	*A* = -9.999E+9 to +9.999E+9	Sets vertical axis lower end value of FFT display.	FFT	67
:FFTLow? *G$*	*G$, A* <NR3>	Queries the current vertical axis lower end value of FFT display.		
:FFTPrint *A$*	*A$* = WAVE, DATA	Sets FFT data printer output.	FFT	68
:FFTPrint?	*A$*	Queries FFT data printer output.		

③ TRIGger command (Setting and querying trigger source, level, etc.)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:TRIGger	(*ch$* = CH1 to CH4)			
:PAGE *A$*	*A$* = NORMal, SPECial	Switches the trigger screen.	MEM FFT	68
:PAGE?	*A$*	Queries the current trigger screen.		
:SOURce *A$*	*A$* = OR, AND	Sets trigger logical operator to AND or OR.	All	69
:SOURce?	*A$*	Queries trigger logical operator (AND or OR).		
:KIND *ch$, A$*	(in normal triggers) *ch$* = CH1 to CH4 *A$* = OFF, LEVEl, LOGIc, WINDow, TIMEout, GLITch (in special triggers) *ch$* = S1, S2 *A$* = OFF, EVENt, DELAy	Sets type of trigger for the indicated channel.	All	69
:KIND? *ch$*	*ch$, A$*	Queries type of trigger for the indicated channel.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:TRIGger	(*ch$* = CH1 to CH4)			
:LEVEl *ch$, A*	*A* = 0 to 100 (unit %)	Sets the trigger level of the indicated channel (in normal triggers).	All	70
:LEVEl? *ch$*	*ch$, A* <NR1>	Queries the trigger level.		
:SLOPe *ch$, A$*	*A$* = UP, DOWN	Sets the trigger direction (slope).	All	70
:SLOPe? *ch$*	*ch$, A$*	Queries the trigger direction (slope).		
:WIDTh *ch$, A$*	*A* = 2 to 4000	Sets glitch width or time out width.	All	70
:WIDTh? *ch$*	*ch$, A* <NR1>	Queries glitch width or time out width.		
:FILTer *ch$, A*	*ch$* = CH1, CH2 *A$* = OFF, ON	Enables and disables filter of level or logic trigger.	MEM FFT	71
:FILTer? *ch$*	*ch$, A$*	Queries enablement of filter of level or logic trigger.		
:FILTWidth *ch$, A*	*ch$* = CH1, CH2 *A* = 2 to 4000	Sets filter width of level or logic trigger.	MEM FFT	71
:FILTWidth? *ch$*	*ch$, A* <NR1>	Queries filter width of level or logic trigger.		
:UPPEr *ch$, A*	*ch$* = CH1, CH2 *A* = 1 to 100 (unit %)	Sets upper limit level of window trigger.	All	72
:UPPEr? *ch$*	*ch$, A* <NR1>	Queries upper limit level of window trigger.		
:LOWEr *ch$, A*	*ch$* = CH1, CH2 *A* = 0 to 99 (unit %)	Sets lower limit level of window trigger.	All	72
:LOWEr? *ch$*	*ch$, A* <NR1>	Queries lower limit level of window trigger.		
:LOGPat *ch$, 'A$'*	*ch$* = CH1 to CH3 *A$* = XXXXXXXX trigger pattern (X, 0, 1)	Sets the trigger pattern for a logic trigger.	All	73
:LOGPat? *ch$*	*ch$,* "*A$*"	Queries the trigger pattern for a logic trigger.		
:LOGAnd *ch$, A$*	*ch$* = CH1 to CH3 *A$* = OR, AND	Sets AND/OR for the logic trigger pattern.	All	73
:LOGAnd? *ch$*	*ch$, A$*	Queries AND/OR for the logic trigger pattern.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:TRIGger	(*ch$* = CH1 to CH4)			
:SPCHannel *A$, B$, C$*	*A$* = S1, S2 *B$*: start channel *B$* = OFF, CH1 to CH4 (OFF; event trigger only) *C$*: count channel, check channel *C$* = *ch$*	Sets the source channels for special trigger.	MEM FFT	74
:SPCHannel? *A$*	*A$, B$, C$*	Queries the source channels for special trigger.		
:SPLEvel *A$, B, C*	*A$* = S1, S2 *B* = 0 to 100(%) (start) *C* = 0 to 100(%) (count, check)	Sets levels of special trigger.	MEM FFT	74
:SPLEvel *A$*	*A$, B* <NR1>, *C* <NR1>	Queries levels of special trigger.		
:SPSLope *A$, B$, C$*	*A$* = S1, S2 *B$* = direction (slope) of start *C$* = direction (slope) of count or check *B$, C$* = UP, DOWN	Sets the directions (slopes) of special trigger.	MEM FFT	75
:SPSLope? *A$*	*A$, B$, C$*	Queries the directions (slopes) of special trigger.		
:SPCOunt *A$, B*	*A$* = S1, S2 *B* = 2 to 4000	Sets count for event trigger.	MEM FFT	75
:SPCOunt? *A$*	*A$, B* <NR1>	Queries count for event trigger.		
:DELAy *A$, B$, C*	*A$* = S1, S2 *B$* = GREATer, LESS *C* = 2 to 4000	Sets delay width of delay trigger.	MEM FFT	76
:DELAy? *A$*	*A$, B$, C* <NR1>	Queries delay width of delay trigger.		
:TVTRigger *A$*	*A$* = OFF, CH1, CH2	Sets TV trigger.	MEM FFT	76
:TVTRigger?	*A$*	Queries TV trigger.		
:TVFOrmat *A$*	*A$* = NTSC, PAL	Selects NTSC or PAL.	MEM FFT	77
:TVFOrmat?	*A$*	Queries NTSC or PAL.		
:TVFIeld *A*	*A* = 1, 2	Sets field for TV trigger.	MEM FFT	77
:TVFIeld?	*A* <NR1>	Queries field for TV trigger.		
:TVLIne *A*	*A* = 1 to 263 (NTSC) *A* = 1 to 313 (PAL)	Sets line for TV trigger.	MEM FFT	77
:TVLIne?	*A* <NR1>	Queries line for TV trigger.		
:EXTErnal *A$*	*A$* = OFF, ON	Enables and disables external trigger.	All	78
:EXTErnal?	*A$*	Queries external trigger enablement.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:TRIger	(*ch$* = CH1 to CH4)			
:TIMEr *A$*	*A$* = OFF, ON	Sets timer trigger.	All	78
:TIMEr?	*A$*	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	79
:TMSTArt?	*month, day, hour, min* all <NR1>	Queries start time of timer trigger.		
:TMSTOp *month, day, hour, min*	Same as :TMSTARrt	Sets stop time of timer trigger.	All	79
:TMSTOp?	Same as :TMSTArt?	Queries stop time of timer trigger.		
:TMINTvl *hour, min, sec*	*hour* = 0 to 23 *min* = 0 to 59 *sec* = 0 to 59	Sets time interval for timer trigger.	All	80
:TMINTvl?	*hour, min, sec* all <NR1>	Queries time interval for timer trigger.		
:MODE *A$*	*A$* = SINGle, REPEat (REC) SINGle, REPEat, AUTO (MEM, FFT)	Sets trigger mode.	MEM REC FFT	80
:MODE?	*A$*	Queries trigger mode.		
:PRETrig *A*	*A* = 0, 2, 5, 10, ... 90, 95, 100, and -950 to -50 in 50 % steps.	Sets pre-trigger.	MEM FFT	81
:PRETrig?	*A* <NR1> (unit %)	Queries pre-trigger.		
:TIMIng *A$*	*A$* = START, STOP, S_S	Sets trigger timing.	REC XYC	81
:TIMIng?	*A$*	Queries trigger timing.		
:TRGTime? (*A*)	*A* = block number in memory segmentation (0 to maximum number of blocks) *hour, min, sec* (all <NR1>)	Queries the currently set time point for trigger detection.	All	82
:TRGDate? (*A*)	*A* = block number in memory segmentation (0 to maximum number of blocks) *year, month, day* (all <NR1>)	Queries the currently set date for trigger detection.	All	82

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

④ UNIT command (Setting and querying input channel)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:UNIT	(*ch$* = CH1 to CH4)			
:TYPE *ch$, A$*	*ch$* = CH1 to CH3 *A$* = ANALog, LOGIc	Sets switching between analog and logic.	All	83
:TYPE? *ch$*	*ch$, A$*	Queries switching between analog and logic.		
:RANGe *ch$, A*	*A* = voltage axis range (unit volts)	Sets input channel voltage axis range.	All	83
:RANGe? *ch$*	*ch$, A* <NR3>	Queries input channel voltage axis range.		
:POSItion *ch$, A*	*A* = Position value (unit %)	Sets the origin position for an input channel.	All	84
:POSItion? *ch$*	*ch$, A* <NR1>	Queries the origin position for an input channel.		
:COUPling *ch$, A$*	*A$* = GND, DC, AC	Sets input channel coupling.	All	84
:COUPling? *ch$*	*ch$, A$*	Queries input channel coupling.		
:FILTer *ch$, A*	*A* = 0, 2.0E7, 10 (0; OFF)	Sets input channel filter.	All	85
:FILTer? *ch$*	*ch$, A* <NR3>	Queries input channel filter.		
:ADJUST		Carries out zero adjustment for the input units.	All	85
:LOGLevel *ch$, A*	*ch$* = CH1 to CH3 *A* = -6.2 to 6.2	Sets the logic threshold level.	All	85
:LOGLevel? *ch$*	*ch$, A* <NR2>	Queries the logic threshold level.		

MEM	memory recorder function	REC	recorder function
XYC	X-Y recorder function	FFT	FFT function
All	all MEM, REC, XYC, and FFT function.		

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

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DISPlay	(*ch$* = CH1 to CH4) (*G$* = G1, G2)			
:CHANge *A$*	*A$* = SYSTem, STATus, TRIGger, DISPlay	Changes over the display screen.	All	86
:CHANge?	*A$*	Queries the display screen.		
:DRAWing *ch$, A$*	*A$* = OFF, LIGHt, DARK	Sets display and recording intensity for waveform.	MEM REC XYC	86
:DRAWing? *ch$*	*ch$, A$*	Queries display and recording of a waveform.		
:PAGE *A*	*A* = 1 to 8 (system screen) *A* = 1, 2 (status screen)	Changes over the page of the screen.	All	87
:PAGE?	*A* <NR1>	Queries the page of the screen.		
:GRAPh *ch$, G$*		Sets waveform display graph in DUAL and QUAD format.	MEM REC	87
:GRAPh? *ch$*	*ch$, G$*	Queries waveform display graph in DUAL and QUAD format.		
:XMAG *A$*	(MEM) *A$* = X 10, X 5, X 2, X 1, X 1_2, X 1_5, X 1_10, X 1_20, X 1_50, X 1_100, X 1_200, X 1_500, X 1_1000 X 1_2000, X 1_5000, X 1_10000, X 1_20000, X 1_40000 (REC) *A$* = X 1, X 1_2, X 1_5, X 1_10, X 1_20, X 1_50, X 1_100, X 1_200	Sets the magnification/ compression factor on the time axis.	MEM REC	88
:XMAG?	*A$*	Queries the magnification/ compression factor on the time axis.		
:ZOOM *A$*	*A$* = OFF, ON	Enables and disables the zoom function.	MEM	88
:ZOOM?	*A$*	Queries the zoom function enablement.		
:ZOOMMag *A$*	Same as *A$* = XMAG	Sets the zoom magnification.	MEM	89
:ZOOMMag?	*A$*	Queries the zoom magnification.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DISPlay	(*ch$* = CH1 to CH4)			
:YMAG *ch$, A$*	*A$* = X 1_2, X 1, X 2	Sets the magnification/ compression factor on the voltage axis.	MEM REC	89
:YMAG? *ch$*	*ch$, A$*	Queries the magnification/ compression factor on the voltage axis.		
:YZOOm *ch$, A*	*A* = 0 to 100 %	Sets the waveform display position on the voltage axis.	MEM REC	90
:YZOOm? *ch$*	*ch$, A* <NR1>	Queries the waveform display position on the voltage axis.		
:WAVE *A$*	*A$* = ACUR (A-cursor), TRIG (trigger point), POINT (the point set with :MEMory:POINt)	Executes waveform display.	MEM	90
:DIVMap *A$*	*A$* = OFF, ON	Enables and disables the memory segmentation screen.	MEM	91
:DIVMap?	*A$*	Queries the memory segmentation screen enablement.		
:CALCEdit *A$*	*A$* = OFF, ON	Enables and disables the waveform processing calculation setting screen.	MEM	91
:CALCEdit?	*A$*	Queries enablement of the waveform processing calculation setting screen.		
:MEASEdit *A$*	*A$* = OFF, ON	Enables and disables the waveform parameter calculation setting screen.	MEM	92
:MEASEdit?	*A$*	Queries enablement of the waveform parameter calculation setting screen.		
:XAXIs *ch$*		In X-Y format, sets the X-axis.	MEM XYC	92
:XAXIs?	*ch$*	In X-Y format, queries the X-axis.		
:XYCLr *A$*	*A$* = OFF, ON	Sets the display clear function in the X-Y recorder function off or on.	XYC	93
:XYCLr?	*A$*	Queries the setting of the display clear function in the X-Y recorder function.		

MEM	memory recorder function	REC	recorder function
XYC	X-Y recorder function	FFT	FFT function
All	all MEM, REC, XYC, and FFT function.		

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DISPlay	(*ch$* = CH1 to CH4) (*G$* = G1, G2)			
:FFTCH *G$*, *ch$*	*G$* = G1, G2	Sets the FFT analysis channel.	FFT	93
:FFTCH? *G$*	*G$*, *ch$*	Queries the FFT analysis channel.		

⑥ CURSor command (Cursor setting and reading)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CURSor	(*ch$* = CH1 to CH4)			
:MODE *A$*	*A$* = OFF, TIME, VOLT (REC) OFF, TIME, VOLT, TRACe (MEM excluding XY format) OFF, Xcur, Ycur (XY format in MEM, or XYC) OFF, ON (FFT)	Sets the A and B cursor type.	All	94
:MODE?	*A$*	Queries the A and B cursor type.		
:ABCUrsor *A$*	*A$* = A, A_B	Chooses between the A and the A&B cursors.	MEM REC XYC	94
:ABCUrsor?	*A$*	Queries between the A and the A&B cursors.		
:ACHAnnel *ch$*		Sets the A cursor channel.	MEM REC XYC	95
:ACHAnnel?	*ch$*	Queries the A cursor channel.		
:BCHAnnel *ch$*		Sets the B cursor channel.	MEM REC XYC	95
:BCHAnnel?	*ch$*	Queries the B cursor channel.		
:YDISp *A$*	*A$* = PAEK, RMS	Sets the FFT cursor readout value as peak or RMS value.	FFT	95
:YDISp?	*A$*	Queries the FFT cursor readout value as peak or RMS value.		
:APOSition *A*	(vertical cursor, trace cursor) *A* = 0 to amount of stored data (MEM, REC) 0 to 400 (XYC) (horizontal cursor) *A* = 0 to 250 (MEM, REC, XYC)	Sets the position of the A cursor.	MEM REC XYC	96
:APOSition?	*A* <NR1>	Queries the position of the A cursor.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CURSor				
:BPOSition *A*	Same as :APOSition	Sets the position of the B cursor.	MEM REC XYC	96
:BPOSition?	*A* <NR1>	Queries the position of the B cursor.		
:FPOSition *A*	*A* = 0 to 799 (STORAGE) 0 to 399 (except STORAGE)	Sets the position of the FFT cursor.	FFT	97
:FPOSition?	*A* <NR1>	Queries the position of the FFT cursor.		
:DTREad?	*A$* = readout value (t, Δt) *B$* = readout value (1/t, 1/Δt)	Queries the cursor readout value (t).	MEM REC	98
:DVREad?	*A$* = readout value (V, ΔV)	Queries the cursor readout value (V).	MEM REC XYC	98
:FFTRead?	*A$* = x-axis readout value. *B$* = y-axis readout value.	Queries the cursor readout value.	FFT	99

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

Command	Data (for a query, response data)	Explanation	Function	Ref page
:MEMory	(*ch$* = CH1 to CH4)			
:POINt *ch$*, *A*	*A* = 0 to recording length × 40 (16000000 max.)	Sets point in memory for input and output.	MEM	99
:POINt?	*ch$*, *A* <NR1>	Queries point in memory for input and output.		
:MAXPoint?	*A* <NR1> = 0: not stored 600 to 16000000 (÷ 40 = number of divisions)	Queries the amount of data stored.	MEM	100
:ADATa *B*, *C*,...	*B*, *C*,... = -3 to 252	Input data to memory (ASCII).	MEM	100
:ADATa? *A*	*A* = 1 to 40 (number of output units) *B*, *C*,...<NR1> = -3 to 252	Output data from memory (ASCII).		
:VDATa *B*, *C*,...	*B*, *C*,... = voltage values (units V)	Input data to memory (voltage values).	MEM	101
:VDATa? *A*	*A* = 1 to 10 (amount of data) *B*, *C*,...<NR3> = voltage value (units V)	Output stored data (voltage values).		

MEM	memory recorder function	REC	recorder function
XYC	X-Y recorder function	FFT	FFT function
All	all MEM, REC, XYC, and FFT function.		

Command	Data (for a query, response data)	Explanation	Function	Ref page
:MEMory	(*ch$* = CH1 to CH4)			
:LDATa *B, C,...*	*B, C,...* = 0 to 255	Input logic data from memory.	MEM	102
:LDATa? *A*	*A* = 1 to 40 (amount of output data) Response data <NR1> = 0 to 255	Output logic data from memory.		
:AREAl? *ch$*	*A* <NR1> = -3 to 252	Output stored data. Real time data output (ASCII)	MEM	103
:VREAl? *ch$*	*A* <NR3> = voltage value (units V)	Real time data output (voltage value)	MEM	103
:LREAl? *ch$*	*A* <NR1> = 0 to 255	Logic real time data output	MEM	103
:BDATa? *A*	*A* = 1 to 125 (amount of output data) Response data, binary, integer data	Performs binary transfer for stored data.	MEM	104
:BREAl? *ch$*	Response data, binary, integer data	Real time data output (binary)	MEM	104
:FFTPOint *A*	*A* = 0 to 799 (stored waveform) 0 to 399 (except stored waveform)	Sets the output point for FFT data.	FFT	105
:FFTPOint?	*A* <NR1>	Queries the current output point for FFT data.		
:FFTData?	*A* <NR3> = y-axis data	Output FFT data.	FFT	105

⑧ SYSTem command (Setting and querying the system screen)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:SYSTem				
:TIME *hour, min, sec*	*hour* = 0 to 23 *min* = 0 to 59 *sec* = 0 to 59	Sets the time.	All	106
:TIME?	*hour, min, sec* (all <NR1>)	Queries the current time.		
:DATE *year, month, day*	*year* = 0 to 99 *month* = 1 to 12 *day* = 1 to 31	Sets the calendar.	All	106
:DATE?	*year, month, day* (all <NR1>)	Queries the calendar.		
:DATAClear		Clear data.	All	106
:CRTOff *A$*	*A$* = ON, OFF	Enables and disables the screen saver.	All	107
:CRTOff?	*A$*	Queries enablement of the screen saver.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:SYSTem	(*ch$* = CH1 to CH4)			
:GRID *A$*	*A$* = OFF, NORMal, FINE	Sets the grid type.	All	107
:GRID?	*A$*	Queries the grid type.		
:STARt *A$*	*A$* = ON, OFF	Enables and disables start key backup.	All	107
:STARt?	*A$*	Queries start key backup enablement.		
:CHMArk *A$*	*A$* = ON, OFF	Enables and disables channel markers.	All	108
:CHMArk?	*A$*	Queries enablement of channel markers.		
:BEEPer *A$*	*A$* = ON, OFF	Enables and disables the beep sound.	All	108
:BEEPer?	*A$*	Queries beep sound enablement.		
:LIST *A$*	*A$* = OFF, LIST, GAUGE, L_G	Sets list and gauge functions.	All	108
:LIST?	*A$*	Queries list and gauge functions.		
:USECH *A*	*A* = 1, 2, 4	Sets number of channels used.	All	109
:USECH?	*A* <NR1>	Queries number of channels used.		
:COPYSize *A$*	*A$* = LARGE, SMALL	Sets the screen dump size.	All	109
:COPYSize?	*A$*	Queries the screen dump size.		
:SCSI *A$, B*	*A$* = 8852, SCSI (HDD) *B* = 0 to 7	Sets the SCSI interface device address ID.	All	109
:SCSI? *A$*	*A$, B* <NR1>	Queries the SCSI interface device address ID.		
:COPYPlot *A$*	*A$* = PRINter, PLOTter, FD, SCSI	Sets the screen dump output destination.	All	110
:COPYPlot?	*A$*	Queries the screen dump output destination.		
:PEN *A$, B*	*A$* = AREA, FRAME, CHAR, CH1 to CH4 *B* = 0 to 8 (0; OFF)	Sets the plotter pen.	All	110
:PEN? *A$*	*A$, B* <NR1>	Queries the plotter pen.		
:PROBE *ch$, A$*	*A$* = 10_1, 1_1	Sets the probe ratio.	All	111
:PROBE? *ch$*	*ch$, A$*	Queries the probe ratio.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:SYSTem	(*ch$* = CH1 to CH4)			
:BMPKind *A$*	*A$* = MONO, COLOR	Sets the color mode of the bitmap file.	All	111
:BMPKind?	*A$*	Queries the color mode of the bitmap file.		
:BMPColor *A$* to *D$*	*A$* to *D$* = BLACK, BLUE, RED, MAGENTA, GREEN, CYAN, YELLOW, ORANGE	Sets the colors of the bitmap file.	All	112
:BMPColor?	*A$* to *D$*	Queries the colors of the bitmap file.		
:DISKMode *A$*	*A$* = FD, SCSI, FD_SCSI	Sets the FD key.	All	112
:DISKMode?	*A$*	Queries the FD key.		

⑨ SCALing command (Setting and querying scaling)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:SCALing	(*ch$* = CH1 to CH4)			
:SET *ch$, A$*	*A$* = OFF, SCI, ENG	Enables and disables scaling.	All	113
:SET? *ch$*	*ch$, A$*	Queries scaling enablement.		
:VOLT *ch$, A*	*A* = -9.999E+9 to +9.999E+9	Sets the scaling conversion value.	All	113
:VOLT? *ch$*	*ch$, A* <NR3>	Queries the scaling conversion value.		
:OFFSet *ch$, A*	*A* = -9.999E+9 to +9.999E+9	Sets scaling offset.	All	114
:OFFSet? *ch$*	*ch$, A* <NR3>	Queries scaling offset.		
:UNIT *ch$, 'A$'*	*A$* = scaling unit (7 characters)	Sets scaling unit.	All	114
:UNIT? *ch$*	*ch$,* "*A$*"	Queries scaling unit.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

⑩ COMMent command (Setting and querying comments)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:COMMent	(*ch$* = CH1 to CH4)			
:TITLe *A$*, '*B$*'	*A$* = ON, OFF *B$* = comment string (up to 20 characters)	Sets a title comment.	All	115
:TITLe?	*A$*, "*B$*"	Queries a title comment.		
:CH *ch$*, *A$*, '*B$*'	*A$* = ON, OFF *B$* = comment string (up to 20 characters)	Sets a comment for a particular channel.	All	116
:CH? *ch$*	*ch$*, *A$*, "*B$*"	Queries comment for a particular channel.		

⑪ CALCulate command (Calculation setting and querying)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate	(*ch$* = CH1 to CH4)			
:WVCALc *A$*	*A$* = ON, OFF, EXEC (execute)	Enables and disables waveform processing calculation.	MEM	116
:WVCALc?	*A$*	Queries enablement of waveform processing calculation.		
:Z1 *A$*, *B$*, *C$*, *D$*	*A$*, *B$*, *C$* = A to P *D$* = PLUS, MINUs, MULTi, DIVI	Sets the coefficients for the waveform processing calculation equation for Z1.	MEM	117
:Z1?	*A$*, *B$*, *C$*, *D$*	Queries the coefficients for the waveform processing calculation equation for Z1.		
:Z2 *A$*, *B$*, *C$*, *D$*	*A$*, *B$*, *C$* = A to P *D$* = PLUS, MINUs, MULTi, DIVI	Sets the coefficients for the waveform processing calculation equation for Z2.	MEM	117
:Z2?	*A$*, *B$*, *C$*, *D$*	Queries the coefficients for the waveform processing calculation equation for Z2.		
:Z3 *A$*, *B$*, *C$*, *D$*	*A$*, *B$*, *C$* = A to P *D$* = PLUS, MINUs, MULTi, DIVI	Sets the coefficients for the waveform processing calculation equation for Z3.	MEM	118
:Z3?	*A$*, *B$*, *C$*, *D$*	Queries the coefficients for the waveform processing calculation equation for Z3.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
ALL all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate	(*ch$* = CH1 to CH4)			
:Z4 *A$, B$, C$, D$*	*A$, B$, C$* = A to P *D$* = PLUS, MINUs, MULTi, DIVI	Sets the coefficients for the waveform processing calculation equation for Z4.	MEM	118
:Z4?	*A$, B$, C$, D$*	Queries the coefficients for the waveform processing calculation equation for Z4.		
:X1 *A$, ch$, B$*	*A$* = OFF (*ch$, B$* are disregarded) PAR, ABS, EXP, LOG, SQR, MOV, SLI, DIF, INT, DIF2, INT2 *B$* = A to P (when *A$* = MOV, a value from 1 to 4000, when *A$* = SLI, a value from -4000 to 4000)	Sets calculation equation for X1.	MEM	119
:X1?	*A$, ch$, B$*	Queries calculation equation for X1.		
:X2 *A$, ch$, B$*	Same as X1 (*ch$* = CH1 to CH4, Z1)	Sets calculation equation for X2.	MEM	120
:X2?	*A$, ch$, B$*	Queries calculation equation for X2.		
:X3 *A$, ch$, B$*	Same as X1 (*ch$* = CH1 to CH4, Z1, Z2)	Sets calculation equation for X3.	MEM	120
:X3?	*A$, ch$, B$*	Queries calculation equation for X3.		
:X4 *A$, ch$, B$*	Same as X1 (*ch$* = CH1 to CH4, Z1 to Z3)	Sets calculation equation for X4.	MEM	121
:X4?	*A$, ch$, B$*	Queries calculation equation for X4.		
:Y1 *A$, ch$, B$*	Same as X1 (*ch$* = CH1 to CH4)	Sets calculation equation for Y1.	MEM	122
:Y1?	*A$, ch$, B$*	Queries calculation equation for Y1.		
:Y2 *A$, ch$, B$*	Same as X1 (*ch$* = CH1 to CH4, Z1)	Sets calculation equation for Y2.	MEM	123
:Y2?	*A$, ch$, B$*	Queries calculation equation for Y2.		

MEM	memory recorder function	REC	recorder function
XYC	X-Y recorder function	FFT	FFT function
All	all MEM, REC, XYC, and FFT function.		

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate	(*ch$* = CH1 to CH4) (*NO$* = NO1 to NO4)			
:Y3 *A$, ch$, B$*	Same as X1 (*ch$* = CH1 to CH4, Z1, Z2)	Sets calculation equation for Y3.	MEM	123
:Y3?	*A$, ch$, B$*	Queries calculation equation for Y3.		
:Y4 *A$, ch$, B$*	Same as X1 (*ch$* = CH1 to CH4, Z1 to Z3)	Sets calculation equation for Y4.	MEM	124
:Y4?	*A$, ch$, B$*	Queries calculation equation for Y4.		
:FACTor *A$, B*	*A$* = A to P *B* = -9.999E+9 to +9.999E+9	Sets the value of calculation equation coefficient a to p.	MEM	124
:FACTor? *A$*	*A$, B* <NR3>	Queries the value of calculation equation coefficient a to p.		
:Z1DIsplay *ch$, A$ upper, lower*	*ch$* = CH1 to CH4, NONE *A$* = AUTO, MANUal (for MANUal) *upper, lower* = -9.999E+9 to +9.999E+9 (units V)	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z1.	MEM	125
:Z1DIsplay?	*ch$, A$, upper, lower*	Queries the channel for receipt of the calculated result of the waveform treatment calculation equation for Z1.		
:Z2DIsplay *ch$, A$, upper, lower*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z2.	MEM	125
:Z2DIsplay?	*ch$, A$, upper, lower*	Queries the channel for receipt of the calculated result of the waveform treatment calculation equation for Z2.		
:Z3DIsplay *ch$, A$, upper, lower*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z3.	MEM	126
:Z3DIsplay?	*ch$, A$, upper, lower*	Queries the channel for receipt of the calculated result of the waveform treatment calculation equation for Z3.		

MEM	memory recorder function	REC	recorder function
XYC	X-Y recorder function	FFT	FFT function
All	all MEM, REC, XYC, and FFT function.		

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate	(*ch$* = CH1 to CH4) (*NO$* = NO1 to NO4)			
:Z4DIsplay *ch$, A$, upper, lower*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z4.	MEM	126
:Z4DIsplay?	*ch$, A$, upper, lower*	Queries the channel for receipt of the calculated result of the waveform treatment calculation equation for Z4.		
:MEASure *A$*	*A$* = ON, OFF, EXEC (execute)	Enables and disables waveform parameter calculation.	MEM	127
:MEASure?	*A$*	Queries enablement of waveform parameter calculation.		
:MEASPrint *A$*	*A$* = OFF, PRINter, FD, SCSI	Sets the output destination of waveform parameter calculation values.	MEM	127
:MEASPrint?	*A$*	Queries the output destination of waveform parameter calculation values.		
:MEASSet *NO$, A$, ch$*	*A$* = OFF, AVE, RMS, PP, MAX, MAXT, MIN, MINT, AREA, PERI, FREQ, RISE, FALL, XYAREA *ch$* = ALL, CH1 to CH4	Sets waveform parameter calculation.	MEM	128
:MEASSet? *NO$*	*NO$, A$, ch$*	Queries waveform parameter calculation.		
:ANSWer? *NO$, ch$*	*A$* = OFF, AVE, RMS, PP, MAX, MAXT, MIN, MINT, AREA, PERI, FREQ, RISE, FALL, XYAREA *B* <NR3> = calculation result NONE, 0 (when there is no calculation result.)	Queries a waveform parameter calculation result.	MEM	129
:COMP *NO$, A$*	*A$* = ON, OFF	Enables or disables waveform parameter decision calculations.	MEM	129
:COMP? *NO$*	*NO$, A$*	Queries enablement of waveform parameter decision calculations.		
:COMPArea *NO$, upper, lower*	*upper, lower* = -9.999E+9 to +9.999E+9	Sets upper limit and lower limit values for waveform parameter calculation decision.	MEM	130
:COMPArea? *NO$*	*NO$, upper* <NR3>, *lower* <NR3>	Queries upper limit and lower limit values for waveform parameter calculation decision.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

⑫ DISK command (Setting and querying relating to the FD, HD, and MO)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DISK	(*ch$* = CH1 to CH4)			
:MODE *A$*	*A$* = OFF, FD, SCSI (HD)	Enables or disables the FD screen or SCSI screen.	All	130
:MODE?	*A$*	Queries enablement of the FD screen or SCSI screen.		
:SAVE '*NAME1$*.*NAME2$*', *A$*, *B$*	*NAME1$* = file name (up to 8 characters) *NAME2$* = file extension (up to 3 characters) *A$* = type of data to save Wave: measurement data (MEM, FFT) Func: unit settings Area: waveform decision area (MEM, FFT) (when *A$* = Wave in MEM, FFT) *B$* = channels to save ALL, CH1 to CH4	Saves a file.	All	131
:LOAD *NO* (,*ch$*)	*NO* = file number	Load a file.	All	131
:DELEte *NO*	*NO* = file number	Deletes a file.	All	132
:FORMat (*A$*)	*A$* = 2HD, 2HC (*A$* is effective for 2HD FDs.)	Formats a FD, HD, or MO.	All	132
:MKDIR '*NAME$*'	*NAME$* = directory name (up to 12 characters)	Creates a directory.	All	132
:CHDIR *NO*	*NO* = file number	Changes the current directory.	All	133
:DIR?	*A$* = directory name	Queries the current directory.	All	133
:INFOr? '*NAME$*'	*NAME$* = file name (response) *In the file on the FD, HD, or MO: FILE, "*NAME$*", *A*, *B$*, "*DATE$*", "*TIME$*", *C* *In the HD or MO directory: "*NAME$*", *A*, "*DATE$*", "*TIME$*" *NAME$* = file name *A* = file number (if no file exists, then -1) *B$* = type of data saved WAVE: measurement data FUNC: conditions of creation AREA: waveform decision area N: no such file *DATE$* = year/month/day of save *TIME$* = hour:min:sec of save *C* = file size	Queries information about a file.	All	133

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
All all MEM, REC, XYC, and FFT function.

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DISK				
:NINFor? *NO*	*NO*, "*NAME$*" (response) *NO* = file number *NAME$* = file name	Queries filename.	All	134
:FILE?	*A* <NR1> = number of files	Queries how many files are saved.	All	134
:FREE?	*A* <NR1> = allowable number of clusters	Queries the allowable number of clusters.	All	134

⑬ GRAPh command (Commands relating to the graphics editor)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:GRAPh				
:EDIT *A$*	*A$* = OFF, ON	Enables and disables the editor.	MEM FFT	135
:EDIT?	*A$*	Queries editor enablement.		
:STORage		Loads a waveform into the editor.	MEM FFT	135
:PARAllel *high, low, right, left*	*high* = 0 to 9.96 (div) *low* = 0 to 9.96 (div) *right* = 0 to 14.975 (div) *left* = 0 to 14.975 (div)	Carries out a parallel movement of the drawing.	MEM FFT	135
:LINE *X1, Y1, X2, Y2*	*X1, X2* = x-coordinates *Y1, Y2* = y-coordinates	Draws a line from (X1, Y1) to (X2, Y2).	MEM FFT	136
:PAINT *X, Y*	*X* = x-coordinate, *Y* = y-coordinate	Begins solid fill from the point specified by (X, Y).	MEM FFT	137
:REVErse		Reverses the drawing.	MEM FFT	137
:ERASe *X1, Y1, X2, Y2*	*X1, X2* = x-coordinates *Y1, Y2* = y-coordinates	Erases from (X1, Y1) to (X2, Y2).	MEM FFT	137
:CLEAr *X1, Y1, X2, Y2*	*X1, X2* = x-coordinates *Y1, Y2* = y-coordinates all NR1	Clears the rectangle with the points (X1, Y1) and (X2, Y2) at diagonally opposite corners.	MEM FFT	137
:ALLClear		Clears the entire drawing.	MEM FFT	138

MEM	memory recorder function	REC	recorder function
XYC	X-Y recorder function	FFT	FFT function
All	all MEM, REC, XYC, and FFT function.		

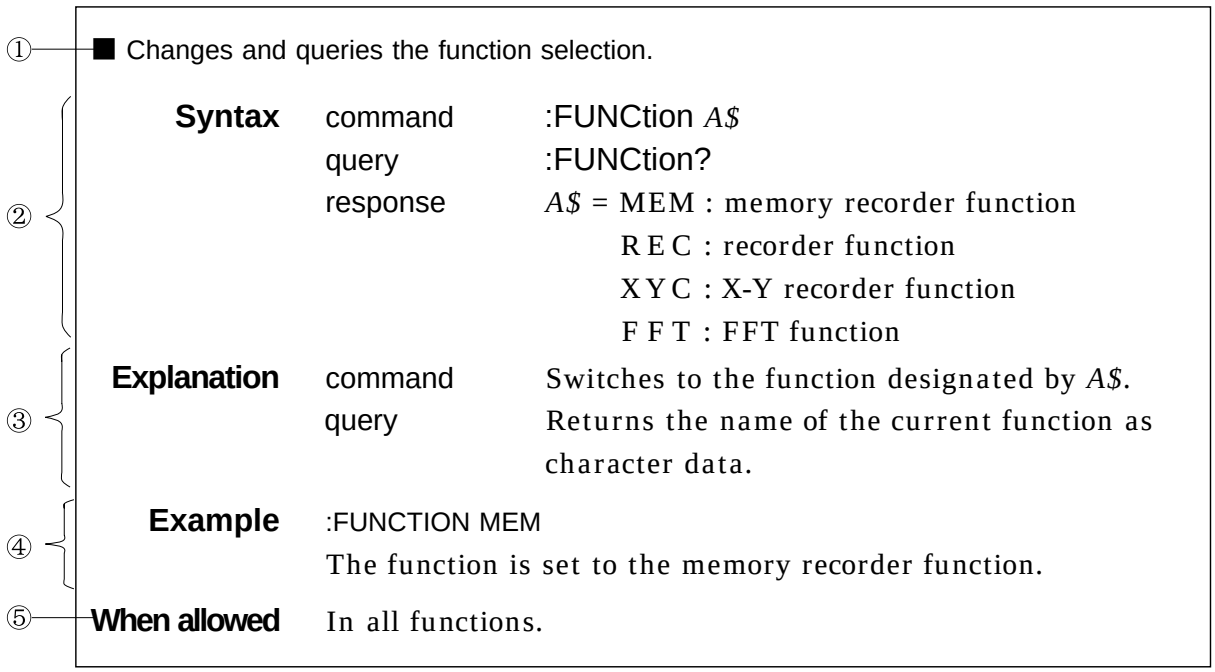
Command	Data (for a query, response data)	Explanation	Function	Ref page
:GRAPh				
:UNDO		Reverses the effect of the immediately previous editor command.	MEM FFT	138
:SAVE		Saves the decision area created with the editor.	MEM FFT	138
:POINt *X*, *Y*, *A*	*X* = x-coordinates, *Y* = y-coordinates *A* = 0, 1	Sets waveform decision area data.	MEM FFT	138
:POINt? *X*, *Y*	*X*, *Y*, *A* all NR1	Queries waveform decision area data.		

MEM memory recorder function REC recorder function
XYC X-Y recorder function FFT FFT function
ALL all MEM, REC, XYC, and FFT function.

4.2 Command Reference

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, A_B, C1)
"*A*", "*A$*",...	Character string data (e.g. "1.5", "mA")

The format of numerical data follows the formats <NR1>, <NR2>, and <NR3>, described "(8) Data format" in Section 3.3. If no format is mentioned, <NRf> format (i.e. any of the above) is accepted.

A <NR1>	Numerical parameter in NR1 format
B	Numerical parameter in NRf format

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

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

When the 8852 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, according to the setting made by the :HEADER command.

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
XYC	X-Y recorder function
FFT	FFT function
All	Any of the MEM, REC, XYC and FFT functions

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 8852 in its current state are not executed but generate execution errors. This happens, for example, when in recorder function it is attempted to execute an FFT mode setting.

Further, almost all commands cannot be executed during measurement operation.

4.3 Standard Commands Stipulated by IEEE 488.2

1. System data commands and queries

*IDN? command

■ Queries device ID.

Syntax

query	*IDN?	
response	HIOKI, 8852, 0, V1. 00	
	First field	Manufacturer's name
	Second field	Model name
	Third field	Serial number (not used: 0)
	Fourth field	Software version

2. Internal operation commands and queries

*RST command

■ Device initial setting.

Syntax command *RST

Explanation Initializes the 8852 (same as system reset).
However, it does not clear GP-IB related items.
(the event registers and the enable registers)
(the input buffer and the output queue)

*TST? command

■ Queries the result of the self-test.

Syntax

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

Explanation The result of the self-test of the 8852 is returned as an NR1 numerical value.

3. Synchronous commands and queries

*OPC command

■ 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? command

■ 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 command

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

Syntax command *WAI

Example :FUNC MEM; *WAI;:CONF:TDIV +500.0E-6
A$* *B$
The command following *WAI is not executed until the execution of the commands *A$* and *B$* is completed.

4. Status and event control commands and queries

*CLS command

■ 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. Accordingly, it also clears the status byte register. However, because it does not clear the output queue, it has no effect upon bit 4 (MAV) of the status byte.

*ESE command

■ Writes the standard event status enable register (SESER).

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? command

■ Reads the standard event status register (SESER).

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

■ 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 command

■ Writes the service request enable register (SRER).

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. The initial value (when the power is turned on) is 0.

Example *SRE 33

Bits 5 and 0 of SRER are set.

*SRE? command

■ Reads the service request enable register (SRER).

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? command

■ 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. However, bit 6 is not RQS, but is MSS. (Refer to the description of the status byte and the event register.)

:ESE0 command (Command specific to the 8852)

■ Writes event status enable register 0 (ESER0).

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? command (Command specific to the 8852)

■ 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? command (Command specific to the 8852)

■ 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.

4.4 Commands Specific to the 8852

1. Execution control commands (common to all functions)

■ Performs starting.

Syntax command :STARt

Explanation Same as the START key of the 8852.
Starts waveform sampling operation.

When allowed In all functions.

■ Performs stopping.

Syntax command :STOP

Explanation Same as the STOP key of the 8852.
Terminates at the instant that waveform sampling operation is completed. (With :STOP command, printer operation is not stopped, use :ABORT command to stop operation.)

When allowed In all functions.

■ Aborts processing.

Syntax command :ABORT

Explanation Same as the STOP key of the 8852. Forced halt. Terminates even if waveform sampling operation is not yet completed. Also stops printer operation.

When allowed In all functions.

■ Performs printing.

Syntax command :PRINt

Explanation Same as the PRINT key of the 8852.

When allowed In all functions.

■ Screen dump function.

Syntax command :HCOPy

Explanation Same as the COPY key of the 8852. Produces a hard copy of the screen.

When allowed In all functions.

■ Feeds printer paper.

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 in the data portion.

When allowed In all functions.

■ Performs automatic range setting.

Syntax command :AUTO

Explanation Same as the AUTO key of the 8852. Sets the time axis and the voltage axis automatically.

When allowed In the memory recorder function.

■ Queries the 8852 error number.

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

Explanation The number of error or warning that has occurred on the 8852 is returned in <NR1> as a numerical value. (See 8852 instruction manual appendix 1.)
If an error occurs during execution of :ERROR? then the error number is cleared.

When allowed In all functions.

■ Enables and disables headers, and queries header enablement.

Syntax

command	:HEADer *A$*
query	:HEADer?
response	*A$*

A$ = OFF, ON

Explanation

command	Sets header enablement. When headers are enabled, responses to queries are prefixed by headers; when headers are disabled, responses are not so prefixed.
query	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 When headers are disabled: response to :HEADER? is OFF.
When headers are enabled: response to :HEADER? is :HEADER ON.

When allowed In all functions.

■ Changes and queries the function selection.

Syntax

command	:FUNCtion *A$*
query	:FUNCtion?
response	*A$*

A$ = MEM : memory recorder function
REC : recorder function
XYC : X-Y recorder function
FFT : FFT function

Explanation

command	Switches to the function designated by *A$*.
query	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.

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

■ Sets and queries the time/div.

Syntax

command	:CONFigure:TDIV *A*
query	:CONFigure:TDIV?
response	*A* <NR3>

Explanation

command	Sets the time per division to a numerical value (unit seconds).
query	Returns the currently set value of the time per division as an NR3 numerical value.
	(If an attempt is made to set the time per division to a non-permitted value, it will be set to the next range above that value.)

Example

```
:CONFIGURE:TDIV +500.0E-6
```

Sets the time per division to 500 µs.

When allowed In the memory recorder function and the recorder function.

■ Sets and queries the recording length.

Syntax

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

Explanation

command	Sets the numerical value of the recording length (unit divisions).
query	Returns the currently set value of the recording length as an NR1 numerical value. (For the recorder function, 0 = CONT.)

Example

```
:CONFIGURE:SHOT 15
```

Sets the recording length to 15 divisions.

When allowed In the memory recorder function and the recorder function.

■ Sets and queries the format.

Syntax

command :CONFigure:FORMat *A$*
query :CONFigure:FORMat?
response *A$*
A$ = SINGle, DUAL, QUAD, XY : MEM
SINGle, DUAL, QUAD : REC
SINGle, DUAL : FFT

Explanation

command Sets the format.
query Returns the current format as character data.

Example

```
:CONFIGURE:FORMAT SINGLE
```

Sets the format to SINGLE.

When allowed In the memory recorder function, the recorder function, and the FFT function.

■ Sets and queries the interpolation function.

Syntax

command :CONFigure:DOTLine *A$*
query :CONFigure:DOTLine?
response *A$*
A$ = DOT, LINE

Explanation

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

Example

```
:CONFIGURE:DOTLINE DOT
```

Sets the interpolation function to DOT.

When allowed In the memory recorder function, the XY recorder function, and the FFT function.

■ Sets and queries the waveform superimposition function.

Syntax

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

Explanation

command Enables and disables screen waveform superimposition.
query Returns the current setting of the waveform superimposition enablement as character data.

Example

```
:CONFIGURE:OVWRITE ON
```

Sets the screen waveform superimposition to ON.

When allowed In the memory recorder function.

CONFigure

■ Sets and queries the auto print function.

Syntax

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

Explanation

command: Toggles the auto print function on and off.
query: Returns the current setting of the auto print function as character data.

Example

```
:CONFIGURE:ATPRINT ON
```

Sets the auto print function to ON.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries the auto save function.

Syntax

command :CONFigure:ATSAve *A$*
query :CONFigure:ATSAve?
response *A$*
A$ = OFF
FD: Auto save to the floppy disk
SCSI (HD) : Auto save to the SCSI device (HD, MO)

Explanation

command: Toggles the auto save function on and off.
query: Returns the current setting of the auto save function as character data.

Example

```
:CONFIGURE:ATSAVE FD
```

Sets the auto save function to the floppy disk to ON.

When allowed In the memory recorder function and the FFT function.

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

Syntax

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

Explanation

command	Enables and disables the smooth printing function.
query	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 the memory recorder function.

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

Syntax

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

Explanation

command	Enables and disables the roll mode function.
query	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 the memory recorder function.

■ Sets and queries the count for averaging.

Syntax

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

Explanation

command	Sets the count for averaging.
query	Returns the current setting of the count for averaging as an NR1 numerical value.

Example

```
:CONFIGURE:AVERAGE 32
```

Sets the count for averaging to 32.

When allowed In the memory recorder function and the FFT function.

CONFigure

■ Sets and queries memory segmentation.

Syntax

command :CONFigure:MEMDiv *A$*
query :CONFigure:MEMDiv?
response *A$*
A$ = OFF
SEQ : sequential save
MULTI : multi-block

Explanation

command: Sets the method of memory segmentation recording.
query: 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 the memory recorder function.

■ Sets and queries the number of memory blocks.

Syntax

command :CONFigure:MAXBlock *A*
query :CONFigure:MAXBlock?
response *A* <NR1>
A = 2 to 255

Explanation

command: Sets the number of memory blocks (memory segmentations).
query: Returns the current number of memory blocks as an NR1 numerical value.

Example

```
:CONFIGURE:MAXBLOCK 15
```

Sets the number of memory blocks to 15.

When allowed In the memory recorder function, when the multi-block function is in use. Query is also possible, when the sequential save function is in use.

■ Sets and queries the start block.

Syntax

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

Explanation

command	Sets the sequential save start block.
query	Returns the current start block as an NR1 numerical value.

Example

```
:CONFIGURE:STTBLOCK 1
```

Sets the start block to 1.

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

■ Sets and queries the end block.

Syntax

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

Explanation

command	Sets the sequential save end block.
query	Returns the current end block as an NR1 numerical value.

Example

```
:CONFIGURE:ENDBLOCK 15
```

Sets the end block to 15.

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

■ Sets and queries the memory block used.

Syntax

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

Explanation

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

Example

```
:CONFIGURE:USEBLOCK 15
```

Sets the block used to 15.

When allowed In the memory recorder function, when the memory segmentation function is in use.

CONFigure

■ Sets and queries the reference block.

Syntax

command :CONFigure:REFBlock *A*
query :CONFigure:REFBlock?
response *A* <NR1>
A = 1 to number of memory segmentations
0 : OFF

Explanation

command: In multi-block mode, sets the reference block ("ref block").
query: Returns the current reference block as an NR1 numerical value.

Example

```
:CONFIGURE:REFBLOCK 15
```

Sets the reference block to 15.

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

■ Enables and disables, and queries the waveform display in the sequencial save function.

Syntax

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

Explanation

command: Sets whether the waveform is displayed or not in the sequential save function.
query: Returns the current enablement state of the waveform display as character data.

Example

```
:CONFIGURE:SEQUDISP ON
```

Sets so that the waveform is displayed.

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

■ Sets and queries the waveform decision mode.

Syntax

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

Explanation

command: Sets the waveform decision mode.
query: Returns the current waveform decision mode as character data.

Example

```
:CONFIGURE:WVCOMP OUT
```

Sets the waveform decision mode to OUT.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries the waveform decision stop mode.

Syntax

command :CONFigure:CMPStop *A$*
query :CONFigure:CMPStop?
response *A$*
A$ = GO, NG, G_N

Explanation

command: Sets the stop mode during waveform decision.
query: Returns the current stop mode as character data.

Example

```
:CONFIGURE:CMPSTOP GO
```

Sets the stop mode during waveform decision to GO.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries printer output.

Syntax

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

Explanation

command: Sets the printer output.
query: 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 the recorder function.

CONFigure

■ Sets and queries the FFT frequency range.

Syntax

command	:CONFigure:MAXFreq *A*
query	:CONFigure:MAXFreq?
response	*A* <NR3>

Explanation

command	Sets the frequency range as a numerical value (unit: Hz).
query	Returns the currently set frequency range as a numerical value in <NR3> format.
	If an attempt is made to set an unacceptable value, then the frequency range is set to the next higher value.

Example

```
:CONFIGURE:MAXFREQ 100
```

The frequency range is set to 100 Hz.

When allowed In the FFT function.

■ Sets and queries the FFT window function.

Syntax

command	:CONFigure:FFTWind *A$*
query	:CONFigure:FFTWind?
response	*A$*
	A$ = RECTan : rectangular window
	HANNing : hanning window

Explanation

command	Sets the window function as indicated by *A$*.
query	Returns the current window function as character data.

Example

```
:CONFIGURE:FFTWIND HANNING
```

The window function is set to hanning window.

When allowed In the FFT function.

■ Sets and queries the FFT reference data.

Syntax

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

Explanation

command	Designates the FFT reference data as specified by *A$*.
query	Returns the current FFT reference data as character data.

Example

```
:CONFIGURE:FFTREF NEW
```

New data is used as FFT data.

When allowed In the FFT function.

■ Sets and queries the FFT analysis mode.

Syntax

command :CONFigure:FFTMode *G$, A$*
query :CONFigure:FFTMode? *G$*
response *G$, A$*
G$ = G1, G2 : graph 1, graph 2
A$ = STORage : stored waveform
PSPMDB : power spectrum (indicated in decibels)
PSPMAG : power spectrum
LINMAG : linear spectrum (magnitude)
LINIMG : linear spectrum (imaginary axis amplitude)
LINREAL : linear spectrum (real axis amplitude)
PHASE : phase
HISTogram : histogram

Explanation

command Sets the FFT analysis mode.
query Returns the current FFT analysis mode as character data.
G2 can be designated even if the display format is SINGLE, but this does not affect the display.

Example

```
:CONFIGURE:FFTMODE G1, PSPMDB
```

The FFT analysis mode for graph 1 is set to power spectrum (indicated in decibels).

When allowed In the FFT function.

CONFigure

■ Sets and queries the FFT x-axis.

Syntax

command :CONFigure:FFTXaxis *G$, A$*
query :CONFigure:FFTXaxis? *G$*
response *G$, A$*
G$ = G1, G2 : graph 1, graph 2
A$ = LINhz, LOGhz : linear frequency axis, logarithm
(time) : in the FFT analysis mode; stored waveform (response only)
(volt) : in the FFT analysis mode; histogram (response only)

Explanation

command Sets the x-axis of the graph number designated by *A$*.
query Returns the present x-axis setting as character data.
When the analysis mode is stored waveform or histogram, the setting is not available.

Example

```
:CONFIGURE:FFTXAXIS G1, LINHZ
```

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

When allowed In the FFT function.

■ Sets and queries the FFT display scaling method.

Syntax

command :CONFigure:FFTSCale *G$, A$*
query :CONFigure:FFTSCale? *G$*
response *G$, A$*
G$ = G1, G2 : graph 1, graph 2
A$ = AUTO, MANUal

Explanation

command Sets the display scaling method for the graph number designated by *G$*.
query Returns the current display scaling method for the graph number designated by *G$* as character data.

Example

```
:CONFIGURE:FFTSCALE G1, AUTO
```

The scaling method for graph 1 is set to automatic.

When allowed In the FFT function.

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

Syntax

command	:CONFigure:FFTUp *G$, A*
query	:CONFigure:FFTUp? *G$*
response	*G$, A* <NR3>
	G$ = G1, G2 : graph 1, graph 2
	A = -9.999E+9 to +9.999E+9

Explanation

command	Sets the FFT display scale vertical axis upper limit for the graph number designated by *G$* to the value designated by *A*.
query	Returns the current FFT display scale vertical axis upper limit for the graph number designated by *G$* 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 the FFT function.

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

Syntax

command	:CONFigure:FFTLow *G$, A*
query	:CONFigure:FFTLow? *G$*
response	*G$, A* <NR3>
	G$ = G1, G2 : graph 1, graph 2
	A = -9.999E+9 to +9.999E+9

Explanation

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

Example

```
:CONFIGURE:FFTLOW G2, 100
```

The FFT display scale vertical axis lower limit for graph 2 is set to 100.

When allowed In the FFT function.

CONFigure

■ Sets and queries the FFT data printer output style.

Syntax

command: :CONFigure:FFTPrint *A$*

query: :CONFigure:FFTPrint?

response: *A$*

A$ = WAVE : waveform data
DATA : numerical data

Explanation

command: Sets the printer output style to be waveform or logging (numerical data).

query: Returns the current setting of the printer output style.

Example

```
:CONFIGURE:FFTPRINT WAVE
```

Sets the printer output style to be waveform.

When allowed

In the FFT function.

3. TRIGger command (Sets and queries trigger source, level, etc.)

■ Sets and queries switching the trigger screen.

Syntax

command: :TRIGger:PAGE *A$*

query: :TRIGger:PAGE?

response: *A$*

A$ = NORMal : trigger screen source(NORMAL)
SPECial : trigger screen source(SPECIAL)

Explanation

command: Sets switching the trigger screen source(NORMAL) or source(SPECIAL).

query: Returns the current trigger screen as character data.

When allowed

In the memory recorder function and the FFT function.

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

Syntax

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

Explanation

command: Sets the logical operator determining whether the internal and external triggers are ORed or ANDed.
query: 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 the kind of trigger.

Syntax

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

・On the trigger screen SOURCE(NORMAL)
ch$ = CH1 to CH4
A$ = OFF
LEVEl : level trigger
LOGIc : logic trigger
WINDow : window trigger
TIMEout : time out trigger (in MEM and FFT only)
GLITch : glitch detection trigger (in MEM and FFT only)

・On the trigger screen SOURCE(SPECIAL)
(in MEM and FFT only)
ch$ = S1, S2
A$ = OFF
EVENt : event trigger
DELAy : delay trigger

Explanation

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

Example

```
:TRIGGER:KIND CH1, LEVEL
```

Sets channel 1 to level trigger.

When allowed In all functions.

:TRIGger

■ Sets and queries trigger level.

Syntax

command	:TRIGger:LEVEl *ch$, A*
query	:TRIGger:LEVEl? *ch$*
response	*ch$, A* <NR1>
	ch$ = CH1 to CH4
	A = 0 to 100 (%)

Explanation

command	Sets the trigger level of the level, glitch detection, or time out trigger, of the channel designated by *ch$*.
query	Returns the current trigger level as an NR1 numerical value.

Example

```
:TRIGGER:LEVEL CH1, 50
```

Sets the trigger level of channel 1 to 50%.

When allowed In all functions.

■ Sets and queries trigger direction (slope).

Syntax

command	:TRIGger:SLOPe *ch$, A$*
query	:TRIGger:SLOPe? *ch$*
response	*ch$, A$*
	ch$ = CH1 to CH4
	A$ = UP (rising: ↑)
	DOWN (falling: ↓)

Explanation

command	Sets the trigger direction of the level, time out, or glitch detection trigger, of the channel designated by *ch$*.
query	Returns the current trigger direction as a character value.

Example

```
:TRIGGER:SLOPE CH1, UP
```

Sets the trigger direction of channel 1 to rising.

When allowed In all functions.

■ Sets and queries time out width or glitch width.

Syntax

command	:TRIGger:WIDTh *ch$, A$*
query	:TRIGger:WIDTh? *ch$*
response	*ch$, A*
	ch$ = CH1 to CH4
	A = 2 to 4000

Explanation

command	Sets the glitch width or width for a time out trigger (time out width) of the channel designated by *ch$*.
query	Returns the current glitch width or time out width as an NR1 numerical value.

Example

```
:TRIGGER:WIDTH CH1, 100
```

Sets the glitch width of channel 1 to 100 samples.

When allowed In the memory recorder and the FFT function.

■ Sets and queries the filter.

Syntax

command	:TRIGger:FILTer *ch$, A$*
query	:TRIGger:FILTer? *ch$*
response	*ch$, A$*
	ch$ = CH1, CH2
	A$ = OFF, ON

Explanation

command	Enables or disables the filter of a level or logic trigger of the channel designated by *ch$*.
query	Returns the current filter enablement state as character data.

Example

```
:TRIGGER:FILTER CH1, ON
```

Sets the filter of channel 1 to ON.

When allowed In the memory recorder and the FFT function.

■ Sets and queries the filter width.

Syntax

command	:TRIGger:FILTWidth *ch$, A*
query	:TRIGger:FLITWidth? *ch$*
response	*ch$, A* <NR1>
	ch$ = CH1, CH2
	A = 2 to 4000

Explanation

command	Sets the filter width for a level or logic trigger of the channel designated by *ch$*.
query	Returns the current filter width as an NR1 numerical value.

Example

```
:TRIGGER:FILTWIDTH CH1, 10
```

Sets the filter width of channel 1 to 10.

When allowed In the memory recorder function and the FFT function.

TRIGger

■ Sets and queries upper limit level for a window trigger.

Syntax

command	:TRIGger:UPPEr *ch$, A*
query	:TRIGger:UPPEr? *ch$*
response	*ch$, A* <NR1>
	ch$ = CH1, CH2
	A = 1 to 100 (%)

Explanation

command	Sets the upper limit level of the window trigger of the channel designated by *ch$*.
query	Returns the current upper limit value of the window trigger as an NR1 numerical value.

Example

```
:TRIGGER:UPPER CH1, 80
```

Sets the upper limit level of the window trigger of channel 1 to 80%.

When allowed In all functions.

■ Sets and queries lower limit level for a window trigger.

Syntax

command	:TRIGger:LOWEr *ch$, A*
query	:TRIGger:LOWEr? *ch$*
response	c*h$, A* <NR1>
	ch$ = CH1, CH2
	A = 0 to 99 (%)

Explanation

command	Sets the lower limit level of the window trigger of the channel designated by *ch$*.
query	Returns the current lower limit value of the window trigger as an NR1 numerical value.

Example

```
:TRIGGER:LOWER CH1, 20
```

Sets the lower limit level of the window trigger of channel 1 to 20%.

When allowed In all functions.

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

Syntax

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

ch$ = CH1 to CH3
A$ = XXXXXXXX : trigger pattern (X, 0, 1)

Explanation

command	Sets the trigger pattern for the logic trigger of the channel designated by *ch$* to that specified by the given character data. (Characters other than X, 0 and 1 are X.)
query	Returns the current trigger pattern for the logic trigger as that specified by the given character data. Double quotation marks (") can be used instead of single quotation marks (').

Example

```
:TRIGGER:LOGPAT CH1, '10XX10XX'
```

Sets the trigger pattern for channel 1 to '10XX10XX'.

When allowed In all functions.

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

Syntax

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

ch$ = CH1 to CH3
A$ = OR, AND

Explanation

command	Sets the AND/OR logical operator for the trigger pattern of a logic trigger.
query	Returns the present AND/OR setting as a character string.

Example

```
:TRIGGER:LOGAND CH1, OR
```

Sets the AND/OR logical operator for the trigger pattern of channel 1 to OR.

When allowed In all functions.

:TRIGger

■ Sets and queries the source channels for a special trigger.

Syntax

command :TRIGger:SPCHannel *A$, B$, C$*
query :TRIGger:SPCHannel? *A$*
response *A$, B$, C$*
A$ = S1, S2 : trigger source S1, S2
B$ = OFF, CH1 to CH4 : start channel
(OFF; event trigger only)
C$ = CH1 to CH4 : count or check channel

Explanation

command: Sets the start channel and count or check channel for the special trigger (event or delay) of the trigger source designated by *A$*.
query: Returns as character data the current settings for the start channel and count or check channel for the special trigger.

Example

```
:TRIGGER:SPCHANNEL S1, CH1, CH2
```

Sets the start channel to channel 1, and the count or check channel to channel 2 for the trigger source S1 .

When allowed In the memory recorder function and the FFT function.

■ Sets and queries levels of a special trigger.

Syntax

command :TRIGger:SPLEvel *A$, B, C*
query :TRIGger:SPLEvel? *A$*
response *A$, B* <NR1>, C <NR1>
A$ = S1, S2 : trigger source S1, S2
B = 0 to 100 (%) : trigger level of the start channel
C = 0 to 100 (%) : trigger level of the count or check channel

Explanation

command: Sets the trigger levels of the start channel and count or check channel for the special trigger (event or delay) of the trigger source designated by *A$*.
query: Returns as NR1 numerical values the current trigger levels of the start channel and count or check channel for the special trigger.

Example

```
:TRIGGER:SPLEVEL S1, 30, 70
```

Sets the trigger level of the start channel to 30%, and the trigger level of the count or check channel to 70% for the trigger source S1.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries trigger directions (slopes) of a special trigger.

Syntax

command :TRIGger:SPSLope *A$, B$, C$*

query :TRIGger:SPSLope? *A$*

response *A$, B$, C$*

A$ = S1, S2 : trigger source S1, S2

B$ = UP : start channel rising

DOWN : start channel falling

C$ = UP : count or check channel rising

DOWN : count or check channel falling

Explanation

command: Sets the trigger directions of the start channel and count or check channel for the special trigger (event or delay) of the trigger source designated by *A$*.

query: Returns as character data the current trigger directions of the start channel and count or check channel for the special trigger.

Example

```
:TRIGGER:SPSLOPE S1, UP, DOWN
```

Sets the trigger direction of the start channel to rising, and the trigger direction of the count or check channel to falling for the trigger source S1.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries count for an event trigger.

Syntax

command :TRIGger:SPCOunt *A$, B*

query :TRIGger:SPCOunt? *A$*

response *A$, B* <NR1>

A$ = S1, S2 : trigger source S1, S2

B = 2 to 4000 : number of counts for the event trigger

Explanation

command: Sets the number of counts for the event trigger of the trigger source designated by *A$*.

query: Returns the current number of counts for the event trigger as an NR1 numerical value.

Example

```
:TRIGGER:SPCOUNT S1, 10
```

Sets to 10 the number of counts for the event trigger of the trigger source S1.

When allowed In the memory recorder function and the FFT function.

:TRIGger

■ Sets and queries delay width of a delay trigger.

Syntax

command :TRIGger:DELAy *A$, B$, C*
query :TRIGger:DELAy? *A$*
response *A$, B$, C* <NR1>
A$ = S1, S2 : trigger source S1, S2
B$ = GREATer : A → B > t
LESS : A → B ≦ t
C = 2 to 4000 : delay width

Explanation

command: Sets the delay trigger of the trigger source designated by *A$* to the triggering method designated by *B$*, and sets the delay width.

query: Returns the triggering method for the delay trigger of the trigger source designated by *A$* as character data, and the delay width as an NR1 numerical value.

Example

```
:TRIGGER:DELAY S1, LESS, 100
```

Sets the delay trigger of the trigger source S1 to A → B ≦ t, and the delay width to 100.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries a TV trigger.

Syntax

command :TRIGger:TVTRigger *A$*
query :TRIGger:TVTRigger?
response *A$*
A$ = OFF
CH1 : positive synchronous signal
CH2 : negative synchronous signal

Explanation

command: Sets the source channel for the TV trigger. Input the video signal to channel 1 if it is positive synchronous, and to channel 2 if it is negative synchronous.

query: Returns the current setting for the source channel for the TV trigger as character data.

Example

```
:TRIGGER:TVTRIGGER CH2
```

Sets the source channel for the TV trigger to channel 2.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries NTSC/PAL for the TV trigger.

Syntax

command :TRIGger:TVFOrmat *A$*
query :TRIGger:TVFOrmat?
response *A$*
A$ = NTSC, PAL

Explanation

command Sets NTSC/PAL for the TV trigger.
query Returns the current NTSC/PAL setting as character data.

Example

```
:TRIGGER:TVFORMAT NTSC
```

Sets the TV trigger to be activated in the NTSC signal.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries the field for the TV trigger.

Syntax

command :TRIGger:TVFIeld *A*
query :TRIGger:TVFIeld?
response *A* <NR1>
A = 1 : odd number field
2 : even number field

Explanation

command Sets the field for the TV trigger.
query Returns the current field setting as an NR1 numerical value.

Example

```
:TRIGGER:TVFIELD 1
```

Sets the TV trigger to be activated in the odd number field.

When allowed In the memory recorder function and the FFT function.

■ Sets and queries the line number for the TV trigger.

Syntax

command :TRIGger:TVLIne *A*
query :TRIGger:TVLIne?
response *A* <NR1>
A = 1 to 263 (NTSC)
1 to 313 (PAL)

Explanation

command Sets the line number for the TV trigger.
query Returns the current line number setting as an NR1 numerical value.

Example

```
:TRIGGER:TVLINE 20
```

Sets the TV trigger to be activated in the 20th line.

When allowed In the memory recorder function and the FFT function.

TRIGger

■ Sets and queries external trigger.

Syntax

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

Explanation

command	Enables and disables external trigger.
query	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.

■ Sets and queries whether the timer trigger is on or off.

Syntax

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

Explanation

command	Enables or disables the timer trigger.
query	Returns the current enablement state of the timer trigger as character data.

Example

```
:TRIGGER:TIMER ON
```

Sets the timer trigger to ON.

When allowed In all functions.

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

Syntax

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

Explanation

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

Example

```
:TRIGGER:TMSTART 7, 5, 9, 30
```

Sets the start instant for the timer trigger to 09:30 on July 5th.

When allowed In all functions.

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

Syntax

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

Explanation

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

Example

```
:TRIGGER:TMSTOP 7, 5, 10, 30
```

Sets the stop instant for the timer trigger to 10:30 on July 5th.

When allowed In all functions.

:TRIGger

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

Syntax

command	:TRIGger:TMINTvl *hour, min, sec*
query	:TRIGger:TMINTvl?
response	*hour* <NR1>, *min* <NR1>, *sec* <NR1>
	hour = 0 to 23
	min = 0 to 59
	sec = 0 to 59

Explanation

command	Sets the time interval for the timer trigger.
query	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 trigger mode.

Syntax

command	:TRIGger:MODE *A$*
query	:TRIGger:MODE?
response	*A$*
	A$ = SINGle, REPEat, AUTO : MEM, FFT
	SINGle, REPEat : REC

Explanation

command	Sets the trigger mode.
query	Returns the current trigger mode as character data.

Example

```
:TRIGGER:MODE REPEAT
```

Sets the trigger mode to repeat.

When allowed In the memory recorder function, the recorder function, and the FFT function.

■ Sets and queries pre-trigger.

Syntax

command :TRIGger:PRETrig *A*
query :TRIGger:PRETrig?
response *A* <NR1>
A = 0, 2, 5, 10, 20,..., 80, 90, 95, 100 (unit %)
-950 to -50; 50% step

Explanation

command: Sets pre-trigger value to a numerical value (in percent). If an attempt is made to set a value which cannot be set on the 8852, setting is performed to the next higher permitted value.

query: 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 the memory recorder function and the FFT function.

■ Sets and queries trigger timing.

Syntax

command :TRIGger:TIMIng *A$*
query :TRIGger:TIMIng?
response *A$*
A$ = START
STOP
S_S (START&STOP)

Explanation

command: Sets the trigger timing.

query: The currently set trigger timing is returned as a character string.

Example

```
:TRIGGER:TIMING START
```

Sets the trigger timing to START.

When allowed In the recorder function and the X-Y recorder function.

:TRIGger

■ Queries the time point for trigger detection.

Syntax

query :TRIGger:TRGTime? (*A*)

response *hour* <NR1>, *min* <NR1>, *sec* <NR1>

A = block number during memory segmentation
hour = 0 to 23
min = 0 to 59
sec = 0 to 59

Explanation

query

Returns the currently set time point for trigger detection as a numerical value in NR1 format.
During memory segmentation, returns the time point for trigger detection in the memory block whose block number is specified.

Example

```
:TRIGGER:TRGTIME?
```

The currently set time point for trigger detection is queried.

When allowed In all functions.

■ Queries the date for trigger detection.

Syntax

query :TRIGger:TRGDate? (*A*)

response *year* <NR1>, *month* <NR1>, *day* <NR1>

A = block number during memory segmentation
year = 0 to 99
month = 1 to 12
day = 1 to 31

Explanation

query

Returns the currently set date for trigger detection as a numerical value in NR1 format.
During memory segmentation, returns the date for trigger detection in the memory block whose block number is specified.

Example

```
:TRIGGER:TRGDATE?
```

The currently set date for trigger detection is queried.

When allowed In all functions.

4. UNIT command (Sets and queries input channel (voltage axis range, filter etc.).)

■ Sets and queries switching between analog and logic.

Syntax

command	:UNIT:TYPE *ch$, A$*
query	:UNIT:TYPE? *ch$*
response	*ch$, A$*
	ch$ = CH1 to CH3
	A$ = ANALog : used as analog channel
	LOGIc : used as logic channel

Explanation

command	Sets switching between analog and logic for the channel designated by *ch$*.
query	Returns the current analog or logic setting as character data.

Example

```
:UNIT:TYPE CH1, ANALOG
```

Sets channel 1 as analog channel.

When allowed In all functions.

■ Sets and queries the voltage axis range of an input channel.

Syntax

command	:UNIT:RANGe *ch$, A*
query	:UNIT:RANGe? *ch$*
response	*ch$, A* <NR3>
	ch$ = CH1 to CH4
	A = voltage axis range (unit V)

Explanation

command	Sets the voltage axis range for the channel designated by *ch$* to a numerical value.
query	Returns the current voltage axis range for the channel designated by *ch$* as an NR3 numerical value.

Example

```
:UNIT:RANGE CH1, +20.E-3
```

Sets the voltage axis range for channel 1 to 20 mV.

When allowed In all functions.

UNIT

■ Sets and queries input channel origin position.

Syntax

command	:UNIT:POSItion *ch$, A*
query	:UNIT:POSItion? *ch$*
response	*ch$, A* <NR1>
	ch$ = CH1 to CH4
	A = -100 to 100 (%)

Explanation

command	Sets the origin position for the channel designated by *ch$* in the range.
query	Returns the current origin position for the channel designated by *ch$* 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 input coupling for an input channel.

Syntax

command	:UNIT:COUPling *ch$, A$*
query	:UNIT:COUPling? *ch$*
response	*ch$, A$*
	ch$ = CH1 to CH4
	A$ = GND, AC, DC

Explanation

command	Sets the input coupling for the channel designated by *ch$*.
query	Returns the current input coupling for the channel designated by *ch$* 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 the filter for an input channel.

Syntax

command	:UNIT:FILTer *ch$, A*
query	:UNIT:FILTer? *ch$*
response	*ch$, A* <NR3>
	ch$ = CH1 to CH4
	A = 0, 2.0E7, 10 (0 : OFF)

Explanation

command	Sets the filter for the channel designated by *ch$*.
query	Returns the current filter setting for the channel designated by *ch$* as an NR3 numerical value.

Example

```
:UNIT:FILTER CH1, 10
```

Sets the filter for channel 1 to 10 Hz.

When allowed In all functions.

■ Carries out zero adjustment for the input units.

Syntax

command	:UNIT:ADJUST

Explanation

command	Carries out zero adjustment for the input units.

When allowed In all functions.

■ Sets and queries the logic threshold level.

Syntax

command	:UNIT:LOGLevel *ch$, A*
query	:UNIT:LOGLevel? *ch$*
response	*ch$, A* <NR2>
	ch$ = CH1 to CH3
	A = -6.2 to 6.2 (unit V)

Explanation

command	Sets the logic threshold level for the channel designated by *ch$*.
query	Returns the current threshold level setting for the channel designated by *ch$* as an NR2 numerical value.

Example

```
:UNIT:LOGLEVEL CH1, 2.5
```

Sets the logic threshold level for channel 1 (channel A) to 2.5 V.

When allowed In all functions.

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

■ Sets and queries the screen mode.

Syntax

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

A$ = SYSTem
STATus
TRIGger
DISPlay

Explanation

command	Changes the screen mode.
query	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 waveform display style.

Syntax

command	:DISPlay:DRAWing *ch$, A$*
query	:DISPlay:DRAWing? *ch$*
response	*ch$, A$*

ch$ = CH1 to CH4
A$ = OFF, LIGHt, DARK

Explanation

command	Sets the waveform display style for the channel designated by *ch$* to OFF, LIGHT (low intensity), or DARK (high intensity).
query	Returns the current waveform display style setting for the channel designated by *ch$* as character data.

Example

```
:DISPLAY:DRAWING CH1, DARK
```

Displays the channel 1 waveform the DARK.

When allowed In the memory recorder function, the recorder function, and the X-Y recorder function.

DISPlay

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

Syntax

command :DISPlay:PAGE *A*

query :DISPlay:PAGE?

response *A* <NR1>

・On status screen

A = 1 : page 1

2 : page 2 (no X-Y recorder function)

・On system screen

A = 1 : INITIALIZE

2 : SCALING

3 : COMMENT

4 : SETUP

5 : INTERFACE

6 : CRT COPY

7 : PROBE RATIO

8 : SELF CHECK

Explanation

command: Changes over the page of the status or system screen according to the corresponding numerical value.

query: Returns the current page of the status or system screen as a corresponding NR1 numerical value.

Example

```
:DISPLAY:CHANGE SYSTEM
:DISPLAY:PAGE 4
```

Changes over the SETUP screen accessed from the system screen.

When allowed In all functions.

■ Sets and queries waveform display graph in DUAL and QUAD format.

Syntax

command :DISPlay:GRAPh *ch$, G$*

query :DISPlay:GRAPh? *ch$*

response *ch$, G$*

ch$ = CH1 to CH4

G$ = G1, G2 : graph 1, graph 2

Explanation

command: Sets the waveform display graph on the screen.

query: On the screen, returns the current waveform display graph for a channel as character data.

Example

```
:DISPLAY:GRAPH CH1, G1
```

Displays the channel 1 waveform in display graph 1.

When allowed In the memory recorder function and the recorder function.

DISPlay

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

Syntax

command :DISPlay:XMAG *A$*
query :DISPlay:XMAG?
response *A$*
MEM:
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, X1_20000 (*), X1_40000 (*) (*: 8852-01only)
REC:
A$ = X1, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, X1_200

Explanation

command: Sets the magnification/compression factor on the time axis according to character data. When the zoom function is used, sets the magnification/compression factor on the time axis for the lower graph.
query: 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 the memory recorder function and the recorder function.

■ Enables and disables, and queries the zoom function.

Syntax

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

Explanation

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

Example

```
:DISPLAY:ZOOM ON
```

Enables the zoom function.

When allowed In the memory recorder function.

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

Syntax

command :DISPlay:ZOOMMag *A$*
query :DISPlay:ZOOMMag?
response *A$*
A$ = 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, X1_20000 (*), X1_40000 (*)
(*: 8852-01only)

Explanation

command: Sets the magnification/compression factor on the time axis for the upper graph, when the zoom function is used.
query: Returns as character data the current magnification/compression factor on the time axis for the upper graph in the zoom function.

Example

```
:DISPLAY:ZOOMMAG X1_100
```

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

When allowed In the memory recorder function.

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

Syntax

command :DISPlay:YMAG *ch$, A$*
query :DISPlay:YMAG? *ch$*
response *ch$, A$*
ch$ = CH1 to CH4
A$ = X1_2, X1, X2

Explanation

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

Example

```
:DISPLAY:YMAG X2
```

Sets the magnification ratio along the voltage axis to be X2.

When allowed In the memory recorder function and the recorder function.

DISPlay

■ Sets and queries waveform display position on the voltage axis.

Syntax

command :DISPlay:YZOOm *ch$, A*
query :DISPlay:YZOOm? *ch$*
response *ch$, A* <NR1>
ch$ = CH1 to CH4
A = 1 to 100 (%)

Explanation

command: Sets the waveform display position on the voltage axis. Sets the percentage of the position displayed in the center of the display screen with respect to the full scale.

query: Returns the current waveform display position on the voltage axis as an NR1 numerical value.

Explanation

```
:DISPLAY:YZOOM CH1, 40
```

Displays the position of 40% of the full scale on channel 1 in the center of the display screen.

When allowed In the memory recorder function and the recorder function.

■ Performs waveform display.

Syntax

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

Explanation

command: Displays the waveform on the screen from the position indicated by *A$*.
Displays the waveform from the position of the last 60 points within the last 60 points of data.

Example

```
:DISPLAY:WAVE ACUR
```

Displays the waveform from the position of A cursor.

When allowed In the memory recorder function (when *A$* = ACUR, the A cursor must be displayed).

■ Enables and disables the memory segmentation screen.

Syntax

command :DISPlay:DIVMap *A$*
query :DISPlay:DIVMap?
response *A$*
A$ = ON : Enter the memory segmentation screen.
OFF : Exit from the memory segmentation screen.

Explanation

command: Enables and disables the memory segmentation screen.
query: Returns the current memory segmentation screen enablement as character data.

Example

```
:DISPLAY:DIVMAP ON
```

Displays the memory segmentation screen.

When allowed

In the memory recorder function (on page 2 of the status screen and the memory must be segmented).

■ Enables and disables the waveform processing calculation setting screen.

Syntax

command :DISPlay:CALCEdit *A$*
query :DISPlay:CALCEdit?
response *A$*
A$ = ON : Enter the waveform processing calculation setting screen.
OFF : Exit from the waveform processing calculation setting screen.

Explanation

command: Enables and disables the waveform processing calculation setting screen.
query: Returns the current waveform processing calculation setting screen enablement as character data.

Example

```
:DISPLAY:CALCEDIT ON
```

Displays the waveform processing calculation setting screen.

When allowed

In the memory recorder function (on page 2 of the status screen).

DISPlay

■ Enables and disables the waveform parameter calculation setting screen.

Syntax

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

A$ = ON : Enter the waveform parameter calculation setting screen.
OFF : Exit from the waveform parameter calculation setting screen.

Explanation

command	Enables and disables the waveform parameter calculation setting screen.
query	Returns the current waveform parameter calculation setting screen enablement as character data.

Example

```
:DISPLAY:MEASEDIT ON
```

Displays the waveform parameter calculation setting screen.

When allowed In the memory recorder function (on page 2 of the status screen).

■ Sets and queries the X-axis, in the X-Y format.

Syntax

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

ch$ = CH1 to CH4

Explanation

command	Sets the X-axis channel in the X-Y format.
query	Returns the current X-axis channel in the X-Y format.

Example

```
:DISPLAY:XAXIS CH1
```

Sets channel 1 to the X-axis.

When allowed In the memory recorder function (in X-Y format) and in the X-Y recorder function.

■ Sets and queries display clearing in the X-Y recorder function.

Syntax

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

Explanation

command	Enables or disables display clearing in the X-Y recorder function.
query	In the X-Y recorder function, returns the enablement of display clearing as character data.

Example

```
:DISPLAY:XYCLR ON
```

Sets the display clearing to ON.

When allowed In the X-Y recorder function.

■ Sets and queries the FFT analysis channel.

Syntax

command	:DISPlay:FFTCH *G$, ch$*
query	:DISPlay:FFTCH? *G$*
response	*G$, ch$*
	G$ = G1, G2 : graph 1, graph 2
	ch$ = CH1 to CH4

Explanation

command	Sets the FFT analysis channel for the graph designated by *G$*.
query	Returns the current FFT analysis channel for the graph designated by *G$* as character data.

Example

```
:DISPLAY:FFTCH G1, CH1
```

Sets the FFT analysis channel for graph 1 to channel 1.

When allowed In the FFT function.

6. CURSor command (Cursor setting and reading)

■ Turns on and off, and queries, the A and B cursors.

Syntax

command :CURSor:MODE *A$*
query :CURSor:MODE?
response *A$*
A$ = OFF, TIME, VOLT, TRACe : MEM
OFF, Xcur, Ycur : MEM (XY format)
OFF, TIME, VOLT : REC
OFF, Xcur, Ycur : XYC
OFF, ON : FFT

Explanation

command: Sets the A and B cursor type (vertical cursor, horizontal cursor, trace cursor). TIME and Xcur relate to the vertical cursor, VOLT and Ycur relate to the horizontal cursor, and TRACe relates to the trace cursor. In the FFT function, ON relates to the trace cursor enablement.
query: Returns the current A and B cursor type as character data.

Example

```
:CURSOR:MODE TIME
```

Sets vertical cursors.

When allowed In all functions.

■ Selects between, and queries, A only or A and B cursors.

Syntax

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

Explanation

command: Selects between A only or A and B cursors.
query: Returns whether currently the A cursor only or both A and B cursors are in use, as character data.

Example

```
:CURSOR:ABCURSOR A
```

Sets A cursor.

When allowed In the memory recorder function, the recorder function, and the X-Y recorder function.

■ Sets and queries the channel for the A cursor.

Syntax

command	:CURSor:ACHAnnel *ch$*
query	:CURSor:ACHAnnel?
response	*ch$*
	ch$ = CH1 to CH4

Explanation

command	Sets the channel for the A cursor.
query	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.

■ Sets and queries the channel for the B cursor.

Syntax

command	:CURSor:BCHAnnel *ch$*
query	:CURSor:BCHAnnel?
response	*ch$*
	ch$ = CH1 to CH4

Explanation

command	Sets the channel for the B cursor.
query	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.

■ Sets and queries the FFT trace cursor readout value as peak or RMS value.

Syntax

command	:CURSor:YDISp *A$*
query	:CURSor:YDISp?
response	*A$*
	A$ = PEAK, RMS

Explanation

command	Sets the FFT trace cursor readout value as peak or RMS value.
query	Returns the current FFT trace cursor readout value setting as character data.

Example

```
:CURSOR:YDISP RMS
```

Sets the FFT trace cursor readout value as RMS value.

When allowed In the FFT function.

CURSor

■ Sets and queries the position of the A cursor.

Syntax

command :CURSor:APOSition *A*
query :CURSor:APOSition?
response *A* <NR1>
(vertical cursor, trace cursor)
A = 0 to (number of stored data values)
(40 × recording length) : MEM
A = 0 to (number of stored data values) : REC
A = 0 to 400 : XYC, MEM (XY format)
(horizontal cursor)
A = 0 to 250 : MEM, REC, XYC

Explanation

command Sets the A cursor position (refer to next page).
query Returns the current A cursor position as an NR1 numerical value.

Example

```
:CURSOR:APOSITION 400
```

Move the A cursor position to 400 points (10DIV).

When allowed In the memory recorder function, the recorder function, and the X-Y recorder function.

■ Sets and queries the position of the B cursor.

Syntax

command :CURSor:BPOSition *A*
query :CURSor:BPOSition?
response *A* <NR1>
A is the same as in APOSition.

Explanation

command Sets the B cursor position (refer to next page).
query Returns the current B cursor position as an NR1 numerical value.

Example

```
:CURSOR:BPOSITION 400
```

Move the B cursor position to 400 points (10DIV).

When allowed In the memory recorder function, the recorder function, and the X-Y recorder function.

CURSor

■ Sets and queries the position of the FFT trace cursor.

Syntax

command	:CURSor:FPOSition *A*
query	:CURSor:FPOSition?
response	*A* <NR1>

A = 0 to 799 : analysis mode; STORAGE
0 to 399 : except analysis mode; STORAGE

Explanation

command	Sets the FFT trace cursor position.
query	Returns the current FFT trace cursor position as an NR1 numerical value.

Example

```
:CURSOR:FPOSITION 100
```

Move the FFT trace cursor position to 100 points.

When allowed In the FFT function.

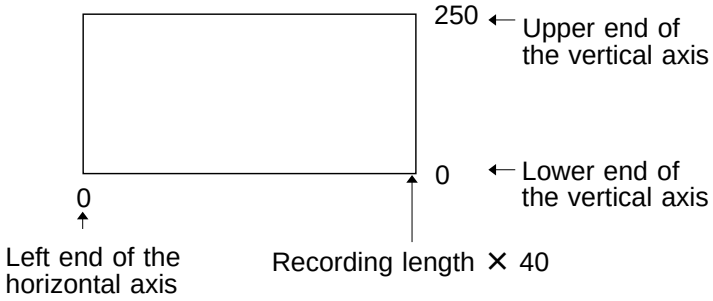
∗The cursor position has the following meaning:

In the memory recorder function and the recorder function, when the vertical cursor or the trace cursor is in use, the cursor position is an indication of the number of the current points in memory.

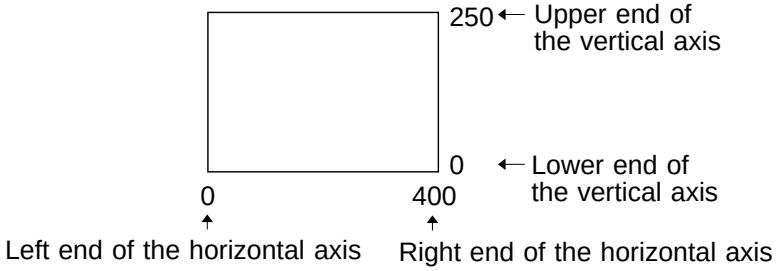
(In the 8852, the stored data values per one division are 40 points, so when recording length is 15 division, the number of stored data values is 600 points (15 divisions × 40 points). Therefore, the cursor position indication lies in the range from 0 to 600.)

The standard cursor position is the left end or the lower end.

- In the memory recorder function and the recorder function

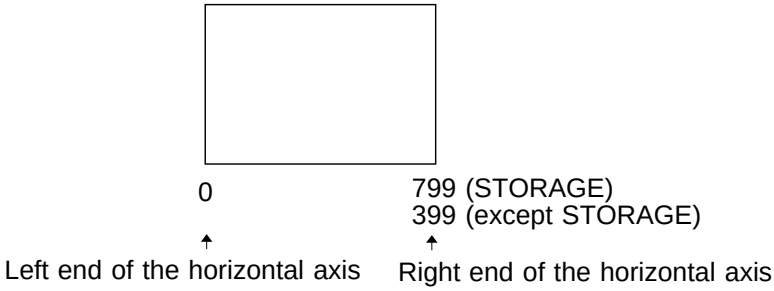


- In the X-Y recorder function and the memory recorder function (X-Y format)



CURSor

- In the FFT function



■ Queries the cursor readout value (t).

Syntax
query :CURSor:DTREad?
response "*A$* unit" (, "*B$* unit")
A$ = t or Δt readout value
B$ = 1/t or 1/Δt readout value (vertical cursor only)

Explanation
query Returns the cursor readout value (t, 1/t) as a line of character data.

Example
```
:CURSOR:DTREAD?
```
Queries the cursor readout value.

When allowed Provided that (t or Δt) is being shown on the display.

■ Queries the cursor readout value (V).

Syntax
query :CURSor:DVREad?
response "*A$* unit"
A$ = v or Δv readout value

Explanation
query Returns the cursor readout value (v) as a line of character data.

Example
```
:CURSOR:DVREAD?
```
Queries the cursor readout value.

When allowed Provided that (v or Δv) is being shown on display.

■ Queries the FFT cursor readout position.

Syntax

query	:CURSor:FFTRead?
response	"*A$* unit", "*B$* unit" *A$* = x-axis readout position *B$* = y-axis readout position

Explanation

query: Returns the current cursor readout position in the FFT function as a line of character data.

Example

```
:CURSOR:FFTREAD?
```

Queries the FFT cursor readout position.

When allowed

In the FFT function (provided that the cursor is on).

7. MEMory command (Sets and queries input and output, etc., from the memory.)

■ Sets and queries the point in memory for input/output.

Syntax

command	:MEMory:POINt *ch$, A*
query	:MEMory:POINt?
response	*ch$, A* <NR1> *ch$* = CH1 to CH4 *A* = 0 to 4000000 (8852) 0 to 16000000 (8852-01)

Explanation

command: Sets the input/output point in memory.

query: 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 the memory recorder function.

MEMory

■ Queries the number of data samples stored.

Syntax

query :MEMory:MAXPoint?

response *A* <NR1>

A = 0 : no data stored

600 to 4000000 (divided by 40 gives the number of divisions: 8852)

600 to 16000000 (divided by 40 gives the number of divisions: 8852-01)

Explanation

query Returns the number of data samples stored in the memory.

Example

query :MEMORY:MAXPOINT?

response :MEMORY:MAXPOINT 600 (when headers are on)

The number of data samples stored in the memory is 600 (15 divisions).

When allowed In the memory recorder function.

■ Inputs data to memory, and outputs stored data.

Syntax

command :MEMory:ADATa *B*, *C*,...

query :MEMory:ADATa? *A*

response *B*, *C*,... all <NR1>

B, *C*,... = -3 to 252 (data for storage)

A = 1 to 40 (number of data values to be output)

Explanation

command 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.

query 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.

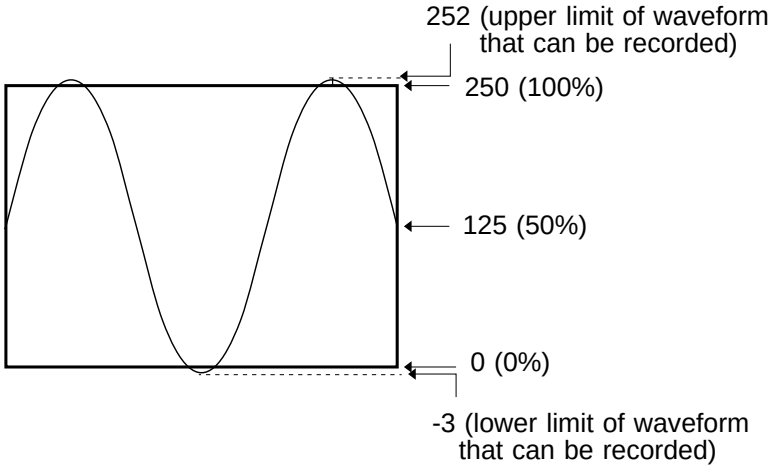
Example

: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 the memory recorder function, provided that stored data is present, and provided that the input/output point is lower than the amount of data stored.

∗Relationship between data values in memory and measured voltages

The following figure illustrates the relationship between the data values (-3 to 252) input and output using the :MEMory:ADATa command and the measured voltage values.



■ Input voltage data to memory, and output voltage data from memory.

Syntax

command	:MEMory:VDATa *B*, *C*,...
query	:MEMory:VDATa? *A*
response	*B*, *C*,... all (NR3>
	B, *C*,... = voltage values (unit volts)
	A = 1 to 10 (amount of data)

Explanation

command

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.

query

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 the memory recorder function, provided that stored data is present, and provided that the input/output point is lower than the amount of data stored.

MEMory

■ Input logic data to memory, and output logic data from memory.

Syntax

command :MEMory:LDATa *B, C,...*

query :MEMory:LDATa? *A*

response *B, C,...* all <NR1>

B, C,... = 1 to 255 (logic data)

A = 1 to 40 (number of data values to be output)

Explanation

command 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.

query 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:

CH1-----CHA1 to A8
CH2-----CHB1 to B8
CH3-----CHC1 to C8

The eight logic channels in each group are encoded as binary bits in the NR1 data value, as shown in the following example.

7	6	5	4	3	2	1	0
A8	A7	A6	A5	A4	A3	A2	A1

Example

```
:MEMORY:POINT CH1, 0
:MEMORY:LDATA? 1
```

If the response is :MEMORY:LDATA 10, then channels A1 to A8 are as follows;

7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0
A8	A7	A6	A5	A4	A3	A2	A1

LOW : 0
HIGH : 1

When allowed In the memory recorder function, 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 (in ASCII).

Syntax
query :MEMory:AREAl? *ch$*
response *A* <NR1>
ch$ = CH1 to CH4
A = -3 to 252

Explanation
query Returns the value input on the channel designated by *ch$*.

Example
query **:MEMORY:AREAL? CH1**
response **:MEMORY:AREAL 125** (HEADER ON)

When allowed Providing that measurement operation is not taking place.

■ Outputs real time data (voltage values).

Syntax
query :MEMory:VREAl? *ch$*
response *A* <NR3>
ch$ = CH1 to CH4
A = a voltage value (unit V)

Explanation
query Returns as a voltage value the value input on the channel designated by *ch$*.

Example
query **:MEMORY:VREAL? CH1**
response **:MEMORY:VREAL 5.5E-2** (HEADER ON)

When allowed Providing that measurement operation is not taking place.

■ Outputs real time data (logic).

Syntax
query :MEMory:LREAl? *ch$*
response *A* <NR1>
ch$ = CH1 to CH4
A = 0 to 255

Explanation
query Returns as an NR1 numerical value, the value input on the channel designated by *ch$*.
The correspondence between the logic channel groups and the response data is the same as that of LDAT on the previous page.

Example
query **:MEMORY:LREAL? CH1**
response **:MEMORY:LREAL 10** (HEADER ON)
Indicates that the current logic data for CHA8 to CHA1 is 00001010.

When allowed Providing that measurement operation is not taking place.

MEMory

■ Binary transfer of stored data.

Syntax

query :MEMory:BDATa? *A*

response #0∗∗∗∗∗∗ • • • •

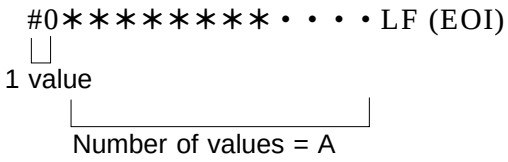
A = 1 to 125

Explanation

query 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 one byte), is transmitted.
- The data is followed by LF (0AH) + EOI.



- 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 the memory recorder function, 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 (binary)

Syntax

query :MEMory:BREAl? *ch$*

response #0∗

ch$ = CH1 to CH4

Explanation

query Outputs in binary format the value input on the channel designated by *ch$*.

When allowed Providing that measurement operation is not taking place.

■ Sets and queries the output point for FFT data.

Syntax

command :MEMory:FFTPOint *A*
query :MEMory:FFTPOint?
response *A* <NR1>
A = 0 to 799 : in analysis mode STORAGE
0 to 399 : except in analysis mode STORAGE

Explanation

command: Sets the output point for FFT data. In DUAL format, sets the output point only for the graph 1.
query: Returns the current output point for FFT data as a numerical value in <NR1> format.

Example

```
:MEMORY:FFTPOINT 100
```

Sets the output point for FFT data to 100.

When allowed

In the FFT function.

■ Outputs FFT data (in ASCII).

Syntax

query :MEMory:FFTData?
response *A* <NR3>
A = y-axis data

Explanation

query: Outputs the y-axis data according to an NR3 numerical value from the point set by the "MEMory:FFTPOint" command. When this command is executed, the set point is increased by one.

Example

```
:MEMORY:FFTPOINT 100
:MEMORY:FFTDATA?
```

Returns the y-axis data for the FFT data of 100 points.

When allowed

In the FFT function.

8. SYSTem command (Sets and queries the system screen.)

■ Sets the time, and queries the current time.

Syntax

command	:SYSTem:TIME *hour, min, sec*
query	:SYSTem:TIME?
response	*hour, min, sec* *hour* = 0 to 23 *min* = 0 to 59 *sec* = 0 to 59

Explanation

command	Sets the time.
query	Returns the current time.

Example

```
:SYSTEM:TIME 10, 0, 0
```

Sets the internal clock to 10:00.

When allowed In all functions.

■ Sets the calendar date, and queries the current calendar date.

Syntax

command	:SYSTem:DATE *year, month, day*
query	:SYSTem:DATE?
response	*year, month, day* *year* = 0 to 99 *month* = 1 to 12 *day* = 1 to 31

Explanation

command	Sets the date on the internal calendar.
query	Returns the current date.

Example

```
:SYSTEM:DATE 94, 4, 25
```

Sets the internal calendar to April 25th, 1994.

When allowed In all functions.

■ Clearing waveform data.

Syntax

command	:SYSTem:DATAClear

Explanation

command	Clear the waveform data.

When allowed In all functions (on the system screen).

■ Enables and disables, and queries the screen auto off (screen saver) function.

Syntax

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

Explanation

command	Enables or disables the screen saver function.
query	Returns the current enablement state of the screen saver function as character data.

Example

```
:SYSTEM:CRTOFF ON
```

Sets the screen saver function to ON.

When allowed In the memory recorder function.

■ Sets and queries the grid type.

Syntax

command	:SYSTem:GRID *A$*
query	:SYSTem:GRID?
response	*A$*
	A$ = OFF, NORMal, FINE

Explanation

command	Sets the type of grid displayed.
query	Returns the current grid setting as character data.

Example

```
:SYSTEM:GRID NORMAL
```

Sets the grid type to NORMAL.

When allowed In all functions.

■ Enables and disables, and queries the start key backup function.

Syntax

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

Explanation

command	Enables and disables the start key backup function.
query	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.

SYSTem

■ Enables and disables, and queries the channel marker.

Syntax

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

Explanation

command	Makes the corresponding channel marker setting.
query	Returns the current channel marker setting as character data.

Example

```
:SYSTEM:CHMARK ON
```

Sets the channel marker to ON.

When allowed In all functions.

■ Enables and disables, and queries the sound of beeper.

Syntax

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

Explanation

command	Enables and disables the beeper sound.
query	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 list function and the gauge function.

Syntax

command	:SYSTem:LIST *A$*
query	:SYSTem:LIST?
response	*A$*
	A$ = OFF, LIST, GAUGE, L_G (LIST&GAUGE)

Explanation

command	Sets the list function and the gauge function according to a character string.
query	Returns the current settings for the list function and the gauge function as a character string.

Example

```
:SYSTEM:LIST LIST
```

Sets the list function.

When allowed In all functions.

■ Sets and queries the number of channels used.

Syntax

command	:SYSTem:USECH *A*
query	:SYSTem:USECH?
response	*A* <NR1>

A = 1, 2, 4

Explanation

command	Sets the number of channels used to a numerical value.
query	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 all functions.

■ Sets and queries the screen dump size.

Syntax

command	:SYSTem:COPYSize *A$*
query	:SYSTem:COPYSize?
response	*A$*

A$ = LARGE, SMALL

Explanation

command	Sets the screen dump size.
query	Returns the current screen dump size as character data.

Example **:SYSTEM:COPYSIZE SMALL**

Sets the screen dump size to SMALL.

When allowed In all functions.

■ Sets and queries the SCSI interface device address ID.

Syntax

command	:SYSTem:SCSI *A$, B*
query	:SYSTem:SCSI? *A$*
response	*A$, B* <NR1>

A$ = 8852 : 8852

SCSI (HDD) : hard disk drive or magneto-optical disk drive

B = 0 to 7 : device address ID

Explanation

command	Sets the device address ID designated by *A$*.
query	Returns as an NR1 numerical value the setting for the device address ID designated by *A$*.

Example **:SYSTEM:SCSI 8852, 1**

Sets the SCSI interface device address ID for the 8852 to 1.

When allowed In all functions.

SYSTem

■ Sets and queries the screen dump output destination.

Syntax

command :SYSTem:COPYPlot *A$*
query :SYSTem:COPYPlot?
response *A$*
A$ = PRINter
PLOTter
FD: floppy disk
SCSI: SCSI interface

Explanation

command Sets the screen dump output device.
query Returns the screen dump output device setting as character data.

Example

```
:SYSTEM:COPYPLOT PLOTTER
```

Sets so that the screen dump is output to the plotter.

When allowed In all functions.

■ Sets and queries the plotter pen.

Syntax

command :SYSTem:PEN *A$, B*
query :SYSTem:PEN? *A$*
response *A$, B* <NR1>
A$ = AREA : waveform decision area
FRAME
CHAR : character
CH1 to CH4
B = 0 to 8 (0; OFF)

Explanation

command Sets the plotter pen number for the setting designated by *A$*.
query Returns as character data the pen number setting for the setting designated by *A$*.

Example

```
:SYSTEM:PEN AREA 1
```

Uses the plotter pen 1 to draw the waveform decision area.

When allowed In all functions.

■ Sets and queries the probe ratio.

Syntax

command	:SYSTem:PROBE *ch$, A$*
query	:SYSTem:PROBE? *ch$*
response	*ch$, A$*

ch$ = CH1 to CH4
A$ = 10_1 : probe ratio 10:1
1_1 : probe ratio 1:1

Explanation

command	Sets the probe ratio for the channel designated by *ch$*.
query	Returns as character data the probe ratio setting for the channel designated by *ch$*.

Example

```
:SYSTEM:PROBE CH1, 10_1
```

Sets the probe ratio for channel 1 to 10:1.

When allowed In all functions.

■ Sets and queries the color mode of the bitmap file (.bmp).

Syntax

command	:SYSTem:BMPKind *A$*
query	:SYSTem:BMPKind?
response	*A$*

A$ = MONO : monochrome
COLOR

Explanation

command	Sets the color mode (monochrome or color) of bitmap file (.bmp) output.
query	Returns the color mode as character data.

Example

```
:SYSTEM:BMPKIND MONO
```

Bitmap files are output as monochrome files.

When allowed In all functions.

SYSTem

■ Sets and queries the colors of the bitmap file (.bmp).

Syntax

command :SYSTem:BMPColor *A$, B$, C$, D$*

query :SYSTem:BMPColor?

response *A$, B$, C$, D$*

:BMPColor *A$, B$, C$, D$*

char dark light cursor

A$* to *D$ = BLACK, BLUE, RED, MAGENTA, GREEN, CYAN, YELLOW or ORANGE

Explanation

command: Sets the colors used for bitmap file (.bmp) output when color bitmap file (.bmp) output is selected.

query: Returns the color setting as character data.

Example

```
:SYSTEM:BMPCOLOR BLACK, BLUE, RED, CYAN
```

Sets char to black, dark to blue, light to red, and cursor to cyan.

When allowed In all functions.

■ Sets and queries the FD key.

Syntax

command :SYSTem:DISKMode *A$*

query :SYSTem:DISKMode?

response *A$*

A$ = FD : FD screen
SCSI : SCSI screen
FD_SCSI : FD or SCSI screen

Explanation

command: Sets the screen that is displayed when the FD key is pressed.

query: Returns the FD key setting as character data.

Example

```
:SYSTEM:DISKMODE SCSI
```

Sets so that the SCSI screen is displayed when the FD key is pressed.

When allowed In all functions.

9. SCALing command (Sets and queries scaling.)

■ Enables and disables, and queries the scaling function.

Syntax

command	:SCALing:SET *ch$, A$*
query	:SCALing:SET? *ch$*
response	*ch$, A$* *ch$* = CH1 to CH4 *A$* = OFF, SCI, and ENG

Explanation

command	Enables or disables the scaling function for the channel designated by *ch$*. A setting SCI produces conventional scientific floating-point notation. The setting ENG produces floating-point notation using powers of 1000.
query	Returns the current state of enablement of the scaling function for the channel designated by *ch$* as character data.

Example

```
:SCALING:SET CH1, SCI
```

Sets the scaling function for channel 1 to SCI.

When allowed In all functions.

■ Sets and queries the scaling conversion value.

Syntax

command	:SCALing:VOLT *ch$, A*
query	:SCALing:VOLT? *ch$*
response	*ch$, A* <NR3> *ch$* = CH1 to CH4 *A* = scaling conversion value (EU/volts) (-9.999E+9 to +9.999E+9)

Explanation

command	Sets the scaling conversion value for the channel designated by *ch$*.
query	Returns the scaling conversion value for the channel designated by *ch$* 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.

SCALing

■ Sets and queries the scaling offset.

Syntax

command :SCALing:OFFSet *ch$, A*
query :SCALing:OFFSet? *ch$*
response *ch$, A* <NR3>
ch$ = CH1 to CH4
A = scaling offset (EU offset)
(-9.999E+9 to +9.999E+9)

Explanation

command: Sets the scaling offset for the channel designated by *ch$*.
query: Returns the scaling offset for the channel designated by *ch$* 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.

■ Sets and queries the scaling unit.

Syntax

command :SCALing:UNIT *ch$*, '*A$*'
query :SCALing:UNIT? *ch$*
response *ch$*, "*A$*"
ch$ = CH1 to CH4
A$ = scaling unit (up to 7 characters)

Explanation

command: Sets the scaling unit for the channel designated by *ch$* (up to 7 characters allowed).

Characters that can be used are as follows:
(Characters other than the following are replaced by spaces.)

A to Z	a to z	+	−	∗	/	%	~.(•)
=	(space)	^2 (^{2})	^3 (^{3})	~u (**µ**)	~c (°)	~o (**Ω**)	

Double quotation marks (") can be used instead of single quotation marks (').

query: Returns the scaling unit for the channel designated by *ch$* as character string data.

Example

```
:SCALING:UNIT CH1, 'mA'
```

Sets the scaling unit for channel 1 to milliamps.

When allowed In all functions.

10. COMMent command (Sets and queries comments.)

■ Enables and disables, and queries title comments, and inputs comment characters.

Syntax

command :COMMent:TITLe *A$*, '*B$*'
query :COMMent:TITLe?
response *A$*, "*B$*"
A$ = OFF, ON
B$ = comment characters (up to 20 characters)

Explanation

command: Enables and disables comments, and inputs a string of comment characters.

Characters that can be used are as follows:
(Characters other than the following are replaced by spaces.)

A to Z	a to z	0 to 9	+	−	∗	/
%	=	(	)	#	&	.
^	,	~u (μ)	~c (゜)	(space)		

Double quotation marks (") can be used instead of single quotation marks(').
Comments may be omitted.

query: Returns the current enablement state of title comments, and the characters of the comment if any, as character data.

Example

```
:COMMENT:TITLE ON, 'HIOKI 8852'
```

Inputs "HIOKI 8852" as a title comment.

When allowed In all functions.

COMMent

■ For each channel, enables and disables and queries comments, and inputs comment characters.

Syntax

command :COMMent:CH *ch$*, *A$*, *'B$'*
query :COMMent:CH? *ch$*
response *ch$*, *A$*, "*B$*"
ch$ = CH1 to CH4
A$ = OFF, ON
B$ = comment characters (up to 20 characters)

Explanation

command: Enables and disables comments for the channel specified by *ch$*, and inputs a string of comment characters (may be omitted).
Characters that can be used are the same as in :TITLe.
Double quotation marks (") can be used instead of single quotation marks (').

query: Returns the enablement state of comments for the channel specified by *ch$*, and the characters of the comment if any, as character data.

Example

```
:COMMENT:CH, CH1, ON, 'ch1 = TEST'
```

Sets the comment display for channel 1 to "ch1 = TEST".

When allowed In all functions.

11. CALCulate command (Calculation setting and querying)

■ Enables and disables, and queries waveform processing calculation.

Syntax

command :CALCulate:WVCALc *A$*
query :CALCulate:WVCALc?
response *A$*
A$ = OFF, ON, EXEC (execute)

Explanation

command: Enables or disables, according to character data, the execution of waveform processing calculation.

query: Returns, as character data, whether execution of waveform processing calculation is enabled or disabled.
Only valid when execution (EXEC) is enabled.

Example

```
:CALCULATE:WVCALC ON
```

Sets the waveform processing calculation to ON.

When allowed In the memory recorder function.

■ Sets and queries the coefficients for the waveform processing calculation equation for Z1.

Syntax

command	:CALCulate:Z1 *A$, B$, C$, D$*
query	:CALCulate:Z1?
response	*A$, B$, C$, D$*

A$, B$, C$ = A to P

D$ = PLUS : +
MINUs : -
MULTi : ∗
DIVI : /

The syntax is "Z1 = *A$* X1 *D$ B$* Y1 + *C$*".

(Syntax of :Z2 to :Z4 commands are same as the :Z1 command.)

Explanation

command	Sets the coefficients for the waveform processing calculation equation for Z1 according to the character data.
query	Returns the current coefficients for the waveform processing calculation equation for Z1 as character data.

Example

```
:CALCULATE:Z1 A, B, C, PLUS
```

Sets up the calculation equation for Z1 to be Z1 = aX1+bY1+c.

When allowed In the memory recorder function.

■ Sets and queries the coefficients for the waveform processing calculation equation for Z2.

(For details, refer to the explanation for the :Z1 command.)

Syntax

command	:CALCulate:Z2 *A$, B$, C$, D$*
query	:CALCulate:Z2?
response	*A$, B$, C$, D$*

Explanation

command	Sets the coefficients for the waveform processing calculation equation for Z2 according to the character data.
query	Returns the current coefficients for the waveform processing calculation equation for Z2 as character data.

When allowed In the memory recorder function.

CALCulate

■ Sets and queries the coefficients for the waveform processing calculation equation for Z3.

(For details, refer to the explanation for the :Z1 command.)

Syntax

command	:CALCulate:Z3 *A$, B$, C$, D$*
query	:CALCulate:Z3?
response	*A$, B$, C$, D$*

Explanation

command	Sets the coefficients for the waveform processing calculation equation for Z3 according to the character data.
query	Returns the current coefficients for the waveform processing calculation equation for Z3 as character data.

When allowed In the memory recorder function.

■ Sets and queries the coefficients for the waveform processing calculation equation for Z4.

(For details, refer to the explanation for the :Z1 command.)

Syntax

command	:CALCulate:Z4 *A$, B$, C$, D$*
query	:CALCulate:Z4?
response	*A$, B$, C$, D$*

Explanation

command	Sets the coefficients for the waveform processing calculation equation for Z4 according to the character data.
query	Returns the current coefficients for the waveform processing calculation equation for Z4 as character data.

When allowed In the memory recorder function.

■ Sets up and queries the calculation equation for X1.

Syntax

command :CALCulate:X1 *A$, ch$, B$*

query :CALCulate:X1?

response *A$, ch$, B$*

A$* = OFF (in this case, *ch$ and *B$* are disregarded)
PAR : (
ABS : Absolute value
EXP : Exponential
LOG : Common logarithm
SQR : Square root
MOV : Moving average
DIF : Differentiation once
INT : Integration once
DIF2 : Differentiation twice
INT2 : Integration twice
SLI : Parallel displacement

ch$ = CH1 to CH4

B$ = A to P
(when *A$* is set to MOV)
a numerical value from 1 to 4000
(when *A$* is set to SLI)
a numerical value from -4000 to 4000.

The syntax is "X1 = *A$* (*ch$* + *B$*)", however when *A$* = MOV or SLI:
"X1 = [MOV or SLI] (*ch$, B$*)" (See Section 12.2.2, "Method of Calculation" in the 8852 Instruction Manual.)

(Syntax of the :X2 to :X4 commands are same as :X1 command except "*ch$*.")

Explanation

command Sets the X1 calculation equation for the waveform processing calculation equation for Z1 according to the character or numerical data.

query Returns the current X1 calculation equation for the waveform processing calculation equation for Z1 as character or numerical data.

Example 1

```
CALCULATE:X1 ABS, CH1, A
```

Sets up the calculation equation for X1 to be X1 = ABS (ch1 + a)

Example 2

```
CALCULATE:X1 MOV, CH1, 50
```

Sets up the calculation equation for X1 to be X1 = MOV (ch1, 50)

When allowed In the memory recorder function.

CALCulate

■ Sets up and queries the calculation equation for X2.

(For details, refer to the explanation for the :X1 command.)

Syntax

command	:CALCulate:X2 *A$, ch$, B$*
query	:CALCulate:X2?
response	*A$, ch$, B$*
	ch$ = CH1 to CH4, Z1

Explanation

command	Sets the X2 calculation equation for the waveform processing calculation equation for Z2 according to the character or numerical data.
query	Returns the current X2 calculation equation for the waveform processing calculation equation for Z2 as character or numerical data.

When allowed In the memory recorder function.

■ Sets up and queries the calculation equation for X3.

(For details, refer to the explanation for the :X1 command.)

Syntax

command	:CALCulate:X3 *A$, ch$, B$*
query	:CALCulate:X3?
response	*A$, ch$, B$*
	ch$ = CH1 to CH4, Z1, Z2

Explanation

command	Sets the X3 calculation equation for the waveform processing calculation equation for Z3 according to the character or numerical data.
query	Returns the current X3 calculation equation for the waveform processing calculation equation for Z3 as character or numerical data.

When allowed In the memory recorder function.

■ Sets up and queries the calculation equation for X4.

(For details, refer to the explanation for the :X1 command.)

Syntax

command	:CALCulate:X4 *A$, ch$, B$*
query	:CALCulate:X4?
response	*A$, ch$, B$* *ch$* = CH1 to CH4, Z1 to Z3

Explanation

command	Sets the X4 calculation equation for the waveform processing calculation equation for Z4 according to the character or numerical data.
query	Returns the current X4 calculation equation for the waveform processing calculation equation for Z4 as character or numerical data.

When allowed In the memory recorder function.

CALCulate

■ Sets up and queries the calculation equation for Y1.

Syntax

command :CALCulate:Y1 *A$, ch$, B$*

query :CALCulate:Y1?

response *A$, ch$, B$*

A$* = OFF (in this case, *ch$ and *B$* are disregarded)
PAR : (
ABS : Absolute value
EXP : Exponential
LOG : Common logarithm
SQR : Square root
MOV : Moving average
DIF : Differentiation once
INT : Integration once
DIF2 : Differentiation twice
INT2 : Integration twice
SLI : Parallel displacement

ch$ = CH1 to CH4

B$ = A to P
(when *A$* is set to MOV)
a numerical value from 1 to 4000
(when *A$* is set to SLI)
a numerical value from -4000 to 4000

The syntax is "Y1 = *A$* (*ch$* + *B$*)", however when *A$* = MOV or SLI: "Y1 = [MOV or SLI] (*ch$, B$*)" (See Section 12.2.2, "Method of Calculation" in the 8852 Instruction Manual.)

(Syntax of the :Y2 to :Y4 commands are same as :Y1 command except "*ch$*.")

Explanation

command: Sets the Y1 calculation equation for the waveform processing calculation equation for Z1 according to the character or numerical data.

query: Returns the current Y1 calculation equation for the waveform processing calculation equation for Z1 as character or numerical data.

Example 1

```
:CALCULATE:Y1 ABS, CH1, A
```

Sets up the calculation equation for Y1 to be Y1 = ABS (ch1 + a)

Example 2

```
:CALCULATE:Y1 MOV, CH1, 50
```

Sets up the calculation equation for Y1 to be Y1 = MOV (ch1, 50)

When allowed In the memory recorder function.

■ Sets up and queries the calculation equation for Y2.

(For details, refer to the explanation for the :Y1 command.)

Syntax

command	:CALCulate:Y2 *A$, ch$, B$*
query	:CALCulate:Y2?
response	*A$, ch$, B$* *ch$* = CH1 to CH4, Z1

Explanation

command	Sets the Y2 calculation equation for the waveform processing calculation equation for Z2 according to the character or numerical data.
query	Returns the current Y2 calculation equation for the waveform processing calculation equation for Z2 as character or numerical data.

When allowed In the memory recorder function.

■ Sets up and queries the calculation equation for Y3.

(For details, refer to the explanation for the :Y1 command.)

Syntax

command	:CALCulate:Y3 *A$, ch$, B$*
query	:CALCulate:Y3?
response	*A$, ch$, B$* *ch$* = CH1 to CH4, Z1, Z2

Explanation

command	Sets the Y3 calculation equation for the waveform processing calculation equation for Z3 according to the character or numerical data.
query	Returns the current Y3 calculation equation for the waveform processing calculation equation for Z3 as character or numerical data.

When allowed In the memory recorder function.

CALCulate

■ Sets up and queries the calculation equation for Y4.

(For details, refer to the explanation for the :Y1 command.)

Syntax

command :CALCulate:Y4 *A$, ch$, B$*
query :CALCulate:Y4?
response *A$, ch$, B$*
ch$ = CH1 to CH4, Z1 to Z3

Explanation

command: Sets the Y4 calculation equation for the waveform processing calculation equation for Z4 according to the character or numerical data.
query: Returns the current Y4 calculation equation for the waveform processing calculation equation for Z4 as character or numerical data.

When allowed In the memory recorder function.

■ Sets and queries numerical values for coefficients a to p of the waveform processing calculation equation.

Syntax

command :CALCulate:FACTor *A$, B*
query :CALCulate:FACTor? *A$*
response *A$, B* <NR3>
A$ = A to P
B = -9.999E+9 to +9.999E+9

Explanation

command: Sets to the given numerical value the one of the coefficients a to p which is designated in *A$*.
query: Returns as an <NR 3> numerical value the current value of that one of the coefficients a to p which is designated in *A$*. (Refer to Chapter 12, "Calculation Functions.")

Example

```
:CALCULATE:FACTOR A, +1.234E+1
```

Sets the coefficient a to be equal to +1.234E+1

When allowed In the memory recorder function.

■ Sets and queries the display channel for the calculated result of the waveform processing calculation equation for Z1.

Syntax

command :CALCulate:Z1DIsplay *ch$, A$, upper, lower*

query :CALCulate:Z1DIsplay?

response *ch$, A$, upper, lower*

ch$ = CH1 to CH4, NONE

A$ = MANUal, AUTO

upper, lower = -9.999E+9 to +9.999E+9

(if *A$* = AUTO, *upper, lower* may be omitted.)

(Syntax of :Z2DIsplay to :Z4DIsplay commands are same as :Z1DIsplay.)

Explanation

command: Displays the calculated result of the waveform processing calculation equation for Z1 on the channel designated by *ch$* within the range from lower to upper (unit volts - however, if scaling is being performed, in those units).

query: Returns the currently set display channel, scale setting lower limit, and upper limit for display of the calculated result of the waveform processing calculation equation for Z1.

Example

```
:CALCULATE:Z1DISPLAY CH1, MANUAL, +5.000E+0, +0.000E+0
```

Displays the calculated result of the waveform processing calculation equation for Z1 on channel 1 within the range from 0 volts to 5 volts.

When allowed In the memory recorder function.

■ Sets and queries the display channel for the calculated result of the waveform processing calculation equation for Z2.

(For details, refer to the explanation for the :Z1DIsplay command.)

Syntax

command :CALCulate:Z2DIsplay *ch$, A$, upper, lower*

query :CALCulate:Z2DIsplay?

response *ch$, A$, upper, lower*

Explanation

command: Displays the calculated result of the waveform processing calculation equation for Z2 on the channel designated by *ch$* within the range from lower to upper (unit volts - however, if scaling is being performed, in those units).

query: Returns the currently set display channel, scale setting lower limit, and upper limit for display of the calculated result of the waveform processing calculation equation for Z2.

When allowed In the memory recorder function.

CALCulate

■ Sets and queries the display channel for the calculated result of the waveform processing calculation equation for Z3.

(For details, refer to the explanation for the :Z1DIsplay command.)

Syntax

command	:CALCulate:Z3DIsplay *ch$, A$, upper, lower*
query	:CALCulate:Z3DIsplay?
response	*ch$, A$, upper, lower*

Explanation

command	Displays the calculated result of the waveform processing calculation equation for Z3 on the channel designated by *ch$* within the range from lower to upper (unit volts - however, if scaling is being performed, in those units).
query	Returns the currently set display channel, scale setting lower limit, and upper limit for display of the calculated result of the waveform processing calculation equation for Z3.

When allowed In the memory recorder function.

■ Sets and queries the display channel for the calculated result of the waveform processing calculation equation for Z4.

(For details, refer to the explanation for the :Z1DIsplay command.)

Syntax

command	:CALCulate:Z4DIsplay *ch$, A$, upper, lower*
query	:CALCulate:Z4DIsplay?
response	*ch$, A$, upper, lower*

Explanation

command	Displays the calculated result of the waveform processing calculation equation for Z4 on the channel designated by *ch$* within the range from lower to upper (unit volts - however, if scaling is being performed, in those units).
query	Returns the currently set display channel, scale setting lower limit, and upper limit for display of the calculated result of the waveform processing calculation equation for Z4.

When allowed In the memory recorder function.

CALCulate

■ Enables and disables, and queries waveform parameter calculation.

Syntax

command	:CALCulate:MEASure *A$*
query	:CALCulate:MEASure?
response	*A$*

A$ = OFF, ON, EXEC (execute)

Explanation

command	Enables or disables, according to character data, the execution of waveform parameter calculation.
query	Returns, as character data, whether execution of waveform parameter calculation is enabled or disabled. Only valid when execution (EXEC) is enabled.

Example

```
:CALCULATE:MEASURE ON
```

Sets the waveform parameter calculation to ON.

When allowed In the memory recorder function.

■ Sets and queries the output destination of waveform parameter calculation values.

Syntax

command	:CALCulate:MEASPrint *A$*
query	:CALCulate:MEASPrint?
response	*A$*

A$ = OFF : no output
PRINter
FD : floppy disk
SCSI : SCSI interface

Explanation

command	Sets the output destination of waveform parameter calculation values according to the character data.
query	Returns the output destination of waveform parameter calculation values as character data.

Example

```
:CALCULATE:MEASPRINT PRINTER
```

Outputs the result of waveform parameter calculation to the printer.

When allowed In the memory recorder function.

CALCulate

■ Sets and queries waveform parameter calculations.

Syntax

command :CALCulate:MEASSet *NO$, A$, ch$*

query :CALCulate:MEASSet? *NO$*

response *NO$, A$, ch$*

NO$ = NO1 to NO4

A$ = OFF

MIN : minimum value
MAX : maximum value
MINT : time to minimum value
MAXT : time to maximum value
PP : peak value
AVE : average value
RMS : effective value
AREA : area value
PERI : period
FREQ : frequency
RISE : rise time
FALL : fall time
XYAREA : X-Y area value

ch$ = CH1 to CH4, ALL

During XYAREA:

ch$ = x-axis channel, y-axis channel

Explanation

command: Sets the channel and the calculation item of the waveform parameter calculation designated by *NO$*.

query: Returns the channel and the calculation item of the waveform parameter calculation designated by *NO$*.

Example 1

```
:CALCULATE:MEASSET NO1, MAX, CH1
```

Sets the calculation to be of the maximum value on channel 1 for the calculation NO1.

Example 2

```
:CALC:MEASS NO2, XYAREA, CH1, CH2
```

If the x-axis is channel 1 and the y-axis is channel 2, sets X-Y area value calculation for the calculation NO2.

When allowed

In the memory recorder function.

■ Queries result of waveform parameter calculation.

Syntax

query :CALCulate:ANSWer? *NO$, ch$*

response *A$, B* <NR 3>

NO$ = NO1 to NO4

ch$ = CH1 to CH4

A$ = OFF, AVE, RMS, PP, MAX, MAXT, MIN, MINT, AREA, PERI, FREQ, RISE, FALL, XYAREA
NONE : no calculation result

B = calculation result

Explanation

query 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 the memory recorder function.

■ Enables and disables, and queries decision for waveform parameter calculation.

Syntax

command :CALCulate:COMP *NO$, A$*

query :CALCulate:COMP? *NO$*

response *NO$, A$*

NO$ = NO1 to NO4

A$ = OFF, ON

Explanation

command Enables and disables, according to the character data, the decision of the calculation result of waveform parameter calculation.

query Returns, as character data, the enablement state of the decision of the calculation result of waveform parameter calculation.

Example

```
:CALCULATE:COMP NO1, ON
```

Sets the decision of the calculation result of NO1 to ON.

When allowed In the memory recorder function.

CALCulate

■ Sets and queries upper and lower limits for the decision value for waveform parameter calculation.

Syntax

command :CALCulate:COMPArea *NO$, upper, lower*
query :CALCulate:COMPArea? *NO$*
response *NO$, upper, lower*
NO$ = NO1 to NO4
upper, lower = -9.999E+9 to +9.999E+9

Explanation

command: Sets, according to the numerical values supplied, the upper limit and the lower limit used when performing a decision on the waveform parameter calculated value designated by *A$*.

query: Returns, as <NR 3> numerical values, the upper limit and the lower limit used when performing a decision on the waveform parameter calculated value designated by *A$*.

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 the memory recorder function.

12. DISK command (Setting and querying relating to the FD, HD, and MO)

■ Enables and disables, and queries the floppy disk control screen, and the SCSI control screen.

Syntax

command :DISK:MODE *A$*
query :DISK:MODE?
response *A$*
A$ = FD : floppy disk control (FD) screen
SCSI (HD) : SCSI control screen
OFF : except FD or SCSI screen

Explanation

command: Enters the FD or SCSI screen.

query: Returns the currently set screen as character data.

Example

```
:DISK:MODE FD
```

Enters the floppy disk control screen.

When allowed In all functions.

■ Saves a file.

Syntax command :DISK:SAVE '*NAME1$. NAME2$*', *A$, B$* (when *A$* = Wave)
:DISK:SAVE '*NAME1$. NAME2$*', *A$* (when *A$* = Func or Area)
NAME1$ = file name (8 characters)
NAME2$ = extension (3 characters)
A$ = type of saved information
Wave : measurement data (MEM and FFT only)
Func : setting data
Area : waveform decision area (MEM and FFT only)
B$* = saved channels (only when *A$ = Wave in MEM)
ALL, CH1 to CH4

Explanation command 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 `:DISK:SAVE 'TEST. DAT', WAVE, ALL`
Saves all channels of measurement data under the file name 'TEST. DAT'.

When allowed When the FD or SCSI screen is displayed.

■ Loads a file.

Syntax command :DISK:LOAD *NO* (, *ch$*)
NO = file number
ch$ = CH1 to CH4 (only when a file is WAVE)

Explanation command Loads the data in the file numbered *NO*.
When a file is WAVE (measurement data), and a saved channel (refer to the :DISK:SAVE command) is CH1 to CH4, the measurement data is loaded to the channel specified by *ch$*. When *ch$* is omitted, it is loaded to the saved channel.

Example `:DISK:LOAD 1`
Loads the data of the file numbered 1.

When allowed When the FD or SCSI screen is displayed.

DISK

■ Deletes a file.

Syntax command :DISK:DELEte *NO*
NO = file number

Explanation command Deletes the file whose number is specified by *NO*.

Example
```
:DISK:DELETE 1
```
Deletes the file of No.1.

When allowed When the FD or SCSI screen is displayed.

■ Formats a floppy disk, hard disk or magneto-optical disk.

Syntax command :DISK:FORMat (*A$*)
Effective only for 2HD floppy disks.
A$ = 2HD (1.2 MB)
2HC (1.44 MB)

Explanation command Formats a floppy disk, hard disk, or magneto-optical disk. Select 2HD (1.2 M-byte) or 2HC (1.44 M-byte) format for 2HD floppy disks.

Example
```
:DISK:FORMAT 2HD
```
Formats in 2HD (1.2 M-byte) format.

When allowed When the FD or SCSI screen is displayed.

■ Creates a directory on the hard disk or magneto-optical disk.

Syntax command :DISK:MKDIR '*NAME$*'
NAME$ = subdirectory name (up to 12 characters)

Explanation command Creates a subdirectory in the current directory on the hard disk or magneto-optical disk.
Double quotation marks (") can be used instead of single quotation marks (').

Example
```
:DISK:MODE SCSI
:DISK:MKDIR 'TEST'
```
Creates a subdirectory called TEST on the hard disk or magneto-optical disk.

When allowed When the SCSI screen is displayed.

DISK

■ Changes the current directory on the hard disk or magneto-optical disk.

Syntax

command :DISK:CHDIR *NO*

NO = file number (directory)

Explanation

command Changes the current directory to the directory specified by *NO* on the hard disk or magneto-optical disk.

When allowed When the SCSI screen is displayed.

■ Queries the current directory on the hard disk or magneto-optical disk.

Syntax

query :DISK:DIR?

response *A$*

A$ = directory name

Explanation

query Returns the current directory name on the hard disk or magneto-optical disk as character data.

When allowed When the SCSI screen is displayed.

■ Queries information about a file.

Syntax

query :DISK:INFOr? '*NAME$*'

response FILE, "*NAME$*", *A*, *B$*, "*DATE$*", "*TIME$*", *C* (file),
"*NAME$*", *A*, "*DATE$*", "*TIME$*" (directory)

NAME$ = file name

A = file number (if no such file exists, -1)

B$ = type of information saved:

WAVE : measurement data

FUNC : conditions of creation

AREA : waveform decision area

N : no such file

DATE$ = date of save "year-month-day"

TIME$ = time of save "hour:minute:second"

C = size of file

Explanation

query Returns information about the file whose name is specified in *NAME$*.
If no such file exists, returns: "*NAME$*", -1, N, "----", "-:-:-", 0
Double quotation marks (") can be used instead of single quotation marks (').

When allowed When the FD or SCSI screen is displayed.

DISK

■ Queries the filename.

Syntax
query :DISK:NINFor? *NO*
response *NO, NAME$*
NO = file number
NAME$ = name of the file

Explanation
query Returns the filename of the file whose number is specified in *NO*.

Example
query **:DISK:NINFOR? 1**
response **:DISK:NINFOR 1, "TEST. DAT"**

When allowed When the FD or SCSI screen is displayed.

■ Queries the number of files.

Syntax
query :DISK:FILE?
response *A* <NR1>
A = number of files

Explanation
query Returns the total number of files which are currently saved on the floppy disk.
Returns the number of files (including directories) in the current directory on the hard disk or magneto-optical disk.

When allowed When the FD or SCSI screen is displayed.

■ Queries the allowable number of clusters for the floppy disk, hard disk or magneto-optical disk.

Syntax
query :DISK:FREE?
response *A* <NR1>
A = allowable number of clusters

Explanation
query Returns the allowable number of clusters for the floppy disk, hard disk or magneto-optical disk.

When allowed When the FD or SCSI screen is displayed.

13 GRAPh command (Commands relating to graphics editor)

■ Enables and disables, and queries the enablement of the graphics editor.

Syntax

command	:GRAPh:EDIT *A$*
query	:GRAPh:EDIT?
response	*A$*

A$ = OFF, ON

Explanation

command	Enables and disables the graphic editor mode.
query	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 the memory recorder function (SINGLE, X-Y format) and the FFT function (SINGLE format).

■ Loads a waveform into the editor.

Syntax

command	:GRAPh:STORage

Explanation

command	Loads a waveform into the editor.

When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

■ Parallel Command

Syntax

command	:GRAPh: PARAllel *high*, *low*, *right*, *left*

high = 0 to 9.960 (div)
low = 0 to 9.960 (div)
right = 0 to 14.975 (div)
left = 0 to 14.975 (div)

Explanation

command	Carries out a parallel movement of the drawing. The *high* and *low* parameters are set in units of 0.04 steps, and the *right* and *left* parameters in units of 0.025 steps.

When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

GRAPh

■ Line command

Syntax command :GRAPh: LINE *X1, Y1, X2, Y2*

X1, X2 = x-coordinates

Y1, Y2 = y-coordinates

Explanation command Draws a line from (*X1, Y1*) to (*X2, Y2*).

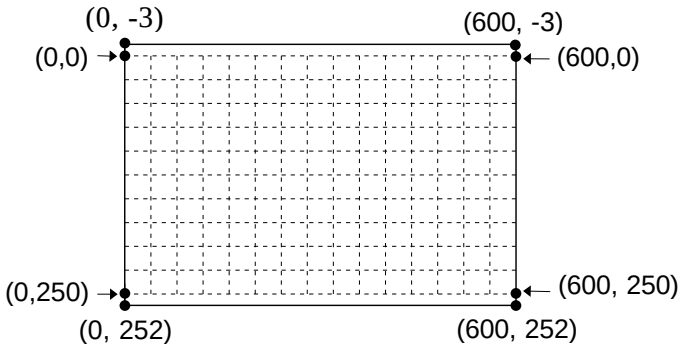
Example **:GRAPH: LINE 10, 20, 100, 200**

Draws a line from (10, 20) to (100, 200).

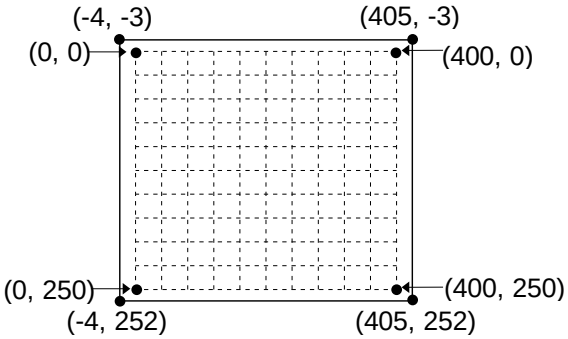
When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

The x- and y-coordinates

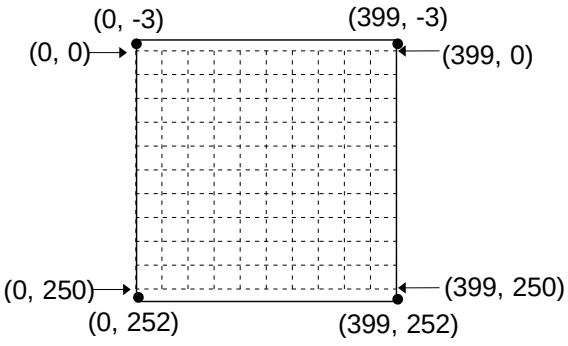
・In the memory recorder function (SINGLE format)



・In the memory recorder function (X-Y format)



・In the FFT function (SINGLE format)



■ Paint command

Syntax command :GRAPh:PAINT *X, Y*

X = x-coordinate

Y = y-coordinate

Explanation command 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 the memory recorder function and the FFT function, when in the graphics editor mode.

■ The reverse command.

Syntax command :GRAPh:REVErse

Explanation command Reverses the video of the drawing.

When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

■ Erase command

Syntax command :GRAPh:ERASe *X1, Y1, X2, Y2*

X1, X2 = x-coordinates

Y1, Y2 = y-coordinates

Explanation command Erases the line from (*X1, Y1*) to (*X2, Y2*).
Refer to the :GRAPh:LINE command for details of X and Y.

When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

■ Clear command

Syntax command :GRAPh: CLEAr *X1, Y1, X2, Y2*

X1, X2 = x-coordinates

Y1, Y2 = y-coordinates

Explanation command 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 the memory recorder function and the FFT function, when in the graphics editor mode.

GRAPh

■ The all clear command.

Syntax command :GRAPh:ALLClear

Explanation command Clears the entire drawing.

When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

■ Undo command

Syntax command :GRAPh:UNDO

Explanation command Reverses the effect of the immediately previous editor command.

When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

■ Saves the drawing (decision area)

Syntax command :GRAPh:SAVE

Explanation command Saves the decision area created with the editor.

When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

■ Sets and queries decision area data points.

Syntax
command :GRAPh:POINt *X*, *Y*, *A*
query :GRAPh:POINt? *X*, *Y*
response *X*, *Y*, *A* all <NR1>
X = x-coordinate
Y = y-coordinate
A = 0 (point outside the decision area)
1 (point within it)

Explanation
command Writes the value *A* at the coordinates indicated by *X* and *Y*.
query Returns the value *A* at the coordinates indicated by *X* and *Y*.

A is 1 for a point within the decision area, 0 for a point outside it.

When allowed In the memory recorder function and the FFT function, when in the graphics editor mode.

Chapter 5
Example Programs

The programs in this chapter run on an IBM-PC(VGA) series computer.

Example 1 Using a setting command

Send the command in the format specified, when the conditions for the command to be acceptable are met.

Line 150	Set ADRS%(0) to address of 8852.
Lines 160-170	Send interface clear, and switch to remote mode.
Line 180	Select memory recorder function.
Line 190	Time/division is 500 µs.
Line 200	Recording length is 15 divisions.
Line 210	Enter measurement operation mode.
Line 220	End remote mode.

```
100 ' ------------------------------------
110 ' 8852 Set command
120 ' You must merge this code with DECL.BAS
130 ' ------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
160 CALL SENDIFC(BOAD%)                          'Clear interface
170 CALL ENABLEREMOTE(BOAD%, ADRS%(0))           'Enable remote
180 GOSUB 270                                    'Function MEM
190 GOSUB 270                                    'Time/Div 500us
200 GOSUB 270                                    '15DIV
210 GOSUB 270                                    '< START >
220 CALL ENABLELOCAL(BOAD%, ADRS%(0))            'Enable operations
230 END
240 ' ------------------------------------
250 ' Send data
260 ' ------------------------------------
270 READ COMMAND$
280 CALL SEND(BOAD%, ADRS%(0), COMMAND$, NLEND%)
290 RETURN
300 ' ------------------------------------
310 ' data table
320 ' ------------------------------------
330 DATA ":FUNCTION MEM"
340 DATA ":CONFIGURE:TDIV +500.e-6"
350 DATA ":CONFIGURE:SHOT 15"
360 DATA ":START"
```

Example 2 Using a query

(1) Send the query in the format specified, when the conditions for the query to be acceptable are met.
Next switch the 8852 to be the talker, and receive the output data.
(2) The response data from the query is returned in the format specified for the corresponding command.

Line 150	Set ADRS%(0) to address of 8852.
Lines 160-170	Send interface clear, and switch to remote mode.
Line 180	Disable headers.
Lines 190-200	Ask function, and load into ANS$.
Lines 210-220	Ask current time, and load into TM$.
Line 240	Release talker.
Line 250	End remote mode.

```
100 ' ---------------------------------------
110 ' 8852 Receive command
120 ' You must merge this code with DECL.BAS
130 ' ---------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
160 CALL SENDIFC(BOAD%)                          'Clear interface
170 CALL ENABLEREMOTE(BOAD%, ADRS%(0))           'Enable remote
180 GOSUB 300                                    'Header OFF
190 GOSUB 300                                    'Read FUNCTION
200 GOSUB 360:ANS$ = READING$
210 GOSUB 300                                    'Read TIME
220 GOSUB 360:TM$ = READING$
230 PRINT ANS$, TM$
240 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%, UNT$)    'UN TALK
250 CALL ENABLELOCAL(BOAD%, ADRS%(0))            'Enable operations
260 END
270 ' ---------------------------------------
280 ' Send data
290 ' ---------------------------------------
300 READ COMMAND$
310 CALL SEND(BOAD%, ADRS%(0), COMMAND$, NLEND%)
320 RETURN
330 ' ---------------------------------------
340 ' Receive data
350 ' ---------------------------------------
360 READING$ = SPACE$(30)
370 CALL RECEIVE(BOAD%, ADRS%(0), READING$, STOPEND%)
380 LENGS% = IBCNT% - 1
390 READING$ = LEFT$(READING$, LENGS%)
400 RETURN
410 ' ---------------------------------------
420 ' data table
430 ' ---------------------------------------
440 DATA ":HEADER OFF"
450 DATA ":FUNCTION?"
460 DATA ":SYSTEM:TIME?"
```

Example 3 Using service requests

(1) 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.

(2) The service request interrupt handling routine uses serial polling to read the 8852 status byte, then carries out appropriate processing depending on the value of the status byte.
It then re-enables the service request interrupt, and returns.

Line 150	Set ADRS%(0) to address of 8852.
Lines 160-170	Send interface clear, and switch to remote mode.
Line 180	Set jump address for service request.
Line 200	Enable bit 5 (ESB) of the status byte by the service request enable register.
Line 210	Enable bits 2, 3, 4, and 5 of the standard event status register by the standard event status enable register.
Line 220	Clear the status byte associated queue.
Line 230	Enable the service request interrupt.
Line 250	Set the function.
Line 280	Set the averaging. (Error source)
Lines 330-340	Serial polling to read the status byte.
Line 380	Enable service request interrupt.
Lines 390-400	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8852 Service request
120 ' You must merge this code with DECL.BAS
130 ' ------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
160 CALL SENDIFC(BOAD%)                          'Clear interface
170 CALL ENABLEREMOTE(BOAD%,ADRS%(0))            'Enable remote
180 ON PEN GOSUB 330
190 SRE$="*SRE 32":ESE$="*ESE 60":SCL$="*CLS"
200 CALL SEND(BOAD%,ADRS%(0),SRE$,NLEND%)        'Mask SRQ
210 CALL SEND(BOAD%,ADRS%(0),ESE$,NLEND%)        'Mask SESER
220 CALL SEND(BOAD%,ADRS%(0),SCL$,NLEND%)        'Clear statusbyte
230 PEN ON
240 FUN$=":FUNCTION MEM"
250 CALL SEND(BOAD%,ADRS%(0),FUN$,NLEND%)        'Set FUNCTION
260 I% = 0
270 AVR$=":CONFIGURE:AVERAGE "+STR$(I%)
280 CALL SEND(BOAD%,ADRS%(0),AVR$,NLEND%)        'Set AVERAGE
290 I% = I% + 50:GOTO 270
300 ' ------------------------------------
310 ' Service request operation
320 ' ------------------------------------
330 CALL IBRSP(ADRS%,S%)
340 DCL$ = CHR$(DCL%):CALL IBCMD(BOAD%,DCL$)     'Clear buffer
350 PRINT "SQR=";S%
360 CALL SEND(BOAD%,ADRS%(0),SRE$,NLEND%)        'Mask SRQ
370 CALL SEND(BOAD%,ADRS%(0),ESE$,NLEND%)        'Mask SESER
380 PEN ON
390 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%,UNT$)     'UN TALK
400 CALL ENABLELOCAL(BOAD%,ADRS%(0))             'Enable operations
410 END
```

Example 4 Outputting stored data

(1) 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.
(2) 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.
(3) 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 40 using :ADATA? and 1 to 10 using the :VDATA? query.

Note: Outputting data in bigger sets reduces the overall processing time.

Program example

Read data (1200 samples) for channel 1 when stored with a 30-division recording length.

Line 170	Set ADRS%(0) to address of 8852.
Lines 180-190	Send interface clear, and switch to remote mode.
Line 210	Set memory recorder function and 30-division recording length.
Line 230	Enter measurement operation mode.
Lines 240-260	Wait for end of measurement operation.
Lines 270-280	Disable headers, and read number of stored data samples into MAX%.
Line 300	Set output data to be from channel 1, point 0.
Lines 310-370	Set size of output data set to be 10 samples, and read as voltage values.
Lines 410-420	Release talker and remote mode.

```
100 ' ---------------------------------------
110 ' 8852 Data out
120 ' You must merge this code with DECL.BAS
130 ' ---------------------------------------
140 DIM D(1201)
150 ESR$ = ":ESR0?":VDT$ = ":MEMORY:VDATA? 10"
160 BOAD% = 0
170 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
180 CALL SENDIFC(BOAD%)                          'Clear interface
190 CALL ENABLEREMOTE(BOAD%,ADRS%(0))            'Enable remote
200 GOSUB 470                                    'Enable ESR0
210 GOSUB 470                                    'MEM,30DIV
220 GOSUB 470                                    'Trigger mode SINGLE
230 GOSUB 470                                    '<START>
240 CALL SEND(BOAD%,ADRS%(0),ESR$,NLEND%)
250 GOSUB 530:STS% = VAL(READING$)
260 IF (STS% AND 2) = 0 THEN 240                 '<START> stopped?
270 GOSUB 470                                    'Check STORAGE data
280 GOSUB 530:MAX% = VAL(READING$)
290 IF MAX% <> 1200 THEN 410
300 GOSUB 470                                    'Set point ch1,0
310 FOR I% = 0 TO MAX% - 10 STEP 10
320 CALL SEND(BOAD%,ADRS%(0),VDT$,NLEND%)
330 GOSUB 530
340 FOR II% = 0 TO 9
350 D(I%+II%) = VAL(MID$(READING$,(12*II%+1),11))
360 NEXT II%
370 NEXT I%
380 GOSUB 470
390 GOSUB 530:D(1200) = VAL(READING$)            'Last Data
400 FOR I% = 0 TO 1200:PRINT D(I%):NEXT I%       'Print data
410 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%,UNT$)     'UN TALK
420 CALL ENABLELOCAL(BOAD%,ADRS%(0))             'Enable operations
430 END
440 ' ---------------------------------------
450 ' Send data
460 ' ---------------------------------------
470 READ COMMAND$
480 CALL SEND(BOAD%,ADRS%(0),COMMAND$,NLEND%)
490 RETURN
500 ' ---------------------------------------
510 ' Receive data
520 ' ---------------------------------------
530 READING$ = SPACE$(128)
540 CALL RECEIVE(BOAD%,ADRS%(0),READING$,STOPEND%)
550 LENGS% = IBCNT% - 1
560 READING$ = LEFT$(READING$,LENGS%)
570 RETURN
580 ' ---------------------------------------
590 ' data table
600 ' ---------------------------------------
610 DATA ":ESE0 2"
620 DATA ":FUNCTION MEM;:CONFIGURE:SHOT 30"
630 DATA ":TRIGGER:MODE SINGLE"
640 DATA ":START"
650 DATA ":HEADER OFF;:MEMORY:MAXPOINT?"
660 DATA ":MEMORY:POINT CH1,0"
670 DATA ":MEMORY:VDATA? 1"
```

Example 5 Inputting storage data

(1) Using the :MEMORY:MAXPOINT? query, this program checks whether data can be input to memory. If this query returns zero, the state is such as not to store data, and it cannot therefore be input.
(2) 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.

Note: As with output, it is more efficient to input data in bigger sets.

Program example

With the unit storing with a 30-division recording length, write sine wave data into memory for channel 1.

Line 190	Set ADRS%(0) to address of 8852.
Lines 200-210	Send interface clear, and switch to remote mode.
Lines 220-250	Read maximum number of data samples in memory into MAX%.
Line 270	Set input data to be to channel 1, point 0.
Lines 280-320	Write the sine wave.
Lines 340-350	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8852 Data input
120 ' You must merge this code with DECL.BAS
130 ' ------------------------------------
140 BOAD% = 0
150 HEA$ = ":HEADER OFF;:MEMORY:MAXPOINT?"
160 ADT$ = ":MEMORY:ADATA"
170 PNT$ = ":MEMORY:POINT CH1,0"
180 WAV$ = ":DISPLAY:CHANGE DISPLAY"
190 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
200 CALL SENDIFC(BOAD%)                          'Clear interface
210 CALL ENABLEREMOTE(BOAD%,ADRS%(0))            'Enable remote
220 CALL SEND(BOAD%,ADRS%(0),HEA$,NLEND%)        'Header off
230 MXP$ = SPACE$(10)
240 CALL RECEIVE(BOAD%,ADRS%(0),MXP$,STOPEND%)   'Maxpoint?
250 MAX% = VAL(MXP$)
260 IF MAX% <> 1200 THEN 340
270 CALL SEND(BOAD%,ADRS%(0),PNT$,NLEND%)        'Set point CH1,0
280 FOR I% = 0 TO MAX%
290 VOLT% = 125 * SIN(3.14 * I% / 500) + 125
300 SND$ = ADT$ + STR$(VOLT%)
310 CALL SEND(BOAD%,ADRS%(0),SND$,NLEND%)
320 NEXT I%
330 CALL SEND(BOAD%,ADRS%(0),WAV$,NLEND%)        'Wave display
340 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%,UNT$)     'UN TALK
350 CALL ENABLELOCAL(BOAD%,ADRS%(0))             'Enable operations
360 END
```

Example 6 Making storage condition settings

Line 150	Set ADRS%(0) to address of 8852.
Lines 170-180	Send interface clear, and switch to remote mode.
Lines 200-310	Set the 8852 function, trigger conditions, etc.
Line 330	Enter measurement operation mode with the conditions set.
Line 350	Release talker.
Line 360	End remote mode.

```
100 ' ---------------------------------------
110 ' 8852 Sample program No.1
120 ' You must merge this code with DECL.BAS
130 ' ---------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
160 '
170 CALL SENDIFC(BOAD%)                          'Clear interface
180 CALL ENABLEREMOTE(BOAD%, ADRS%(0))           'Enable remote
190 '
200 GOSUB 410                                    'FUNCTION MEM
210 GOSUB 410                                    'TIME/DIV 1ms
220 GOSUB 410                                    'SHOT 15DIV
230 '
240 GOSUB 410                                    'Trigger source OR
250 GOSUB 410                                    'LEVEL trigger
260 GOSUB 410                                    'Pre-trigger 5%
270 GOSUB 410                                    'LEVEL 60%
280 GOSUB 410                                    'SLOPE UP
290 GOSUB 410                                    'CH2 trigger OFF
300 GOSUB 410                                    'CH3 trigger OFF
310 GOSUB 410                                    'CH4 trigger OFF
320 '
330 GOSUB 410                                    '<START>
340 '
350 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%, UNT$)    'UN TALK
360 CALL ENABLELOCAL(BOAD%, ADRS%(0))            'Enable operations
370 END
380 ' ---------------------------------------
390 ' Send data
400 ' ---------------------------------------
410 READ COMMAND$
420 CALL SEND(BOAD%, ADRS%(0), COMMAND$, NLEND%)
430 RETURN
440 ' ---------------------------------------
450 ' data table
460 ' ---------------------------------------
470 DATA ":FUNCTION MEM"
480 DATA ":CONFIGURE:TDIV 1.E-3"
490 DATA ":CONFIGURE:SHOT 15"
500 DATA ":TRIGGER:SOURCE OR"
510 DATA ":TRIGGER:KIND CH1,LEVEL"
520 DATA ":TRIGGER:PRETRIG 5"
530 DATA ":TRIGGER:LEVEL CH1,60"
540 DATA ":TRIGGER:SLOPE CH1,UP"
550 DATA ":TRIGGER:KIND CH2,OFF"
560 DATA ":TRIGGER:KIND CH3,OFF"
570 DATA ":TRIGGER:KIND CH4,OFF"
580 DATA ":START"
```

Example 7 Start measurement operation mode, and if no trigger is detected execute a STOP.

Line 150	Set ADRS%(0) to address of 8852.
Lines 160-170	Send interface clear, and switch to remote mode.
Lines 190-260	Set the function and trigger conditions. Clear event status register 0. Clear the standard event status register.
Line 280	Enter measurement operation mode.
Lines 320-360	At fixed intervals, check whether the trigger has been applied. Read event status register 0, and check if bit 2 is set. When it is, go to line 410.
Lines 370-390	If no trigger has been detected, abort measurement.
Lines 410-440	If a trigger has been detected, read event status register 0, and check that bit 1 is set, confirming that measurement operation has started.
Lines 460-470	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8852 Sample program No.2
120 ' You must merge this code with DECL.BAS
130 ' ------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
160 CALL SENDIFC(BOAD%)                          'Clear interface
170 CALL ENABLEREMOTE(BOAD%,ADRS%(0))            'Enable remote
180 '
190 GOSUB 520                                    'Enable SESER bit
200 GOSUB 520                                    'TIME/DIV 1ms,SHOT
15DIV
210 GOSUB 520                                    'Trigger source OR
220 GOSUB 520                                    'LEVEL trigger
CH1,CH2
230 GOSUB 520                                    'Trigger OFF CH3,CH4
240 GOSUB 520                                    'Trigger CH1,60%,UP
250 GOSUB 520                                    'Trigger CH2,60%,UP
260 GOSUB 520                                    'Trigger MODE SINGLE
270 '
280 GOSUB 520                                    '<START>
290 '
300 ESR$ = ":ESR0?"
310 '
320 FOR W% = 1 TO 100
330 CALL SEND(BOAD%,ADRS%(0),ESR$,NLEND%)
340 GOSUB 580
350 IF (ESR0% AND &H4) <> 0 THEN 410
360 NEXT W%
370 PRINT "Not Trigger"
380 GOSUB 520
390 GOTO 460
400 '
410 CALL SEND(BOAD%,ADRS%(0),ESR$,NLEND%)
420 GOSUB 580
430 IF (ESR0% AND &H2) = 0 THEN 410
440 PRINT "Storage end"
450 '
460 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%,UNT$)     'UN TALK
470 CALL ENABLELOCAL(BOAD%,ADRS%(0))             'Enable operations
480 END
490 ' ------------------------------------
500 ' Send data
510 ' ------------------------------------
520 READ COMMAND$
530 CALL SEND(BOAD%,ADRS%(0),COMMAND$,NLEND%)
540 RETURN
550 ' ------------------------------------
560 ' Receive data
570 ' ------------------------------------
580 READING$ = SPACE$(10)
590 CALL RECEIVE(BOAD%,ADRS%(0),READING$,STOPEND%)
600 ESR0% = VAL(READING$)
610 RETURN
620 ' ------------------------------------
630 ' data table
640 ' ------------------------------------
650 DATA "*CLS;:ESE0 6;:FUNCTION MEM"
660 DATA ":CONFIGURE:TDIV 1.E-3;SHOT 15"
670 DATA ":TRIGGER:SOURCE OR"
680 DATA ":TRIGGER:KIND CH1,LEVEL;KIND CH2,LEVEL"
690 DATA ":TRIGGER:KIND CH3,OFF;KIND CH4,OFF"
700 DATA ":TRIGGER:LEVEL CH1,60;SLOPE CH1,UP"
710 DATA ":TRIGGER:LEVEL CH2,60;SLOPE CH2,UP"
720 DATA ":TRIGGER:MODE SINGLE"
730 DATA ":START"
740 DATA ":ABORT"
```

Example 8 Displaying input ranges for the channels on the screen.

Line 160	Set ADRS%(0) to address of 8852.
Lines 170-180	Send interface clear, and switch to remote mode.
Line 240	Disable headers.
Lines 250-350	Screen display.
Lines 370-480	Read real time data for the channels into variables.
Line 500-670	Read real time data for the channels and display.
Lines 700-710	Release talker and remote mode.

```
100 ' ---------------------------------------
110 ' 8852 Sample program No.3
120 ' You must merge this code with DECL.BAS
130 ' ---------------------------------------
140 SCREEN 9
150 BOAD% = 0
160 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
170 CALL SENDIFC(BOAD%)                          'Clear interface
180 CALL ENABLEREMOTE(BOAD%, ADRS%(0))           'Enable remote
190 HEA$ = ":HEADER OFF"
200 CH1$ = ":MEMORY:AREAL? CH1"
210 CH2$ = ":MEMORY:AREAL? CH2"
220 CH3$ = ":MEMORY:AREAL? CH3"
230 CH4$ = ":MEMORY:AREAL? CH4"
240 CALL SEND(BOAD%, ADRS%(0), HEA$, NLEND%)     'Header OFF
250 CLS
260 LOCATE 3,5:PRINT "<LEVEL MONITOR>"
270 LOCATE 4,1:PRINT "100"
280 LOCATE 13,1:PRINT " 50"
290 LOCATE 22,1:PRINT "  0"
300 LOCATE 1,52:PRINT "CH1             CH2"
310 LOCATE 2,52:PRINT "CH3             CH4"
320 LINE (30,57)-(620,307),7,B,&HCCCC            'Frame
330 FOR Y% = 82 TO 282 STEP 25
340 LINE (30,Y%)-(620,Y%),7,,&H1010
350 NEXT Y%
360 '
370 LINE (440,8)-(490,10),6,B
380 CALL SEND(BOAD%, ADRS%(0), CH1$, NLEND%)     'CH1 ADATA
390 GOSUB 760:Y10% = ADT%
400 LINE (560,8)-(610,10),5,B
410 CALL SEND(BOAD%, ADRS%(0), CH2$, NLEND%)     'CH2 ADATA
420 GOSUB 760:Y20% = ADT%
430 LINE (440,24)-(490,26),4,B
440 CALL SEND(BOAD%, ADRS%(0), CH3$, NLEND%)     'CH3 ADATA
450 GOSUB 760:Y30% = ADT%
460 LINE (560,24)-(610,26),3,B
470 CALL SEND(BOAD%, ADRS%(0), CH4$, NLEND%)     'CH4 ADATA
480 GOSUB 760:Y40% = ADT%
490 '
500 FOR X% = 30 TO 618 STEP 2
510 CALL SEND(BOAD%, ADRS%(0), CH1$, NLEND%)     'CH1 ADATA
520 GOSUB 760
530 LINE (X%,307-Y10%)-(X%+2,307-ADT%),6
540 Y10% = ADT%
550 CALL SEND(BOAD%, ADRS%(0), CH2$, NLEND%)     'CH2 ADATA
560 GOSUB 760
570 LINE (X%,307-Y20%)-(X%+2,307-ADT%),5
580 Y20% = ADT%
590 CALL SEND(BOAD%, ADRS%(0), CH3$, NLEND%)     'CH3 ADATA
600 GOSUB 760
610 LINE (X%,307-Y30%)-(X%+2,307-ADT%),4
620 Y30% = ADT%
630 CALL SEND(BOAD%, ADRS%(0), CH4$, NLEND%)     'CH4 ADATA
640 GOSUB 760
650 LINE (X%,307-Y40%)-(X%+2,307-ADT%),3
660 Y40% = ADT%
670 NEXT X%
680 IF INKEY$ = "" GOTO 250
690 SCREEN 0
700 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%, UNT$)    'UN TALK
710 CALL ENABLELOCAL(BOAD%, ADRS%(0))            'Enable operations
720 END
730 ' ---------------------------------------
740 ' Receive data
750 ' ---------------------------------------
760 READING$ = SPACE$(32)
770 CALL RECEIVE(BOAD%, ADRS%(0), READING$, STOPEND%)
780 ADT% = VAL(READING$)
790 RETURN
```

Example 9 Saving stored data onto drive 2 (sequential file)

Line 150	Set ADRS%(0) to address of 8852.
Lines 180-190	Send interface clear, and switch to remote mode.
Line 200	Set the jump addresses for if an error occurs, to ensure that the program does not exit with the file left open.
Lines 250-260	Disable headers, and read the number of stored data values into MAX%.
Lines 310-330	Input the channels to be saved and the filename.
Line 390	Set the stored data output point.
Line 410	Write the number of data values saved, at the beginning of the file.
Lines 420-460	Read the stored data from the 8852, and save sequentially.
Lines 530-540	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8852 Sample program No.4
120 ' You must merge this code with DECL.BAS
130 ' ------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
160 HEA$ = ":HEADER OFF;:MEMORY:MAXPOINT?"
170 ADT$ = ":MEMORY:ADATA? 1"
180 CALL SENDIFC(BOAD%)                          'Clear interface
190 CALL ENABLEREMOTE(BOAD%,ADRS%(0))            'Enable remote
200 ON ERROR GOTO 500
210 '
220 CLS:LOCATE 2,10
230 PRINT "< Storage Data SAVE >"
240 PRINT:PRINT
250 CALL SEND(BOAD%,ADRS%(0),HEA$,NLEND%)        'Header OFF
260 GOSUB 590:MAX% = VALUE%                      'Max point?
270 IF MAX% <> 0 THEN 300                        'Output ready?
280 PRINT "No storage data !!"
290 GOTO  520
300 '
310 PRINT " Max point=";MAX%:PRINT
320 INPUT " Channel(CH1-CH4)";CH$                'Input channel No.
330 INPUT " File name";NA$                       'Input
(drive)+filename
340 PRINT:PRINT
350 '
360 OPEN NA$ FOR OUTPUT AS #1                    'Open file
370 '
380 PNT$ = ":MEMORY:POINT "+CH$+",0"             'Set output point
390 CALL SEND(BOAD%,ADRS%(0),PNT$,NLEND%)
400 '
410 PRINT #1,MAX%                                'Save max point
420 FOR I% = 0 TO MAX%
430 CALL SEND(BOAD%,ADRS%(0),ADT$,NLEND%)
440 GOSUB 590                                    'Get ADATA
450 PRINT #1,VALUE%                              'Save ADATA
460 NEXT I%
470 PRINT " Completed."
480 GOTO 520
490 '
500 PRINT "ERROR !!"
510 '
520 CLOSE #1                                     'Close file
530 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%,UNT$)     'UN TALK
540 CALL ENABLELOCAL(BOAD%,ADRS%(0))             'Enable operations
550 END
560 ' ------------------------------------
570 ' Receive data
580 ' ------------------------------------
590 READING$ = SPACE$(30)
600 CALL RECEIVE(BOAD%,ADRS%(0),READING$,STOPEND%)
610 VALUE% = VAL(READING$)
620 RETURN
```

Example 10 Reading the data saved in Example 9, and loading it into the 8852.

Line 150	Set ADRS%(0) to address of 8852.
Lines 180-190	Send interface clear, and switch to remote mode.
Line 200	Set the jump addresses for if an error occurs, to ensure that the program does not exit with the file left open.
Lines 250-260	Specify the filename to be opened and channel.
Line 310	Set the stored data input point.
Lines 340-350	Read the number of stored data values into VALUE%.
Lines 380-420	Read the data from the file, and write to memory on the 8852.
Lines 500-510	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8852 Sample program No.5
120 ' You must merge this code with DECL.BAS
130 ' ------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%              'GP-IB Address = 5
160 HEA$ = ":HEADER OFF;:MEMORY:MAXPOINT?"
170 DIS$ = ":DISPLAY:CHANGE DISPLAY"
180 CALL SENDIFC(BOAD%)                          'Clear interface
190 CALL ENABLEREMOTE(BOAD%,ADRS%(0))            'Enable remote
200 ON ERROR GOTO 470
210 '
220 CLS:LOCATE 2,10
230 PRINT "< Storage Data LOAD >"
240 PRINT:PRINT
250 INPUT " Channel(CH1-CH4)";CH$                'Input channel No.
260 INPUT " File name";NA$                       'Input
(drive)+filename
270 '
280 OPEN NA$ FOR INPUT AS #1                     'Open file
290 '
300 PNT$ = ":MEMORY:POINT "+CH$+",0"             'Set output point
310 CALL SEND(BOAD%,ADRS%(0),PNT$,NLEND%)
320 '
330 INPUT #1,MAX%                                'Load max point
340 CALL SEND(BOAD%,ADRS%(0),HEA$,NLEND%)        'Header OFF
350 GOSUB 560                                    'Max point?
360 IF VALUE% <> MAX% THEN 470                   'Input ready?
370 '
380 FOR I% = 0 TO MAX%
390 INPUT #1,DAT%                                'Load ADATA
400 ADT$ = ":MEMORY:ADATA "+STR$(DAT%)
410 CALL SEND(BOAD%,ADRS%(0),ADT$,NLEND%)        'Set ADATA
420 NEXT I%
430 PRINT " Completed."
440 CALL SEND(BOAD%,ADRS%(0),DIS$,NLEND%)        'Display wave
450 GOTO 490
460 '
470 PRINT "ERROR !!"
480 '
490 CLOSE #1                                     'Close file
500 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%,UNT$)     'UN TALK
510 CALL ENABLELOCAL(BOAD%,ADRS%(0))             'Enable operations
520 END
530 ' ------------------------------------
540 ' Receive data
550 ' ------------------------------------
560 READING$ = SPACE$(30)
570 CALL RECEIVE(BOAD%,ADRS%(0),READING$,STOPEND%)
580 VALUE% = VAL(READING$)
590 RETURN
```

Example 11 Setting measurement conditions, and starting measurement operation after synchronizing with the *OPC command.

Line 150	Set ADRS%(0) to address of 8852.
Lines 160-170	Send interface clear, and switch to remote mode.
Line 180	Set jump address for if a service request is received.
Line 190	Enable bit 5 (ESB) of the status byte by the service request enable register.
Line 200	Enable bit 0 of the standard event status register by the standard event status enable register.
Line 210	Clear the status byte associated queue.
Line 220	Enable the service request interrupt.
Lines 240-300	Set the measurement conditions.
Line 310	Wait for a service request.
Lines 330-340	Serial polling to read the status byte.
Line 360	After confirming the completion of condition setting, start measurement operation.
Line 380	Disable service request interrupt.
Lines 390-400	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8852 Sample program No.6
120 ' You must merge this code with DECL.BAS
130 ' ------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%            'GP-IB Address = 5
160 CALL SENDIFC(BOAD%)                        'Clear interface
170 CALL ENABLEREMOTE(BOAD%,ADRS%(0))          'Enable remote
180 ON PEN GOSUB 330
190 GOSUB 450                                  'Mask SRQ
200 GOSUB 450                                  'Mask SESER
210 GOSUB 450                                  'Clear statusbyte
220 PEN ON
230 '
240 GOSUB 450                                  'Set FUNCTION
250 GOSUB 450                                  'TIME/DIV 1ms
260 GOSUB 450                                  'SHOT 15DIV
270 '
280 GOSUB 450                                  'CH1 <- LEVEL TRIG.
290 GOSUB 450                                  'Pre-TRIG. 5%
300 GOSUB 450                                  'LEVEL 60%,SLOPE UP
310 GOTO 310
320 '
330 CALL IBRSP(ADRS%,S%)
340 DCL$ = CHR$(DCL%):CALL IBCMD(BOAD%,DCL$)   'Clear buffer
350 PRINT "START OK "
360 GOSUB 450                                  '< START >
370 '
380 PEN OFF
390 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%,UNT$)   'UN TALK
400 CALL ENABLELOCAL(BOAD%,ADRS%(0))           'Enable operations
410 END
420 ' ------------------------------------
430 ' Send data
```

```
440 ' -------------------------------------
450 READ COMMAND$
460 CALL SEND(BOAD%,ADRS%(0),COMMAND$,NLEND%)        'Mask SRQ
470 RETURN
480 ' -------------------------------------
490 ' DATA table
500 ' -------------------------------------
510 DATA "*SRE 32"
520 DATA "*ESE 1"
530 DATA "*CLS"
540 DATA ":FUNCTION MEM"
550 DATA ":CONFIGURE:TDIV 1.E-3"
560 DATA ":CONFIGURE:SHOT 15"
570 DATA ":TRIGGER:KIND CH1,LEVEL"
580 DATA ":TRIGGER:PRETRIG 5"
590 DATA ":TRIG:LEVEL CH1,60;SLOPE CH1,UP;*OPC"
600 DATA ":START"
```

Example 12 Using service requests to display errors

Line 150	Set ADRS%(0) to address of 8852.
Lines 160-170	Send interface clear, and switch to remote mode.
Line 180	Set jump address for if a service request is received.
Line 210	Enable bit 5 (ESB) of the status byte by the service request enable register.
Line 220	Enable bits 2, 3, 4, and 5 of the standard event status register by the standard event status enable register.
Line 230	Clear the status byte associated queue.
Line 240	Enable the service request interrupt.
Line 260	Set the function.
Line 290	Set averaging. (Error source)
Line 340	Serial polling to read the status byte.
Line 380	Read the standard event status register.
Lines 400-430	From the value read, determine the error, and display it.
Line 440	Disable service request interrupt.
Lines 440-460	Release talker and remote mode.

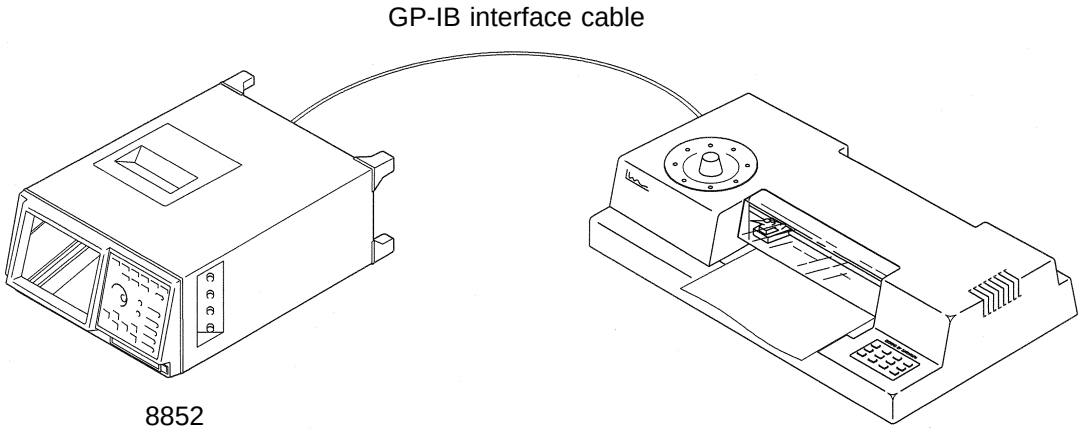
```
100 ' ------------------------------------
110 ' 8852 Sample program No.7
120 ' You must merge this code with DECL.BAS
130 ' ------------------------------------
140 BOAD% = 0
150 ADRS%(0) = 5:ADRS%(1) = NOADDR%            'GP-IB Address = 5
160 CALL SENDIFC(BOAD%)                        'Clear interface
170 CALL ENABLEREMOTE(BOAD%,ADRS%(0))          'Enable remote
180 ON PEN GOSUB 340
190 SRE$="*SRE 32":ESE$="*ESE 60"
200 SCL$="*CLS":ESR$="*ESR?"
210 CALL SEND(BOAD%,ADRS%(0),SRE$,NLEND%)      'Mask SRQ
220 CALL SEND(BOAD%,ADRS%(0),ESE$,NLEND%)      'Mask SESER
230 CALL SEND(BOAD%,ADRS%(0),SCL$,NLEND%)      'Clear statusbyte
240 PEN ON
250 FUN$=":FUNCTION MEM"
260 CALL SEND(BOAD%,ADRS%(0),FUN$,NLEND%)      'Set FUNCTION
270 I% = 5
280 AVR$=":CONFIGURE:AVERAGE "+STR$(I%)
290 CALL SEND(BOAD%,ADRS%(0),AVR$,NLEND%)      'Set AVERAGE
300 I% = I% + 50:GOTO 280
310 ' ------------------------------------
320 ' Service request operation
330 ' ------------------------------------
340 CALL IBRSP(ADRS%,S%)
350 DCL$ = CHR$(DCL%):CALL IBCMD(BOAD%,DCL$)   'Clear buffer
360 CALL SEND(BOAD%,ADRS%(0),ESR$,NLEND%)      'ERROR kind?
370 CMD$ = SPACE$(8)
380 CALL RECEIVE(BOAD%,ADRS%(0),CMD$,STOPEND%) 'receive ERROR
390 B = VAL(CMD$)
400 IF (B AND &H4) <> 0 THEN PRINT "Query ERROR"
410 IF (B AND &H8) <> 0 THEN PRINT "Machine ERROR"
420 IF (B AND &H10) <> 0 THEN PRINT "Execute ERROR"
430 IF (B AND &H20) <> 0 THEN PRINT "Command ERROR"
440 PEN OFF
```

```
450 UNT$ = CHR$(UNT%):CALL IBCMD(BOAD%,UNT$)     'UN TALK
460 CALL ENABLELOCAL(BOAD%,ADRS%(0))             'Enable operations
470 END
```

Chapter 6
Plotter Output

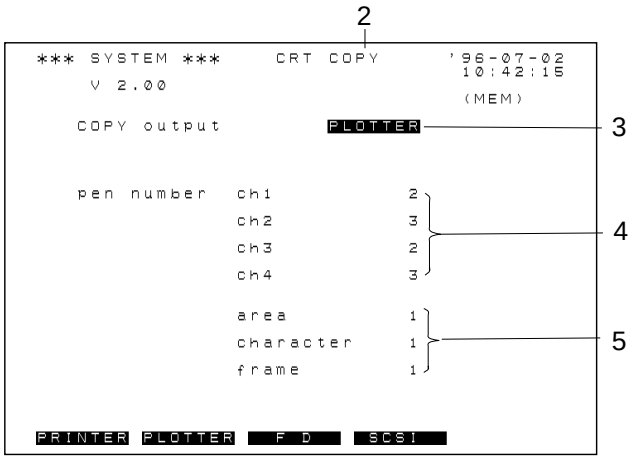
Settings for plotter output

It is possible to use an HP-GL compatible plotter to output the waveform from the 8852 display screen.



Connect the plotter to the 8852 using the GP-IB interface cable.
The settings for the plotter are all carried out using the system screen (CRT COPY) on the 8852.

Procedure



System screen (CRT COPY)

1. Press the SYSTEM key, to display the system screen.
2. Press the **CRT COPY** soft key to get the plotter settings screen.

   Next, use the CURSOR keys to move the flashing cursor through the items in the order listed left, and use the soft keys to make the settings.
3. Plotter output settings.

   Press the **PLOTTER** soft key in the "COPY OUTPUT" item.

   [PRINTER, PLOTTER, FD, SCSI]

4. Pen number for each channel
   - Set the pen number to be used by each channel.
   - Choose from pens 1 through 8 to assign a color for each waveform.
   - Use the ↓ and ↑ soft keys or the rotary knob to make the settings. [—, 1 to 8]

     —: No pen.

NOTE

- Logic channel groups CHA through CHC get the same pen assignments as the corresponding analog channels 1 through 3.
- If there is a comment appended to a channel, it will also appear on the plot. (For more details of comments, see Section 13.4, "Adding Comments" in the instruction manual for the 8852.)

5. Area, character, and frame pen number

   area — Sets the pen number to be used for the waveform decision area.

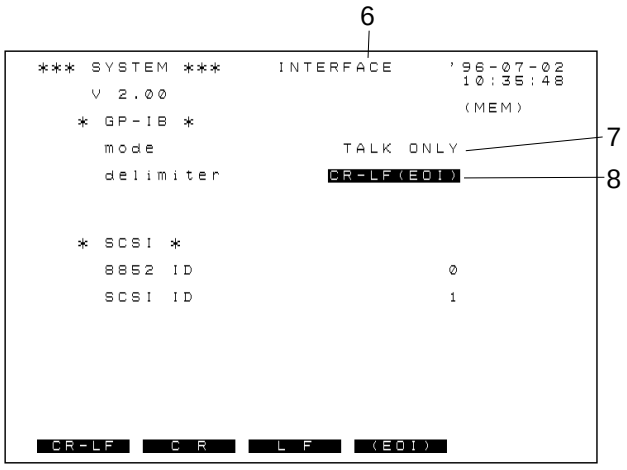
   character — Sets the pen number to be used for the characters, trigger mark, cursor readout value, waveform parameter calculation result, and comments.

   frame — Sets the pen number to be used for the frame, grids, and A and B cursors.

   The setting procedure is the same as in step 4.

This completes settings in the system screen (CRT COPY).

Next carry out the GP-IB settings.



System screen (INTERFACE)

6. Press the **INTER** soft key, to display the interface settings screen.
7. GP-IB operation mode

   Press the **TALK** soft key to select talk only mode for the plotter.

   [ADDRESSABLE, TALK ONLY, DISABLE]
8. Delimiter selection.

   The delimiter sequence required depends on the plotter being used. Consult the documentation accompanying the plotter, then use the soft keys to make the selection.

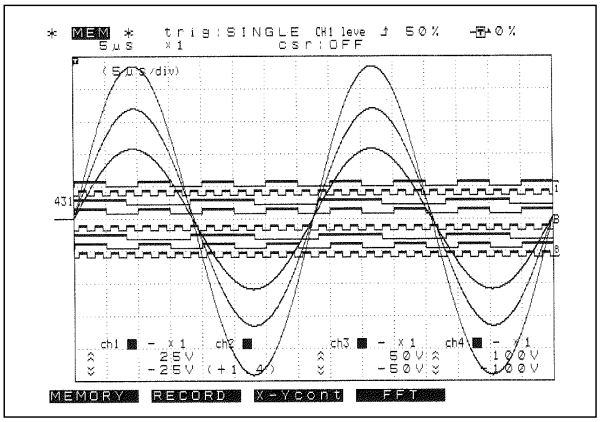
   [CR-LF(EOI), CR(EOI), LF(EOI), (EOI)]

Set the plotter to listen-only mode.

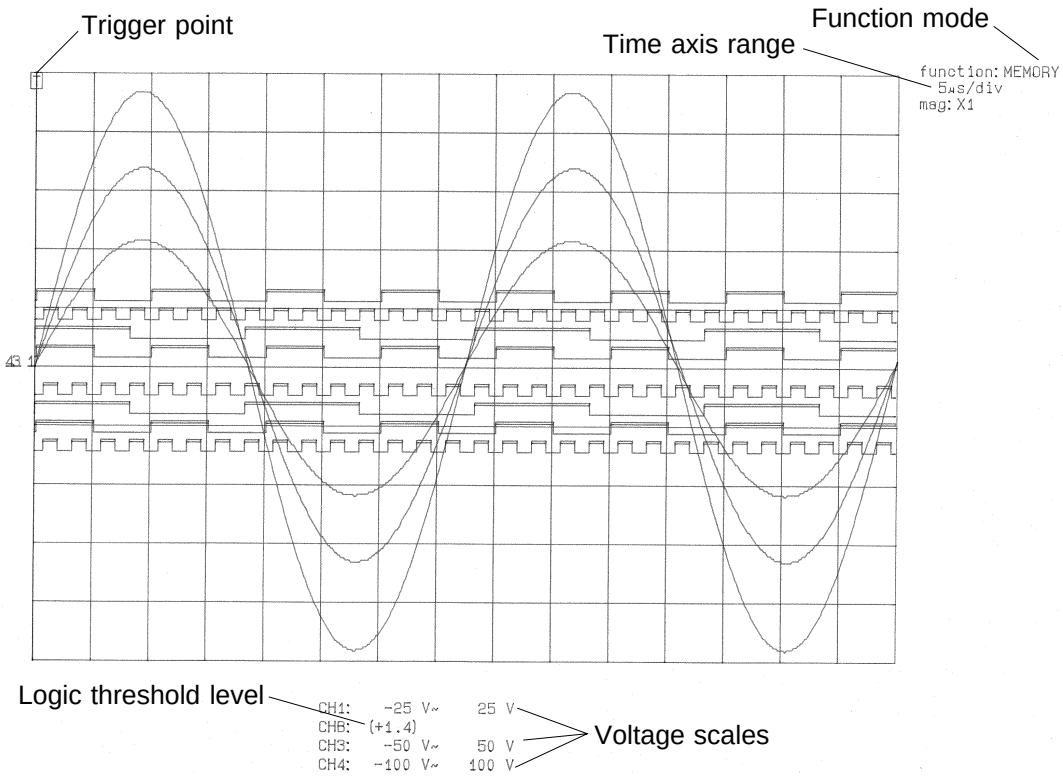
This completes the 8852 settings.

9. Press the DISP key to show the display screen.
10. Press the COPY key to begin plotter output.

The plotter output appears as shown in the following figures.



Display screen



NOTE

- In overwriting mode, only the last waveform captured will be printed.
- The waveform in the recorder function and the compression waveform are traced twice.

Chapter 7
Device Compliance Statement

The following information relates to the compliance with the IEEE 488.2 standard.

(1) IEEE 488.1 interface functions

These are detailed in Section 2.2, "Interface functions".

(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 commands :MEMORY:POINT, and :MEMORY:FFTPOINT are all reinitialized, and all other items are preserved.

(5) List of message exchange options

(a) Input buffer capacity and operation

The 8852 has an input buffer of 512 bytes capacity. If the data accumulated in this buffer exceeds 512 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

On the 8852, 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 4.3, "Standard Commands Stipulated by IEEE 488.2", and Section 4.4, "Commands Specific to the 8852."

(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 4.1, "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 4.3, "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 8852 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 3.6, "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

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?
	Is the GP-IB address of the 8852 unit correctly set? Does it clash the address of other equipment on the same bus?
	Are all the devices that are connected powered on?
The 8852 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 8852? 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.
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.

Index

For the GP-IB commands, refer to "Ref page" in Section 4.1, "Command Summary."

HIOKI 8852 GP-IB INTERFACE

Instruction Manual

Publication date: March Revised edition 2

Edited and published by HIOKI E.E. CORPORATION
Technical Sales Support Section

All inquiries to Sales and Marketing International Department
81 Koizumi, Ueda, Nagano, 386-11, Japan
FAX: 0268-28-0568 TEL: 0268-28-0562
E-mail: os-com@hioki.co.jp

Printed in Japan 8852A983-02

HIOKI

HIOKI E. E. CORPORATION

HEAD OFFICE 81 Koizumi, Ueda, Nagano 386-11, Japan
FAX. 0268-28-0568 / TEL. 0268-28-0562
E-mail: os-com@hioki.co.jp

8852A983-02 97-03-0001H