HIOKI

INSTRUCTION MANUAL

8845・8846

MEMORY HiCORDER

9537

GP-IB INTERFACE

HIOKI E. E. CORPORATION

Contents

Introduction

Thank you for purchasing this HIOKI "9537 GP-IB INTERFACE." To get the maximum performance from the unit, please read this manual first, and keep this at hand.

Safety Notes

This Instruction Manual provides information and warnings essential for operating this equipment in a safe manner and for maintaining it in safe operating condition. Before using this equipment, be sure to carefully read the following safety notes.

⚠ DANGER	**During high voltage measurement, incorrect measurement procedures could result in injury or death, as well as damage to the equipment. Please read this manual carefully and be sure that you understand its contents before using the equipment. The manufacturer disclaims all responsibility for any accident or injury except that resulting due to defect in its product.**

Safety symbols

	• This symbol is affixed to locations on the equipment where the operator should consult corresponding topics in this manual (which are also marked with the ⚠ symbol) before using relevant functions of the equipment. • In the manual, this mark indicates explanations which it is particularly important that the user read before using the equipment.

The following symbols are used in this Instruction Manual to indicate the relative importance of cautions and warnings.

⚠ DANGER	Indicates that incorrect operation presents extreme danger of accident resulting in death or serious injury to the user.
⚠ WARNING	Indicates that incorrect operation presents significant danger of accident resulting in death or serious injury to the user.
⚠ CAUTION	Indicates that incorrect operation presents possibility of injury to the user or damage to the equipment.
NOTE	Denotes items of advice related to performance of the equipment or to its correct operation.

Notes on Use

In order to ensure safe operation and to obtain maximum performance from the unit, observe the cautions listed below.

⚠ WARNING

- To avoid electric shock, before replacing the input units, turn the power off and disconnect the all input cables and power cord.
- The fixing screws must be firmly tightened or the input unit may not function up to specification, or may even fail.
- To avoid the danger of electric shock, never operate the unit with an input unit removed. If you should wish to use the unit after removing an input unit, fit a blank panel over the opening of the removed unit.
- The 9537 GP-IB INTERFACE is not isolated from the 8845, 8846 units. (common with ground)

Chapter Summary

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.

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

The 9537 GP-IB INTERFACE is applied to the following standards.

IEEE Standard 488.1-1987
IEEE Standard 488.2-1987

2.2 Interface Functions

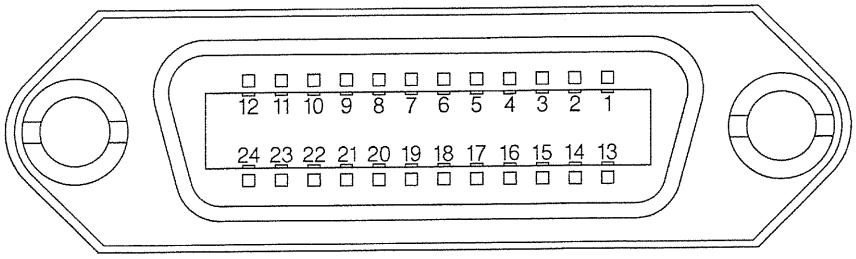
Function	Implementation
SH1	SH (Source Handshake) - All Functions
AH1	AH (Acceptor Handshake) - All Functions
T5	Basic Talk Function, Serial Poll Function, Talk Only Function MLA (My Listen Address) Talk Release Function
L4	Basic Listener Function MTA (My Talk Address) Listen Release Function
SR1	SR (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 8845 RC10(F)-224R-LNA (make by hirose) or compatible.
On the cable 57-10240 (made by DDK) or compatible.



Pin Arrangement Diagram for the GP-IB Interface Connector on the 8845

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

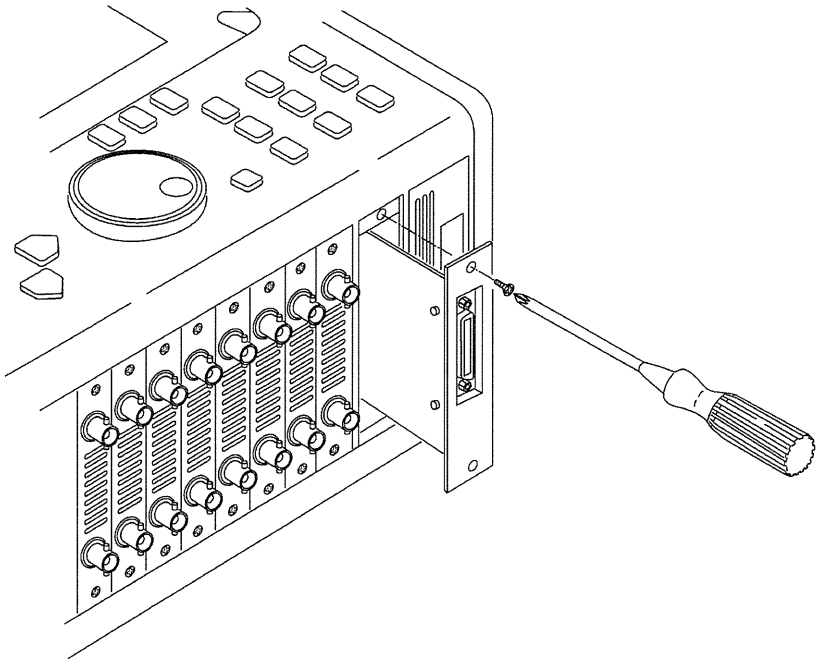
Chapter 3
Installing the 9537 GP-IB INTERFACE

3.1 Installing Method

⚠ WARNING

- To prevent electrical shock, before adding or replacing the input unit, check that the power for the unit is off and the power cord and input cables are disconnected.
- The fixing screws must be firmly tightened or the input unit may not function up to to specification, or may even fail.
- To avoid the danger of electric shock, never operate the unit with the 9537 GP-IB INTERFACE removed. If you should wish to use the unit after removing an input unit, fit a blank panel over the opening of the removed unit.

1. Remove the input cables and thermocouples from all input units.
2. Power off the 8845, 8846 main unit, and disconnect the power cord.
3. Grasp the connector and insert the extension slot.
4. Fix the screws with a Phillips screwdriver, as shown in the figure below.

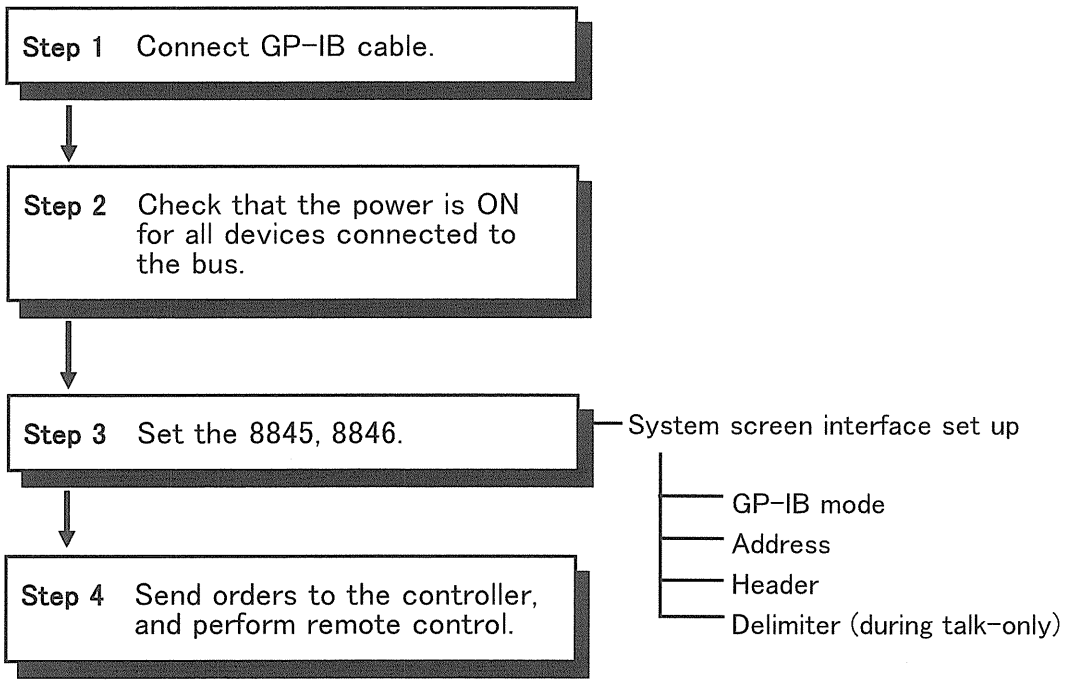


NOTE To remove the unit, remove the two fixing screws on the panel, connect the GP-IB cable again, and then pull out the unit after fixing the screws on the connector.

Chapter 4
Method of Operation

4

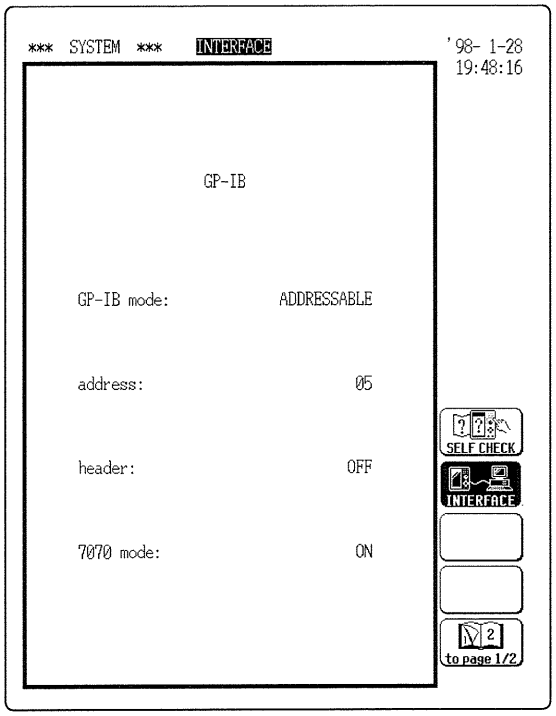
4.1 Basic Operational Procedure



4.2 Setup Procedure

- On the 8845, 8846, set the GP-IB address for the unit, and select whether or not to use headers mode, and delimiter in messages output by the 8845, 8846.
- Use the interface setting screen, accessed from the "system" screen.

Method (Screen: SYSTEM "INTERFACE")



1. Press the **SYSTEM** key to display the system screen.
2. Press the F5 [to page 2/2], and select F2 [INTERFACE].
3. Set the GP-IB mode.
   Set the GP-IB operation mode, address for this unit on the bus.

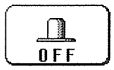
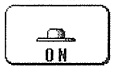
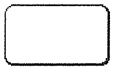
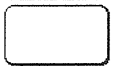
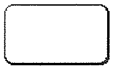
Function key display	Meaning
ADDRESS	: Assign a device address, so this unit can be used both as talker and listener.
TALK ONLY	: Use this unit as talker only.
DISABLE	: Do not use the GP-IB interface.

- When the "ADDRESSABLE" mode is selected, sets the address, and enables or disables the headers mode.
- When the "TALK ONLY" mode is selected, sets the delimiter.

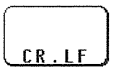
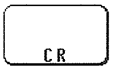
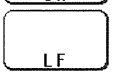
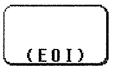
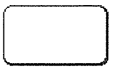
4. Set the GP-IB device address.
   Using the function keys or jog control, adjust the address of the device.

Function key display	Meaning
+	1 to 30
−	

5. Enable or disable the headers. (during addressable)
   Select whether or not this unit as talker should output an identifying header at the beginning of each message it sends.

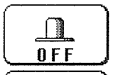
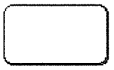
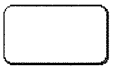
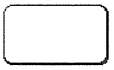
Function key display	Meaning
	: Sets header to off.
	: Sets header to on.
	
	
	

6. Set the GP-IB delimiter for talk-only mode.
   Select the appropriate delimiter sequence for the plotter being used.

Function key display	Meaning
	: CR+LF (EOI)
	: CR (EOI)
	: LF (EOI)
	: (EOI)
	

7. By connectingthe HIOKI 7070WAVEFORM GENERATOR and the 8846 can be transferred to the 7070.

   The HIOKI 7070 WAVEFORM GENERATOR can be purchased in Japan only.

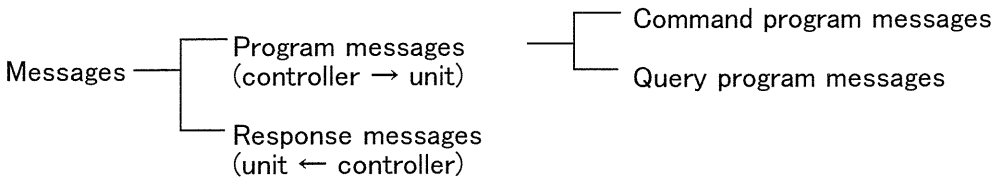
Function key display	Meaning
	: Sets the 7070 mode to off.
	: Sets the 7070 mode to on.
	
	
	

4.3 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:



Command messages orders for control of the device, such as for making settings or for reset or the like.

Query messages orders for responses relating to the results of operation, results of measurement, or the state of device settings.

Response messages 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 8845, 8846 accept 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 8845, 8846 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.

① Simple command type header

The first word constitute the header.

(Example) :HEADer ON

Simple command type header — Data

② 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

③ 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 unit, 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 25

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 25

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 25

(Example 2) :CONF:TDIV 1. E-3;SHOT 25

Both Example 1 and Example 2 are messages setting TIME/DIV to 1 ms and recording length to 25 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:

```
:CONFIGURE:TDIV 1.E-3
```

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

(8) Data format

The 8845 and 8846 use character data, decimal data and character string data.

① Character data

- The first character must be alphabetic.
- The characters after the first character can only be alphabetic characters, numerals, or underline characters (_).
- As alphabetic characters, during sending only upper case letters are used, but during receiving both upper case and lower case letters are permitted.

② 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 8845 and 8846 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 8845, 8846 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.

③ Character string data

- Character string data is enclosed within quotation marks.
- The data is composed of 8 bit ASCII characters.
- Characters which cannot be handled by the 8845, 8846 are replaced by spaces.
- When the 8845, 8846 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.

4.4 Remote Control

(1) Local state

This is the state in which the 8845, 8846 is controlled by its keys. When the power is turned on, the 8845, 8846 always comes up in local state.

(2) Remote state

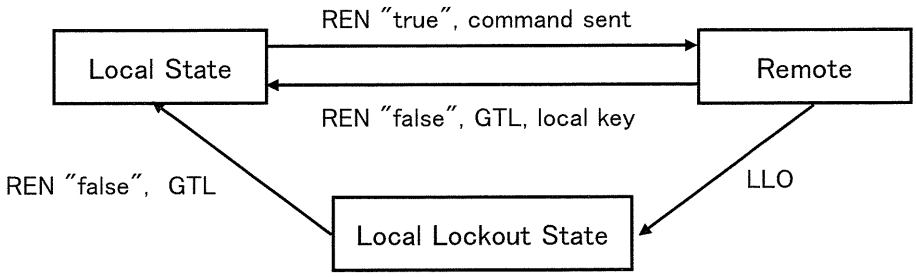
In this state the 8845, 8846 is controlled from the GP-IB interface (the REN line is "true"), and its keys are disabled. When in the remote state, the 8845, 8846 returns to local state if the local key (F5) is pressed.

(3) 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 8845, 8846 is prevented from returning to the local state. This state is called the local lockout state.

In order to return the 8845, 8846 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 8845, 8846 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

4.5 Device Clear

When the 8845, 8846 receive 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

4.6 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.

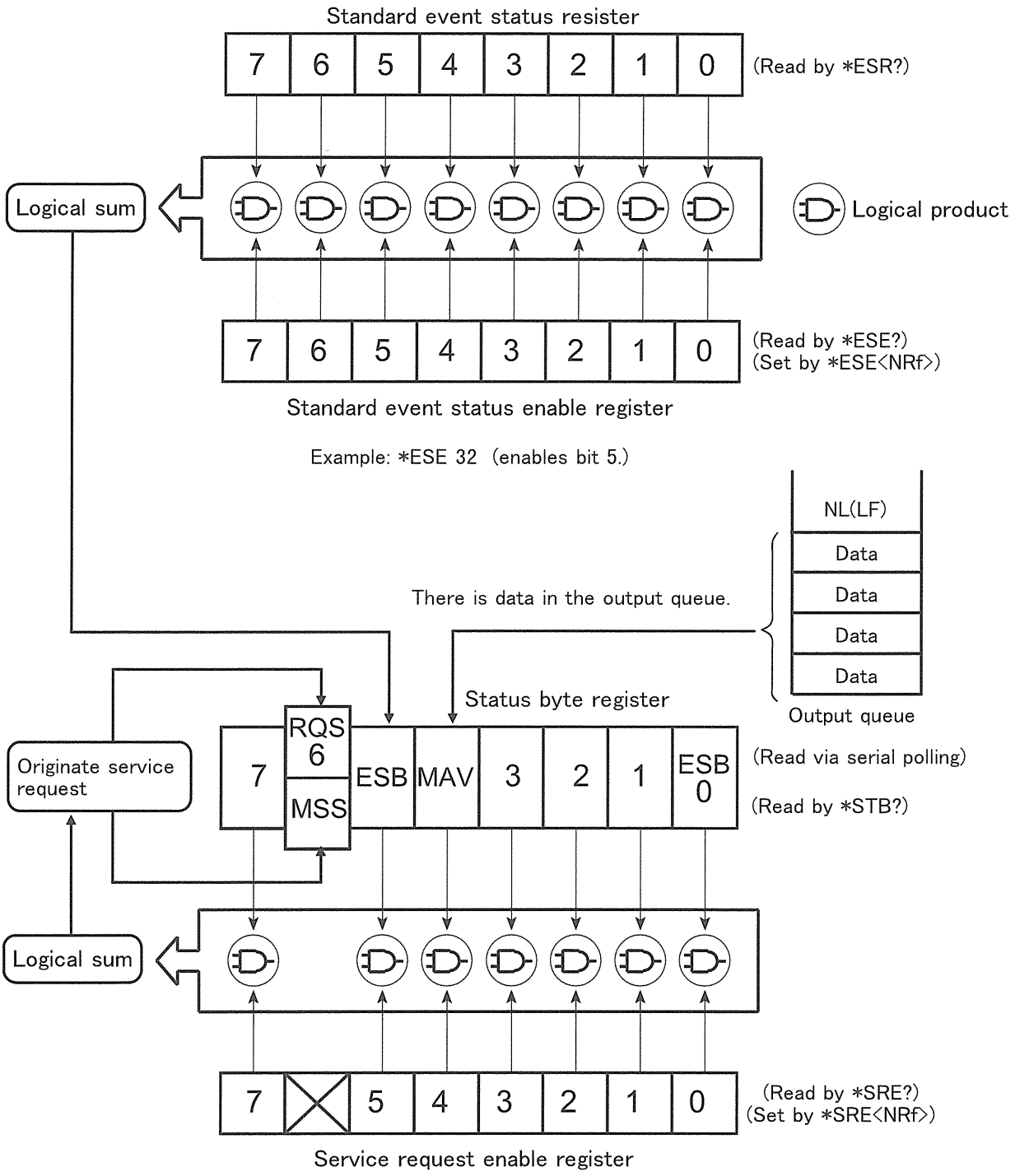
1. When the *CLS command is received.
2. When the contents have been read by an :ESR0? query.
3. When the power is turned off and turned on again.

The bits of event status register 0

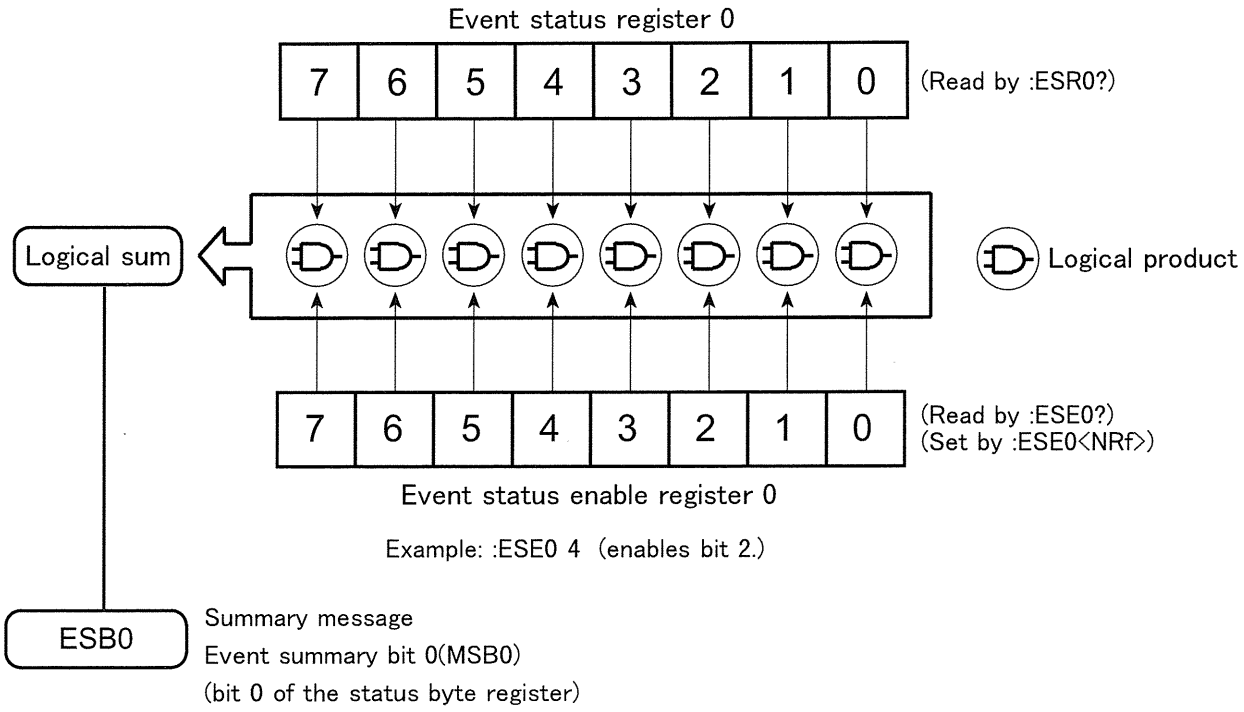
bit 7	Waveform decision fail (NG).
bit 6	Unused.
bit 5	Waveform parameter calculation finished.
bit 4	Waveform processing calculation finished.
bit 3	Printer operation finished (print, or copy output).
bit 2	Trigger wait finished (set when the trigger event occurs).
bit 1	Measurement operation concluded (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

4.7 The Input Buffer and the Output Queue

(1) Input buffer

The 8845, 8846 have an input buffer of 1024 bytes capacity.
Messages which are received are put into this buffer and executed in order.
However, an ABORT command is executed instantly as soon as it is received.

(2) Output queue

The 8845, 8846 have an output queue of 512 bytes capacity.
Response messages are accumulated in this queue and are read out from the controller.

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.

If the length of a response message has exceeded 512 bytes, a query error occurs.

4.8 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 5
Command Summary

5.1 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.	59
*OPT?	Whether channel 1 and 2 input units exist Whether channel 3 and 4 input units exist ..... Whether channel 15 and 16 input units exist	Queries device option provision.	59
*RST		Device initial setting.	60
*TST?	*A* <NR1> (0 = normal, 1 = failure)	Queries the result of the self-test.	60
*OPC		Sets the LSB of SESR after all action has been completed.	60
*OPC?	*A* <NR1>	Queries whether all action has been completed. ASCII [1] is the response.	61
*WAI		Wait until action fully completed.	61
*CLS		Clears the status byte and associated queues.	61
*ESE *A*	*A* = 0 to 255	Sets SESER.	62
*ESE?	*A* <NR1> *A* = 0 to 255	Queries SESER.	
*ESR?	*A* <NR1> *A* = 0 to 255	Queries SESR.	62
*SRE *A*	*A* = 0 to 255	Sets SRER.	63
*SRE?	*A* <NR1> *A* = 0 to 63, 128 to 191	Queries SRER.	
*STB?	*A* <NR1> *A* = 0 to 255	Reads the STB and the MSS bit, without performing serial polling.	63
:ESE0 *A* #	*A* = 0 to 255	Writes ESER0.	64
:ESE0? #	*A* <NR1> *A* = 0 to 255	Reads ESER0.	
:ESR0? #	*A* <NR1> *A* = 0 to 255	Reads ESR0.	64

#: specific to the 8845, 8846.

5.2 Commands Specific to the 8845, 8846

5.2.1 Execution Control etc. (common to all functions)

Command	Data (for a query, response data)	Explanation	Ref page
:STARt		Same as the START key.	65
:STOP		Same as the STOP key.	65
:ABORT		Forced halt.	65
:PRINt		Same as the PRINT key.	65
:HCOPy		Same as the COPY key.	66
:FEED *A*	*A* = 1 to 255 (unit mm)	Feeds the paper the specified distance.	66
:AUTO		Sets the time axis and the voltage axis automatically. (Only the memory recorder function)	66
:LIGHt *A$*	*A$* = OFF, ON	Enables and disables LCD back light.	66
:LIGHt?	*A$*	Queries LCD back light.	
:ERRor?	*A* <NR1> error number	Queries the unit error number.	67
:HEADer *A$*	*A$* = OFF, ON	Enables and disables headers.	67
:HEADer?	*A$*	Queries header enablement.	
:FUNCtion *A$*	*A$* = MEM, REC, FFT	Changes the function.	67
:FUNCtion?	*A$*	Queries the function.	
:A4PRint		Same as the FEED key + COPY key on the main unit.	68
:MODE 7070 *A$*	*A$* = OFF, ON	Enables and disables 7070 mode.	68
:MODE 7070 ?	*A$*	Queries the 7070 mode enablement.	

5.2.2 Setting and Querying the Time Axis Range, Recording Length, etc. (CONFigure command)

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	69
:TDIV?	*A* <NR3> (unit seconds)	Queries the time axis range.		
:SHOT *A* *(B, C, D)*	MEM: *A* = recording length (unit DIV) REC: *A* = day, *B* = hour, *C* = minutes, *D* = seconds	Sets the recording length.	MEM REC	69
:SHOT?	MEM: *A* <NR1> (unit DIV) REC: *A, B, C, D* <NR1>	Queries the recording length.		
:FORMat *A$*	*A$* = SINGle, DUAL, QUAD, OCT, XYSingle, XYDual (MEM) SINGle, DUAL, QUAD, OCT (REC) SINGle, DUAL, NYQuist (FFT)	Sets the format.	All	70
:FORMat?	*A$*	Queries the format.		
:DOTLine *A$*	*A$* = DOT, LINE	Sets the interpolation function.	All	70
:DOTLine?	*A$*	Queries the interpolation function.		
:ROLL *A$*	*A$* = OFF, ON	Enables and disables roll mode.	MEM	70
:ROLL?	*A$*	Queries roll mode enablement.		
:SPIMpose *A$*	*A$* = OFF, ON	Enables and disables waveform superimposition.	MEM	71
:SPIMpose?	*A$*	Queries waveform superimposition enablement.		
:PRKInd *A$*	*A$* = WAVE, LOGGing	Specifies the printer output style.	All	71
:PRKInd?	*A$*	Queries the printer output style.		
:LOGGing *A*	*A* = 1 to 10000 (MEM, REC)	Specifies the logging output interval.	MEM REC	72
:LOGGing?	*A* <NR2>	Queries the logging output interval.		

MEM memory recorder function
REC recorder function
FFT FFT function
All MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CONFigure				
:PRINt *A$*	*A$* = OFF, PRINter, DAT	Sets printer output.	REC	72
:PRINt?	*A$*	Queries printer output.		
:MIC *A$*	*A$* = OFF, CH1 to CH15 (odd number channel)	Sets micro-phone channel.	REC	72
:MIC?	*ch$*	Queries micro-phone channel.		
:SMOOth *A$*	*A$* = OFF, ON	Enables and disables smooth printing.	MEM REC	73
:SMOOth?	*A$*	Queries smooth printing enablement.		
:ATPRint *A$*	*A$* = OFF, ON, TRIGger	Enables and disables auto print.	MEM FFT	73
:ATPRint?	*A$*	Queries auto print enablement.		
:ATSAve *A$*	*A$* = OFF, ON, TRIGger	Enables and disables auto save function.	MEM FFT	73
:ATSAve?	*A$*	Queries auto save function enablement.		
:ATFILe *"A$"*	*A$* = file name (16 characters):8845 (8 characters):8846	Sets the file name of auto save.	All (8845) REC (8846)	74
:ATFILe?	*A$*	Queries the file name of auto save.		
:ATTYpe *A$*	*A$* = BIN, TXT	Sets the save format.	MEM FFT (8846 only)	74
:ATFILe?	*A$*	Queries the save format.		
:ATHEad *A$*	*A$* = OFF, ON	Enables and disables header.	MEM FFT (8846 only)	74
:ATHEad?	*A$*	Queries the header enablement.		
:ATINt *A$*	*A$* = OFF X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, 1_200, X1_500, X1_1000	Set the intermittent.	MEM (8846 only)	75
:ATINt?	*A$*	Queries the intermittent.		
:MEMDiv *A$*	*A$* = OFF, SEQ, MULTI	Sets the memory segmentation function.	MEM	75
:MEMDiv?	*A$*	Queries the memory segmentation function.		

MEM memory recorder function
FFT FFT function
REC recorder function
All MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CONFigure				
:MAXBlock *A*	*A* = 3, 7, 15, 31, 63 (in multi-block function) *A* = 2 to 63 (in sequential save function)	Sets the memory block number (in multi-block function and multi-block function).	MEM	76
:MAXBlock?	*A* <NR1>	Queries the memory block number.		
: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	76
: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	77
:REFBlock?	*A* <NR1>	Queries the reference block.		
:WVCOmp *A$*	*A$* = OFF, OUT, ALLOut	Sets the waveform decision mode.	MEM FFT	77
:WVCOmp?	*A$*	Queries the waveform decision mode.		
:CMPStop *A$*	*A$* = GO, NG, G_N	Sets the waveform decision stop mode.	MEM FFT	77
:CMPStop?	*A$*	Queries the waveform decision stop mode.		
:AVERage *A*	*A* = 0, 2, 4, 8, 16, 32, 64, 128, 256 (0; OFF)	Sets the count for averaging.	MEM	78
:AVERage?	*A* <NR1>	Queries the current setting of the count for averaging.		
:FFTAVERage *A*	*A* = 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096	Sets the count for averaging in the FFT function.	FFT	78
:FFTAVERage?	*A* <NR1>	Queries the current setting of the count for averaging in the FFT function.		
:FFTAVKind *A$*	*A$* = OFF, T_EXP, F_EXP, T_LIN, F_LIN, F_PEAK	Sets the averaging method.	FFT	79
:FFTAVKind?	*A$*	Queries the currently set averaging method.		
:FFTMode *A*, *ch$*, (,*ch2$*)	*A$* = 1, 2 *ch1$*, *ch2$* = CH1 to CH16	Sets the FFT channel mode.	FFT	79
:FFTMode?	*A* <NR1>, *ch1$*, *ch2$*	Queries the current FFT channel mode.		

MEM memory recorder function
FFT FFT function
REC recorder function
All MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CONFigure				
:FFTWind *A$* (,*B*)	*A$* = RECTan, HANNing, EXPOnential *B* = 0 to 99 (%)	Sets the window function.	FFT	80
:FFTWind?	*A$, B* <NR1>	Queries the current window function.		
:FFTFunction *A$, B$*	*A$* = G1, G2 *B$* = STR, LIN, RMS, PSP, ACR, HIS, TRF, CSP, CCR, IMP, COH, OCT	Sets the FFT analysis mode.	FFT	80
:FFTFunction? *A$*	*A$, B$*	Queries the current FFT analysis mode.		
:FFTRef *A$*	*A$* = NEW, MEM	Designates the source for FFT analysis data.	FFT	81
:FFTRef?	*A$*	Queries the current FFT analysis data source.		
:FFTSCale *A$, B$*	*A$* = G1, G2 *B$* = AUTO, MANUaL	Sets the display scaling method for a graph.	FFT	82
:FFTSCale? *A$, B$*	*A$, B$*	Queries the current display scaling method for a graph.		
:FFTUp *A$, B*	*A$* = G1, G2 *B$* = -9.9999E+29 to +9.9999E+29	Sets the vertical axis upper limit for a graph.	FFT	82
:FFTUp? *A$*	*A$, B* <NR3>	Queries the current vertical axis upper limit for a graph.		
:FFTLow *A$, A*	*A$* = G1, G2 *B$* = -9.9999E+29 to +9.9999E+29	Sets the vertical axis lower limit for a graph.	FFT	83
:FFTLow? *A$*	*A$, B* <NR3>	Queries the current vertical axis lower limit for a graph.		
:FFTXaxis *A$, B$*	*A$* = G1, G2 *B$* = 1_1oct, 1_3oct (octave analysis) LINhz, LOGhz (otherwise)	Sets the x-axis.	FFT	83
:FFTXaxis? *A$*	*A$, B$*	Queries the current x-axis setting.		
:FFTYaxis *A$, B$*	*A$* = G1, G2 *B$* = LINMAg, LINREal, LINIMag, LOGMAg, PHASE	Sets the y-axis.	FFT	84
:FFTYaxis? *A$*	*A$, B$*	Queries the current y-axis setting.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CONFigure				
:FREQ *A*	*A* = 80000, 40000, 32000, 20000, 16000, 800, 4000, 800, 400, 200, 80, 40, 20, 8, 4, 2, 0.667, 0.333, 0.133	Sets the frequency range.	FFT	85
:FREQ ?	*A* <NR3>	Queries the currently set frequency range.		
:OCTFilter *A$*	*A$* = NORMal, SHARp	Sets the type of octave filter.	FFT	86
:OCTFilter ?	*A$*	Queries the currently set type of octave filter.		
:PEAK *A$*	*A$* = OFF, PEAK, MAX	Sets the peak value display.	FFT	86
:PEAK?	*A$*	Queries the currently peak value display.		

MEM memory recorder function
FFT FFT function
REC recorder function
All MEM, REC, and FFT

5.2.3 Setting and Querying Trigger Source, Level, etc. (TRIGger command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:TRIGger	(*ch$* = CH1 to CH16)			
:SOURce *A$*	*A$* = OR, AND	Sets trigger logical operator to AND or OR.	All	87
:SOURce?	*A$*	Queries trigger logical operator (AND or OR).		
:KIND *ch$, A$*	*A$* = OFF, LEVEl, IN, OUT, LOGIc	Sets type of trigger for the specified channel.	All	87
:KIND? *ch$*	*ch$, A$*	Queries type of trigger for the specified channel.		
:LEVEl *ch$, A*	*A* = trigger level (unit: V, ℃, με)	Sets the trigger level of the specified channel.	All	88
:LEVEl? *ch$*	*ch$, A* <NR3>	Queries the trigger level of the specified indicated channel.		
:SLOPe *ch$, A$*	*A$* = UP, DOWN	Sets the trigger direction (slope) of the specified channel.	All	88
:SLOPe? *ch$*	*ch$, A$*	Queries the trigger direction (slope) of the specified channel.		
:UPPEr *ch$, A*	*A* = upper limit level (unit: V, ℃, με)	Sets upper limit level of window trigger.	All	89
:UPPEr? *ch$*	*ch$, A* <NR3>	Queries upper limit level of window trigger.		
:LOWEr *ch$, A*	*A* = lower limit level (unit: V, ℃, με)	Sets lower limit level of window trigger.	All	89
:LOWEr? *ch$*	*ch$, A* <NR3>	Queries lower limit level of window trigger.		
:LOGPat *ch$,* "*A$*"	*A$* = XXXX trigger pattern (X, 0, 1)	Sets the trigger pattern for a logic trigger.	All	90
:LOGPat? *ch$*	*ch$,* "*A$*"	Queries the trigger pattern for a logic trigger.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:TRIGger	(*ch$* = CH1 to CH16)			
:LOGAnd *ch$, A$*	*ch$* = CHA to CHD *A$* = OR, AND	Sets AND/OR for the logic trigger pattern.	All	90
:LOGAnd? *ch$*	*ch$, A$*	Queries AND/OR for the logic trigger pattern.		
:FILTer *ch$, A*	*A* = 0 (OFF), 10, 20, 50, 100, 150, 200, 250, 500, 1000 samples	Enables and disables filter of level or logic trigger.	All	91
:FILTer? *ch$*	*ch$, A* <NR2>	Queries enablement of filter of level or logic trigger.		
:EXTErnal *A$*	*A$* = OFF, ON	Enables and disables external trigger.	All	91
:EXTErnal?	*A$*	Queries external trigger enablement.		
:MODE *A$*	*A$* = SINGle, REPEat (REC) SINGle, REPEat, AUTO, AUTOStop (MEM, FFT)	Sets trigger mode.	All	92
:MODE?	*A$*	Queries trigger mode.		
:PRETrig *A*	*A* = 0, 2, 5, 10, ... 90, 95, 100, and -95	Sets pre-trigger.	MEM FFT	92
:PRETrig?	*A* <NR1> (unit %)	Queries pre-trigger.		
:TIMEr *A$*	*A$* = OFF, ON	Sets timer trigger.	All	93
:TIMEr?	*A$*	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	93
: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	94
:TMSTOp?	Same as :TMSTArt?	Queries stop time of timer trigger.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:TRIGger	(*ch$* = CH1 to CH16)			
:TMINTvl *day, hour, min, sec*	*day* = 0 to 10 *hour* = 0 to 23 *min* = 0 to 59 *sec* = 0 to 59	Sets time interval for timer trigger.	All	94
:TMINTvl?	*day, hour, min, sec* all <NR1>	Queries time interval for timer trigger.		
:DETECTTime *hour, min, sec*	*hour* = 0 to 23 *min* = 0 to 59 *sec* = 0 to 59	Sets the time point for trigger detection.	All	95
:DETECTTime?	*hour, min, sec* all <NR1>	Queries the currently set time point for trigger detection.		
:DETECTDate *year, month, day*	*year* = 0 to 99 *month* = 1 to 12 *day* = 1 to 31	Sets the date for trigger detection.	All	95
:DETECTTime?	*hour, min, sec* all <NR1>	Queries the currently set date for trigger detection.		
:STOPTime *hour, min, sec*	*hour* = 0 to 23 *min* = 0 to 59 *sec* = 0 to 59	Sets the termination time of start operation.	All	96
:STOPTime?	*hour, min, sec* all <NR1>	Queries the termination time of start operation.		
:STOPDate *year, month, day*	*year* = 0 to 99 *month* = 1 to 12 *day* = 1 to 31	Sets the date of termination.	All	96
:STOPDate?	*year, month, day* all <NR1>	Queries the date of termination.		

MEM memory recorder function
FFT FFT function
REC recorder function
All MEM, REC, and FFT

5.2.4 Setting and Querying Input Channel (UNIT command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:UNIT	(*ch$* = CH1 to CH16)			
:RANGe *ch$, A*	*A* = voltage axis range (unit V) temperature range (unit ℃) strain range (unit με)	Sets input channel voltage axis range.	All	97
:RANGe? *ch$*	*ch$, A* <NR3>	Queries input channel voltage axis range.		
:POSItion *ch$, A*	*A* = Position value (unit DIV)	Sets the origin position for an input channel.	All	97
:POSItion? *ch$*	*ch$, A* <NR2>	Queries the origin position for an input channel.		
:COUPling *ch$, A$*	*A$* = GND, DC (8916, 8919, 8927 units) DC, GND, RMS, R_G (8917 unit)	Sets input channel coupling.	All	98
:COUPling? *ch$*	*ch$, A$*	Queries input channel coupling.		
:FILTer *ch$, A*	*A* = cutoff frequency (unit Hz)	Sets input channel filter.	All	98
:FILTer? *ch$*	*ch$, A* <NR3>	Queries input channel filter.		
:ADJUST		Carries out zero adjustment for the input units.	All	99
:SENSor *ch$, A$*	*A$* = K, J, T	Sets the type of a temperature input unit sensor.	All	99
:SENSor? *ch$*	*ch$, A$*	Queries the type of a temperature input unit sensor.		
:AAFilter *ch$, A*	*ch$* *A* = OFF, ON	Turns on or off the fft anti-aliasing filter.	All	99
:AAFilter? *ch$*	*ch$, Ah*	Queries the current on or off state of the FFT anti-aliasing filter.		
:BALAnce		Carries out auto-balancing for all strain amplifiers.	All	100
:CHBAlance	*ch$*	Carries out auto-balancing for strain amplifiers in each channel.	All	100
:LOGIc *A$*	*A$* = OFF, CH1 to CH15 (odd number channel)	Sets the logic waveform channel.	MEM REC	100
:LOGIc?	*A$*	Queries the logic waveform channel.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

5.2.5 Setting and Querying Changeover of the Screen Mode and Waveform Display (DISPlay command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DISPlay	(*ch$* = CH1 to CH16)			
:CHANge *A$*	*A$* = SYSTem, STATus, CHANnel, DISPlay	Changes over the display screen.	All	101
:CHANge?	*A$*	Queries the display screen.		
:DRAWing *ch$, A$*	*A$* = OFF, 16 colors	Sets display and recording intensity for waveform.	MEM REC	101
:DRAWing? *ch$*	*ch$, A$*	Queries display and recording of a waveform.		
:GRAPh *ch$, G$*	*A* = 1, 2, 3, 4, 5, 6, 7, 8 (for DUAL format, 1, 2) (for QUAD format, 1 to 4)	Sets waveform display graph in DUAL and QUAD format.	MEM REC	102
:GRAPh? *ch$*	*ch$, A* <NR1>	Queries waveform display graph in DUAL and QUAD format.		
:LOGDraw *ch$, A$*	*A$* = OFF, 16 colors *ch$* = CHA to CHD	Sets the display and recording of logic waveform.	MEM REC	102
:LOGDraw? *ch$*	*A$*	Queries the display and recording of logic waveform.		
:LOGPosi *ch$, A*	*A* = 1, 2, 3, 4, 5, 6, 7, 8 *ch$* = CHA to CHD	Sets the display position of logic waveform.	MEM REC	103
:LOGPosi? *ch$*	*A* <NR1>	Queries the display position of logic waveform.		

MEM memory recorder function
REC recorder function
FFT FFT function
All MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DISPlay	(*ch$* = CH1 to CH16)			
:XMAG *A$*	(MEM) *A$* = X10, X5, X2, X1, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, X1_200, X1_500, X1_1000 (REC) *A$* = X10, X5, X2, X1, X1_2, X1_3, X1_4, X1_5, X1_6, X1_8, X1_10, X1_12, X1_15, X1_16, X1_20, X1_24, X1_25, X1_30, X1_40, X1_50, X1_60, X1_80, X1_100, X1_120, X1_150, X1_160, X1_180, X1_200, X1_240, X1_250, X1_300, X1_360, X_400, X1_500, X1_600, X1_720, X1_800, X1_1000, X1_1200, X1_1500, X1_1600, X1_1800, X1_2000, X1_2400, X1_2500, X1_3000, X1_3600, X1_4000, X1_5000, X1_6000, X1_7200, X1_8000, X1_10000, X1_12000, X1_15000, X1_16000, X1_18000, X1_20000, X1_24000, X1_30000, X1_36000, X1_48000, X1_50000, X1_60000, X1_72000, X1_96000, X1_100000, X1_120000, X1_150000, X1_180000, X1_240000, X1_300000, X1_360000, X1_480000, X1_600000, X1_720000, X1_960000, X1_1440000, X1_1800000, X1_2880000	Sets the magnification/ compression factor on the time axis.	MEM REC	103
:XMAG?	*A$*	Queries the magnification/ compression factor on the time axis.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DISPlay	(*ch$* = CH1 to CH16)			
:YMAG *ch$, A$*	*A$* = X1_10, X1_5, X1_2, X1, X2, X5, X10, X20	Sets the magnification/ compression factor on the voltage axis.	MEM REC	104
:YMAG?	*ch$, A$*	Queries the magnification/ compression factor on the voltage axis.		
:XYDRawing *A, B$*	*A$* = 1 to 4 *B$* = OFF, 1 to 16	Sets the drawing level for an X-Y plot.	MEM	104
:XYDRawing? *ch$*	*A* <NR1>, *B$*	Queries the drawing level for an X-Y plot.		
:XAXIs *A, ch$*	*A* = 1 to 4	In X-Y format, sets the X-axis.	MEM	104
:XAXIs? *A*	*A* <NR1>, *ch$*	In X-Y format, queries the X-axis.		
:YAXIs *A, ch$*	*A* = 1 to 4	In X-Y format, sets the Y-axis.	MEM	105
:YAXIs? *A*	*A* <NR1>, *ch$*	In X-Y format, queries the Y-axis.		
:WAVE *A$*	*A$* = ACUR (A-cursor), TRIG (trigger point), POINT (the point set with :MEMory:POINt)	Executes waveform display.	MEM	105
:VARIable *ch$, A$*	*A$* = OFF, ON	Sets the variable function.	All	105
:VARIable? *ch$*	*ch$, A$*	Queries the variable function.		
:VARIUPLOw *ch$, B, C*	*B* = C = -9.9999E+29 to +9.9999E+29	Sets the upper and lower limit values of the variable.	All	106
:VARIUPLOw? *ch$*	*ch$, B* <NR3>, *C* <NR3>	Queries the upper and lower limit values of the variable.		
:TRSEarch *A, B*	*A* = start data number *B* = stop data number	Search trigger.	REC	106

MEM memory recorder function
FFT FFT function
REC recorder function
All MEM, REC, and FFT

5.2.6 Cursor Setting and Reading (CURSor command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CURSor	(*ch$* = CH1 to CH16)			
:MODE *A$*	*A$* = OFF, Xcur, Ycur, TRACe (XYSingle, XYQuad in MEM) OFF, TIME, VOLT, TRACe (excluding XYSingle and XYQuad in MEM, REC) OFF, TRACe (FFT)	Sets the A and B cursor type.	All	107
:MODE?	*A$*	Queries the A and B cursor type.		
:ABCUrsor *A$*	*A$* = A, B, A_B, OFF	Chooses the cursor among A, B, and A&B.	All	107
:ABCUrsor?	*A$*	Queries the cursor among A, B, and A&B.		
:ACHAnnel *ch$*	For XY format, *ch$* = X1 to X4	Sets the A cursor channel.	All	108
:ACHAnnel?	*ch$*	Queries the A cursor channel.		
:BCHAnnel *ch$*	For XY format, *ch$* = X1 to X4	Sets the B cursor channel.	All	108
:BCHAnnel?	*ch$*	Queries the B cursor channel.		
:APOSition *A*	(vertical cursor, trace cursor) *A* = 0 to amount of stored data (MEM, REC) 0 to 999 (FFT) [STR, ACR, CCR, IMP] 0 to 400 (FFT) [LIN, RMS, PSP, TRF, COH, CSP, HIS, OCT] (horizontal cursor) *A* = 0 to 639 (MEM, REC)	Sets the position of the A cursor.	All	109
:APOSition?	*A* <NR1>	Queries the position of the A cursor.		
:BPOSition *A*	Same as :APOSition	Sets the position of the B cursor.	All	109
:BPOSition?	*A* <NR1>	Queries the position of the B cursor.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CURSor	(*ch$* = CH1 to CH16)			
:DTREad?	*B$* *A$* = A, B, A_B *B$* = readout value (t)	Queries the cursor readout value (t).	MEM REC	110
:DVREad?	*B$* *A$* = A, B, A_B *B$* = readout value (V, ℃, $\mu\varepsilon$)	Queries the cursor readout value (V).	MEM REC	111
:ABCHAnnel *A$*	*A$* = G1, G2	Sets the graph for the A and B cursors.	FFT	111
:ABCHAnnel?	*A$*	Queries the graph setting for the A and B cursors.		
:DFREad?	"*B, C*" *A$* = A, B, A_B *B* = x-axis data *C* = y-axis data	Queries the current cursor readout position.	FFT	112

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

5.2.7 Setting and Querying Input and Output, etc., from the Memory (MEMory command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:MEMory	(*ch$* = CH1 to CH16, CHA to CHD)			
:POINt *ch$, A*	*A* = 0 to 2000000	Sets point in memory for input and output.	MEM	112
:POINt?	*A* <NR1> = 0 to 2000000	Queries point in memory for input and output.		
:MAXPoint?	*A* <NR1> = 0: not stored, 2500 to 2000000 (÷100 = number of divisions)	Queries the number of stored data.	MEM	113
:PREPare		Prepares the memory for receipt of waveform data.	MEM	113
:ADATa *B, C,...*	*B, C,...* = 0 to 16383	Input data to memory (ASCII).	MEM	114
:ADATa? *A*	*A* = 1 to 80 (number of output units) *B, C,...*<NR1> = 0 to 16383	Output data from memory (ASCII).		
:VDATa *B, C,...*	*B, C,...* = voltage values (units V, ℃, με)	Input data to memory (voltage values).	MEM	115
:VDATa? *A*	*A* = 1 to 35 (amount of data) *B, C,...*<NR3> = voltage value (units V, ℃, με)	Output stored data (voltage values).		
:AREAI? *ch$*	*A* <NR1> = 0 to 16383	Output the real time data (ASCII)	MEM	116
:VREAI? *ch$*	*A* <NR3> = voltage value (units V, ℃, με)	Output the real time data (voltage value)	MEM	116
:LDATa *A, B,...*	*A, B,...* = 0 to 15	Input logic data from memory.	MEM	116
:LDATa? *A*	*A* = 1 to 160 (amount of output data) Response data <NR1> = 0 to 15	Output logic data from memory.		
:BDATa? *A*	*A* = 1 to 250 (amount of output data) Response data, binary, integer data	Outputs binary data to memory.	MEM	118
:LBDAta? *A*	*A* = 1 to 500 (amount of output data) Response data, binary, integer data	Outputs logic binary data to memory.	MEM	119

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:MEMory	(*ch$* = CH1 to CH16)			
:FFTPOint *A$, B*	*A$* = G1, G2 *B* = 0 to 999 (STR, ACR, CCR, IMP) 0 to 400 (LIN, RMS, PSP, TRF, COH, CSP, HIS, OCT)	Sets the output point for FFT data.	FFT	120
:FFTPOint?	*A$, B* <NR3>	Queries the current output point for FFT data.		
:FFTData?	"*A, B*" *A* = X-axis data <NR3> *B* = Y-axis data <NR3>	Output FFT data.	FFT	120
:RECPOint *ch$, A*	*A* = 0 to 2000000	Sets point in memory for input and output.	REC	121
:RECPOint?	*A* <NR1> = 0 to 2000000	Queries point in memory for input and output.		
:RECMAXPoint?	*A* <NR1> = 0 (not stored), 2500 to 2000000 (÷100 = number of divisions)	Queries the memory for receipt of waveform data.	REC	121
:RECAData? *A*	*A* = 1 to 80 (number of output) *B, C,...* <NR1> = 0 to 16383	Outputs the stored data (ASCII).	REC	122
:RECVData? *A*	*A* = 1 to 35 (number of data) *B, C,...* <NR3> = voltage value (units V, ℃, με)	Outputs the stored data (voltage value).	REC	122
:RECLData? *A*	*A* = 1 to 160 (number of output) Response data <NR1> = 0 to 15	Outputs the stored data (logic).	REC	123
:RECBData? *A*	*A* = 1 to 250 (number of output) Response data binary, integral data	Outputs the stored data (binary).	REC	123
:RECLBdata? *A*	*A* = 1 to 500 (number of output) Response data binary, integral data	Outputs the stored data (logic binary).	REC	124
:RECTomem		Converts the recorder waveform to memory waveform data.	REC	124

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

5.2.8 Setting and Querying the System Screen (SYSTem command)

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	125
: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	125
:DATE?	*year, month, day* (all <NR1>)	Queries the calendar.		
:DATAClear		Clear data.	All	125
:USEUNit *A*	*A* = 1, 2, 4, 8	Sets the number of units used.	All	126
:USEUNit ?	*A* <NR1>	Queries the number of units used.		
:STARt *A$*	*A$* = OFF, ON	Enables and disables start key backup.	All	126
:STARt?	*A$*	Queries start key backup enablement.		
:GRID *A$*	*A$* = OFF, STANdard, FINE, D_STANdard, D_FINE, S_STANdard, S_FINE	Sets the grid type.	All	126
:GRID?	*A$*	Queries the grid type.		
:CHMArk *A$*	*A$* = OFF, ON	Enables and disables channel markers.	All	127
:CHMArk?	*A$*	Queries enablement of channel markers.		
:TMAXis *A$*	*A$* = TIME, TIME (60), DIV, DATE	Sets the the axis display.	All	127
:TMAXis?	*A$*	Queries the time axis display.		

MEM memory recorder function
REC recorder function
FFT FFT function
All MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:SYSTem				
:LIST *A$*	*A$* = OFF, LIST, GAUGE, L_G	Sets list and gauge functions.	All	127
:LIST?	*A$*	Queries list and gauge functions.		
:CRTOff *A$*	*A$* = ON, OFF	Enables and disables the screen saver.	All	128
:CRTOff?	*A$*	Queries enablement of the screen saver.		
:LCDDisp *A*	*A* = 1 to 32	Sets the LCD display.	All	128
:LCDDisp?	*A* <NR1>	Queries the LCD display.		
:VOLUme *A$*	*A$* = HIGH, MEDium, LOW	Sets the volume.	All	128
:VOLUme?	*A$*	Queries the volume.		
:INTPrint *A$*	*A$* = OFF, ON	Enables or disables the intermittent function.	All	129
:INTPrint ?	*A$*	Queries the intermittent function enablement.		
:CPYOut *A$*	*A$* = PRINter, MO_Mono, MO_256	Set the copy key output destination.	All (8846 only)	129
:CPYOut?	*A$*	Queries the copy key output destination.		
:LANGuage *A$*	*A$* = JAPAnese, ENGLish	Set the language display.	All (8846 only)	129
:LANGuage?	*A$*	Queries the language display.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

5.2.9 Setting and Querying Scaling (SCALing command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:SCALing				
:KIND *ch$, A$*	*A$* = RATIO, POINT	Sets the type of scaling.	All	130
:KIND?	*A$*	Queries the type of scaling.		
:SET *ch$, A$*	*ch$* = CH1 to CH16 *A$* = OFF, SCI, ENG	Enables and disables scaling.	All	130
:SET? *ch$*	*ch$, A$*	Queries scaling enablement.		
:VOLT *ch$, B*	*ch$* = CH1 to CH16 *A* = -9.9999E+9 to +9.9999E+9	Sets the scaling conversion value.	All	130
:VOLT? *ch$*	*ch$, B* <NR3>	Queries the scaling conversion value.		
:OFFSet *ch$, B*	*ch$* = CH1 to CH16 *B* = -9.9999E+9 to +9.9999E+9	Sets scaling offset.	All	131
:OFFSet? *ch$*	*ch$, B* <NR3>	Queries scaling offset.		
:UNIT *ch$, "B$"*	*ch$* = CH1 to CH16 *B$* = scaling unit (7 characters)	Sets scaling unit.	All	131
:UNIT? *ch$*	*ch$, "B$"*	Queries scaling unit.		
:VOUPLOw *ch$, B, C*	*ch$* = CH1 to CH16 *B* = *C* = -9.9999E+29 to +9.9999E+29	Sets the scaling VOLT UP, VOLT LOw.	All	132
:VOUPLOw? *ch$*	*ch$, B* <NR3>, *C* <NR3>	Queries VOLT UP, VOLT LOw.		
:SCUPLOw *ch$, B, C*	*ch$* = CH1 to CH16 *B* = *C* = -9.9999E+29 to +9.9999E+29	Sets the scaling SC UP, SC LOw.	All	132
:SCUPLOw? *ch$*	*ch$, B* <NR3>, *C* <NR3>	Queries the scaling SC UP, SC LOw.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

5.2.10 Setting and Querying Comments (COMMent command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:COMMent				
:TITLe *A$*, "*B$*"	*A$* = OFF, SETTing, COMMent, S_C *B$* = comment string (up to 20 characters)	Sets a title comment.	All	133
:TITLe?	"*B$*"	Queries a title comment.		
:EACHch *A$*	*A$* = OFF, SETTing, COMMENT, S_C	Enables or disables comments for all channels.	All	133
:EACHch? *ch$*	*A$*	Queries the comments for all channels enablement.		
:CH *ch$*, "*A$*"	*ch$* = CH1 to CH16, CHA to CHD *A$* = comment string (up to 20 characters)	Sets a comment for a particular channel.	All	134
:CH? *ch$*	*ch$*, "*A$*"	Queries comment for a particular channel.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

5.2.11 Calculation Setting and Querying (CALCulate command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate				
:WVCALc *A$*	*A$* = OFF, ON, EXEC (execute)	Enables and disables waveform processing calculation.	MEM	135
:WVCALc?	*A$*	Queries enablement of waveform processing calculation.		
:Z1 "*A$*"	*A$* = calculation equation (80 characters) alphabets: small characters	Sets the coefficients for the waveform processing calculation equation for Z1.	MEM	135
:Z1?	*A$* (80 characters)	Queries the coefficients for the waveform processing calculation equation for Z1.		
:Z2 "*A$*"	Same as Z1.	Sets the coefficients for the waveform processing calculation equation for Z2.	MEM	135
:Z2?	*A$* (80 characters)	Queries the coefficients for the waveform processing calculation equation for Z2.		
:Z3 "*A$*"	Same as Z1.	Sets the coefficients for the waveform processing calculation equation for Z3.	MEM	135
:Z3?	*A$* (80 characters)	Queries the coefficients for the waveform processing calculation equation for Z3.		
:Z4 "*A$*"	Same as Z1.	Sets the coefficients for the waveform processing calculation equation for Z4.	MEM	135
:Z4?	*A$* (80 characters)	Queries the coefficients for the waveform processing calculation equation for Z4.		
:Z5 "*A$*"	Same as Z1.	Sets the coefficients for the waveform processing calculation equation for Z5.	MEM	135
:Z5?	*A$* (80 characters)	Queries the coefficients for the waveform processing calculation equation for Z5.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	ALL	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate	(*ch$* = CH1 to CH16)			
:Z6 "*A$*"	Same as Z1.	Sets the coefficients for the waveform processing calculation equation for Z6.	MEM	135
:Z6?	*A$* (80 characters)	Queries the coefficients for the waveform processing calculation equation for Z6.		
:Z7 "*A$*"	Same as Z1.	Sets the coefficients for the waveform processing calculation equation for Z7.	MEM	135
:Z7?	*A$* (80 characters)	Queries the coefficients for the waveform processing calculation equation for Z7.		
:Z8 "*A$*"	Same as Z1.	Sets the coefficients for the waveform processing calculation equation for Z8.	MEM	135
:Z8?	*A$* (80 characters)	Queries the coefficients for the waveform processing calculation equation for Z8.		
:FACTor *A$, B*	*A$* = A to P *B* = -9.9999E+29 to +9.9999E+29	Sets the value of calculation equation coefficient a to p.	MEM	136
:FACTor? *A$*	*A$, B* <NR3>	Queries the value of calculation equation coefficient a to p.		
:MOVE *Z$, B*	*Z$* = Z1 to Z8 *B* = 0 to 4000 <NR1>	Sets the moving average.	MEM	136
:MOVE? *Z$*	*A$, B*	Queries the moving average.		
:SLIDe *Z$, B*	*Z$* = Z1 to Z8 *B* = -4000 to 4000	Sets the parallel movement.	MEM	137
:SLIDe? *Z$*	*Z$, B*	Queries the parallel movement.		
:Z1DIsplay *ch$, A$*	*ch$* = NONE, CH1 to CH16 *A$* = MANUal, AUTO	Sets the channel for receipt of the calculated result of the waveform processing calculation equation for Z1.	MEM	137
:Z1DIsplay?	*ch$, A$*	Queries the channel for receipt of the calculated result of the waveform processing calculation equation for Z1.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate	(*ch$* = CH1 to CH16)			
:Z2DIsplay *ch$, A$*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform processing calculation equation for Z2.	MEM	137
:Z2DIsplay?	*ch$, A$*	Queries the channel for receipt of the calculated result of the waveform processing calculation equation for Z2.		
:Z3DIsplay *ch$, A$*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform processing calculation equation for Z3.	MEM	137
:Z3DIsplay?	*ch$, A$*	Queries the channel for receipt of the calculated result of the waveform processing calculation equation for Z3.		
:Z4DIsplay *ch$, A$*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z4.	MEM	137
:Z4DIsplay?	*ch$, A$*	Queries the channel for receipt of the calculated result of the waveform processing calculation equation for Z4.		
:Z5DIsplay *ch$, A$*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z5.	MEM	137
:Z5DIsplay?	*ch$, A$*	Queries the channel for receipt of the calculated result of the waveform processing calculation equation for Z5.		
:Z6DIsplay *ch$, A$*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z6.	MEM	137
:Z6DIsplay?	*ch$, A$*	Queries the channel for receipt of the calculated result of the waveform processing calculation equation for Z6.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate	(*ch$* = CH1 to CH16)			
:Z7DIsplay *ch$, A$, upper, lower*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z7.	MEM	137
:Z7DIsplay?	*ch$, A$*	Queries the channel for receipt of the calculated result of the waveform processing calculation equation for Z7.		
:Z8DIsplay *ch$, A$, upper, lower*	Same as Z1DIsplay	Sets the channel for receipt of the calculated result of the waveform treatment calculation equation for Z8.	MEM	137
:Z8DIsplay?	*ch$, A$*	Queries the channel for receipt of the calculated result of the waveform processing calculation equation for Z8.		
:MEASure *A$*	*A$* = ON, OFF, EXEC (execute)	Enables and disables waveform parameter calculation.	MEM	138
:MEASure?	*A$*	Queries enablement of waveform parameter calculation.		
:MEASPrint *A$*	*A$* = OFF, ON	Enables and disables output of waveform parameter calculation values.	MEM	138
:MEASPrint?	*A$*	Queries the output enablement of waveform parameter calculation values.		
:MEASSet *NO$, A$, ch$*	*NO$* = NO1 to NO4 *A$* = OFF, MAX, MIN, MINT, PP, AVE, RMS, PERI, FREQ, RISE, FALL, STD, AREA, XYAREA *ch$* = CH1 to CH16, ALL	Sets waveform parameter calculation.	MEM	139
:MEASSet? *NO$*	*NO$, A$, ch$*	Queries waveform parameter calculation.		
:ANSWer? *NO$, ch$*	*NO$* = NO1 to NO4 *ch$* = CH1 to CH16 *A$* = OFF, MIN, MAX, MINT, MAXT, PP, AVE, RMS, PERI, FREQ, RISE, FALL, STD, AREA, XYAREA *B* <NR3> = calculation results	Queries a waveform parameter calculation result. Response a waveform parameter calculation result.	MEM	140

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:CALCulate	(*ch$* = CH1 to CH16) (*NO$* = NO1 to NO4)			
:COMP *NO$, A$* ↓	*NO$* = NO1 to NO4 *A$* = OFF, OUT, IN	Enables or disables waveform parameter decision calculations.	MEM	140
:COMP? *NO$*	*NO$, A$*	Queries enablement of waveform parameter decision calculations.		
:COMPArea *NO$, upper, lower*	*NO$* = NO1 to NO4 *upper, lower* = -9.9999E+29 to +9.9999E+29	Sets upper limit and lower limit values for waveform parameter calculation decision.	MEM	141
:COMPArea? *NO$*	*NO$, upper* <NR3>, *lower* <NR3>	Queries upper limit and lower limit values for waveform parameter calculation decision.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

5.2.12 Setting and Querying Relating to DAT (DAT command) (8845 only)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DAT				
:MODE *A$*	*A$* = OFF, ON	Enables or disables the DAT mode.	All	141
:MODE?	*A$*	Queries enablement of the DAT mode.		
:INFOr? *NO*	*NO* = file number (response) *NO*, "*NAME$*", *B$*, "*DATE$*", "*TIME$*" *NAME$* = file name *NO* = file number (if no file exists, then -1) *B$* = type of data saved W: measurement data F: conditions of creation A: waveform decision area N: no such file *DATE$* = year/month/day of save *TIME$* = hour:min:sec of save	Queries information about the file on a tape.	All	142
:LOAD *NO* (,"*S_TIME$*")	*NO* = file number *S_TIME$* = start time start time: day-hour:min:sec day: 000 - 999 hour: 00 - 23 min, sec: 00 - 59	Load from a tape.	All	142
:SAVE "*NAME$*", *A$*, (,*B$*)	(When MEM, FFT) *NAME$* = file name (up to 16 characters) *A$* = type of data to save W: measurement data F: unit settings A: waveform decision area *B$* = saving method none: usual saving ALL: all blocks saving during memory segmentation (When REC) *A$* = F	Saves on a tape.	All	143
:EJECt		Ejects a tape.	All	143
:READy		Prepares recording.	All	143
:ALLDelete		Deletes all file data.	All	144
:DELEte *NO*	*NO* = file number	Deletes the file specified and following files.	All	144

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:DAT				
:FORMat?		Formats a tape.	All	144
:RENAme *NO*, "*NAME$*"	*NO* = file number *NAME$* = file name (up to 16 characters)	Renames a file.	All	144
:FILE?	*A* <NR1> = number of files	Queries how many files are saved.	All	145
:PLAY *NO*, "*S_TIME$*", "*E_TIME$*", *MODE$*, *SPE$*	*NO* = file number *S_TIME$*=start time (day-hour:min:sec) *E_TIME$*=stop time (day-hour:min:sec) day = 000 to 999 hour = 00 to 23 min, sec = 00 to 59 *MODE$* = replay mode: SINGLe, REPEat *SPE$* = speaker output channel: OFF, CH1 to CH16	Replays from the tape drive (only recorder waveform)	All	145
:DATTomem *A$*, *NO*, "*S_TIME$*" "*E_TIME$*"	*A$* = OFF, ON When on: *NO* = file number *S_TIME$*=start time (day-hour:min:sec) *E_TIME$*=stop time (day-hour:min:sec) day = 000 to 999 hour = 00 to 23 min, sec = 00 to 59	Sets DATtoMEM.	All	146
:DATTomem?	*A$*	Queries DATtoMEM setting.		
:DATTOFft *A$*, *NO*, "*S_TIME$*" "*E_TIME$*"	*A$* = OFF, ON When on: *NO* = file number *S_TIME$*=start time (day-hour:min:sec) *E_TIME$*=stop time (day-hour:min:sec) day = 000 to 999 hour = 00 to 23 min, sec = 00 to 59	Sets DATtoFFT.	All	146
:DATTOFft ?	*A$*	Queries DATtoFFT setting.		

MEM	memory recorder function	REC	recorder function
FFT	FFT function	All	MEM, REC, and FFT

5.2.13 Setting and Querying Relating to MO (MO command) (8846 only)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:MO				
:MODE *A$*	*A$* = OFF, ON	Enables or disables the MO mode.	All	147
:MODE?	*A$*	Queries enablement of the MO mode.		
:INFOr? *"NAME$"*	*NAME$* = file name (response) *"NAME$"*, *A$*, *"DATE$"*, *"TIME$"* *NAME$* = file name *A$* = type of data saved W: measurement data F: conditions of creation A: waveform decision area N: no such file *DATE$* = year/month/day of save *TIME$* = hour:min:sec of save	Queries information about the file on a MO.	All	147
:LOAD *"NAME$"* (,*"S_TIME$"*)	*NAME$* = file name *S_TIME$* = start time start time: day-hour:min:sec day: 000 - 999 hour: 00 - 23 min, sec: 00 - 59	Load from a MO.	All	148
:SAVE *"NAME$"*, *A$*, (,*B$*)	(When MEM, FFT) *NAME$* = file name (up to 8 characters) *A$* = type of data to save W: measurement data F: unit settings A: waveform decision area *B$* = saving method none: usual saving ALL: all blocks saving during memory segmentation (When REC) *A$* = F	Saves on a MO.	All	148
:MKDIr *"NAME$"*	*NAME* = directory name	Directory name	All	149
:EJECt		Ejects a MO.	All	149
:DELEte *"NAME$"*	*NAME* = file name	Deletes specified file name.	All	149
:RMDIr *"NAME$"*	*NAME* = directory name	Deletes specified directory name.	All	149

MEM memory recorder function
REC recorder function
FFT FFT function
All MEM, REC, and FFT

Command	Data (for a query, response data)	Explanation	Function	Ref page
:MO				
:FORMat (*A$*)	*A$* = P P: physical format (not set): normalformat	Formats a MO.	All	150
:RENAme "*OLD$*", "*NEW$*"	*OLD$* = file name before changing *NEW$* = file name after changing	Renames a file.	All	150
:FILE?	*A* <NR1> = number of files	Queries how many files are saved.	All	150
:DIR?	*A* <NR1> = number of directories	Queries how many directories are saved.	All	151
:CHDIr "*NAME$*"	*NAME* = path name	Moves directory.	All	151
:PLAY *"NAME$",* *"S_TIME$"* *"E_TIME$",* *MODE$, SPE$*	*NAME$* = file name *S_TIME$*=start time (day-hour:min:sec) *E_TIME$*=stop time (day-hour:min:sec) day = 000 to 999 hour = 00 to 23 min, sec = 00 to 59 *MODE$* = replay mode: SINGLe, REPEat *SPE$* = speaker output channel: OFF, CH1 to CH16	Replays from the MO (only recorder waveform).	All	151
:FILETomem *A$, "NAME$",* *"S_TIME$"* *"E_TIME$"*	*A$* = OFF, ON When on: *NAME$* = file name *S_TIME$*=start time (day-hour:min:sec) *E_TIME$*=stop time (day-hour:min:sec) day = 000 to 999 hour = 00 to 23 min, sec = 00 to 59	Sets FILEtoMEM.	All	152
:FILETomem?	*A$*	Queries FILEtoMEM setting.		
:FILETOFft *A$, "NAME$",* *"S_TIME$"* *"E_TIME$"*	*A$* = OFF, ON When on: *NAME$* = file name *S_TIME$*=start time (day-hour:min:sec) *E_TIME$*=stop time (day-hour:min:sec) day = 000 to 999 hour = 00 to 23 min, sec = 00 to 59	Sets FILEtoFFT.	All	152
:FILETomem?	*A$*	Queries FILEtoFFT setting.		

MEM memory recorder function REC recorder function
FFT FFT function All MEM, REC, and FFT

5.2.14 Commands Relating to the Graphics Editor (GRAPh command)

Command	Data (for a query, response data)	Explanation	Function	Ref page
:GRAPh				
:EDIT *A$*	*A$* = OFF, ON	Enables and disables the editor.	MEM FFT	153
:EDIT?	*A$*	Queries editor enablement.		
:LINE *X1,Y1, X2, Y2*	*X1, X2* = x-coordinates *Y1, Y2* = y-coordinates	Erases from (X1, Y1) to (X2, Y2).	MEM FFT	153
:PARAllel *high, low, right, left*	*high* = 0 to 20.00 (div) *low* = 0 to 20.00 (div) *right* = 0 to 20.00 (div) *left* = 0 to 20.00 (div)	Carries out a parallel movement of the drawing.	MEM FFT	154
:PAINT *X, Y*	*X* = x-coordinate, *Y* = y-coordinate	Begins solid fill from the point specified by (X, Y).	MEM FFT	154
:ERASe *X1, Y1, X2, Y2*	*X1, X2* = x-coordinates *Y1, Y2* = y-coordinates	Erases from (X1, Y1) to (X2, Y2).	MEM FFT	154
:STORage		Loads a waveform into the editor.	MEM FFT	154
:REVErse		Reverses the drawing.	MEM FFT	155
:ALLClear		Clears the entire drawing.	MEM FFT	155
:CLEAr *X1, Y1, X2, Y2*	*X1, X2* = x-coordinates *Y1, Y2* = y-coordinates	Clears the rectangle with the points (X1, Y1) and (X2, Y2) at diagonally opposite corners.	MEM FFT	155
:UNDO		Reverses the effect of the immediately previous editor command.	MEM FFT	155
:SAVE		Saves the decision area created with the editor.	MEM FFT	156
:POINt *X, Y, A*	*X* = x-coordinates, *Y* = y-coordinates *A* = 0, 1	Sets waveform decision area data.	MEM FFT	156
:POINt? *X, Y*	*X, Y, A*	Queries waveform decision area data.		

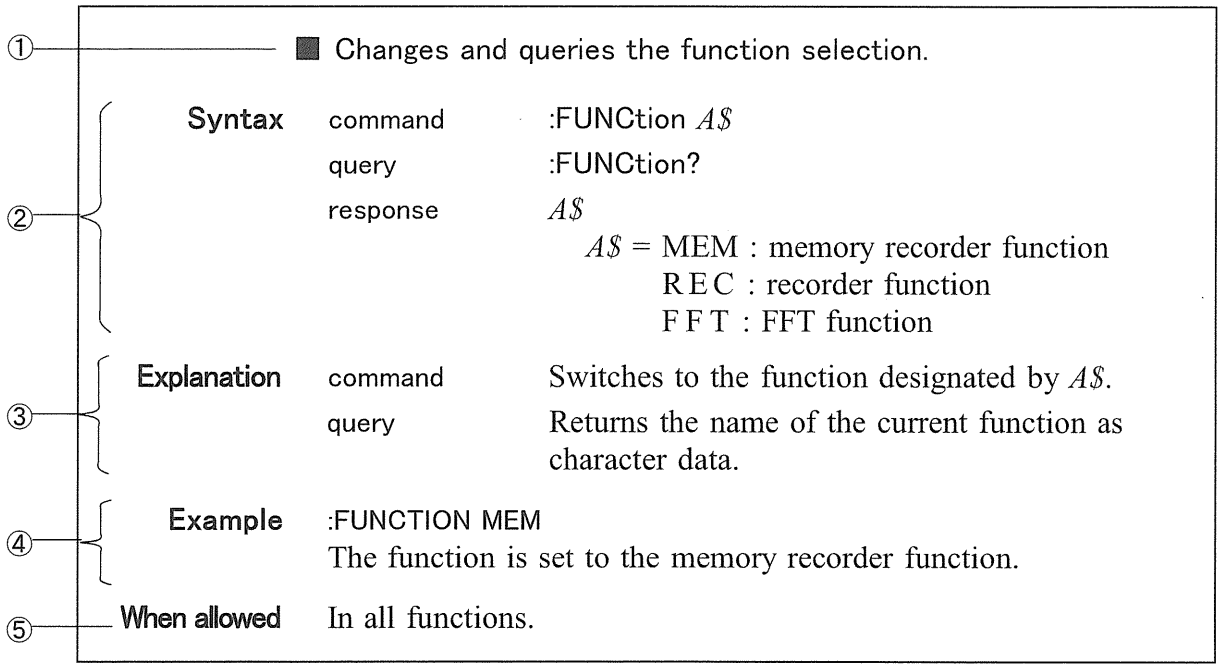
MEM memory recorder function
FFT FFT function
REC recorder function
All MEM, REC, and FFT

Chapter 6
Command Reference

6.1 Command Reference Explanation

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

Example



① Command function

② Command syntax (See next page.)

③ Explanation of the command function.

④ Example of command use.

⑤ Functions in which the command may be used (See next page.)

Command syntax

command	the syntax of a command program message
query	the syntax of a query program message
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 4.3. 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

Format	Meaning	Example
NR1	integer data	+15, −20, 25
NR2	fixed point numbers	+1.23, −4.56, 7.89
NR3	floating point numbers	+1.0E−3, −2.3E+3
NRf	includes all these three formats.	

When the 8845 or 8846 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.

Functions in which the command may be used.

MEM	memory recorder function
REC	recorder function
FFT	FFT function
All	Any of the MEM, REC 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 8845 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.

6.2 Standard Commands Stipulated by IEEE 488.2

6.2.1 System Data Commands and Queries

*IDN? command

■ Queries device ID.

Syntax

query *IDN?

response HIOKI, 8845, 0, V1. 00

First field	Manufacturer's name
Second field	Model name
Third field	Serial number (not used: 0)
Fourth field	Software version

*OPT? command

■ Queries device option provision.

Syntax

query *OPT?

response Whether or not channel 1 or 2 input unit present; whether or not channel 3 or 4 input unit present; whether or not channel 15, 16 input unit present; memory capacity.

0: not present
1: analog input unit (8916 ANALOG UNIT) present
2: temperature input unit (8918 TEMPERATUE UNIT) present
3: DC/RMS input unit (8917 DC/RMS UNIT) present
4: FFT analog unit (8919 FFT ANALOG UNIT) present
5: Analog unit (8927 ANALOG UNIT) present
6: Strain unit (8928 STRAIN UNIT) present

6.2.2 Internal Operation Commands and Queries

*RST command

■ Device initial setting.

Syntax command *RST

Explanation Initializes the 8845 (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 8845 is returned as an NR1 numerical value.

6.2.3 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$* 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 (*B$*) following *WAI is not executed until the execution of the commands *A$* is completed.

6.2.4 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 8845 and 8846)

■ 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 8845 and 8846)

■ 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 8845 and 8846)

■ 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.

6.3 Commands Specific to the 8845 and 8846

6.3.1 Execution Control Commands (common to all functions)

■ Performs starting.

Syntax command :STARt

Explanation Same as the START key of the 8845 and 8846.
Starts waveform sampling operation.

When allowed In all functions

■ Performs stopping.

Syntax command :STOP

Explanation Same as the STOP key of the 8845 and 8846.
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 8845 and 8846. 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 8845 and 8846.

When allowed In all functions

■ Screen dump function.

Syntax command :HCOPy

Explanation Same as the COPY key of the 8845 and 8846. 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 8845 and 8846. Sets the time axis and the voltage axis automatically.

When allowed In the memory recorder function.

■ Enables and disables LCD back light, and queries back light enablement.

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

Explanation
command Sets the LCD back light key to on or off.
query Returns the setting of the current back light key as character data.

When allowed In the memory recorder function.

■ Queries the 8845 or 8846 error number.

Syntax

query	:ERRor?
response	*A* <NR1>
	A = error number

Explanation The number of error or warning that has occurred on the 8845 or 8846 is returned in <NR1> as a numerical value. (See 8845 or 8846 Instruction Manual, Chapter 24 "Error and Warning Messages".)
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
	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

■ A4 print

Syntax command :A4PRint

Explanation Sets as the FEED key + COPY key on the 8845 and 8846 unit.

When allowed In the memory recorder function and recorder function.

■ Sets and queries the 7070 mode.

Syntax
command :MODE7070 *A$*
query :MODE7070?
response *A$*
A$ = OFF, ON

Explanation
command: Enables and disables the 7070 mode.
When *A$* is set to ON, the data from the 8845 or 8846 can be transferred to the 7070.
query: Returns the current 7070 mode settings.

Example :MODE7070 ON
Sets the 7070 mode to ON.

When allowed In all functions

7070 mode

By connecting the HIOKI 7070 WAVEFORM GENERATOR and the 8846 can be transferred to the 7070.

Method
1. Connect the 7070 and the 8845 or 8846 by using the GP-IB interface.
2. On the interface setting of the system screen for the 8845, set the 7070 mode to ON.
3. Set the destination to load of the 7070 data to 8840.

The HIOKI 7070 WAVEFORM GENERATOR can be purchased in Japan only.

6.3.2 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.) (when the external sampling, set to OUT)

Example

:CONFIGURE:TDIV +1.0E-3

Sets the time per division to 1 ms.

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, set day, hour, minutes, second as numerical value.)

Example

:CONFIGURE:SHOT 25

Sets the recording length to 25 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, OCT, XYSingle, XYDual : MEM
	SINGle, DUAL, QUAD, OCT : REC
	SINGle, DUAL, NYQuist : 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 all functions

■ 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 all functions

■ 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 waveform superimposition function.

Syntax

command	:CONFigure:SPIMpose *A$*
query	:CONFigure:SPIMpose?
response	*A$* *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:SPIMPOSE ON

Sets the screen waveform superimposition to ON.

When allowed

In the memory recorder function.

■ Sets and queries the printer output style.

Syntax

command	:CONFigure:PRKInd *A$*
query	:CONFigure:PRKInd?
response	*A$* *A$* = WAVE (waveform) LOGGing (numerical)

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:PRKIND WAVE

Sets the printer output style to be waveform.

When allowed

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

■ Sets and queries the logging output interval.

Syntax

command	:CONFigure:LOGGing *A*
query	:CONFigure:LOGGing?
response	*A* <NR2>
	A = 1 to 10000 (memory recorder function and recorder function)

Explanation

command	Sets the numerical value of the logging output interval.
query	Returns the current setting of the logging output interval. For the memory recorder function and recorder function, the values are 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000 (samples)

Example :CONFIGURE:LOGGING 10

Sets the logging output interval to 10 samples.

When allowed In the memory recorder function and recorder 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 current setting of the printer output as character data.

Example :CONFIGURE:PRINT ON

Sets the printer output to ON.

When allowed In the recorder function.

■ Sets and queries the microphone channel.

Syntax

command	:CONFigure:MIC *A$*
query	:CONFigure:MIC?
response	*A$*
	A$ = OFF, CH1 to CH15 (odd number channels)

Explanation

command	Sets the microphone channel to record the voice.
query	Returns the current setting of the microphone channel as character data.

Example :CONFIGURE:MIC CH3

Sets the microphone channel to ON.

When allowed In the recorder 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 and recorder function.

■ Sets and queries the auto print function.

Syntax

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

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, ON, TRIGger

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 ON

Sets the auto save function to ON.

When allowed

In the memory recorder function and the FFT function.

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

Syntax

command	:CONFigure:ATFIle *A$*
query	:CONFigure:ATFIle?
response	*A$*
	A$ = file name (up to 16 characters) (8845)
	(up to 8 characters) (8846)

Explanation

command	Sets the file name of the auto save function.
query	Returns the current setting of the file name of the auto save function as character data.
	Single quotation marks (') can be used instead of double quotation marks (").

Example :CONFIGURE:ATFILE '8845 DATA'

Sets the file name of auto save function to "8845 DATA".

When allowed In all functions (8845), recorder function (8846)

■ Sets and queries the auto save format.

Syntax

command	:CONFigure:ATTYpe *A$*
query	:CONFigure:ATTYpe?
response	*A$*
	A$ = BIN: binary
	TXT: text

Explanation

command	Sets the auto save format.
query	Returns the current setting of the auto save format.

Example :CONFIGURE:ATTYPE BIN

Sets the auto save format to binary.

When allowed In the memory recorder function and the FFT function (8846 only).

■ Sets and queries the header of the auto save function.

Syntax

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

Explanation

command	Sets the header of auto save function on and off.
query	Returns the current setting of the header of auto save function.

Example :CONFIGURE:ATTYPE ON

Sets the header of auto save function to the ON.

When allowed In the memory recorder function and the FFT function (8846 only).

■ Sets and queries the intermittent of the auto save function.

Syntax

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

A$ = OFF, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, 1_200, X1_500, X1_1000

Explanation

command	Sets the intermittent of auto save function on and off.
query	Returns the current setting of the intermittent of auto save function.

Example

:CONFIGURE:ATINT X1_100

Sets the intermittent of auto save function to X1/100.

When allowed

In the memory recorder function (8846 only).

■ 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 63 (when multi-block, 3, 7, 15, 31, 63)

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.

■ 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 memory block used ("using block").
query	Returns the currently used memory 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.

■ 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 memory block.
query	Returns the current reference memory 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.

■ 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 the count for averaging.

Syntax

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

Explanation

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

Example :CONFIGURE:AVERAGE 32

Sets the count for averaging to 32.

When allowed In the memory recorder function.

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

Syntax

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

Explanation

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

Example :CONFIGURE:FFTAVERAGE 2048

Sets the count for averaging to 2048.

When allowed In the FFT function.

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

Syntax

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

Explanation

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

Example

:CONFIGURE:FFTAVKIND T_EXP
Sets time axis exponential averaging.

When allowed

In the FFT function.

■ Sets and queries the FFT channel mode.

Syntax

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

Explanation

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

Example

:CONFIGURE:FFTMODE 2, CH3, CH5
The channel mode is set to the two-channel FFT mode, and the object channels for FFT mode are set to be channel 3 and channel 5.

When allowed

In the FFT function.

■ Sets and queries the FFT window function.

Syntax

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

Explanation

command: Sets the window function as indicated by *A$*. If the exponential window function is designated by *A$*, its exponential function coefficient can be set by using *B*.

query: Returns the current window function as character data, and the current exponential function coefficient as a numerical value in NR1 format. If the window function is currently rectangular window or Hanning window, *B* is returned as a dummy zero.

Example

:CONFIGURE:FFTWIND HANN
The window function is set to Hanning window.

When allowed

In the FFT function.

■ Sets and queries the FFT analysis mode.

Syntax

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

Explanation

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

query — Returns the current FFT analysis mode as character data.

Example

```
:CONFIGURE:FORMAT DUAL
:CONFIGURE:FFTMODE 2, CH1, CH3
:CONFIGURE:FFTFUNCTION G1, IMP
:CONFIGURE:FFTFUNCTION G2, TRF
```

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

When allowed

In the FFT function.

■ Sets and queries the FFT data source.

Syntax

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

Explanation

command — Designates the source for FFT data as specified by *A$*
query — Returns the current FFT data source 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 display scaling method.

Syntax

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

Explanation

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

Example :CONFIGURE:FFTSCALE G1,AUTO

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

When allowed In the FFT function.

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

Syntax

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

Explanation

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

Example :CONFIGURE:FFTUP G2,100

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

When allowed In the FFT function.

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

Syntax

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

Explanation

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

Example :CONFIGURE:FFTLOW G2,100

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

When allowed In the FFT function.

■ Sets and queries the FFT x-axis.

Syntax

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

Explanation

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

Example :CONFIGURE:FFTXAXIS G1, LINHZ

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

When allowed In the FFT function.

■ Sets and queries the FFT y-axis.

Syntax

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

Explanation

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

Example

:CONFIGURE:FFTYAXIS G1,LINMAG
The setting for the y-axis of graph 1 is set to LINMAG.

When allowed

In the FFT function.

Display settings available on the x-axis

Analysis mode	X-axis				
	Linear-Hz	Log-Hz	1/10 octave	1/3 octave	Fixed scale
STR					TIME
LIN	●	●			
RMS	●	●			
PSP	●	●			
ACR					TIME
HIS					VOLT
TRF	●	●			
CSP	●	●			
CCR					TIME
IMP					TIME
COH	●	●			
OCT			●	●	

Display settings available on the y-axis

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

■ Sets and queries the FFT frequency range

Syntax

command :CONFigure:FREQ *A*

query :CONFigure:FREQ?

response *A* <NR3>

A = 80000, 40000, 32000, 20000, 16000, 8000, 4000, 2000, 800, 400, 200, 80, 40, 20, 8, 4, 2, 0.667, 0.333, 0.133 (units:Hz)

Explanation

command Sets the frequency range. If an attempt is made to set an unacceptable value, i.e. a value which is not one of the above, then the frequency range is set to the next higher one of the above values.

query Returns the currently set frequency range as a numerical value in NR3 format.

Example

:CONFIGURE:FREQ 80

The frequency range is set to 80 Hz.

When allowed

In the FFT function.

■ Sets and queries octave filter type.

Syntax

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

Explanation

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

Example :CONFIGURE:OCTFILGER NORMAL

Sets the octave filter type to NORMAL.

When allowed In the FFT function.

■ Sets and queries peak value display

Syntax

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

Explanation

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

Example :CONFIGURE:PEAK PEAK

Sets the peak value display to PEAK.

When allowed In the FFT function.

6.3.3 TRIGger Command (Sets and queries trigger source, level, etc.)

■ 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$*
	ch$ = CHA to CH16 (CHA to CHD when logic trigger)
	A$ = OFF
	LEVEl: level trigger
	IN: window trigger
	OUT: window out trigger
	LOGIc: logic 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

■ Sets and queries trigger level.

Syntax

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

ch$ = CH1 to CH16
A = voltage value (V) (temperature value (°C) when temperature unit, strain value ($\mu\varepsilon$) when strain unit)

Explanation

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

Example :TRIGGER:LEVEL CH1, 50E-1

Sets the trigger level of channel 1 to 50 mV.

When allowed In all functions

Note On the 8845 and 8846, the trigger level can be set in units of 0.25% of full scale. For example, for 1 V full scale (50 mV/division), the steps are 2.5 mV, and if a command indicates a setting not corresponding to one of these steps (for example 3 mV), the unit automatically rounds the value (3 mV → 2.5 mV).

■ Sets and queries trigger direction (slope).

Syntax

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

ch$ = CH1 to CH16
A$ = UP (rising: ⊥)
DOWN (falling: ⅂)

Explanation

command	Sets the trigger direction of the level 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 upper limit level for a window trigger.

Syntax

command	:TRIGger:UPPEr *ch$, A*
query	:TRIGger:UPPEr? *ch$*
response	*ch$, A* <NR3> *ch$* = CH1 to CH16 *A* = voltage value (V) (temperature value (℃) when temperature unit, strain value ($\mu\varepsilon$) when strain unit)

Explanation

command	Sets the upper limit level of the window-in or window-out trigger of the channel designated by *ch$* as voltage value.
query	Returns the current upper limit value of the window trigger as an NR3 numerical value.

Example :TRIGGER:UPPER CH1,+1.0E-3

Sets the upper limit level of the window trigger of channel 1 to +1.0 mV.

When allowed In all functions

Note On the 8845 and 8846, the upper levels of the window-in and window-out triggers can be set in units of 0.25% of full scale. Therefore, the 8845 automatically rounds the value as shown in the notes on the previous page.

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

Syntax

command	:TRIGger:LOWEr *ch$, A*
query	:TRIGger:LOWEr? *ch$*
response	*ch$, A* <NR1> *ch$* = CH1 to CH16 *A* = voltage value (V) (temperature value (℃) when temperature unit, strain value ($\mu\varepsilon$) when strain unit)

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 NR3 numerical value.

Example :TRIGGER:LOWER CH1,-1.0E-3

Sets the lower limit level of the window trigger of channel 1 to -1.0 mV.

When allowed In all functions

Note On the 8845 and 8846, the upper levels of the window-in and window-out triggers can be set in units of 0.25% of full scale. Therefore, the 8845 automatically rounds the value as shown in the notes of the ":TRIGger:LEVEL" command.

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

Syntax

command	:TRIGger:LOGPat *ch$, "A$"*
query	:TRIGger:LOGPat? *ch$*
response	*ch$, "A$"*
	ch$ = CHA to CHD
	A$ = XXXX : 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. Single quotation marks (') can be used instead of single quotation marks (‘).

Example :TRIGGER:LOGPAT CHA, 'X001'

Sets the trigger pattern for channel A to 'X001'.

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$*
	ch$ = CHA to CHD
	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 CHA, OR

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

When allowed In all functions

■ Sets and queries filter.

Syntax

command	:TRIGger:FILTer *ch$, A*
query	:TRIGger:FILTer? *ch$*
response	*ch$, A* <NR1> *ch$* = CH1 to CH16, CHA to CHD *A* = 0:OFF 10, 20, 50, 100, 150, 200, 250, 500, 1000 (samples)

Explanation

command	Sets the filter width for a level trigger of the channel designated by *ch$* to a numerical value from 10, 20, 50, 100, 150, 200, 250, 500, 1000.
query	Returns the current filter width as an NR1 numerical value.

Example

:TRIGGER:FILTER CH1,10

Sets the filter width for a trigger of channel 1 to 10 samples.

When allowed In all functions

■ 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 trigger mode.

Syntax

command	:TRIGger:MODE *A$*
query	:TRIGger:MODE?
response	*A$* *A$* = SINGle, REPEat, AUTO, AUTOSTOP : 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 all functions

■ 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, -95 (unit %)

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 8845, 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 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

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

Syntax

command	:TRIGger:TMINTvl *day, hour, min, sec*
query	:TRIGger:TMINTvl?
response	*day* <NR1>, *hour* <NR1>, *min* <NR1>, *sec* <NR1>
	day = 0 to 10
	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 1,1,20,30

Sets the time interval for the timer trigger to one day, one hour, twenty minutes, and thirty seconds for a day.

When allowed In all functions

■ Sets and queries the time point for trigger detections.

Syntax

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

Explanation

command	Sets the time point for trigger detection. During memory partitioning, the time point for the memory block which is currently being displayed (the block in use) is the one referenced.
query	Returns the currently set time point for trigger detection as a numerical value in NR1 format.

Example :TRIGGER:DETECTTIME?

The currently set time point for trigger detection is queried.

When allowed In all functions

■ Sets and queries the date for trigger detection.

Syntax

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

Explanation

command	Sets the date for trigger detection. During memory partitioning, the date for the memory block which is currently being displayed (the block in use) is the one referenced.
query	Returns the currently set date for trigger detection as a numerical value in NR1 format.

Example :TRIGGER:DETECTDATE?

The currently set date for trigger detection is queried.

When allowed In all functions

■ Sets and queries the time for start operating termination.

Syntax

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

Explanation

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

Example :TRIGGER:STOPTIME?

The currently set time for start operating termination is queried.

When allowed In all functions

■ Sets and queries the date for start operating termination.

Syntax

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

Explanation

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

Example :TRIGGER:STOPDATE?

The currently set date for start operating termination is queried.

When allowed In all functions

6.3.4 UNIT Command (Sets and queries input channel (voltage axis range, filter etc.))

■ 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 CH16 *A* = voltage range (unit V) temperature range (unit ℃) strain range (unit $\mu\varepsilon$)

Explanation

command	Sets the voltage axis range for the channel designated by *ch$* to a numerical value. When the channel designated is for the temperature unit, sets the temperature range.
query	Returns the current voltage axis range for the channel designated by *ch$* 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

■ Sets and queries input channel origin position.

Syntax

command	:UNIT:POSItion *ch$, A*
query	:UNIT:POSItion? *ch$*
response	*ch$, A* <NR1> *ch$* = CH1 to CH16 *A* = -15.6 to 35.6 (DIV) (single screen, voltage axis magnification ×1)

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 NR3 numerical value (unit DIV).

Example

:UNIT:POSITION CH1, 10

Sets the origin position for channel 1 to 10 divisions.

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 CH16 *A$* = GND, DC (8916, 8919, 8927) GND, DC, RMS, R_G (8917)

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$, A* <NR2> *ch$* = CH1 to CH16 *A* = 0, 5, 50, 500, 5000 (8916, 8927 ANALOG UNIT) 0, 5, 500 (8917 RMS UNIT) 0, 1.5, 5 (8918 TEMPERATURE UNIT) 0, 5, 50 (8919 FFT UNIT) 0, 10, 30, 300, 3000 (8928 STRAIN UNIT) (0: OFF)

Explanation

command	Sets the filter for the channel designated by *ch$*. If the channel designated is for the temperature unit, set the filter of the temperature unit.
query	Returns the current filter setting for the channel designated by *ch$* as an NR2 numerical value.

Example

:UNIT:FILTER CH1, 5

Sets the filter for channel 1 to 5 Hz.

When allowed In all functions

■ Carries out zero adjustment for the input units.

Syntax

command :UNIT:ADJUST

Explanation

Carries out zero adjustment for the input units, however, the temperature unit and strain unit has no zero adjustment function..

When allowed

In all functions

■ Sets and queries the type of the temperature input unit sensor.

Syntax

command :UNIT:SENSor *ch$, A$*
query :UNIT:SENSor? *ch$*
response *ch$, A$*
ch$ = CH1 to CH16
A$ = K, J, T

Explanation

command Sets the type of the temperature input unit sensor on the channel designated by *ch$*.
query Returns the type of the temperature input unit sensor currently on the channel designated by *ch$* as character data.

Example

:UNIT:SENSOR CH1,K
The temperature input unit sensor on channel 1 is set to "K".

When allowed

In all functions

■ Sets and queries the FFT anti-aliasing filter.

Syntax

command :UNIT:AAFilter *ch$, A$*
query :UNIT:AAFilter? *ch$*
response *ch$, A$*
ch$ = CH1 to CH16
A$ = OFF, ON

Explanation

command Turns on or off the FFT anti-aliasing filter on the channel designated by *A$*.
query Returns the current on or off state of the FFT anti-aliasing filter on the channel designated by *A$*.

Example

:UNIT:AAFILTER CH3,ON
Turns on the FFT anti-aliasing filter for channel 3.

When allowed

In all functions

■ Execution auto-balancing of all strain amplifiers.

Syntax command :UNIT:BALAnce

Explanation Carries out auto-balancing for all strain amplifiers.

When allowed In all functions

■ Execution auto-balancing of strain amplifiers in each channels.

Syntax command :UNIT:CHBAlance *ch$*

Explanation Carries out auto-balancing for strain amplifiers in each channels.

When allowed In all functions

■ Sets and queries the logic waveform channel.

Syntax
command :UNIT:LOGIc *A$*
query :UNIT:LOGIc?
response *A$*
A$ = OFF, CH1 to CH15 (odd number channels)

Explanation
command Sets the logic waveform channel.
query Returns the current setting of the logic waveform channel as characters.

Example :UNIT:LOGIC CH5
Sets the logic waveform channel to CH5.

When allowed In the memory recorder function and recorder function.

6.3.5 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 CHANnel 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 CH16 *A$* = OFF, 1 to 16

Explanation

command	Sets the waveform display style for the channel designated by *ch$* to OFF, 1 to 16.
query	Returns the current waveform display style setting for the channel designated by *ch$* as character data.

Example :DISPLAY:DRAWING CH1,1

Displays the channel 1 waveform with red.

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

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

Syntax

command	:DISPlay:GRAPh *ch$, A*
query	:DISPlay:GRAPh? *ch$*
response	*ch$, G$*

ch$ = CH1 to CH16
A = 1 to 8
(when DUAL, no 3 to 8)
(when QUAD, no 5 to 8)

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 CH1,1

Displays the channel 1 waveform in display graph 1.

When allowed

In the memory recorder function and the recorder function.

■ Enables and and disables, and queries display of logic waveforms.

Syntax

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

ch$ = CHA to CHD
A$ = OFF, ON

Explanation

command	Enables and disables display of logic waveforms.
query	Returns current enablement state of logic waveform display as character data.

Example

:DISPLAY:LOGDRAW CHA, ON

Enables display of the channel A logic waveform.

When allowed

In the memory recorder function and the recorder function.

■ Sets and queries operation of logic waveform display.

Syntax

command	:DISPlay:LOGPosi *ch$, A*
query	:DISPlay:LOGPosi? *ch$*
response	*ch$, A$* *ch$* = CHA to CHD *A* = 1 to 8

Explanation

command	Sets the position of logic waveform display.
query	Returns the position of the current logic waveform display as character data.

Example :DISPLAY:LOGPOSI CHA,1

Sets the position of logic waveform display for channel A to 1.

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

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

Syntax

command	:DISPlay:XMAG *A$*
query	:DISPlay:XMAG?
response	*A$* MEM: *A$* = X10, X5, X2, X1, X1_2, X1_5, X1_10, X1_20, X1_50, X1_100, X1_200, X1_500, X1_1000 REC: *A$* = X10, X5, X2, X1, X1_2, X1_3, X1_4, X1_5, X1_6, X1_8, X1_10, X1_12, X1_15, X1_16, X1_20, X1_24, X1_25, X1_30, X1_40, X1_50, X1_60, X1_80, X1_100, X1_120, X1_150, X1_160, X1_180, X1_200, X1_240, X1_250, X1_300, X1_360, X_400, X1_500, X1_600, X1_720, X1_800, X1_1000, X1_1200, X1_1500, X1_1600, X1_1800, X1_2000, X1_2400, X1_2500, X1_3000, X1_3600, X1_4000, X1_5000, X1_6000, X1_7200, X1_8000, X1_10000, X1_12000, X1_15000, X1_16000, X1_18000, X1_20000, X1_24000, X1_30000, X1_36000, X1_48000, X1_50000, X1_60000, X1_72000, X1_96000, X1_100000, X1_120000, X1_150000, X1_180000, X1_240000, X1_300000, X1_360000, X1_480000, X1_600000, X1_720000, X1_960000, X1_1440000, X1_1800000, X1_2880000

Explanation

command	Sets the magnification/compression factor on the time axis according to character data.
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.

■ 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 CH16 *A$* = X20, X10, X5, X2, X1, X1_2, X1_5, X1_10

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 the recorder function, and FFT function.

■ Sets and queries the drawing level for an X-Y plot.

Syntax

command	:DISPlay:XYDRawing *A, B$*
query	:DISPlay:XYDRawing ? *A*
response	*A* <NR1>, *B$* *A* = 1 to 4 *B$* = OFF, 1 to 16

Explanation

command	Sets the XY waveform display.
query	Returns the current XY waveform display setting.

Explanation :DISPLAY:XYDRAWING 1,1

Sets the graph 1 to red display.

When allowed In the memory recorder function in XY format.

■ Sets and queries the X-axis in the XY format.

Syntax

command	:DISPlay:XAXIs *A, ch$*
query	:DISPlay:XAXIs? *A*
response	*A* <NR1>, *ch$* *A* = 1 to 4 *ch$* = CH1 to CH16

Explanation

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

Explanation :DISPLAY:XAXIS 1,CH1

Sets the graph 1 to the X axis channel 1.

When allowed In the memory recorder function in XY format.

■ Sets and queries the Y-axis in the XY format.

Syntax

command	:DISPlay:YAXIs *A, ch$*
query	:DISPlay:YAXIs? *A*
response	*A* <NR1>, *ch$*
	A = 1 to 4
	ch$ = CH1 to CH16

Explanation

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

Explanation :DISPLAY:YAXIS 1,CH2

Sets the graph 1 to the Y axis channel 2.

When allowed In the memory recorder function in XY format.

■ 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$*.

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, and queries the variable function.

Syntax

command	:DISPlay:VARIable *ch$, A$*
query	:DISPlay:VARIable? *ch$*
response	*ch$, A$*
	ch$ = CH1 to CH16
	A$ = ON, OFF

Explanation

command	Enables or disables the variable function for channel designated by *ch$*.
query	Returns the current state of enablement of the variable function.

When allowed In all functions

■ Sets and queries the upper limit and lower limit values of the variable function.

Syntax

command :DISPlay:VARIUPLOw *ch$, B, C*
query :DISPlay:VARIUPLOw? *ch$*
response *ch$, B* <NR3>, *C* <NR3>
ch$ = CH1 to CH16
B = *C* = -9.9999E+29 to +9.9999E+29
B: upper limit value C: lower limit value

Explanation

command Sets the upper and lower limit values of the waveform on the display screen for CH1 to CH16.
query Returns the current upper and lower limit values of the waveform on the display screen for CH1 to CH16 as an NR3 numerical value.

When allowed In all functions

■ Searches trigger.

Syntax

command :DISPlay:TRSEarch *A, B*
A <NR1> = start data number
B <NR1> = end data number

Explanation

command Runs a trigger search using data between numbers specified by *A* and *B*.

Example :DISPLAY:TRSEARCH 0,2500
Searches the trigger of the data between 0 and 2500.

When allowed In the recorder function (only display screen)

6.3.6 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, REC OFF, Xcur, Ycur, TRACe : MEM (XY format) OFF, TRACe

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.
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 cursor 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 all functions

■ Sets and queries the channel for the A cursor.

Syntax

command	:CURSor:ACHAnnel *ch$*
query	:CURSor:ACHAnnel?
response	*ch$*
	ch$ = CH1 to CH16
	X1 to X4 (when XY format)

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 (excluding FFT function).

■ Sets and queries the channel for the B cursor.

Syntax

command	:CURSor:BCHAnnel *ch$*
query	:CURSor:BCHAnnel?
response	*ch$*
	ch$ = CH1 to CH16
	X1 to X4 (when XY format)

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 (excluding FFT function).

■ 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
(100 × recording length) : MEM
A = 0 to number of stored data values : REC
Analysis mode
A = 0 to 999 (STR, ACR, CCR, IMP) : FFT
A = 0 to 400 (LIN, RMS, PSP, TRF, COH, HIS, CSP, OCT) : FFT
(horizontal cursor)
A = 0 to 639 : MEM, REC

Explanation

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

Example

:CURSOR:APOSITION 1000
Move the A cursor position to 1000 points (10DIV).

When allowed In all functions

■ 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 that for APOSition.

Explanation

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

Example

:CURSOR:BPOSITION 1000
Move the B cursor position to 1000 points (10DIV).

When allowed In all functions

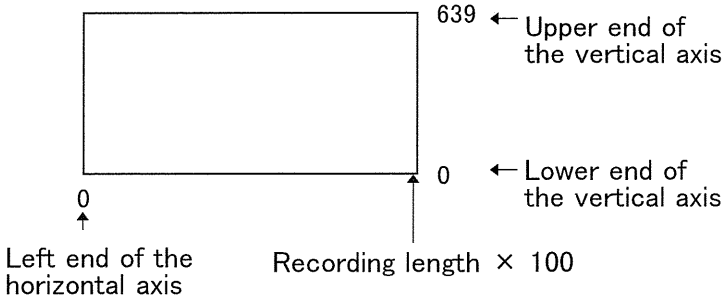
The cursor position

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.

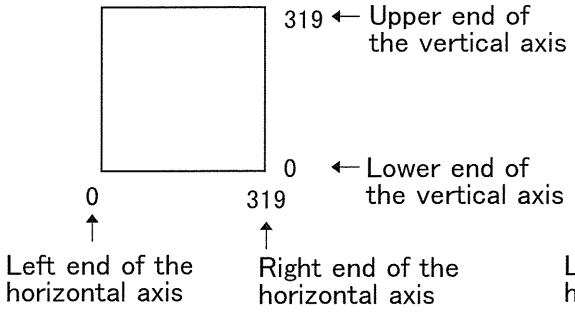
(In the 8845 and 8846, the stored data values per one division are 100 points, so when recording length is 25 division, the number of stored data values is 2500 points (25 divisions × 100 points). Therefore, the cursor position indication lies in the range from 0 to 2500.)

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

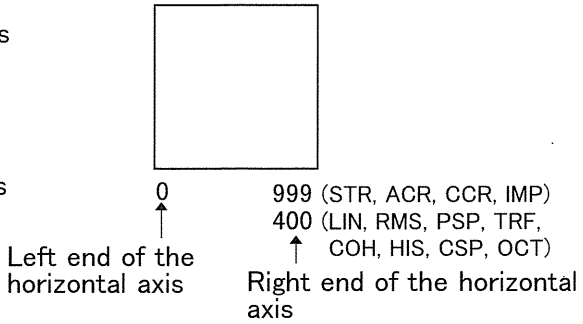
In the memory recorder function and the recorder function



In the memory recorder function (X–Y format)



In the FFT function (trace cursor)



■ Queries the cursor readout value (t).

Syntax

query :CURSor:DTREad? *A$*

response *"B* unit"

A$ = A, B, A_B

B = the readout value (t, 1/t)

Explanation

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

Example

query :CURSOR:DTREAD? A

response :CURSOR:DTREAD"5ms, 200Hz"

Queries the A cursor readout value.

When allowed

Provided that the cursor is not off, and that (t, 1/t) are being shown on the display.

■ Sets and queries the cursor readout value (V).

Syntax

query :CURSor:DVREad? *A$*
response *"B* unit*"*
A$ = A, B, A_B
B$ = the readout value (V)

Explanation

query: Returns the cursor readout value (V) as a line of character data. For the temperature input unit and strain unit, returns the temperature value (℃) or strain value ($\mu\varepsilon$).

Example

query :CURSOR:DVREAD? A
response :CURSOR:DVREAD"385mV"

Queries the A cursor readout value.

When allowed

Provided that the cursor is not off, and that (V) is being shown on the display.

■ Sets and queries the graph for the A and B cursors.

Syntax

command :CURSor:ABCHAnnel *A$*
query :CURSor:ABCHAnnel?
response *A$*
A$ = G1, G2

Explanation

command: Sets the graph for the A and B cursors when the display format is DUAL. If the display format is SINGle or NYQuist, the cursor is displayed on graph 1, whatever setting is made with this command.

query: Returns the current graph setting for the A and B cursors as character data.

Example

```
:CONFIGURE:FORMAT DUAL
:CURSOR:ABCHANNEL G1
:CURSOR:MODE TRACE
```

The A and B cursors are displayed on graph 1.

When allowed

In the FFT function.

■ Queries the cursor readout position for FFT data.

Syntax

query	:CURSor:DFREad? *A$*
response	*"B* unit, *C* unit*"*
	A$ = A, B, A_B
	B = x-axis data
	C = y-axis data

Explanation

query	Returns the current cursor readout position for FFT data as character data.

Example

:CURSOR:DFREAD? A

The A cursor readout position is returned as character data.

When allowed

In the FFT function

6.3.7 MEMory Command (Sets and queries input and output, etc.)

■ 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 CH16, CHA to CHD
	A = 0 to 2000000

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.

■ Queries the number of data samples stored.

Syntax

query :MEMory:MAXPoint?

response *A* <NR1>

A = 0 : no data stored

2500 to 2000000 (divided by 100 gives the number of divisions)

Explanation

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

Example

query :MEMORY:MAXPOINT?

response :MEMORY:MAXPOINT 2500 (when headers are on)

The number of data samples stored in the memory is 2500 (25 divisions).

When allowed In the memory recorder function.

■ Prepares the memory.

Syntax

command only :MEMory:PREPare

Explanation

command only If there is no waveform data in the 8845 or 8846 unit, ensures that the memory is in a state ready and able to receive transmitted data. If waveform data is currently stored in the unit, clears it.

Example

query :MEMORY:PREPARE

Prepares the memory for receipt of waveform data.

When allowed In the memory recorder function.

■ Inputs data to memory, and outputs stored data (in ASCII).

Syntax

command :MEMory:ADATa *B, C,...*
query :MEMory:ADATa? *A*
response *B, C,...* all <NR1>
B, C,... = 0 to 4095 (8916 to 8919)
0 to 16383 (8927)
A = 1 to 80 (number of data)

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.
To display the waveform data input, exit from the display screen once, and then display again.

query: The number of data 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
```

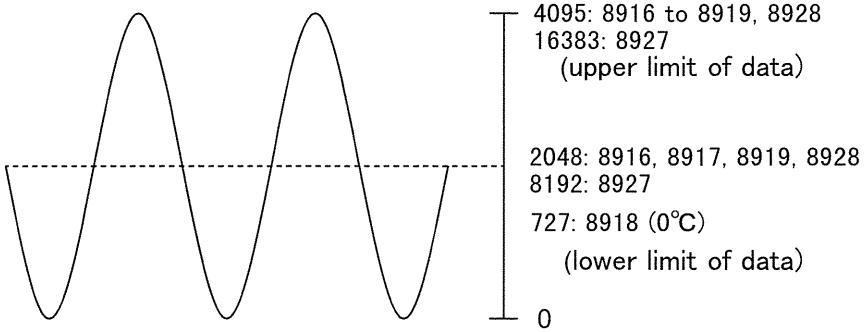
Sets the input/output point to channel 1 and data value zero in memory, then outputs 10 stored data values.

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 (8916 to 8919, 8928: 0 to 4095, 8927: 0 to 16383) input and output using the :MEMory:ADATa command and the measured voltage values.



Measured voltage value = (data value-2048)×(voltage range)/80 (8916 to 8919)
(data value-8192)× (voltage range)/320 (8927)

Example: When voltage range = 1 (V/DIV), data value = 2500
Measurement voltage = (2500 - 2048)×1/80 = 5.65 (V) (8916 to 8919)
(2500 - 8192)×1/320 = -17.79 (V) (8927)

Reference

The transfer rate by the :MEMORY:ADATA? 40 command is:
when recording length is 10,000 DIV, 1085 seconds (8845), 687 seconds (8846).
when recording length is 1,000 DIV, 109 seconds (8845), 79 seconds (8846).
(PC9801 RX in use, receiving LINE INPUT@)

■ 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 35 (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 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.

Reference

The transfer rate by the :MEMORY:VDATA? 10 command is:
when recording length is 1,000 DIV, 441 seconds (8845), 193 seconds (8846).
(PC9801 RX in use, receiving LINE INPUT@)

■ Outputs real time data (in ASCII)

Syntax

query :MEMory:AREAl? *ch$*

response *A* <NR1>

ch$ = CH1 to CH16

A = 0 to 4095 (8916 to 8919, 8928)

0 to 16383 (8927)

Explanation

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

Example

query :MEMORY:AREAL? CH1

response :MEMORY:AREAL 3022 (for 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 CH16

A = a voltage value (unit V), temperature value (unit℃), or strain value (unit $\mu\varepsilon$)

Explanation

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

Example

query :MEMORY:VREAL CH1

response :MEMORY:VREAL 5.5E-2 (for header on)

When allowed Providing that measurement operation is not taking place.

■ 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,... = 0 to 255 (logic data)
A = 1 to 160 (number of data)

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.

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. This cannot be executed during measurement operation.

The following is the correspondence between the channels set by the MEMory:POINt command and the logic channel groups:

CHA-----CHA1 to A4
CHB-----CHB1 to B4
CHC-----CHC1 to C4
CHD-----CHD1 to D4

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	
0	0	0	0	A4	A3	A2	A1	LOW : 0 HIGH : 1

Example

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

This sets the input/output point to channel A, and stored data value to address 0 in memory, then outputs 10 data values in binary format. Then channels A1 to A4 are as follows;

7	6	5	4	3	2	1	0	
0	0	0	0	1	0	1	0	LOW : 0 HIGH : 1
				A4	A3	A2	A1	

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.

Reference The transfer rate by the :MEMORY:LDATA? 80 command is:
when recording length is 1,000 DIV, 64 seconds (8845), 43 seconds (8846).
(PC9801 RX in use, receiving LINE INPUT@)

■ Binary transfer of stored data.

Syntax

query :MEMory:BDATa? *A*

response #0＊＊＊＊＊＊＊＊ ・・・・

A = 1 to 250 (number of data)

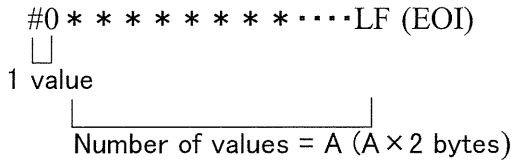
Explanation

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.

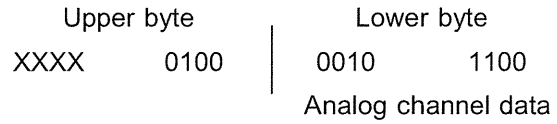
The format of the output data is as follows:

- Initially: "#0" (Indicates binary format.)
- After the "#0", the number of data values specified by *A* (each value is 2 bytes), is transmitted.
- The data is followed by LF (0AH) + EOI.

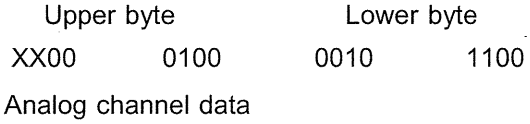


- The data consists of the unaltered binary codes of the data stored in memory. The bits are transmitted most significant bit first.
  When the 8916 to 8919 in use, the first four bits have no meaning.
  When the 8927 in use, the first two bits have no meaning.

  When using the 8916 to 8919, 8928:



  When using the 8927



- The data obtained is the same as that for ADATa?; 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.

Reference

The transfer time in response to the command :MEMORY:BDATA? 125 is as follows:
when recording length is 10,000 DIV, 379 seconds (8845), 242 seconds (8846).
when recording length is 1,000 DIV, 38 seconds (8845), 25 seconds (8846).
(PC 9801RX in use, recording LINE INPUT@)

■ Binary transfer of stored data.

Syntax

query :MEMory:LBDAta? *A*

response #0＊＊＊＊＊＊＊ ・・・・

A = 1 to 500 (number of data)

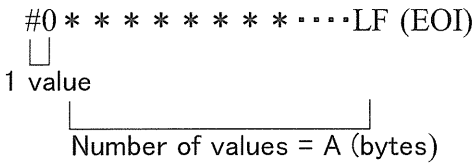
Explanation

query

Outputs the data stored (logic data) 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 "#0", the number of data values specified by *A* (each value is 2 bytes), is transmitted.
- The data is followed by LF (0AH) + EOI.



- The data obtained is the same as that for LDATa?; for details refer to these commands.
  It is not possible to input data in binary format.

Example

```
:MEMORY:POINT CHA, 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.

Reference

The transfer time in response to the command :MEMORY:LBDATA? 250 is as follows:
when recording length is 1,000 DIV, 21 seconds (8845), 14 seconds (8846).
(PC 9801RX in use, recording LINE INPUT@)

■ Sets and queries the output point for FFT data.

Syntax

command	:MEMory:FFTPOint *A$*, *B*
query	:MEMory:FFTPOint?
response	*A$*, *B*<NR1>
	A$ = G1, G2
	B = 0 to 999 : in analysis mode STR, ACR, CCR, or IMP
	0 to 400 : in analysis mode LIN, RMS, PSP, TRF, COH, CSP, HIS, or OCT

Explanation

command	Sets the output point for FFT data on the graph number designated by *A$*.
query	Returns the current output point as an NR1 format.

Example :MEMORY:FFTPOINT G1,100

Sets the output point for FFT data on the graph 1 to 100.

When allowed In the FFT function.

■ Queries the output point for FFT data.

Syntax

query only	:MEMory:FFTData?
response	"*A* unit, *B* unit"
	A = x-axis data (in <NR3> numerical format)
	y-axis data (in <NR3> numerical format)

Explanation

query only	Returns the x-axis and y-axis FFT data at the output point specified by the instruction :MEMORY:FFTPOINT in <NR3> numerical format. When this command is executed, only one output point is calculated, and then the specified output point is increased by one. By executing this command repeatedly, a continuous set of data can be obtained.

Example

:MEMORY:FFTPOINT G1,100
:MEMORY:FFTDATA?

Returns the x-axis and y-axis FFT data at points of 100 on graph 1.

When allowed In the FFT function.

■ Sets and queries the output point for storage data. (recorder function)

Syntax

command	:MEMory:RECPOint *ch$, A*
query	:MEMory:RECPOint?
response	*ch$, A*<NR1> *ch$* = CH1 to CH16, CHA to CHD *A* = 0 to 2000000

Explanation

command	Sets the output point for stored data in recorder function.
query	Returns the current output point for stored data in recorder function as an NR1 format.

Example :MEMORY:RECPOINT CH1,100

Sets the output point in recorder function to channel 1 and data value 100 in memory.

When allowed In the recorder function.

■ Queries the number of stored data.

Syntax

query	:MEMory:RECMAXPoint?
response	*A* <NR1> *A* = 0 : no data stored 2500 to 2000000 (divided by 100 gives the number of divisions)

Explanation

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

Example

query	:MEMORY:RECMAXPOINT?
response	:MEMORY:RECMAXPOINT 2500 (when headers are on)

The number of data samples stored in the memory is 2500 (25 divisions).

When allowed In the recorder function.

■ Outputs stored data (in ASCII). (recorder function)

Syntax

query :MEMory:RECAData? *A*

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

B, C,... = 0 to 4095 (8916 to 8919, 8928)
0 to 16383 (8927)

A = 1 to 80 (number of data)

Explanation

query

The number of data specified by *A* are output from the memory channel and point set by the MEMory:RECPOint command.
The output point is incremented by the number of data values.
This cannot be executed during measurement operation.
This data obtained is the same as that for ADATA?

Example

```
:MEMORY:RECPOINT CH1, 0
:MEMORY:RECADATA? 10
```

Sets the output point to channel 1 and data value zero in memory, then outputs 10 stored data values.

When allowed

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

■ Output voltage data from memory.

Syntax

query :MEMory:RECVData? *A*

response *B, C,...* all (NR3>

B, C,... = voltage values (unit volts), temperature value (unit ℃), or strain value (unit $\mu\varepsilon$)

A = 1 to 35 (amount of data)

Explanation

query

The number of stored data specified by *A* are output as voltage values from the memory channel and point set by the MEMory:RECPOINt command.
The output point is incremented by the number of data values.
When scaling, the scaled values are output. When calculating the waveform, calculated results are input and output.
This cannot be executed during measurement operation.

Example

```
:MEMORY:RECPOINT CH1, 0
:MEMORY:RECVDATA? 10
```

Sets the output point to channel 1 and data value zero in memory, then outputs 10 stored data values as voltage values.

When allowed

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

■ Output logic data from memory. (recorder function)

Syntax

query :MEMory:RECLdata? *A*

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

A = 1 to 160 (number of data)

Explanation

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:RECPOINt command. The output point is incremented by the number of data.
This cannot be executed during measurement operation.
The data obtained in the same as that for LDATa?.

Example

```
:MEMORY:RECPOINT CH1, 0
:MEMORY:RECLDATA? 10
```

Sets the output point to channel 1 and data value zero in memory, then outputs 10 stored data values as voltage values.

When allowed

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

■ Binary transfer of stored data.

Syntax

query :MEMory:RECBData? *A*

response #0＊＊＊＊＊＊＊＊ ・・・・

A = 1 to 250 (number of data)

Explanation

query Outputs the data stored by a MEMory:RECPOint specification in binary format. The output point is incremented by the number of data values.
This cannot be executed during measurement operation.
The data obtained is the same as that for BDATa?;

Example

```
:MEMORY:RECPOINT CH1, 0
:MEMORY:RECBDATA? 10
```

This sets the 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 recorder function, provided that stored data is present, and provided that the output point is lower than the amount of data stored.

■ Binary transfer of stored data (logic). (recorder function)

Syntax

query :MEMory:RECLBdata? *A*

response #0＊＊＊＊＊＊＊＊ ・・・・

A = 1 to 500 (number of data)

Explanation

query Outputs the data stored (logic data) by a MEMory:RECPOINt specification in binary format. The output point is incremented by the number of data values.
This cannot be executed during measurement operation.
The data obtained is the same as that for LBDATa?;

Example

```
:MEMORY:RECPOINT CH1, 0
:MEMORY:RECLBDATA? 10
```

This sets the 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 recorder function, provided that stored data is present, and provided that the output point is lower than the amount of data stored.

■ Conversion data to memory waveform from recorder waveform.

Syntax

command :MEMory:RECTomem

Explanation

command Converts the waveform data captured in the recorder function to data in memory recorder function.

When allowed In the recorder function.

6.3.8 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).

■ Sets and queries the number of channels used.

Syntax

command	:SYSTem:USEUNit *A*
query	:SYSTem:USEUNit?
response	*A* <NR1> *A* = 1, 2, 4, 8

Explanation

command	Sets the number of units used to a numerical value.
query	Returns the current number of units used as as NR1 numerical value.

Example :SYSTEM:USEUNIT 8

Sets the number of units to 8.

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

■ Sets and queries the grid type.

Syntax

command	:SYSTem:GRID *A$*
query	:SYSTem:GRID?
response	*A$* *A$* = OFF, STANdard, FINE, D_STANdard, D_FINE, S_STANdard, S_FINE

Explanation

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

Example :SYSTEM:GRID STANDARD

Sets the grid type to STANDARD.

When allowed In all functions

■ 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

■ Sets and queries the time axis display.

Syntax

command	:SYSTem:TMAXis *A$*
query	:SYSTem:TMAXis?
response	*A$* *A$* = TIME, TIME (60), DIV, DATE

Explanation

command	Sets the time axis display as character string data.
query	Returns the current time axis display setting as character string data.

Example :SYSTEM:TMAXIS TIME

Sets the time axis display to TIME.

When allowed In all functions

■ Sets and queries the list function and the gauge function.

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

■ Enables and disables, an queries, the screen back light 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 back light saver function to ON.

When allowed In all functions

■ Sets and queries the LCD display.

Syntax

command	:SYSTem:LCDDisp *A*
query	:SYSTem:LCDDisp?
response	*A*<NR1>
	A = 1 to 32

Explanation

command	Sets the LCD display.
query	Returns the current LCD display setting as NR1 numerical value.

Example :SYSTEM:LCDDISP 1

Sets the screen dump size to 1.

When allowed In all functions

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

Syntax

command	:SYSTem:VOLUme *A$*
query	:SYSTem:VOLUme ?
response	*A$*
	A$ = OFF, HIGH, MEDium, LOW

Explanation

command	Sets the beeper sound.
query	Returns the current volume setting of the beeper sound as character data.

Example :SYSTEM:VOLUME LOW

Sets the beeper sound to LOW.

When allowed In all functions

■ Enables and disables, and queries intermittent printing function.

Syntax

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

Explanation

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

Example :SYSTEM:INTPRINT ON

Sets the intermittent print function to ON.

When allowed In all functions

■ Sets and queries the copy output destination.

Syntax

command	:SYSTem:CPYOut *A$*
query	:SYSTem:CPYOut ?
response	*A$*
	A$ = PRINter, MO_Mono, MO_256

Explanation

command	Sets the copy output destination.
query	Returns the current setting of copy output destination as character data.

Example :SYSTEM:CPYOUT MO_MONO

Sets the copy output destination to MO in monochrome.

When allowed In all functions (8846 only)

■ Sets and queries the language.

Syntax

command	:SYSTem:LANGuage *A$*
query	:SYSTem:LANGuage ?
response	*A$*
	A$ = JAPAnese, ENGLish

Explanation

command	Sets the language.
query	Returns the current language setting as character data.

Example :SYSTEM:LANGUAGE ENGLISH

Sets the display in English.

When allowed In all functions (8846 only)

6.3.9 SCALing Command (Sets and queries scaling.)

■ Sets and queries the scaling function.

Syntax

command	:SCALing:KIND *A$*
query	:SCALing:KIND?
response	*A$*
	A$ = RATIO, POINT

Explanation

command	Sets the scaling type as character string data.
query	Returns the current scaling type setting as character data.

When allowed In all functions

■ Enables and disables, and queries the scaling function.

Syntax

command	:SCALing:SET *ch$, A$*
query	:SCALing:SET? *ch$*
response	*ch$, A$*
	ch$ = CH1 to CH16
	A$ = OFF, SCI (or ON), and ENG

Explanation

command	Enables or disables the scaling function. A setting SCI or ON produces conventional scientific floating-point notation. The setting ENG produces floating-point notation using powers of 10^3.
query	Returns the current state of enablement of the scaling function 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$, B*
query	:SCALing:VOLT? *ch$*
response	*B* <NR3>
	ch$ = CH1 to CH16
	B = scaling conversion value (EU/volts)
	(-9.9999E+9 to +9.9999E+9)

Explanation

command	Sets the scaling conversion value for CH1 to CH16.
query	Returns the current scaling conversion value for CH1 to CH16 as NR3 numerical value.

Example :SCALING:VOLT CH1,+2.0E-3

Sets the scaling conversion value for channel 1 to +2. 0E-3.

When allowed In all functions

■ Sets and queries the scaling offset.

Syntax

command	:SCALing:OFFSet *ch$, B*
query	:SCALing:OFFSet? *ch$*
response	*ch$, B* <NR3> *ch$* = CH1 to CH16 *B* = scaling offset (EU offset) (-9.9999E+9 to +9.9999E+9)

Explanation

command	Sets the scaling offset for CH1 to CH16.
query	Returns the current scaling offset for CH1 to CH16 as an NR3 numerical value.

Example :SCALING:OFFSET CH1,+1.0E-3

Sets the scaling offset for channel 1 to +1.0E-3.

When allowed In all functions

■ Sets and queries the scaling unit.

Syntax

command	:SCALing:UNIT *ch$, "B$"*
query	:SCALing:UNIT? *ch$*
response	*ch$, "B$"* *ch$* = CH1 to CH16 *B$* = scaling unit (up to 7 characters)

Explanation

command	Sets the scaling unit for CH1 to CH16 (up to 7 characters allowed). ^2 = ^{2} ^3 = ^{3} ~u = μ ~o = Ω ~e = ε ~c = $^\circ$
query	Returns the scaling unit for CH1 to CH16 as character string data. Single quotation marks (') can be used instead of single quotation marks ('').

Example :SCALING:UNIT CH1, "mA"

Sets the scaling unit for channel 1 to milliamps.

When allowed In all functions

■ Sets and queries the scaling VOLT UP and LOW.

Syntax

command	:SCALing:VOUPLOw *ch$, B, C*
query	:SCALing:VOUPLOw? *ch$*
response	*ch$, B* <NR3>, *C* <NR3>, *ch$* = CH1 to CH16 *B* = *C* = -9.9999E+29 to +9.9999E+29

Explanation

command	Sets the scaling VOLT UP and VOLT LOW values for CH1 to CH16.
query	Returns the current setting VOLT UP and VOLT LOW values for CH1 to CH16 as an NR3 numerical value.

When allowed In all functions

■ Sets and queries the scaling SCALE UP and LOW.

Syntax

command	:SCALing:SCUPLOw *ch$, B, C*
query	:SCALing:SCUPLOw? *ch$*
response	*ch$, B* <NR3>, *C* <NR3> *ch$* = CH1 to CH16 *B* = *C* = -9.9999E+29 to +9.9999E+29

Explanation

command	Sets the scaling SC UP and SC LOW values for CH1 to CH16.
query	Returns the current setting SC UP and SC LOW values for CH1 to CH16 as an NR3 numerical value.

When allowed In all functions

6.3.10 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, SETTing, COMMent, S_C (setting &comment)
B$ = comment characters (up to 20 characters)

Explanation

command: Enables and disables comments, and inputs a string of comment characters.
^2 = ^{2}
^3 = ^{3}
~u = μ
~o = Ω
~e = ε
~c = °

Single quotation marks (') can be used instead of double 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 8845'

Inputs "HIOKI 8845" as a title comment.

When allowed

In all functions

■ Enables and disables, and queries, comments for all channels.

Syntax

command	:COMMent:EACHch *A$*
query	:COMMent:EACHch?
response	*A$*

A$ = OFF, SETTing, COMMent, S_C (setting & comment)

Explanation

command: Enables and disables comments for all channels.

query: Returns the current ON/OFF enablement state of comments for all channels as character data.

Example

:COMMENT:EACHCH COMMENT

Sets the comments for all channels to COMMENT.

When allowed

In all functions

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

Syntax

command	:COMMent:CH *ch$*, *"A$"*
query	:COMMent:CH? *ch$*
response	*ch$*, *"A$"*

ch$ = CH1 to CH16, CHA to CHD
A$ = 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. Single quotation marks (') can be used instead of double 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:CH1,'ch1 = TEST'
```

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

When allowed

In all functions

6.3.11 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$*"
query	:CALCulate:Z1?
response	*Z$*, "*A$*" *Z$* = Z1 to Z8 *A$* = calculation equation (up to 80 characters, alphabets in small letter, operator in capital letter) (Syntax of :Z2 to :Z8 commands are same as the :Z1 command.)

Explanation

command	Sets the waveform processing calculation equation for Z1 according to the character data. Single quotation marks(') can be used instead of double quotation marks (").
query	Returns the current for the waveform processing calculation equation for Z1 as character data.

Example

:CALCULATE:Z1 'a+b+ABC (CH1)'

Sets up the calculation equation for Z1 to be Z1 = a+b+ABC (CH1)

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* = -9.9999E+9 to +9.9999E+9

Explanation

command	Sets to the given numerical value the one of the coefficients a to p which is designated by *A$*.
query	Returns as an NR3 numerical value the current value of that one of the coefficients a to p which is designated by *A$*. (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 the moving averaging for the waveform processing calculation.

Syntax

command	:CALCulate:MOVE *Z$*, *A*
query	:CALCulate:MOVE? *Z$*
response	*Z$*, *A* <NR1> *Z$* = Z1 to Z8 *A* = 0 to 4000 <NR1>

Explanation

command	Sets the moving averaging for the waveform processing calculation designated by *Z$*.
query	Returns as an <NR1> numerical value the current setting of the value of the moving averaging for the waveform processing calculation.

Example

:CALCULATE:MOVE Z1, 200

Sets the moving averaging of Z1 equation to 200.

When allowed

In the memory recorder function.

■ Sets the parallel moving averaging for the waveform processing calculation.

Syntax

command	:CALCulate:SLIDe *Z$, A*
query	:CALCulate:SLIDe? *Z$*
response	*Z$, A* <NR1> *Z$* = Z1 to Z8 *A* = -4000 to 4000 <NR1>

Explanation

command	Sets the parallel moving averaging for the waveform processing calculation designated by *Z$*.
query	Returns as an <NR1> numerical value the current setting of the value of the parallel moving averaging for the waveform processing calculation.

Example :CALCULATE:SLIDE Z1,200

Sets the parallel moving averaging of Z1 equation to 200.

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$, A$*
query	:CALCulate:Z1DIsplay?
response	*ch$, A$* *ch$* = CH1 to CH16, NONE *A$* = MANUal, AUTO

(Syntax of :Z2DIsplay to :Z8DIsplay 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$*. When *A$* is MANUal, displays within upper and lower limits of setting values on the variable screen (unit: V or℃). (When scaling, displays in its unit.)
query	Returns the currently set display channel of the calculated result of the waveform processing calculation equation for Z1.

Example :CALCULATE:Z1DISPLAY CH1, MANUAL

Displays the calculated result of the waveform processing calculation equation for Z1 on channel 1. However, the range between upper and lower limits for the channel 1 on the variable screen.

When allowed In the memory recorder function.

■ 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, ON

Explanation

command	Enables or disables the print output of waveform parameter calculation values according to the character data.
query	Returns whether print out is enabled or disabled.

Example

:CALCULATE:MEASPRINT ON

Outputs the result of waveform parameter calculation to the printer.

When allowed

In the memory recorder function.

■ 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
PERI : period
FREQ : frequency
RISE : rise time
FALL : fall time
STD : standard deviation
AREA : area value
XYAREA : X-Y area value

ch$ = CH1 to CH16, 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 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 *NO$, ch$, A$, B* <NR3>

NO$ = NO1 to NO4

ch$ = CH1 to CH16

A$ = OFF, AVE, RMS, PP, MAX, MAXT, MIN, MINT, PERI, FREQ, RISE, FALL, STD, AREA, 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 NO1, CH1, 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, OUT, IN

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, OUT

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

When allowed In the memory recorder function.

■ 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$, upper* <NR3>, *lower* <NR3> *NO$* = NO1 to NO4 *upper, lower* = -9.9999E+29 to +9.9999E+29

Explanation

command	Sets the upper limit and the lower limit used when performing a decision on the waveform parameter calculated value designated by *A$*.
query	Returns the upper limit and the lower limit used when performing a decision on the waveform parameter calculated value designated by *A$* as an NR3 numerical values.

Example

:CALCULATE:COMPAREA NO1, +1.000E+0, −1.000E+0

Sets the decision value for the waveform parameter calculation NO1 to be in the range -1.000E+0 < NO1 < +1.000E+0

When allowed In the memory recorder function.

6.3.12 DAT Command (8845 only)

■ Enables and disables the DAT mode.

Syntax

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

Explanation

command	Enables or disables DAT mode.
query	Returns the current setting of DAT mode.

Example

:DAT:MODE ON

Sets the DAT mode to on.

When allowed In all functions

■ Queries information about a file saved on a tape.

Syntax

query :DAT:INFOr? *NO*

response *NO*, *"NAME$"*, *A$*, *"DATE$"*, *"TIME$"*

NO <NR1> = file number

NAME$ = file name

A$ = type of information saved:

W : measurement data

F : conditions of creation

A : waveform decision area

N : no such file

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

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

Explanation

query Returns information about the file whose name is specified in *NO*.

If no such file exists, returns: -1, "----", N, "-----", "-:-:-".

Example

query :DAT:INFOR? 1

response :DAT:INFOR 1,"8845 DAT ",W,"1997-05-16","10:20:30" (When header on)

Queries the file information of the file number 1.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Loads a file from a tape.

Syntax

command :DAT:LOAD *NO* (,*"S_TIME$"*)

NO <NR1> = file number

S_TIME$ = start time (day-hour:min:sec)

day: 000 to 999

hour: 00 to 23

min, sec: 00 to 59

Explanation

command Loads the data in the file numbered *NO*.

When the start time is not specified, loads normally.

Only the data file in the recorder function, the start time can be specified.

Example

:DAT:LOAD 1

Loads the data of the file number 1.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Saves a file on a tape.

Syntax command :DAT:SAVE *"NAME$"* (, *A$*)
In the memory recorder and FFT functions
NAME$ = file name (up to 16 characters)
A$ = type of saved information
W : measurement data
F : setting data
A : waveform decision area
B$ = method of saving
(none) : usual saving
ALL : all blocks saving during memory segmentation
In the recorder function
A$ = F

Explanation command Saves the information specified by *A$* on a tape.
In the recorder function, the data in memory can be saved when *A$* is W.
Single quotation marks (') can be used instead of double quotation marks (").

Example :DAT:SAVE '8845 WAVE',W
Saves all channels of measurement data under the file name "8845 WAVE".

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Ejects a tape.

Syntax command :DAT:EJECt

Explanation command Ejects a DAT tape.

Example :DAT:EJECT
Ejects a tape.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Prepares for recording on a tape.

Syntax command :DAT:READy
Prepares for recording on a tape.

Example DAT:READY
Prepares for recording on a tape.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Deletes all files on a tape.

Syntax command :DAT:ALLDelete

Explanation command Deletes all files recorded on a tape.

Example :DAT:ALLDELETE
Deletes all files.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Deletes the file specified and following files.

Syntax command :DAT:DELEte *NO*
NO <NR1>= file number

Explanation command Deletes the file specified and following files.

Example :DAT:DELETE 10
Deletes files after file number 10.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Formats a tape

Syntax command :DAT:FORMat?

Explanation command Formats a tape.

Example command :DAT:FORMAT
Formats a tape.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Rename a file

Syntax command :DAT:RENAme *NO, "NAME$"*
NO <NR1> = file number
NAME$ = file name

Explanation command Renames the specified file.

Example command :DAT:RENAME 2, '8845 FUNC'
Renames a file number 2 to "8845 FUNC".

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Queries the number of files saved on a tape.

Syntax

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

Explanation

query Returns the total number of files which are currently saved on the tape.

Example

query :DAT:FILE?
response :DAT:FILE 10 (When header on)

Deletes all files.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Replays from a tape.

Syntax

command :DAT:PLAY *NO,"S_TIME$","E_TIME$",MODE$,SPE$*
NO <NR1> = file number
S_TIME$ = start time (day-hour:min:sec)
E_TIME$ = stop time (day-hour:min:sec)
day: 000 to 999
hour: 00 to 23
min, sec: 00 to 59
MODE$ = (replay mode) SINGLE, REPEat
SPE$ = (speaker output channel) OFF, CH1 to CH16

Explanation

command Replays the voice data of the file designated by *NO.*
Only recorder waveform data can be designated.
When the start and stop time settings are both "000-00:00:00", all data is replayed.

Example

:DAT:PLAY 1,"000-00:00:00","001-10:10:10",SINGLE,CH1

Replays once the voice data for channel 1 from the head to one day 10 hours 10 minutes 10 seconds of the file number 1.

When allowed When the DAT control screen is displayed. (:DAT:MODE ON)

■ Sets and queries DATtoMEM

Syntax

command :DAT:DATTomem *A$,NO,"S_TIME$","E_TIME$"*
query :DAT:DATTomem?
response *A$*
A$ = OFF, ON
NO = file number
S_TIME$ = start time (day-hour:min:sec)
E_TIME$ = stop time (day-hour:min:sec)
day: 000 to 999
hour: 00 to 23
min, sec: 00 to 59

Explanation

command Sets DATtoMEM.
When the time axis is set to external, it is not possible to set the time.
query Returns the current setting for DATtoMEM as character data.

Example

:DAT:DATTOMEM ON,"000-10:20:30","000-11:21:31"

Sets the start time for DATtoMEM to 10 hours 20 minutes 30 seconds and the stop time to 11 hours 21 minutes 31 seconds

When allowed

When the DAT control screen is displayed. (:DAT:MODE ON)
When *A$* is set to OFF, all screen.

■ Sets and queries DATtoFFT

Syntax

command :DAT:DATTOFft *A$,NO,"S_TIME$","E_TIME$"*
query :DAT:DATTOFft?
response *A$*
A$ = OFF, ON
NO = file number
S_TIME$ = start time (day-hour:min:sec)
E_TIME$ = stop time (day-hour:min:sec)
day: 000 to 999
hour: 00 to 23
min, sec: 00 to 59

Explanation

command Sets DATtoFFT.
query Returns the current setting for DATtoFFT as character data.

Example

:DAT:DATTOFFT ON,1,'000-10:20:30','000-11:21:31'

Sets the start time for DATtoFFT to 10 hours 20 minutes 30 seconds and the stop time to 11 hours 21 minutes 31 seconds

When allowed

When the DAT control screen is displayed. (:DAT:MODE ON)
When *A$* is set to OFF, all screen.

6.3.13 MO Command (8846 only)

■ Enables and disables the MO mode.

Syntax

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

Explanation

command	Enables or disables MO mode.
query	Returns the current setting of MO mode.

Example :MO:MODE ON

Sets the MO mode to on.

When allowed In all functions

■ Queries information about a file saved on a tape.

Syntax

query	:MO:INFOr? *"NAME$"*
response	*"NAME$", A$, "DATE$", "TIME$"*
	NAME$ = file name
	A$ = type of information saved:
	W : measurement data
	F : conditions of creation
	A : waveform decision area
	N : no such file
	DATE$ = date of save "year-month-day"
	TIME$ = time of save "hour:minute:second"

Explanation

query	Returns information about the file whose name is specified in *NAME$*. If no such file exists, returns: "----", N, "-----", "-:-:-".

Example

query	:MO:INFOR? 'SAMPLE.MEM'
response	:MO:INFOR? "SAMPLE.MEM",W,"1997-11-29","11:59:12" (When header on)

Queries the file information of the file name, "SAMPLE. MEM".

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Loads a file from a MO.

Syntax command :MO:LOAD *"NAME$"* (,*"S_TIME$"*)
NAME$ = file name
S_TIME$ = start time (day-hour:min:sec)
day: 000 to 999
hour: 00 to 23
min, sec: 00 to 59

Explanation command Loads the data in the file numbered *NAME$*.
Only the data file in the recorder function, the start time can be specified.
When the start time is not specified, loads normally.

Example :MO:LOAD' DATA/SAMPLE.MEM
Loads the data of the SAMPLE.MEM file in the directory DATA.

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Saves a file on a MO.

Syntax command :MO:SAVE *"NAME$"*, *A$* (, *B$*)
In the memory recorder and FFT functions
NAME$ = file name (up to 8 characters)
A$ = type of saved information
W : measurement data
F : setting data
A : waveform decision area
B$ = method of saving
(none) : usual saving
ALL : all blocks saving during memory segmentation
In the recorder function
A$ = F

Explanation command Saves the information specified by *A$* on a MO.
The all channels of measurement data can be saved.
The data is saved in the current directory.

Example :MO:SAVE 'SAMPLE',W
Saves all channels of measurement data under the file name "SAMPLE" in the directory "DATA".

When allowed When the MO control screen is displayed. (:MO:MODE ON)

Make a directory.

Syntax command :MO:MKDIr *"NAME$"*
NAME$ = directory name (up to 8 characters)

Explanation command Make a directory.

Example :MO:MKDIR'DATA'
Make a directory DATA.

When allowed When the MO control screen is displayed. (:MO:MODE ON)

Ejects a MO.

Syntax command :MO:EJECt

Explanation command Ejects a MO disk.

Example :MO:EJECT
Ejects a MO.

When allowed When the MO control screen is displayed. (:MO:MODE ON)

Deletes the specified file .

Syntax command :MO:DELEte *"NAME$"*
NAME$ = file name

Explanation command Deletes the file saved on a MO disk.

Example :MO:DELETE'SAMPLE.MEM'
Deletes the file "SAMPLE.MEM".

When allowed When the MO control screen is displayed. (:MO:MODE ON)

Deletes the specified directory.

Syntax command :MO:RMDIr *"NAME$"*
NAME$ = directory name

Explanation command Deletes the specified file.

Example :MO:RMDIR 'DATA'
Deletes the directory "DATA".

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Formats a MO

Syntax command :MO:FORmat (*A$*)
A$ = P : physical format
(not specified): normal format

Explanation command Formats a MO.

Example command :MO:FORMAT
Formats a MO.

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Rename a file

Syntax command :MO:RENAme *"OLD$", "NEW$"*
OLD$ = file name before rename
NEW$ = file name after rename

Explanation command Renames the specified file.

Example command :MO:RENAME 'SAMPLE.MEM','TEST.MEM'
Renames a file "SAMPLE.MEM" to "TEST.MEM".

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Queries the number of files.

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

Explanation query Returns the total number of files in the current directory as numerical value.

Example query :MO:FILE?
response :MO:FILE 10 (When header on)
Queries the number of files in the current directory.

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Queries the number of directory.

Syntax
query :MO:DIR?
response *A* <NR1>
A = number of directory

Explanation
query Returns the number of directories <NR1> recognized in the current directory.

Example
query :MO:DIR?
response :MO:DIR 10 (When header on)
Queries the number of directory in the current directory.

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Move a directory.

Syntax
command :MO:CHDIr *"NAME$"*
NAME$ = path name

Explanation
command Moves the directory.

Example
command :MO:CHDIR 'DATA'
Moves to the directory "DATA" in the current directory.

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Replays from a MO.

Syntax
command :MO:PLAY *"NAME$","S_TIME$","E_TIME$",MODE$,SPE$*
NAME$ = file number
S_TIME$ = start time (day-hour:min:sec)
E_TIME$ = stop time (day-hour:min:sec)
day: 000 to 999
hour: 00 to 23
min, sec: 00 to 59
MODE$ = (replay mode) SINGLE, REPEat
SPE$ = (speaker output channel) OFF, CH1 to CH16

Explanation
command Replays the voice data of the file designated by *NAME$*.
Only recorder waveform data can be designated.
When the start and stop time settings are both "000-00:00:00", all data is replayed.

Example :MO:PLAY 'SAMPLE.REC','000-00:00:00',001-10:10:10',SINGLE,CH1
Replays once the voice data for channel 1 from the head to one day 10 hours 10 minutes 10 seconds of the file name "SAMPLE.REC".

When allowed When the MO control screen is displayed. (:MO:MODE ON)

■ Sets and queries FILETOMEM

Syntax

command :MO:FILETomem *A$,"NAME$","S_TIME$","E_TIME$"*
query :MO:FILETomem?
response *A$*
A$ = OFF, ON
NAME$ = file name
S_TIME$ = start time (day-hour:min:sec)
E_TIME$ = stop time (day-hour:min:sec)
day: 000 to 999
hour: 00 to 23
min, sec: 00 to 59

Explanation

command: Sets FILETOMEM.
When the time axis is set to external, it is not possible to set the time.
query: Returns the current setting for FILETOMEM as character data.

Example

:MO:FILETOMEM ON,'SAMPLE.MEM','000-10:20:30','000-11:21:31'

Sets the start time for FILETOMEM to 10 hours 20 minutes 30 seconds and the stop time to 11 hours 21 minutes 31 seconds

When allowed

When the MO control screen is displayed. (:MO:MODE ON)
When *A$* is set to OFF, all screen.

■ Sets and queries FILETOFFT

Syntax

command :MO:FILETOFft *A$,"NAME$","S_TIME$","E_TIME$"*
query :MO:FILETOFft?
response *A$*
A$ = OFF, ON
NAME$ = file name
S_TIME$ = start time (day-hour:min:sec)
E_TIME$ = stop time (day-hour:min:sec)
day: 000 to 999
hour: 00 to 23
min, sec: 00 to 59

Explanation

command: Sets FILETOFFT.
query: Returns the current setting for FILETOFFT as character data.

Example

:MO:FILETOFFT ON,'SAMPLE.MEM','000-10:20:30','000-11:21:31'

Sets the start time for FILETOFFT to 10 hours 20 minutes 30 seconds and the stop time to 11 hours 21 minutes 31 seconds

When allowed

When the MO control screen is displayed. (:MO:MODE ON)
When *A$* is set to OFF, all screen.

6.3.14 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 Single format) and the FFT function (single, Nyquist format).

■ Draws a line.

Syntax

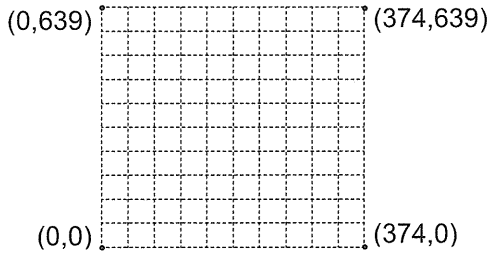
command :GRAPh: LINE *X1, Y1, X2, Y2*

X1, *X2* = x-coordinates

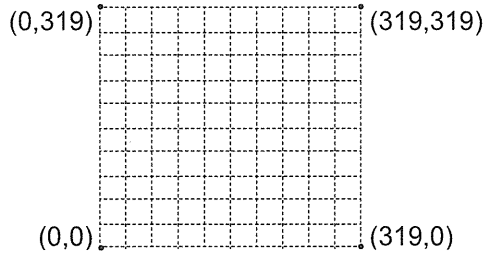
Y1, *Y2* = y-coordinates

Explanation

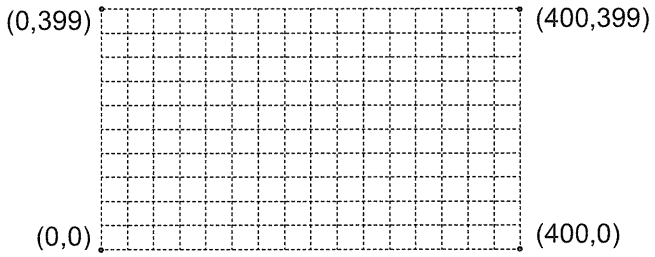
command Draws a line from (*X1*, *Y1*) to (*X2*, *Y2*).



In the memory recorder function (when Single format)



In the memory recorder function (when XYsingle format)



FFT

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 editor mode.

■ Parallel movement

Syntax command :GRAPh: PARAllel *high, low, right, left*
high, low, right, left = 0 to 20.00 (div)

Explanation command Carries out a parallel movement of the drawing.
The *high* and *low* parameters and the *right* and *left* parameters are set in units of 0.05 steps.

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

■ Paints the drawing.

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 editor mode.

■ Erases the line.

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 editor mode.

■ 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 editor mode.

■ Reverses the video of the drawing.

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 editor mode.

■ Clears all drawing.

Syntax command :GRAPh:ALLClear

Explanation command Clears the entire drawing.

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

■ Clears drawing.

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.

■ Undoes the drawing.

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, 1

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 editor mode.

Chapter 7
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 the main unit.
Line 160-170	Send interface clear, and switch to remote mode.
Line 180	Select memory recorder function.
Line 190	Time/division is 1 ms.
Line 200	Recording length is 25 divisions.
Line 210	Enter measurement operation mode.
Line 220	End remote mode.

```
100 ' ----------------------------------------
110 ' 8845 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 CALL SENDIFC( BOAD% )                       'Clear interface
170 CALL ENABLEREMOTE( BOAD%, ADRS%(0) )        'Enable remote
180 GOSUB 270                                   'Function MEM
190 GOSUB 270                                   'Time/Div 1ms
200 GOSUB 270                                   '25DIV
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 +1.E-3"
350 DATA ":CONFIGURE:SHOT 25"
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 8845 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 8845.
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 ' 8845 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 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 8845 status byte, then carries out appropriated 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 off 8845.
Line 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 graph. (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 ' 8845 Sample program No.3
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$, NLEND% )     'Set FUNCTION
260 I% = 0
270 AVR$=":DISPLAY:GRAPH CH1,"+STR$(I%)
280 CALL SEND( BOAD%, ADRS%(0), AVR$, NLEND% )     'Set GRAPH
290 I% = I% + 1: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 one 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 20 using the :VDATA? query.
Note: Outputting data in bigger sets reduces the overall processing time.

Read data (2500 samples) for channel 1when stored with a 25-division recording.

Line 170	Set ADRS%(0) to address off 8845.
Line 180-190	Send interface clear, and switch to remote mode.
Line 210	Set memory recorder function and 25-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 fromchannel 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 ' 8845 Sample program No.4
120 ' You must merge this code with DECL.BAS
130 ' ----------------------------------------
140 DIM D( 2501 )
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,25DIV
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% <> 2500 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$,(12*II%+1),11) )
360 NEXT II%
370 NEXT I%
380 GOSUB 470
390 GOSUB 530:D( 2500 ) = VAL( READING$ )           'Last Data
400 FOR I% = 0 TO 2500: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 RETURN
560 ' ----------------------------------------
570 ' data table
580 ' ----------------------------------------
590 DATA ":ESE0 2"
600 DATA ":FUNCTION MEM;:CONFIGURE:SHOT 25"
610 DATA ":TRIGGER:MODE SINGLE"
620 DATA ":START"
630 DATA ":HEADER OFF;:MEMORY:MAXPOINT?"
640 DATA ":MEMORY:POINT CH1,0"
650 DATA ":MEMORY:VDATA? 1"
```

Example 5 Inputting storage data (when using the 8927)

(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.

With the unit storing with a 25-division recording length, write sine wave data into memory for channel 1.

Line 190	Set ADRS%(0) to address of 8845. 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.
Line 280-320	Write the sine wave.
Line 340-350	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8845 Sample program No.5
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% <> 2500 THEN 340
270 CALL SEND( BOAD%, ADRS%(0), PNT$, NLEND% )   'Set point CH1,0
280 FOR I% = 0 TO MAX%
290 VOLT% = 1500 * SIN( 3.14 * I% / 500 ) + 8192
300 SND$ = 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 8845.
Lines 170 - 180 Send interface clear, and switch to remote mode.
Lines 200-310 Set the 8845 function, trigger conditions, etc.
Line 330 Enter measurement operation mode with the conditions set.
Line 360 End remote mode.

```
100 ' ------------------------------------
110 ' 8845 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 '
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 25DIV
230 '
240 GOSUB 410                                   'Trigger source OR
250 GOSUB 410                                   'LEVEL trigger
260 GOSUB 410                                   'Pre-trigger 5%
270 GOSUB 410                                   'LEVEL 2mV
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 25"
500 DATA ":TRIGGER:SOURCE OR"
510 DATA ":TRIGGER:KIND CH1,LEVEL"
520 DATA ":TRIGGER:PRETRIG 5"
530 DATA ":TRIGGER:LEVEL CH1,2.E-3"
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 8845.
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 ' 8845 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 '
190 GOSUB 520                                   'Enable SESER bit
200 GOSUB 520                                   'TIME/DIV 1ms,SHOT 25DIV
210 GOSUB 520                                   'Trigger source OR
220 GOSUB 520                                   'LEVEL trigger CH1,CH2
230 GOSUB 520                                   'Trigger OFF CH3,CH4
240 GOSUB 520                                   'Trigger CH1,1mV,UP
250 GOSUB 520                                   'Trigger CH2,1mV,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 25"
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 CH1,1.E-3;SLOPE CH1,UP"
710 DATA ":TRIGGER:LEVEL CH2,1.E-3;SLOPE CH2,UP"
720 DATA ":TRIGGER:MODE SINGLE"
730 DATA ":START"
740 DATA ":ABORT"
```

Example 8 Checking the presence of input unit, and displaying input ranges on the screen. (when using with the 8927 only).

Line 1060 Set ADRS%(0) to address of 8845
Lines 1070-1080 Send interface clear, and switch to remote mode.
Line 1260 Disable headers.
Lines 1270-1500 Screen display.
Lines 1520-1670 Read the presence of the input unit into variables CH1 to CH4.
Lines 1690-1900 Display the presence of the input unit on the screen.
Lines 1940-1950 Set screen display

```
1000 ' ---------------------------------------
1010 ' 8845 Sample program No.8
1020 ' You must merge this code with DECL.BAS
1030 ' ---------------------------------------
1040 SCREEN 9
1050 BOAD% = 0
1060 ADRS%(0) = 5:ADRS%(1) = NOADDR%                'GP-IB Address = 5
1070 SP% = 2                                        'Set CONST
1080 CALL SENDIFC( BOAD% )                          'Clear interface
1090 CALL ENABLEREMOTE( BOAD%, ADRS%(0) )           'Enable remote
1200 HEA$ = ":HEADER OFF"
1210 OPT$ = "*OPT?"
1220 CH1$ = ":MEMORY:AREAL? CH1"
1230 CH2$ = ":MEMORY:AREAL? CH2"
1240 CH3$ = ":MEMORY:AREAL? CH3"
1250 CH4$ = ":MEMORY:AREAL? CH4"
1260 CALL SEND( BOAD%, ADRS%(0), HEA$, NLEND% )     'Header OFF
1270 CLS
1280 LOCATE 3,5:PRINT "<LEVEL MONITOR>"
1290 LOCATE 4,1:PRINT "100"
1300 LOCATE 13,1:PRINT " 50"
1310 LOCATE 22,1:PRINT "  0"
1320 LOCATE 1,52:PRINT "CH1            CH2"
1330 LOCATE 2,52:PRINT "CH3            CH4"
1340 '
1350 CALL SEND( BOAD%, ADRS%(0), OPT$, NLEND% )     'Uint?
1360 UNIT$ = SPACE$( 50 )
1370 CALL RECEIVE( BOAD%, ADRS%(0), UNIT$, STOPEND% )
1380 UNI1% = VAL( MID$(UNIT$, 1, 1) )
1390 UNI2% = VAL( MID$(UNIT$, 3, 1) )
1400 UNI3% = VAL( MID$(UNIT$, 5, 1) )
1410 UNI4% = VAL( MID$(UNIT$, 7, 1) )
1420 IF UNI1% = 0 THEN LOCATE 1,57:PRINT "Nothing"
1430 IF UNI2% = 0 THEN LOCATE 1,72:PRINT "Nothing"
1440 IF UNI3% = 0 THEN LOCATE 2,57:PRINT "Nothing"
1450 IF UNI4% = 0 THEN LOCATE 2,72:PRINT "Nothing"
1460 '
1470 LINE (30,57)-(620,307),7,B,&HCCCC              'Frame
1480 FOR Y% = 82 TO 282 STEP 25
1490 LINE (30,Y%)-(620,Y%),7,,&H1010
1500 NEXT Y%
1510 '
1520 IF UNI1% = 0 THEN 1550
1530 LINE (440,8)-(490,10),6,B
1540 CALL SEND( BOAD%, ADRS%(0), CH1$, NLEND% )     'CH1 ADATA
1550 GOSUB 2000:Y10% = ADT% / 64
1560 IF UNI2% = 0 THEN 1590
```

```
1570 LINE (560,8)-(610,10),5,B
1580 CALL SEND( BOAD%,ADRS%(0),CH2$,NLEND% )        'CH2 ADATA
1590 GOSUB 2000:Y20% = ADT% / 64
1600 IF UNI3% = 0 THEN 1630
1610 LINE (440,24)-(490,26),4,B
1620 CALL SEND( BOAD%,ADRS%(0),CH3$,NLEND% )        'CH3 ADATA
1630 GOSUB 2000:Y30% = ADT% / 64
1640 IF UNI4% = 0 THEN 1680
1650 LINE (560,24)-(610,26),3,B
1660 CALL SEND( BOAD%,ADRS%(0),CH4$,NLEND% )        'CH4 ADATA
1670 GOSUB 2000:Y40% = ADT% / 64
1680 '
1690 FOR X% = 30 TO 620-SP STEP SP%
1700  IF UNI1% = 0 THEN 1750
1710  CALL SEND( BOAD%,ADRS%(0),CH1$,NLEND% )       'CH1 ADATA
1720  GOSUB 2000:Y11% = ADT% / 64
1730  LINE (X%,307-Y10%)-(X%+SP%,307-Y11%),6
1740  Y10% = Y11%
1750  IF UNI2% = 0 THEN 1800
1760  CALL SEND( BOAD%,ADRS%(0),CH2$,NLEND% )       'CH2 ADATA
1770  GOSUB 2000:Y21% = ADT% / 64
1780  LINE (X%,307-Y20%)-(X%+SP%,307-Y21%),5
1790  Y20% = Y21%
1800  IF UNI3% = 0 THEN 1850
1810  CALL SEND( BOAD%,ADRS%(0),CH3$,NLEND% )       'CH3 ADATA
1820  GOSUB 2000:Y31% = ADT% / 64
1830  LINE (X%,307-Y30%)-(X%+SP%,307-Y31%),4
1840  Y30% = Y31%
1850  IF UNI4% = 0 THEN 1900
1860  CALL SEND( BOAD%,ADRS%(0),CH4$,NLEND% )       'CH4 ADATA
1870  GOSUB 2000:Y41% = ADT% / 64
1880  LINE (X%,307-Y40%)-(X%+SP%,307-Y41%),3
1890  Y40% = Y41%
1900 NEXT X%
1910 '
1920 IF INKEY$ = "" GOTO 1270
1930 SCREEN 0
1940 UNT$ = CHR$( UNT% ):CALL IBCMD( BOAD%,UNT$ )   'UN TALK
1950 CALL ENABLELOCAL( BOAD%,ADRS%(0) )              'Enable operations
1960 END
1970 ' ---------------------------------------
1980 ' Receive data
1990 ' ---------------------------------------
2000 READING$ = SPACE$(32)
2010 CALL RECEIVE( BOAD%,ADRS%(0),READING$,STOPEND% )
2030 ADT% = VAL(READING$)
2040 RETURN
```

Example 9 Saving stored data onto drive 2 (sequential file)

Line 150	Set ADRS%(0) to address of 8845.
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.
Line 420-460	Read the stored data form the 8845, and save sequentially.
Line 530-540	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8845 Sample program No.9
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$, NLEND% )        'Header OFF
260 GOSUB 590:MAX% = VALUE%                           'Max point?
270 IF MAX% <> 0 THEN 300                             'Output ready?
280 PRINT "No storage data !!"
290 GOTO  520
300 '
310 PRINT " Max point=";MAX%:PRINT
320 INPUT " Channel(CH1-CH16)";CH$                    '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 8845.

Line 150	Set ADRS%(0) to address of 8845.
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 8845.
Lines 500-510	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8845 Sample program No.10
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-CH16)";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 8845.
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 bit5(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 240-30	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 ' 8845 Sample program No.11
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 25DIV
270 '
280 GOSUB 450                                   'CH1 <- LEVEL TRIG.
290 GOSUB 450                                   'Pre-TRIG. 5%
300 GOSUB 450                                   'LEVEL 0%,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 25"
570 DATA ":TRIGGER:KIND CH1,LEVEL"
580 DATA ":TRIGGER:PRETRIG 5"
590 DATA ":TRIG:LEVEL CH1,0;SLOPE CH1,UP;*OPC"
600 DATA ":START"
```

Example 12 Using service requests to display errors

Line 150	Set ADRS%(0) to address of 8845.
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 graph. (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 450-460	Release talker and remote mode.

```
100 ' ---------------------------------------
110 ' 8845 Sample program No.12
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$, NLEND% )     'Set FUNCTION
270 I% = 5
280 AVR$ = ":DISPLAY:GRAPH CH1,"+STR$( I% )
290 CALL SEND( BOAD%, ADRS%(0), AVR$, NLEND% )     'Set GRAPH
300 I% = I% + 1: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
```

Example 13 Outputting stored data (binary 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 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 one only.

(3) After converting the binary data to voltage value obtained by :MEMORY:BDATA?, it is output on the screen. The number of data samples which may be output in one set is 1 to 125 using :BDATA? .

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

Line 150	Set ADRS%(5) to address of 8845.
Lines 160-170	Send interface clear, and switch to remote mode.
Lines 200-240	Read the number of stored data values into MAX%.
Lines 300-340	Queries the range for channel 1 to obtain voltage value by binary data.
Line 350-360	Set the output data to be from channel 1, point to 0.
Line 400-420	Set the size of output data set to be 125 samples, andread as binary data.
Line 440-470	Convert the binary data to voltage values.
Lines 550-560	Release talker and remote mode.

The calculation for line 470 is fo the 8927 unit. For other calculations, refer to Section 6.3.7, "Input/Output of Stored Data".

```
100 ' ---------------------------------------
110 ' 8845 Sample program No.13
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 ERROR GOTO 530
190 '
200 COMMAND$ = ":MEMORY:MAXPOINT?"                 'Max point?
210 READING$ = SPACE$( 30 )
220 CALL SEND( BOAD%,ADRS%(0),COMMAND$,NLEND% )
230 CALL RECEIVE( BOAD%,ADRS%(0),READING$,STOPEND% )
240 MAX% = VAL( READING$ )
250 PRINT "Storage Data Point = ",MAX%
260 IF MAX% <> 0 THEN 300
270 PRINT "No Storage Data"
280 GOTO 550
290 '
300 COMMAND$ = ":UNIT:RANGE? CH1"                  'Range?
310 READING$ = SPACE$( 30 )
320 CALL SEND( BOAD%,ADRS%(0),COMMAND$,NLEND% )
330 CALL RECEIVE( BOAD%,ADRS%(0),READING$,STOPEND% )
340 RANG% = VAL( MID$(READING$,5,LEN(READING$)-4 ) )
341 PRINT "RANGE = ",RANG%
350 COMMAND$ = ":MEMORY:POINT CH1, 0"              'Set output point
360 CALL SEND( BOAD%,ADRS%(0),COMMAND$,NLEND% )
370 N = MAX% / 125
380 COMMAND$ = ":MEMORY:BDATA? 125"
390 FOR J = 1 TO N
400  CALL SEND( BOAD%,ADRS%(0),COMMAND$,NLEND% )
410  ANS$ = SPACE$( 255 )
420  CALL RECEIVE( BOAD%,ADRS%(0),ANS$,STOPEND% ) 'Get BDATA
430  FOR I = 1 TO 125
440   A% = ASC( MID$( ANS$, 2*I+1, 1 ) )
450   B% = ASC( MID$( ANS$, 2*I+2, 1 ) )
460   D% = A%*256 + B%
470   E% = ( D% - 8192 ) * RANG% / 320
480   PRINT E%
490  NEXT I
500 NEXT J
510 GOTO 550
520 '
530 PRINT "ERROR !!"                               'error
540 '
550 UNT$ = CHR$( UNT% ):CALL IBCMD( BOAD%,UNT$ )   'UN TALK
560 CALL ENABLELOCAL( BOAD%,ADRS%(0) )             'Enable operations
570 END
```

Example 14 Saving stored data onto tape. (in the memory recorder function)

Line 150	Set ADRS%(0) to address of 8845.
Lines 160-170	Send interface clear, and switch to remote mode.
Line 180	Set jump address for service request.
Line 190	Enable bit 5 (ESB) of the status byte by the service request enable register.
Line 200	Enable bits 2, 3, 4, and 5 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 240-250	Disable headers, and read the number of stored data values into MAX%.
Lines 310-320	Set DAT mode to on to prepare recording.
Line 330	Set the file name and item to be saved and save on to tape.
Line 580	Serial polling to read the status byte.
Line 620	Read the standard event status register.
Line 660	Disable service request interrupt.
Line 670-680	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8845 Sample program No. 14
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 580
190 GOSUB 730                                          'Mask SRQ
200 GOSUB 730                                          'Mask SESER
210 GOSUB 730                                          'Clear statusbyte
220 PEN ON
230 '
240 GOSUB 730                                          'Header off
250 GOSUB 790
260 MAX% = VAL( ANS$ )                                 'Get max point
270 IF MAX% <> 0 THEN 310
280 PRINT "No Storage Data"
290 GOTO 660
300 '
310 GOSUB 730                                          'DAT mode on
320 GOSUB 730                                          'Ready
330 GOSUB 730                                          'Save
530 GOTO 660
540 '
550 ' ------------------------------------
560 ' Service request operation
570 ' ------------------------------------
580 CALL IBRSP( ADRS%, S% )
590 DCL$ = CHR$( DCL% ):CALL IBCMD( BOAD%, DCL$ )      'Clear buffer
600 ESR$ = "*ESR?"
610 CALL SEND( BOAD%, ADRS%(0), ESR$, STOPEND% )       'ESR?
620 GOSUB 760
630 B = VAL( ANS$ )
640 PRINT "*ESR = ", B
650 '
```

```
660 PEN OFF
670 UNT$ = CHR$( UNT% ):CALL IBCMD( BOAD%, UNT$ )        'UN TALK
680 CALL ENABLELOCAL( BOAD%, ADRS%(0) )                  'Enable operations
690 END
700 ' ---------------------------------------
710 ' Send data
720 ' ---------------------------------------
730  READ COMMAND$
740  CALL SEND( BOAD%, ADRS%(0), COMMAND$, NLEND% )
750  RETURN
760 ' ---------------------------------------
770 ' Receive data
780 ' ---------------------------------------
790  READING$ = SPACE$( 30 )
800  CALL RECEIVE( BOAD%, ADRS%(0), READING$, STOPEND% )
810  ANS$ = READING$
820  RETURN
830 ' ---------------------------------------
840 ' DATA table
850 ' ---------------------------------------
860 DATA "*SRE 32"
870 DATA "*ESE 60"
880 DATA "*CLS"
890 DATA ":HEADER OFF;:MEMORY:MAXPOINT?"
900 DATA ":DAT:MODE ON"
910 DATA ":DAT:READY"
920 DATA ":DAT:SAVE '8845 DATA',W"
```

Example 15 Loading data saved on tape to the 8845.

Line 150	Set ADRS%(0) to address of 8845.
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.
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 240	Set DAT mode to on.
Lines 250-270	Read the number of stored data values into VALUE%.
Line 300	Load the file specified to 8845.
Lines 360	Serial polling to read the status byte.
Line 400	Read the standard event status regiser
Line 440	Disable service request interrupt.
Line 450-460	Release talker and remote mode.

```
100 ' ------------------------------------
110 ' 8845 Sample program No. 15
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 360
190 GOSUB 520                                    'Mask SRQ
200 GOSUB 520                                    'Mask SESER
210 GOSUB 520                                    'Clear statusbyte
220 PEN ON
230 '
240 GOSUB 520                                    'DAT mode on
250 GOSUB 520                                    'File?
260 GOSUB 580
270 NO% = VAL( ANS$ )
280 IF NO% = 0 THEN PRINT "No File !!":GOTO 440
290 COMMAND$ = ":DAT:LOAD "+STR$( NO% )
300 CALL SEND( BOAD%, ADRS%(0), COMMAND$, NLEND% ) 'Load
310 GOTO 440
320 '
330 ' ------------------------------------
340 ' Service request operation
350 ' ------------------------------------
360 CALL IBRSP( ADRS%, S% )
370 DCL$ = CHR$( DCL% ):CALL IBCMD( BOAD%, DCL$ ) 'Clear buffer
380 ESR$ = "*ESR?"
390 CALL SEND( BOAD%, ADRS%(0), ESR$, STOPEND% )    'ESR?
400 GOSUB *RECEIVE
410 B = VAL( ANS$ )
420 PRINT "*ESR = ", B
430 '
440 PEN OFF
```

```
450 UNT$ = CHR$( UNT% ):CALL IBCMD( BOAD%, UNT$ ) 'UN TALK
460 CALL ENABLELOCAL( BOAD%, ADRS%(0) )            'Enable operations
470 END
480 '
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$( 30 )
590 CALL RECEIVE( BOAD%, ADRS%(0), READING$, STOPEND% )
600 ANS$ = READING$
610 RETURN
620 ' ---------------------------------------
630 ' DATA table
640 ' ---------------------------------------
650 DATA "*SRE 32"
660 DATA "*ESE 60"
670 DATA "*CLS"
680 DATA ":DAT:MODE ON"
690 DATA ":DAT:FILE?"
```

Example 16 Saving the data in memory on a MO disk (8846: MEM function)

Line 150	Set ADRS%(0) to address of 8845.
Line 190	Set the header off and read the number of stored data values into MX.
Lines 260-280	Set MO mode to on and make a directory. Move to the directory.
Line 290	Set the file name and save item to save on MO.
Line 310-320	Release talker and remote mode.

```
100 ' ---------------------------------------
110 ' 8846 Sample program No.16
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 380                                          'Header off
200 GOSUB 440
210 MAX% = VAL( ANS$ )                                 'Get max point
220 IF MAX% <> 0 THEN 260
230 PRINT "No Storage Data"
240 GOTO 310
250 '
260 GOSUB 380                                          'MO mode on
270 GOSUB 380                                          'Make dir
280 GOSUB 380                                          'Cange dir
290 GOSUB 380                                          'Save
300 '
310 UNT$ = CHR$( UNT% ):CALL IBCMD( BOAD%, UNT$ )      'UN TALK
320 CALL ENABLELOCAL( BOAD%, ADRS%(0) )                'Enable operations
330 END
340
350 ' ---------------------------------------
360 ' Send data
370 ' ---------------------------------------
380  READ COMMAND$
390  CALL SEND( BOAD%, ADRS%(0), COMMAND$, NLEND% )
400  RETURN
410 ' ---------------------------------------
420 ' Receive data
430 ' ---------------------------------------
440  READING$ = SPACE$( 30 )
450  CALL RECEIVE( BOAD%, ADRS%(0), READING$, STOPEND% )
460  ANS$ = READING$
470  RETURN
480 ' ---------------------------------------
490 ' DATA table
500 ' ---------------------------------------
510 DATA ":HEADER OFF;:MEMORY:MAXPOINT?"
520 DATA ":MO:MODE ON"
530 DATA ":MO:MKDIR 'DATA'"
540 DATA ":MO:CHDIR 'DATA'"
550 DATA ":MO:SAVE 'SAMPLE',W"
```

Example 17 Loading the data stored on MO in example 16

Line 150 Set ADRS%(0) to address of 8845.
Lines 190-200 Set MO mode to on and move to the directory of file to be loaded.
Line 210 Set the file name and load to teh 8846.
Line 230-240 Release talker and remote mode.

```
100 ' ----------------------------------------
110 ' 8846 Sample program No.17
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                                    'MO mode on
200 GOSUB 520                                    'Change dir
210 GOSUB 520                                    'Load
220 '
230 UNT$ = CHR$( UNT% ):CALL IBCMD( BOAD%, UNT$ ) 'UN TALK
240 CALL ENABLELOCAL( BOAD%, ADRS%(0) )           'Enable operations
250 END
260 '
270 ' ----------------------------------------
280 ' Send data
290 ' ----------------------------------------
300 READ COMMAND$
310 CALL SEND( BOAD%, ADRS%(0), COMMAND$, NLEND% )
320 RETURN
330 ' ----------------------------------------
340 ' DATA table
350 ' ----------------------------------------
360 DATA ":MO:MODE ON"
370 DATA ":MO:CHDIR 'DATA'"
380 DATA ":MO:LOAD 'SAMPLE.MEM'"
```

Chapter 8
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, :MEMORY:RECPOINT, 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 8845, 8846 have an input buffer of 1024 bytes capacity. If the data accumulated in this buffer exceeds 1024 bytes the buffer full, and until a space again becomes available in the buffer, the IEEE 488.1 bus goes into the waiting state.

(b) Queries to which multiple response message units are returned

There are no queries to return multiple response messages.

(c) Queries producing responses as syntax checking is performed

On the 8845, 8846, 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 6.2, "Standard Commands Stipulated by IEEE 488.2", and Section 6.3, "Commands Specific to the 8845, 8846."

(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 Chapter 5, "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 6.2, "Standard Commands Stipulated by IEEE 488.2."

(17) Capacity of the user data storage area reserved for when the "*PUD" command and the "*PUD?" query are being executed

The "*PUD" command and the "*PUD?" query are not used. Further, there is no user data storage area.

(18) Resources when the "*RDT" command and the "*RDT?" query are being used

The "*RDT" command and the "*RDT?" query are not used.

(19) Conditions which are influenced when "*RST", "*LRN?", "*RCL", and "*SAV" are used

"*LRN?", "*RCL", and "*SAV" are not used. The "*RST" command returns the 8845, 8846 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 4.6, "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 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 unit keys stop working after using GP-IB communications.	Press the [LOCAL] function key to end the remote operating state.
	Has an LLO (local lock-out) command been sent to the unit? Send a GTL command to return to the local state.
An attempt to read data using the CALL RECEIVE statement causes the GP-IB bus to hang.	Each and every CALL RECEIVE statement must be preceded by a query.
	Is the query transmitted incorrect?
Although a command was transmitted, the unit did not operate.	Use the "*ESR?" query to check the standard event status register for anomalies.
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

HIOKI 9537 GP-IB INTERFACE

Instruction Manual

Publication date: November 1998 Revised edition 4

Edited and published by HIOKI E.E. CORPORATION
Technical Sales Support Section

All inquiries to Sales and Marketing International Department
81 Koizumi, Ueda, Nagano, 386-1192, Japan
FAX: +81-268-28-0568 TEL: +81-268-28-0562
E-mail: os-com@hioki.co.jp

Printed in Japan 9537A981-04

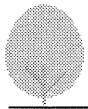
HIOKI

HIOKI E. E. CORPORATION

HEAD OFFICE
81 Koizumi, Ueda, Nagano 386-1192, Japan
TEL +81-268-28-0562 / FAX +81-268-28-0568
E-mail: os-com@hioki.co.jp

HIOKI USA CORPORATION
6 Corporate Drive, Cranbury, NJ 08512, USA
TEL +1-609-409-9109 / FAX +1-609-409-9108

9537A981-04 98-11-0001H



Printed on recycled paper