HIOKI

INSTRUCTION MANUAL

3502

C HiTESTER

9593

RS-232C INTERFACE

HIOKI E.E. CORPORATION

Contents

Introduction

Thank you for purchasing this Hioki 9593 RS-232C interface for the 3502 Capacitance HiTester.

To get the maximum performance from this unit, please read this manual first, and keep this at hand.

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.

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

⚠ 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.

Chapter 1
Before Use

1.1 Check of External Appearance and Accessories

When you receive this product, before use, please check that no abnormality or damage has occurred during delivery.

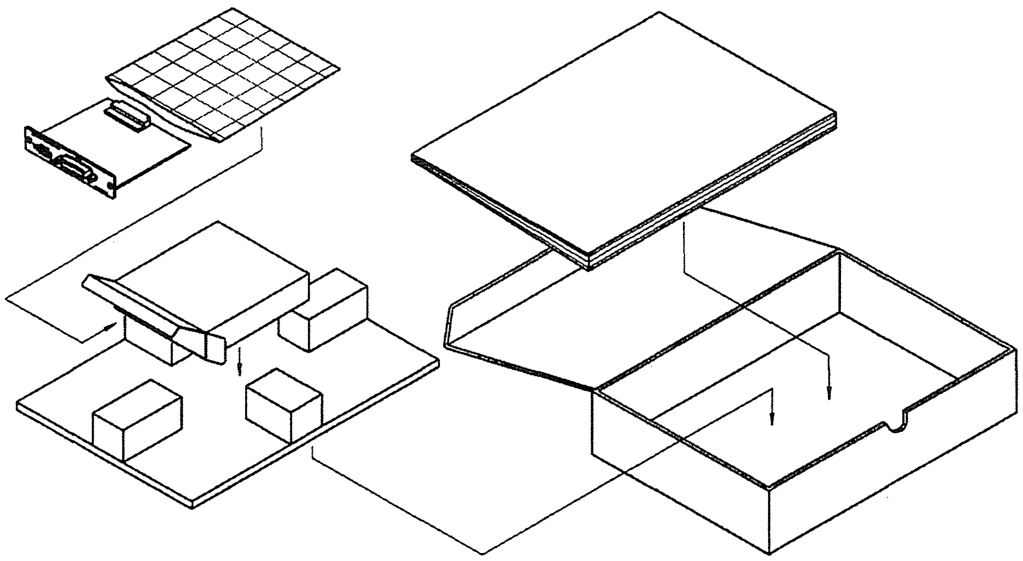
In the unlikely event of any damage, or if the unit does not function according to specification, contact your nearest Hioki service representative without delay.

(1) 9593 RS-232C interface

(2) This instruction manual

1.2 Shipping Precautions

If reshipping the unit, preferably use the original packing.



If this is not available, use the following procedure.

(1) Wrap the unit in plastic sheeting.

(2) After wrapping cushioning material around the unit, pack it into a cardboard box, and then seal up the box with adhesive tape.

1.3 Points for Attention During Use

(1) If you change the communication conditions of the 3502 while using it, you should immediately turn the power off and on again. If you do not do so, the communication conditions will not be changed to the new one.

(2) Always be sure to secure the RS-232C cable to the 9593 unit by tightening up the fixing screws provided.

(3) Program messages sent just after the power has been turned on are executed after the self test has terminated.

(4) It is vital that the proper data format is used when inputting commands with data values to the 3502 unit.

(5) The values given for command processing times are the values measured when the long form of the command and the designated data format are used, and headers for response messages are enabled.

(6) Commands specific to the 3502 unit are all sequential commands.

(7) For details of the various functions, refer to the instruction manual for the 3502 unit.

1.4 Installing the RS-232C Interface

⚠ WARNING

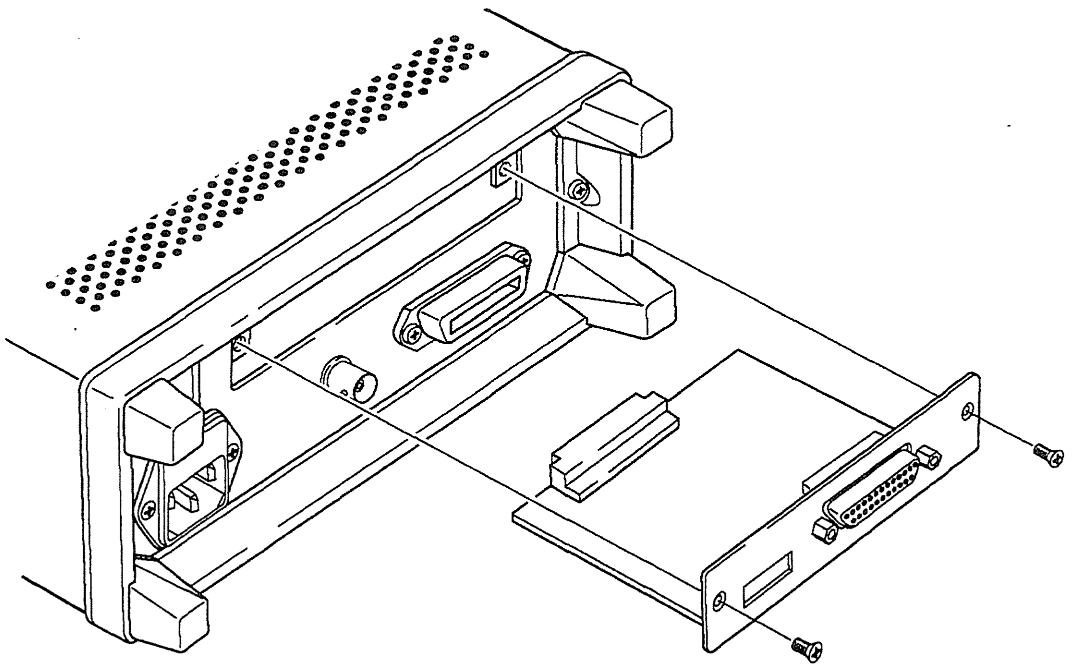
- **When installing the 9593 RS-232C interface unit in the 3502 Capacitance HiTester, be absolutely sure that the power cable and the connectors to the 3502 have been removed first, in order to prevent electric shock to the operator and also damage to the unit.**
- **To prevent electric shock to the operator, never use the 3502 without either the RS-232C interface or the blanking plate in place.**

The space for fitting the 9593 RS-232C interface in the rear panel of the 3502 is covered with a blanking plate. Follow these three steps to install the 9593 interface:

(1) Remove the fixing screws, and take off the blanking plate.

(2) Insert the 9593 RS-232C interface into the exposed slot in the rear of the 3502 in the figure below.

(3) Push the 9593 firmly into place, and fix with the screws removed in step (1).



Chapter 2 Overview

2.1 Introduction to the 9593 RS-232C Interface

By connecting the 9593 RS-232C interface to the 3502 Capacitance HiTester, it is possible to control all the functions of the 3502 (except for powering on and off) via the RS-232C bus.

2.2 Features

(1) All of the functions of the 3502 main unit, except for powering on and off, can be controlled via the RS-232C interface.

(2) When the comparator function is being used, the "D" setting can be made to an accuracy of five decimal places (only when averaging is being performed).

(3) The beeper sound can be turned on and off.

(4) The unit can be reset.

2.3 Specifications

(1) Transfer system	Start-stop synchronization
(2) Baud rate	1200, 2400, 4800, 9600 bps
(3) Data length	7 or 8 bits
(4) Parity	Even, Odd, or None
(5) Stop bits	1 or 2 bits
(6) Delimiter	CR+LF, LF
(7) Handshake	XON/XOFF, Hardware

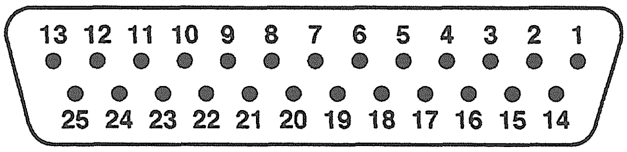
Selected by DIP switch on (1) to (7).

(8) Electrical characteristic

Input voltage levels	+5 V to +15 V -15 V to -5 V	ON OFF
Output voltage levels (load impedance 3 kΩ to 7 kΩ)	+5 V to +9 V -9 V to -5 V	ON OFF

(9) Connector

- RS-232C Interface connector pin assignments
  (D-subminiature 25-pin female)



NOTE The connector on the 9593 is for terminal (DTE).

- Signal assignments and explanation

Connector (Dsub) Pin number	Circuit		Description
	RS-232C	CCITT	
1	AA(FG)	101	Protective Ground
2	BA(TxD)	103	Transmitted Data
3	BB(RxD)	104	Received Data
4	CA(RTS)	105	Request to Send
5	CB(CTS)	106	Clear to Send
7	AB(GND)	102	Signal Ground
20	CD(DTR)	108/2	Data Terminal Ready
Other pins			Unused

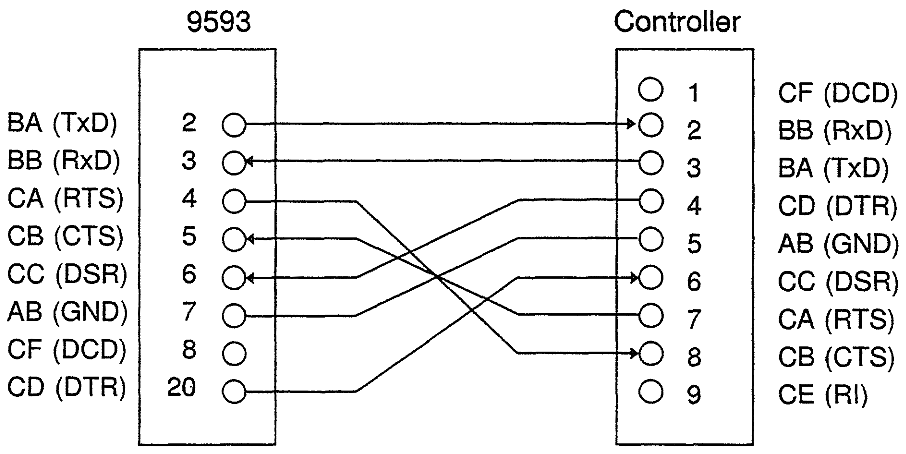
NOTE

- Basically all communications should use the EXT trigger.
  If communications occur during data sampling, the data sampling is repeated.
  Therefore with the INT trigger, there is basically constant sampling, and as a result there may be a delay until correct data can be obtained.
- If the output queue becomes full a query error occurs, and the output queue is cleared. When the controller handshake setting is not the same as the 9593, if many commands are sent and the input buffer becomes full, following read data will be ignored.

(10) Connecting method

When connecting to the controller (DTE), use a cross cable which meets the connector specifications of both sides of the 9593 and the controller.

Example When connecting to the controller using a D-subminiature 9-pin connector.



Specification	D-subminiature 25-pin male to D-subminiature 9-pin female connectors, with "crossed" data connections

(11) Handshake

Buffer flow control

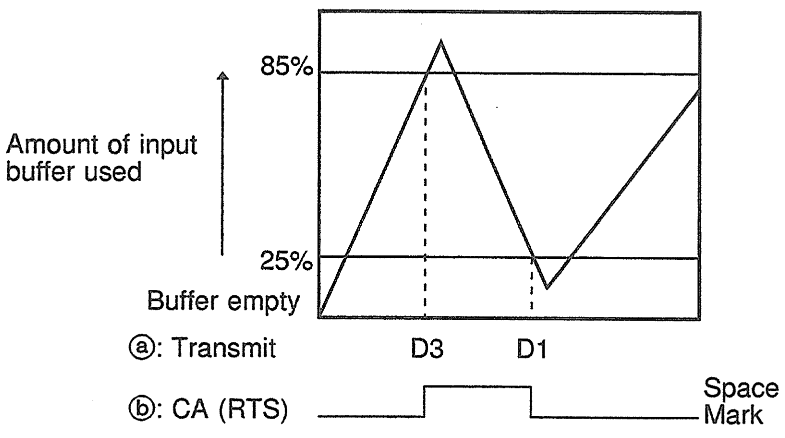
● Controls when receiving

When the receiving buffer is more than 85 % full, to indicate to the controller that the empty buffer capacity is low:

ⓐ Using XON/OFF control	D3 (13H) is transmitted.
ⓑ Using hardware handshake	CA(RTS) is set to Space.

Processing of data in the buffer continues, and when the receiving buffer is less than 25 % full, to indicate to the controller that there is ample buffer capacity:

ⓐ Using XON/OFF control	D1 (11H) is transmitted.
ⓑ Using hardware handshake	CA(RTS) is set to Mark.



● Controls when transmitting

ⓐ Using XON/OFF control

- When a D3 code (13 H) is received, transmission is suspended; when a D1 code (11 H) is received transmission resumes.
- When CB (CTS) is found to be Space, transmission is suspended; it is found to be Mark transmission resumes.

ⓑ Using hardware handshake

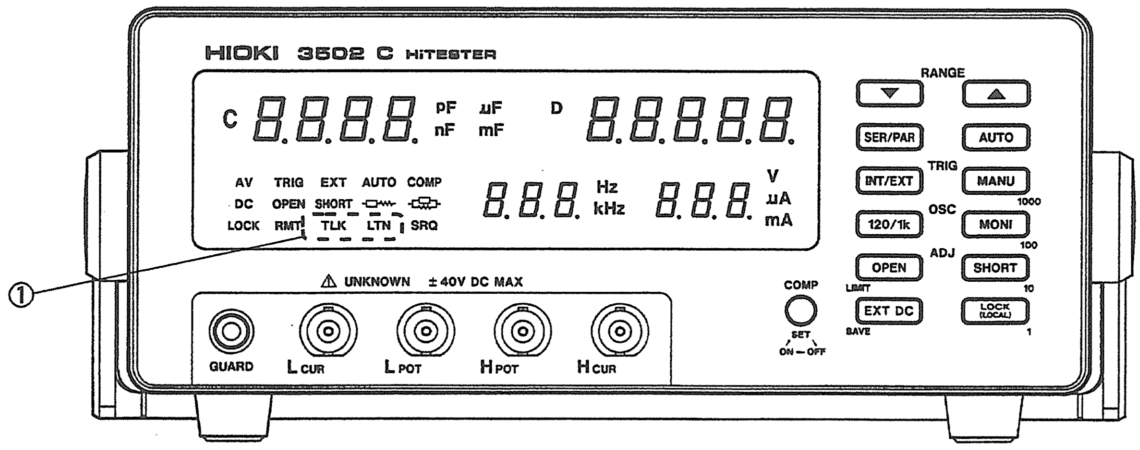
When CB (CTS) is found to be Space, transmission is suspended; it is found to be Mark transmission resumes.

Specifications

Chapter 3 Names of Parts

3.1 Controls and Connections

(1) 3502 front panel



① RS-232C Status display

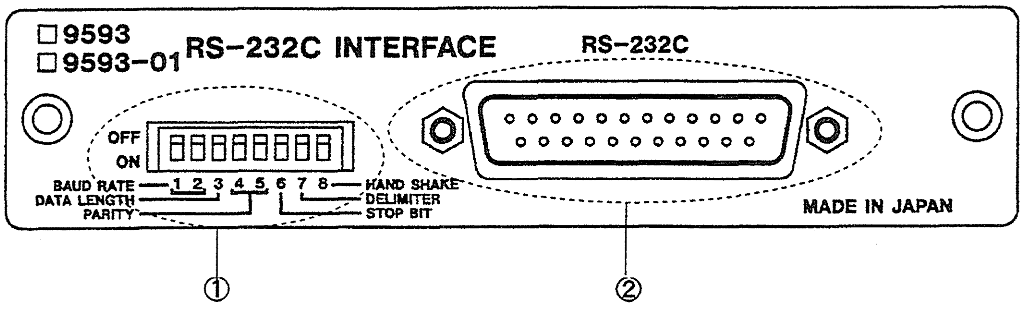
These indicators show the RS-232C control state:
TLK: During transmitting
LTN: During receiving

During both transmitting and receiving, the unit is under RS-232C remote control, and when these indicators appear all key input is invalid.

NOTE

When using the INT trigger or transmitting the *TRG command continuously, because of the method of internal processing, in some cases these indicators may not light.

(2) 9593 RS-232C interface outer panel



① Communication condition setting switches

These are used to set the communication condition of this 3502 unit on the RS-232C bus. For how to set these switches, refer to Section 4.1, "Setting the RS-232C Communication Conditions."

② RS-232C connector

Connect the RS-232C cable to this connector.

⚠ WARNING	**In order to prevent any danger of electric shock to the operator, check carefully that the power cable and the connectors to the 3502 have been removed first, before connecting the RS-232C cable to this connector.**

⚠ CAUTION	In order to prevent damage to the unit, do not short the connector and output terminal, and do not input voltage to the output terminal.

Chapter 4 Operation

4.1 Setting the RS-232C Communication Conditions

- Use the communication condition setting switches on the RS-232C panel to set the communication condition.
- On dispatch from the factory, this address is initially set to 00000000.

NOTE

If you change the communication condition while the 3502 is being used, then you should immediately turn the power off and on again.
If this is not done, the communication condition will not be changed to the new one.

0: OFF 1: ON

Bits	1	2	3	4	5	6	7	8
Description	Baud rate		Data length	Parity		Stop bits	Delimiter	Handshake

Baud rate

Bit 1	2	Baud rate
0	0	9600
0	1	4800
1	0	2400
1	1	1200

Data length

Bit 3	Data length
0	8 bits
1	7 bits

Parity

Bit 4	5	Parity
0	0	None
0	1	None
1	0	Even
1	1	Odd

Stop bits

Bit 6	Stop bits
0	1 bit
1	2 bits

Delimiter

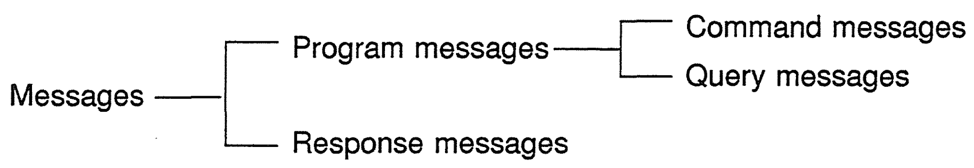
Bit 7	Delimiter
0	CR+LF
1	LF

Handshake

Bit 8	Handshake
0	Hard ware
1	XON/XOFF

4.2 Messages

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



Of these, program messages are those received by the device from the controller, while response messages are those transmitted from the device to the controller.

(1) Program messages

Program messages are command messages or query messages.

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

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

(2) Response messages

After a query message has been received, a response message is produced the moment that its syntax has been checked.

4.3 Command Syntax

- The names of commands for the 3502 are as far as possible mnemonic. Furthermore, all commands have a long form, and an abbreviated short form.

  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.

  For "HEADER", either "HEADer" (the long form) or "HEAD" (the short form) will be accepted. However, any one of "HEADE", or "HEA" is wrong and will generate an error.

- If the unabbreviated long form of the command consists of two or more words, then its abbreviated short form consists of the first letter of the first word (shown with a single upper case letter) followed by a header portion of the second word (the portion shown in upper case letters).
  For example, the abbreviated short form of the "COMParator:First_LIMit" command is as follows:
  COMParator: $\boxed{\text{FLIMit}}$

  (A space is represented by "_" in the examples)

  Response messages generated by the 3502 are in long form and in upper case letters.

4.4 Headers

Whether or not headers are prefixed to response messages is set by the "HEADer" command. It is essential to prefix headers to program messages.

(1) Command program headers

There are three types of command: simple commands, compound commands, and standard commands.

① Simple command header

This header is a sequence of letters.

Example HEADer

② Compound command header

This header is made up from a plurality of simple command type headers marked off by colons (:).

Example COMParator:RANGe

③ Standard command header

This header begins with an asterisk (*) to indicate that it is a standard command.

Example *RST

(2) Query program headers

These are for commands used for interrogating the device about the results of operations, about measured values, or about the current states of settings for the device.

As shown by the following example, they can be recognized as queries by a question mark (?) appearing after the program header.

Example COMParator $\boxed{?}$

4.5 Delimiters

The 3502 recognizes either a linefeed character (LF) or a carriage return plus linefeed (CR+LF) as delimiters.

(1) LF (linefeed only)
(2) CR+LF (carriage return plus linefeed)

4.6 Separators

(1) 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 :AUTO_ON$\boxed{;}$:AVERaging_ON$\boxed{;}$ *IDN?

(2) Header separator

In a message which has a header and data, a space (represented by "_ " in the examples) is used as the header separator to separate the header from the data.

Example :DISPlay:MONItor$\boxed{_}$VOLTage

(3) Data separator

If a message has several data items, commas (,) are required as data separators for separating these data items from one another.

Example :COMParator:SLIMit_<lower limit>$\boxed{,}$<upper limit>

4.7 Data Formats

The 3502 uses character string data and decimal numeric data, and the type used varies according to the command in question.

(1) Character data

Character string data must always begin with an alphabetic character, and the characters following can be either alphabetic characters or numerals. Although in character data either upper case letters or lower case letters are accepted, response messages output by the 3502 are always in upper case letters.

Example :TRIGger_intERNAL

(2) Decimal data

The numeric data values are all represented in decimal, in three formats identified as NR1, NR2 and 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 limit which the 3502 can deal, it is rounded off. (5 and above is rounded up; 4 and below is rounded down).

NR1 format - integer data.

Examples +12, -23, 34

NR2 format - fixed point numbers.

Examples +1.23, -23.45, 3.456

NR3 format - floating point numbers.

Examples +1E-2, -2.3E+4

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

When the 3502 is receiving it accepts NRf format, but when it is sending response messages it utilizes whichever one of the formats NR1 to NR3 is indicated in the particular command.

Examples :COMParator:RANGe_6
:COMParator:RANGe_+6.012
:COMParator:RANGe_0.0006E4

NOTE If the accuracy of a numerical value does not satisfy the limit which the 3502 can deal, an execution error is generated.

4.8 Abbreviation of Compound Commands

When several compound commands have a common head portion (for example, :COMP:RANG and :COMP:TYPE, etc.), then, when and only when writing them directly following on from one another, this common portion (:COMP: in this example) can be omitted from each command except for the first one.

This common portion is called "the current path", by analogy with the general concept of the current directory in the directory structure of UNIX or MSDOS, and until it is cleared the analysis of following commands is performed by deeming them to be preceded by the current path which has been curtailed in the interests of brevity. This manner of using the current path is shown in the following example:

Normal expression

```
:COMParator:RANGe_6;:COMParator:TYPE_3
```

Abbreviated expression

```
[:COMParator:] RANGe_6;TYPE_3
 ↑
 └── This becomes the current path, and can be
     curtailed from the following commands.
```

The current path is cleared when the power is turned on, when a colon (:) appears at the start of a command, and when a delimiter is detected.

Messages of standard command form can be executed without relation to the current path. Further, they have no effect upon the current path.

It is not necessary to prefix a colon (:) at the start of headers of simple commands and compound commands. However, in order to prevent confusion with abbreviated forms and mistakes in operation, it is recommended practice always to prefix ":" to headers.

With the 3502, there are two possible current paths:

:COMParator:

:CORRection:

NOTE The current path cannot be curtailed in the query messages.

4.9 Output Queue

Response messages accumulate in the output queue and all data are received and cleared.

The output queue is also cleared in the following circumstances:

· When the power is turned off and turned on again.

· When the 3502 unit is reset by a key press.

The 3502 has an output queue of 800 bytes capacity. If the response messages overflow this limit of 800 bytes, a query error is generated, and the output buffer is cleared. Further, if a new message is received while the output queue still contains data, the output queue is cleared, and a query error is generated.

4.10 Input Buffer

The 3502 has an input buffer of 300 bytes capacity.
When more than 300 bytes of data are transmitted, when the buffer is full any subsequent bytes received will be ignored.
(When the controller handshake setting is not the same as the 9593.)

4.11 Event Registers

(1) Standard event status register (SESR)

The standard event status register is an 8-bit register.
The standard event status register is cleared in the following three situations:

- When a "*CLS" command is executed.
- When an "*ESR?" query is executed.
- When the unit is powered on.

Standard event status register (SESR) bit assignments

Bit 7 PON	Power on flag. When the power is turned on, or on recovery from a power cut, this bit is set to 1.
Bit 6 URQ	User request. Unused.
Bit 5 CME	Command error. When a command which has been received contains a syntactic or semantic error, this bit is set to 1. · The command is not supported by the 3502. · There is a mistake in a program header. · The number of data parameters is wrong. · The format of the parameters is wrong.
Bit 4 EXE	Execution error. When for some reason a command which has been received cannot be executed, this bit is set to 1. · The designated data value is outside the set range. · The designated data value is not acceptable.
Bit 3 DDE	Device dependent error. When a command cannot be executed due to some cause other than a command error, a query error, or an execution error, this bit is set to 1. · Execution is impossible due to an abnormality inside the 3502. · Execution is impossible because some other function is being performed. · During open or short circuit compensation, valid data cannot be obtained. · When COMParator:TYPE is set to "0" (all parameters off).

Bit 2 QYE	Query error. This bit is set to 1 when a query error is detected by the output queue control. · When an attempt has been made to read the output queue when it is empty. · When the data overflows the output queue. · When data in the output queue has been lost.
Bit 1 RQC	Request for controller authority. Unused.
Bit 0 OPC	Operation terminated. Unused.

(2) Event status registers specific to the 3502 (ESR0 and ESR1)

Two 8-bit event status registers are provided for managing events on the 3502. Event status register 0 and event status register 1 are cleared in the following three situations:

- When a "*CLS" command is executed.
- When an ":ESR0?" query (for event status register 0) or ":ESR1?" query (for event status register 1) is executed.
- When the unit is powered on.

Event status register 0 (ESR0) bit assignments

Bit 7 CEM	Compensation data measurement completed
Bit 6 SOF	Second parameter range overflow
Bit 5 SUF	Second parameter range underflow
Bit 4 FOF	First parameter range overflow
Bit 3 FUF	First parameter range underflow
Bit 2 IDX	Data sampling completed
Bit 1 EOM	Measurement completed
Bit 0	Unused

Event status register 1 (ESR1) bit assignments

Bit 7	Unused
Bit 6 AND	Logical product (AND) of comparison results (bit 1, bit 4)
Bit 5 SLO	Second parameter below lower limit
Bit 4 SIN	Second parameter within limits
Bit 3 SHI	Second parameter above upper limit
Bit 2 FLO	First parameter below lower limit
Bit 1 FIN	First parameter within limits
Bit 0 FHI	First parameter above upper limit

Chapter 5 3502 RS-232C Command Summary

5.1 Standard Commands

Command	Function	Ref page
*CLS	Clears SESR, ESR0, ESR1.	36
*ESR?	Queries ESR.	36
*IDN?	Queries device ID.	37
*RST	Device initialization.	38
*TRG	Performs sampling once.	39
*TST?	Queries the result of the self-test.	40

5.2 Commands Specific to the 3502

Command	Data format ( ):number of data items	Function	Ref page
AUTO	ON/OFF (1)	Sets auto ranging.	41
AUTO?		Queries auto range .	41
AVERaging	ON/OFF (1)	Enables and disables averaging processing.	42
AVERaging?		Queries enablement of averaging processing.	42
BEEPer	ON/OFF (1)	Enables and disables the beep sound.	43
BEEPer?		Queries enablement of the beep sound.	43
BIAS	ON/OFF (1)	Enables and disables DC biasing.	44
BIAS?		Queries enablement of the DC biasing.	44
COMParator	ON/OFF (1)	Enables and disables the comparator function.	45
COMParator?		Queries the comparator function enablement.	45
COMParator:AVERaging	ON/OFF (1)	Enables and disables averaging processing for the comparison.	46
COMParator:AVERaging?		Queries averaging processing for the comparison enablement.	46
COMParator:First LIMit	NR2 numerical data (2)	Sets limit values for first parameter.	47
COMParator:First LIMit?		Queries limit values for first parameter.	48
COMParator:FREQuency	NR1 numerical data (1)	Sets the test frequency for comparison.	49
COMParator:FREQuency?		Queries the test frequency for comparison.	49
COMParator:MODE?		Queries equivalent circuit mode for comparison.	50
COMParator:RANGe	NR1 numerical data (1)	Sets range for comparison.	51
COMParator:RANGe?		Queries range for comparison.	51
COMParator:Secondary LIMit	NR2 numerical data (2)	Sets limit values for second parameter.	52
COMParator:Secondary LIMit?		Queries limit values for second parameter.	53
COMParator:TRIGger	INTernal/EXTernal (1)	Sets trigger mode for comparison.	54
COMParator:TRIGger?		Queries trigger mode for comparison.	54

Command	Data format ( ):number of data item	Function	Ref page
COMParator:TYPE	NR1 numerical data (1)	Selects parameters subject to comparison.	55
COMParator:TYPE?		Queries parameters subject to comparison.	56
CORRection:OPEN	ON/OFF (1)	Enables and disables the open circuit compensation function.	57
CORRection:OPEN?		Queries the open circuit compensation function enablement.	58
CORRection:SHORt	ON/OFF (1)	Enables and disables the short circuit compensation function.	59
CORRection:SHORt?		Queries the short circuit compensation function enablement.	60
DISPlay:MONitor	VOLTage/CURRent/ OFF (1)	Sets monitor item displayed.	60
DISPlay:MONItor?		Queries monitor item displayed.	61
ERRor?		Queries RS-232C error.	62
ESR0?		Queries event status register 0.	63
ESR1?		Queries event status register 1.	64
FREQuency	NR1 numerical data (1)	Sets the test frequency.	65
FREQuency?		Queries the test frequency.	65
HEADer	ON/OFF (1)	Enables and disables headers for the response message.	66
HEADer?		Queries headers enablement.	66
MEASure?	NR3 numerical data (2) *NR1 numerical data (1 to 2)	Queries the data item. *Comparator result	67
MODE?	SERial/PARallel (1)	Queries equivalent circuit mode.	70
MONitor:CURRent?	NR3 numerical data (1)	Queries monitored current.	70
MONitor:VOLTtage?	NR3 numerical data (1)	Queries monitored voltage.	71
RANGe	NR1 numerical data (1)	Sets test range.	72
RANGe?		Queries test range setting.	73
TRANsmit:SEParator	NR1 numerical data (1)	Sets the separator.	74
TRANsmit:SEParator?		Queries the separator.	75
TRIGger	INTernal/EXTernal (1)	Sets the trigger mode.	76
TRIGger?		Queries the trigger mode.	76
USER:IDENtity	A to Z, a to z, 0 to 9, underscore "_", 7 characters (1)	Sets the user ID.	77
USER:IDENtity?		Queries the user ID.	77

5.3 Initialization Items

The following table shows which items are initialized and which not, under various conditions.

Item	Power on	*RST command	*CLS command
RS-232C Communication conditions *1	No	No	No
Device specific functions (ranges etc.)	No	Yes	No
Output queue	Yes	No	No
Input buffer	Yes	No	No
Event registers	Yes *2	No	Yes
Current path	Yes	No	No
Headers on/off	Yes	Yes	No
Separator for response messages	Yes	Yes	No

*1 When the power is turned on, item is discriminated.
*2 Except the PON bit (bit 7).

5.4 Making Comparator Settings

Often errors occur due to making comparator settings improperly or in the wrong order. In order to forestall this, keep to the following specified order when constructing your program.
Further, when making comparator settings, either check first to make sure that the comparator is turned off, or be sure first to turn it off by using the ":COMP" command.

Example procedure for setting the comparator on the 3502

```
① :COMP:TRIG INT
② :COMP:AVER ON
③ :COMP:FREQ 120
④ :COMP:RANG 5
⑤ :COMP:FLIM 10.00, 11.00
⑥ :COMP:SLIM 0.0012, 0.0015
⑦ :COMP:TYPE 3
```

NOTE

- Step ①

  Because the ":COMP:TRIG" command exerts no influence on the other setting conditions, it can be located anywhere in this sequence.

- Steps ② and ⑥

  When the upper and lower limit values for D (dissipation) are to be set using 4 1/2 digit accuracy, be sure to execute the ":COMP:SLIM" command to set the second parameter upper and lower limit values only after turning averaging on with the ":COMP:AVER" command. If the second parameter upper and lower limit values are set with averaging off, then the fifth decimal places of the values set will be rounded up or down from the values input. For example, if Step ⑥ were executed with averaging still turned off, then the second parameter upper limit value would be set to 0.002 and its lower limit value would be set to 0.001.

- Steps ③, ④ and ⑤

  The range within which the upper and lower limit values for capacitance (C) can be set differs depending upon the test frequency and the test range. Accordingly, be sure to execute the ":COMP:FLIM" command to set the first parameter upper and lower limit values only after setting the test frequency and the test range.

- Step ⑦

  Use the ":COMP:TYPE" command to designate the test parameters for comparison only after setting all the comparison conditions.

5.5 Commands Which Cannot be Used During Comparison

Commands Specific to the 3502
AUTO
BIAS
COMParator:AVERaging
COMParator:First LIMit
COMParator:FREQuency
COMParator:RANGe
COMParator:Secondary LIMit
COMParator:TRIGger
COMParator:TYPE
CORRection:OPEN
CORRection:SHORt
DISPlay:MONitor
DISPlay:MONitor?
FREQuency
RANGe
USER:IDENtity

* All standard commands can be used.

* If these commands are used while the comparator function is enabled, a device dependent error occurs.

5.6 Test Range Settings

3502 Test range settings (120 Hz)

Parallel equivalent circuit mode		
Range number	Display value range (for a fixed range)	Output resistance
1	0.0 to 400.0 pF	100 kΩ
2	200.0 to 990.0 pF	100 kΩ
3	0.000 to 4.000 nF	100 kΩ
4	2.000 to 9.900 nF	10 kΩ
5	0.00 to 40.00 nF	10 kΩ
6	20.00 to 99.00 nF	1 kΩ
7	0.0 to 400.0 nF	1 kΩ
8	200.0 to 990.0 nF	100 Ω
9	0.000 to 4.000 μF	100 Ω
10	2.000 to 9.900 μF	10 Ω
11	0.00 to 40.00 μF	10 Ω

Serial equivalent circuit mode		
Range number	Display value range (for a fixed range)	Output resistance
12	0.000 to 4.000 μF	10 kΩ
13	2.000 to 9.900 μF	10 kΩ
14	0.00 to 40.00 μF	1 kΩ
15	20.00 to 99.00 μF	1 kΩ
16	0.0 to 400.0 μF	100 Ω
17	200.0 to 990.0 μF	100 Ω
18	0.000 to 4.000 mF	10 Ω
19	2.000 to 9.900 mF	10 Ω
20	1.00 to 40.00 mF	10 Ω
21	20.00 to 99.00 mF	10 Ω
22	10.0 to 400.0 mF	10 Ω

3502 Test Range Settings (1 kHz)

Parallel equivalent circuit mode		
Range number	Display value range (for a fixed range)	Output resistance
1	0.0 to 40.00 pF	100 kΩ
2	20.00 to 99.00 pF	100 kΩ
3	0.0 to 400.0 pF	100 kΩ
4	200.0 to 990.0 pF	10 kΩ
5	0.000 to 4.000 nF	10 kΩ
6	2.000 to 9.900 nF	1 kΩ
7	0.00 to 40.00 nF	1 kΩ
8	20.00 to 99.00 nF	100 Ω
9	0.0 to 400.0 nF	100 Ω
10	200.0 to 990.0 nF	10 Ω
11	0.000 to 4.000 μF	10 Ω

Serial equivalent circuit mode		
Range number	Display value range (for a fixed range)	Output resistance
12	0.0 to 400.0 nF	10 kΩ
13	200.0 to 990.0 nF	10 kΩ
14	0.000 to 4.000 μF	1 kΩ
15	2.000 to 9.900 μF	1 kΩ
16	0.00 to 40.00 μF	100 Ω
17	20.00 to 99.00 μF	100 Ω
18	0.0 to 400.0 μF	10 Ω
19	200.0 to 990.0 μF	10 Ω
20	0.100 to 4.000 mF	10 Ω
21	2.000 to 9.900 mF	10 Ω
22	1.00 to 40.00 mF	10 Ω

Chapter 6 Command Reference

6.1 Format of Command Explanations

Syntax	Specifies the syntax for the command (a space is represented by "_" in this syntax).
<data>	For a command that has parameters, specifies their format.
Function	Specifies the function of the command.
Note	Specifies points to which attention should be paid when using the command.
Response syntax	Only appears for a command (query) to which a response message is returned. Specifies the syntax for the response message, both when headers are on and when headers are off.
Error	Specifies what types of error may occur. However of course all commands are susceptible to spelling mistakes.
Example	These are simple examples of the use of the command. The examples all show commands in the short form.
Processing time	Specifies the processing time used by the combination of the 3502 and the 9593 to perform analysis and internal processing of the command, in its long form. However, for commands which have data parameters, the processing time may depend on the number of parameters. For query commands, this time is the time taken when headers are on.

NOTE

- If communications occur during data sampling, the sampling is repeated. The test time may be longer than usual.
- The commands which can be used on the 3502 are all of the sequential type.

*RST

■ Performs device initial setting.

Syntax *RST

Function Resets the 3502. The parameters which are reset, the values to which they are reset, and those items which are not affected by this command, are listed below.

Note (1) Parameters which are reset, and their new values:

· First parameter display	capacitance (C) display
· Second parameter display	dissipation (D) display
· Trigger mode	internal trigger
· Test range	auto ranging
· Averaging	off
· Frequency display ratio	1 kHz
· Monitor display	monitor voltage
· Beep sound	off
· Headers	on
· Data separator	semicolon ";"

(2) Items which are not affected by a reset:

· RS-232C communication conditions
· The output queue
· The input buffer
· The delimiter for response messages
· The various event registers (SESR, ESR0, and ESR1)

Processing time About 4.5 ms

*TRG

■ Issues external trigger.

Syntax	*TRG	
Function	In external trigger mode, performs measurement once.	
Example	Transmission	`:TRIG EXT;*TRG;:MEAS?`
	Response	`C +22.96E-09;D +0.0008E+00`

Note This carries out sampling using the *TRG command. If communications occur during data sampling, the data sampling is repeated. The time taken may therefore be longer than the following values.

Processing time

· High speed testing	59 ms (when test frequency is 1 kHz)
	75 ms (when test frequency is 120 Hz)
· Testing with averaging	88 ms (when test frequency is 1 kHz)
	205 ms (when test frequency is 120 Hz)

*TST?

■ Requests execution of, and queries the result of, the self test.

Syntax *TST?

Function Performs the self test of the 3502, and returns the result thereof as a numerical value in NR1 format, from 0 to 31. The various bits of the result have the following meanings:

bit 0: a ROM error occurred
bit 1: a RAM error occurred
bit 2: a display error occurred
bit 3: an I/O error occurred
bit 4: an interrupt error occurred
bit 5: unused
bit 6: unused
bit 7: unused

Note
- No header is affixed to the response message.
- If any error occurs, no response message to this query is produced.
- A back up error (only) can be cleared with the *RST command.

Response syntax Whether headers are on or off <data>

Error If the response message is longer than 800 bytes, a query error occurs.

Example

Transmission	*TST?
Response	10

A RAM error (bit 1) and an I/O error (bit 3) have occurred.

Processing time About 393 ms

6.3 Commands Specific to the 3502

:AUTO

■ Sets automatic or manual setting of test range and equivalent circuit mode.

Syntax	:AUTO_<data>
<data>	ON/OFF (character data)
Function	Switches between automatic and manual setting of test range or equivalent circuit mode.
Error	If <data> is other than character data described above, a command error occurs.
Example	Transmission `:AUTO ON` The test range and the equivalent circuit mode are switched to automatic selection (auto-ranging).
Processing time	About 4.5 ms

:AUTO?

■ Queries whether automatic setting in effect.

Syntax	:AUTO?
Function	Queries whether test range and equivalent circuit mode are currently being automatically set, or not, and returns the result as "ON" or "OFF" (<data>).
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :AUTO_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:AUTO?`	`:AUTO?`
Response	`:AUTO ON`	`ON`

Processing time About 4.4 ms

:AVERaging

■ Turns averaging processing on or off.

Syntax	:AVERaging_<data>
<data>	ON/OFF (character data)
Function	Starts and stops averaging of test data.
Note	If this command is used during execution of the comparator function, although the comparator setting conditions are changed temporarily, they will not be preserved.
Error	If <data> consists of character data other than "ON" or "OFF", a command error occurs.
Example	Transmission `:AVER ON` Averaging processing is initiated.
Processing time	About 5.1 ms

:AVERaging?

■ Queries whether averaging is being performed.

Syntax	:AVERaging?
Function	Queries whether averaging is currently being performed, or not, and returns the result as "ON" or "OFF" (<data>).
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :AVERAGING_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:AVER?`	`:AVER?`
Response	`:AVERAGING ON`	`ON`

Processing time About 5.4 ms

:BEEPer

■ Sets the beep sound on or off.

Syntax	:BEEPer_<data>
<data>	ON/OFF (character data)
Function	Sets the beep sound on or off.
Error	If <data> is set to character data other than "ON" or "OFF", a command error occurs.
Example	Transmission `:BEEP OFF` This turns off the beep sound.
Processing time	About 4.7 ms

:BEEPer?

■ Queries whether the beep sound is on or off.

Syntax	:BEEPer?
Function	Returns the current beep sound setting as character data, ON or OFF.
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :BEEPER_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:BEEP?`	`:BEEP?`
Response	`:BEEPER OFF`	`OFF`

Processing time About 4.8 ms

:BIAS

■ Sets DC biasing on and off.

Syntax	:BIAS_<data>
<data>	ON/OFF (character data)
Function	Sets DC biasing on or off.
Note	To use the comparator function with DC biasing applied, the DC biasing should be turned on before activating the comparator function.
Error	If <data> consists of character data other than "ON" or "OFF", a command error occurs.
Example	Transmission `:BIAS ON` DC biasing is turned on.
Processing time	About 4.6 ms

:BIAS?

■ Queries whether DC biasing is on or off.

Syntax	:BIAS?
Function	Returns whether DC biasing is on or off as character data.
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :BIAS_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:BIAS?`	`:BIAS?`
Response	`:BIAS ON`	`ON`

Processing time About 4.6 ms

:COMParator

■ Enables and disables the comparator function.

Syntax :COMParator_<data>

<data> ON/OFF (character data)

Function Turns the comparator function on and off. (When making settings for the comparator function, refer to Section 5.4, "Making Comparator Settings".)

Error If COMParator:TYPE has been set to "0" (all parameters are set off), or if <data> consists of character data other than "ON" or "OFF", a command error occurs.

Example Transmission `:COMP ON`

The comparator function is turned on.

Processing time About 5.5 ms

:COMParator?

■ Queries the comparator function enablement.

Syntax :COMParator?

Function Returns the current enablement state of the comparator function as character data (<data>), "ON" or "OFF".

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:COMPARATOR_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:COMP?`	`:COMP?`
Response	`:COMPARATOR ON`	`ON`

Processing time About 5.6 ms

:COMParator:AVERaging

■ Turns averaging processing for the comparator function on or off.

Syntax :COMParator:AVERaging_<data>

<data> ON/OFF (character data)

Function Turns averaging processing for the comparator function on or off. (When making settings for the comparator function, refer to Section 5.4, "Making Comparator Settings".)

Error If <data> consists of character data other than "ON" or "OFF", a command error occurs.

Example Transmission `:COMP:AVER OFF`

Averaging processing for the comparator function is turned off.

Processing time About 6.4 ms

:COMParator:AVERaging?

■ Queries whether averaging processing for the comparator function is on or off.

Syntax :COMParator:AVERaging?

Function Returns (in <data>) whether averaging processing for the comparator function is currently on or off.

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	COMPARATOR:AVERAGING_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:COMP:AVER?`	`:COMP:AVER?`
Response	`:COMPARATOR:AVERAGING ON`	`ON`

Processing time About 7.4 ms

:COMParator:First LIMit

■ Sets the lower and upper limit values for the first comparator parameter.

Syntax :COMParator:FLIMit_<data1>,<data2>

<data>

<data1>: lower limit value	(example) 5.22	Both are numerical value in NR2 format
<data2>: upper limit value	(example) 6.11	

Function

- Sets the lower and upper limit values for the first comparator parameter (i.e. the principal measured value) as absolute numerical values. (When making settings for the comparator function, refer to Section 5.4, "Making Comparator Settings".)
- If the lower limit value and the upper limit value are set to be the same, then deviation testing is performed.
- The numerical value can be in NRf format, but rounding is performed for figures beyond the last valid decimal place.
- Refer to Section 5.6 "Test Range Settings" to select valid upper and lower limit values for a given ranges setting.

Note

- The long form and the short form are as shown below. Any variation will cause a command error.
  Long form: FLIMit
  Short form: FLIM
- The lower and upper limit values must be in the same range.

Error

- If <data> is other than NRf format, an execution error occurs.
- If an attempt is made to set the lower and upper limit values, straddling two or more ranges, or to set the upper limit to value lower than the lower limit value, an execution error occurs.

Example

```
Transmission        :COMP:FLIM 0.987,1.234
```

The lower limit value is set to 0.987 nF and the upper limit value is set to 1.234 nF for the first comparator parameter. (For the 3502 in range 5)

Processing time About 9.1 ms

:COMParator:First:LIMit?

■ Queries the lower and upper limit values for the comparator parameter.

Syntax :COMParator:FLIMit?

Function
- Returns the lower and upper limit values <data1>, <data2> for the first comparator function in NR2 format.
- If any error occurs, no response message to this query is produced.
- The long form and the short form are as shown below. Any variation will cause a command error.

  Long form: FLIMit
  Short form: FLIM

Response syntax

If headers are on	:COMPARATOR:FLIMIT_<data1>,<data2>
If headers are off	<data1>,<data2>

Error If the response message is longer than 800 bytes, a query error is generated.

Example If headers are on

```
Transmission    :COMP:FLIM?
Response        :COMPARATOR:FLIMIT 0.876,0.987
```

If headers are off

```
Transmission    :COMP:FLIM?
Response        0.876,0.987
```

Processing time About 7.7 ms

:COMParator:FREQuency

■ Sets the test frequency for the comparator function.

Syntax :COMParator:FREQuency_<data>

<data> 120/1000 (numerical value in NR1 format)
This is the test frequency in Hertz (i.e. either 120 Hz or 1 kHz.)

Function
- Sets the test frequency for the comparator function.
(When making settings for the comparator function, refer to Section 5.4, "Making Comparator Settings".)
- <data> can be in NRf format, but rounding is performed for figures beyond the decimal point.

Error If <data> has a value other than the two values described above, an execution error occurs.

Example Transmission `:COMP:FREQ 120`
This sets the test frequency for the comparator function to 120 Hz.

Processing time About 7.3 ms

:COMParator:FREQuency?

■ Queries the test frequency for the comparator function.

Syntax COMParator:FREQuency?

Function
- Returns (in <data>) a numerical value in NR1 format representing the currently set test frequency (in Hertz) for the comparator function.
- The value of <data> is either 120 (120 Hz) or 1000 (1 kHz).

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:COMPARATOR:FREQUENCY_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	:COMP:FREQ?	:COMP:FREQ?
Response	:COMPARATOR:FREQUENCY 120	120

Processing time About 7.6 ms

:COMParator:MODE?

■ Queries the equivalent circuit mode for the comparator function.

Syntax	:COMParator:MODE?
Function	Returns (in <data>) the currently set equivalent circuit mode for the comparator function as "SERIAL" or "PARALLEL". SERIAL: series equivalent circuit mode PARALLEL: parallel equivalent circuit mode
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :COMPARATOR:MODE_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.
Example	

	If headers are on	If headers are off
Transmission	`:COMP:MODE?`	`:COMP:MODE?`
Response	`:COMPARATOR:MODE PARALLEL`	`PARALLEL`

Processing time About 6.8 ms

:COMParator:RANGe

■ Sets the range for the comparator function.

Syntax :COMParator:RANGe_<data>

<data> 1 to 22

- Refer to Section 5.6 "Test Range Settings" for the relationship between the range data value and the displayed values.
- <data> can be in NRf format, but any digits after the decimal point will be rounded.

Function Sets the range for the comparator function. (When making settings for the comparator function, refer to Section 5.4, "Making Comparator Settings.)

Note When the range is set, the equivalent circuit mode is determined automatically.

Error If <data> is of other than NRf format, an execution error occurs.

Example Transmission `:COMP:RANG 5`

Set the comparator function range to range 5.

Processing time About 6.1 ms

:COMParator:RANGe?

■ Queries the range for the comparator function.

Syntax :COMParator:RANGe?

Function

- Returns (in <data>) the currently set range for the comparator function as a numerical value in NR1 format, from 1 to 22 for the 3502.
- Refer to 5.6, "Test Range Settings" for the relationship between the range data value and the displayed values.

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:COMPARATOR:RANGE_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:COMP:RANG?`	`:COMP:RANG?`
Response	`:COMPARATOR:RANGE 5`	5

Processing time About 6.8 ms

:COMParator:Secondary LIMit

■ Sets the lower and upper limit values for the second comparator parameter.

Syntax :COMParator:SLIMit_<data1>,<data2>

<data>

<data1>: the lower limit value (example) 0.0005	Both are numerical value in NR2 format (during averaging)
<data2>: the upper limit value (example) 0.0015	

Function
- Sets the lower and upper limit values for the second parameter for the comparator function. (When making settings for the comparator function, refer to Section 5.4, "Making Comparator Settings".)
- <data> can be in NRf format, but rounding is performed for figures beyond the last valid decimal place.
- The valid ranges within which the upper and lower limit values can be set are as follows for each parameter: for high speed dissipation (D) testing, from 0.000 to 1.999; for dissipation (D) testing with averaging, from 0.0000 to 1.9999.

Note
- The long form and the short form are as shown below. Any variation will cause a command error.
  Long form: SLIMit
  Short form: SLIM
- Deviation testing cannot be performed for the second comparator parameter.
- For averaging, the dissipation value can be set to four decimal places. However, when manually setting the comparator parameters from the front panel, it is important to note that the dissipation value becomes invalid.

Error
- If <data> is in other than NRf format, an execution error occurs.
- If an attempt is made to set the upper limit value to a value which is lower than the lower limit value, an execution error occurs.

Example

```
Transmission        :COMP:SLIM 0.0026,0.0046
```

This sets the lower limit value for the second comparator parameter (D) to 0.0026, and its upper limit value to 0.0046. (For testing with averaging).

```
Transmission        :COMP:SLIM 0.026,0.046
```

This sets the lower limit value for the second comparator parameter (D - dissipation) to 0.026, and its upper limit value to 0.046. (For high speed testing).

Processing time About 12.4 ms

:COMParator:Secondary LIMit?

■ Queries the lower and upper limit values for the second comparator parameter.

Syntax :COMParator:SLIMit?

Function Returns the current lower limit value (in <data1>) and upper limit value (in <data2>) for the second parameter for the comparator function in NR2 format.

Note
- If any error occurs, no response message to this query is produced.
- The long form and the short form are as shown below. Any variation will cause a command error.

Long form: SLIMit
Short form: SLIM

Response syntax

If headers are on	:COMPARATOR:SLIMIT_<data1>,<data2>
If headers are off	<data1>,<data2>

Error If the response message is longer than 800 bytes, a query error is generated.

Example If headers are on

Transmission	:COMP:SLIM?
Response	:COMPARATOR:SLIMIT 0.0123,0.0077

If headers are off

Transmission	:COMP:SLIM?
Response	0.0123,0.0077

Processing time About 7.4 ms

:COMParator:TRIGger

■ Sets the trigger mode for the comparator function.

Syntax :COMParator:TRIGger_<data>

<data> INTernal/EXTernal (character data)
The parameter indicates internal or external trigger mode.

Function Sets the trigger mode for the comparator function.
(When making settings for the comparator function, refer to Section 5.4, "Making Comparator Settings".)

Error If <data> consists of character data other than "INTernal" or "EXTernal", a command error occurs.

Example Transmission `:COMP:TRIG EXT`
This sets the trigger mode for the comparator function to external triggering.

Processing time About 6.8 ms

:COMParator:TRIGger?

■ Queries the trigger mode for the comparator function.

Syntax :COMParator:TRIGger?

Function Returns (in <data>) the currently set trigger mode for the comparator function as "INTERNAL" or "EXTERNAL".

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:COMPARATOR:TRIGGER_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example If headers are on

Transmission	`:COMP:TRIG?`
Response	`:COMPARATOR:TRIGGER EXTERNAL`

If headers are off

Transmission	`:COMP:TRIG?`
Response	`EXTERNAL`

Processing time About 7.3 ms

:COMParator:TYPE

■ Selects the parameters for which the comparison is performed.

Syntax :COMParator:TYPE_<data>

<data> 1, 2 or 3 (numerical data in NR1 format)

The value of <data> indicates which parameters the comparator function applies to, as follows:

Data	First parameter comparison	Second parameter comparison
1	ON	OFF
2	OFF	ON
3	ON	ON

<data> can be in NRf format, but any digits after the decimal point will be rounded.

Function Selects the parameters (i.e. display values) for which the comparison is performed.
(When making settings for the comparator function, refer to Section 5.4, "Making Comparator Settings".)

Note It is not possible to disable comparison for both parameters (e.g. by setting <data>=0).

Error If the value of <data> is other than specified above, an execution error occurs.

Example Transmission `:COMP:TYPE 3`

This sets the comparator function to operate on both parameters.

Processing time About 6.0 ms

:COMParator:TYPE?

■ Queries the parameters for which the comparison is performed.

Syntax COMParator:TYPE?

Function Returns (in <data>) a value indicating which parameters the comparator function currently applies to, as follows:

Data	First parameter comparison	Second parameter comparison
0	OFF	OFF
1	ON	OFF
2	OFF	ON
3	ON	ON

Note
- Although <data>=0 may be returned as a response to this query, this setting cannot be made.
- If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:COMPARATOR:TYPE_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:COMP:TYPE?`	`:COMP:TYPE?`
Response	`:COMPARATOR:TYPE 3`	3

Processing time About 6.4 ms

:CORRection:OPEN

■ Enables and disables the open circuit compensation function.

Syntax	:CORRection:OPEN_<data>
<data>	ON/OFF (character data)
Function	• Enables and disables the open circuit compensation function. • When the parameter "ON" is specified, the data for open circuit compensation is measured, and then the open circuit compensation is carried out.
Note	• If valid open circuit compensation data cannot be obtained, the following steps take place: (1) The first parameter display on the unit displays "Err" for one second. (2) The open circuit compensation function is turned off. (3) The DDE bit (bit 3) of SESR is set to 1. (4) A device dependent error is generated. • Other commands cannot be executed during measurement of the open circuit compensation data.
Error	If <data> consists of character data other than "ON" or "OFF", a command error occurs.
Example	Transmission `:CORR:OPEN ON` This enables the open circuit compensation function, first measuring the data for open circuit compensation, and subsequently performing open circuit compensation.
Processing time	For "ON", about 7.6 s; for "OFF", about 6.0 ms.

:CORRection:OPEN?

■ Queries the open circuit compensation function enablement.

Syntax	:CORRection:OPEN?
Function	Returns the current setting of open circuit compensation function enablement as character data ,"ON" or "OFF".
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :CORRECTION:OPEN_<data>
	If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`CORR:OPEN?`	`CORR:OPEN?`
Response	`CORR:OPEN ON?`	`ON`

Processing time About 6.5 ms

:CORRection:SHORt

■ Enables and disables the short circuit compensation function.

Syntax :CORRection:SHORt_<data>

<data> ON/OFF (character data)

Function
- Enables and disables the short circuit compensation function.
- When the parameter "ON" is specified, the data for short circuit compensation is measured, and then the short circuit compensation is carried out.

Note
- If valid short circuit compensation data cannot be obtained, the following steps take place:

  (1) The short parameter display on the unit displays "Err" for one second.
  (2) The short circuit compensation function is turned off.
  (3) The DDE bit (bit 3) of SESR is set to 1.
  (4) A device dependent error is generated.
- Other commands cannot be executed during measurement of the short circuit compensation data.

Error If <data> consists of character data other than "ON" or "OFF", a command error occurs.

Example Transmission `:CORR:SHOR ON`

This enables the short circuit compensation function, first measuring the data for short circuit compensation, and subsequently performing short circuit compensation.

Processing time For "ON", about 7.4 s; for "OFF", about 6.2 ms.

:CORRection:SHORt?

■ Queries the short circuit compensation function enablement.

Syntax	:CORRection:SHORt?
Function	Returns the current setting of the short circuit compensation enablement as character data "ON" or "OFF".
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :CORRECTION:SHORT_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	:CORR:SHOR?	:CORR:SHOR?
Response	:CORRECTION:SHORT ON	ON

Processing time About 6.7 ms

:DISPlay:MONitor

■ Sets the monitor parameter for display.

Syntax :DISPlay:MONitor_<data>

<data> VOLTage/CURRent/OFF (character data)
VOLTage: display of the monitored voltage value
CURRent: display of the monitored current value
OFF: no monitor display

Function Sets the test signal monitor display parameter, or stops monitor display.

Error If <data> is set to character data other than "VOLTage" or "CURRent", or "OFF", a command error is generated.

Example Transmission `:DISP:MON CURR`
This sets the display to show the monitored current value.

Transmission `:DISP:MON OFF`
This stops the display of monitored values.

Processing time About 6.5 ms

:DISPlay:MONitor?

■ Queries the voltage and current monitored parameters.

Syntax	:DISPlay:MONitor?
Function	Returns the monitored parameters as "VOLTAGE", "CURRENT", and "OFF" in order. VOLTAGE: display of monitored voltage CURRENT: display of monitored current OFF: no monitor display
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :DISPLAY:MONITOR_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error occurs.

Example

	If headers are on	If headers are off
Transmission	`:DISP:MON?`	`:DISP:MON?`
Response	`:DISPLAY:MONITOR CURRENT`	`CURRENT`

Processing time About 6.6 ms

:ERRor?

■ Reads out RS-232C communication condition errors.

Syntax :ERRor?

Function Returns the value of RS-232C communication condition errors as a numerical value in NR1 format from 0 to 7, and then clears RS-232C communication condition errors.

					4	2	1
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Unused	Unused	Unused	Unused	Unused	Overrun error	Framing error	Parity error

Note
- No header is prefixed to the response message.
- If any error occurs, no response message to this query is produced.

Response syntax Whether headers are on or off <data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

Transmission	:ERR?
Response	4

An overrun error has occurred.

Processing time About 4.5 ms

:ESR0?

■ Reads out event status register 0.

Syntax ESR0?

Function Returns the value of event status register 0 (ESR0) as a numerical value in NR1 format as an even numerical value (<data>) in the range from 0 to 254, and then clears event status register 0.

128	64	32	16	8	4	2	1
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
CEM	SOF	SUF	FOF	FUF	IDX	EOM	Unused

Event status register 0 (ESR0)

Note
- No header is prefixed to the response message.
- If any error occurs, no response message to this query is produced.

Response syntax Whether headers are on or off <data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

Transmission	:ESR0?
Response	2

The measurement completed flag (bit 1) of ESR0 is raised.

Processing time About 4.2 ms

:ESR1?

■ Reads out event status register 1.

Syntax ESR1?

Function Returns the value of event status register 1 (ESR1) as a numerical value in NR1 format, from 0 to 127, and then clears event status register 1.

128	64	32	16	8	4	2	1
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Unused	AND	SLO	SIN	SHI	FLO	FIN	FHI

Event status register 1 (ESR1)

Note
- No header is prefixed to the response message.
- If any error occurs, no response message to this query is produced.

Response syntax Whether headers are on or off <data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

Transmission	:ESR1?
Response	36

The first parameter below lower limit flag (bit 2) and the second parameter below lower limit flag (bit 5) are up.

Processing time About 4.2 ms

:FREQuency

■ Sets the test frequency.

Syntax FREQuency_<data>

<data> 120/1000 (numerical data in NR1 format)
This is the test frequency in Hertz (i.e. either 120 Hz or 1 kHz)

Function
- Sets the testing frequency.
- The numerical value can be in NRf format, but rounding is performed for figures beyond the last valid decimal place.

Error If <data> is other than numerical value described above, an execution error occurs.

Example Transmission `:FREQ 120`
The test frequency is set to 120 Hz.

Processing time About 5.6 ms

:FREQuency?

■ Queries the test frequency.

Syntax :FREQuency?

Function
- Returns the currently test frequency as a numerical value in NR1 format.
- The value of <data> is either 120 (120 Hz) or 1000 (1 kHz).

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:FREQUENCY_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:FREQ?`	`:FREQ?`
Response	`:FREQUENCY 120`	`120`

Processing time About 5.6 ms

:HEADer

■ Enables and disables headers for the response message

Syntax	:HEADer_<data>
<data>	ON/OFF (character data)
Function	Sets whether or not the 3502 will prefix headers to its response messages.
Note	In any case, responses to the queries *IDN?, *TST?, ESR0?, ESR1? *ESR? and ERRor? are not prefixed with any headers.
Error	If <data> is set to character data other than "ON" or "OFF", a command error occurs.
Example	Transmission `:HEADer OFF` No headers are prefixed to response messages.
Processing time	About 4.4 ms

:HEADer?

■ Queries whether or not headers on response messages are enabled.

Syntax :HEADer?

Function Returns whether or not headers on response messages are enabled as character data, "ON" or "OFF" (<data>).

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:HEADER_ON
If headers are off	OFF

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	:HEAD?	:HEAD?
Response	:HEADER ON	OFF

Processing time About 4.8 ms

:MEASure?

■ Queries measured data items.

Syntax :MEASure?

Function
- Returns the measured values of test data items as numerical values in NR3 format.
- When the comparator function is in use, the comparison result is also returned as a numerical value in NR1 format.

Note
- With this query, if any error other than overflow or underflow occurs, no response message is produced.
- No test data is output until testing has been completed.
- By using the ":TRANsmit:SEParator" command, the data separator can be changed from the semicolon ";" to the comma ",".

(For ":TRANsmit:SEParator", refer to ":TRANsmit:SEParator" item.)

Response syntax

(1) During normal testing:

When headers are on:

```
□_ +□□□□□E±□□;□_+□□□□□□E+00
↑       ↑       ↑  ↑↑       ↑      ↑
①      ②      ③ ④⑤      ⑥     ⑦
```

When headers are off:

```
+□□□□□E±□□;_+□□□□□□E+00
    ↑      ↑ ↑      ↑      ↑
   ②     ③ ④     ⑥     ⑦
```

(2) When the comparator function is being used for comparison:

When headers are on:

```
□_+□□□□□E±□□;□_+□□□□□□E+00;_□□;□□
↑      ↑       ↑  ↑↑       ↑       ↑↑  ↑↑ ↑
①     ②      ③ ④⑤      ⑥      ⑦④ ⑧④ ⑨
```

When headers are off:

```
+□□□□□E±□□;_+□□□□□□E+00;_□□;□□
    ↑      ↑ ↑      ↑       ↑↑  ↑↑ ↑
   ②     ③ ④     ⑥      ⑦④ ⑧④ ⑨
```

(3) When the comparator function is being used for deviation testing:

When headers are on:

```
□_±□□□□□E+00;□_±□□□□□□E+00;_□□;□□
↑    ↑        ↑↑↑       ↑       ↑↑  ↑↑ ↑
①   ⑩       ⑪④⑤      ⑥      ⑦④ ⑧④ ⑨
```

When headers are off:

```
±□□□□□E+00;_±□□□□□□E+00;_□□;□□
    ↑      ↑ ↑      ↑       ↑↑  ↑↑ ↑
   ⑩     ⑪ ④     ⑥      ⑦④ ⑧④ ⑨
```

① First parameter header ("C" (capacitance))

② First parameter mantissa (four digits and decimal point)
(position of the decimal point depends upon the set range)

③ First parameter exponent (two digits) (corresponding to the unit prefix p, n, μ or m)

④ Semicolon (the data separator - can be changed to comma)

⑤ Second parameter header ("D" (dissipation))

⑥ Second parameter mantissa (for dissipation (D), five digits and decimal point)
For dissipation (D), the position of the decimal point is: □.□□□□.
When dissipation is being displayed during high speed testing, the last digits is always "0".

⑦ Second parameter exponent (two digits) (always "00")

⑧ First parameter comparator result (sign and one digit)
This represents the actual result as follows:
+0: In   +1: Hi   -1: Lo
With ":COMP:TYPE_2", this is two spaces (refer to Example 3).

⑨ Second parameter comparator result (sign and one digit)
This represents the actual result as follows:
+0: In   +1: Hi   -1: Lo
With ":COMP:TYPE_1", this does not appear (refer to Example 2).

⑩ First parameter deviation testing result (four digits and decimal point)
The decimal point is in the position □□□□., and the value represents the displayed value.

⑪ First parameter exponent (two digits) (always "00")

- First parameter overflow:
  All four digits of ② are "9". (example) +99.99E-03
- First parameter underflow:
  All four digits of ② are "0". (example) +00.00E-03
  For range 1, underflow cannot occur.
- Second parameter overflow:
  All five digits of ⑥ are "9". (example) +9.9999E+00

Error If the response message is longer than 800 bytes, a query error is generated.

Example (1) During normal testing (with the 3502, during averaging):

If headers are on

```
Transmission    :MEAS?
Response        C +18.19E-03;D +0.0011E+00
```

If headers are off

```
Transmission    :MEAS?
Response        +18.19E-03; +0.0011E+00
```

(2) When the comparator function is being used for comparison (:COMP:TYPE_1):

If headers are on

```
Transmission        :MEAS?
Response            C +18.19E-06;D +0.0011E+00; +0
```

If headers are off

```
Transmission        :MEAS?
Response            +18.19E-06; +0.0011E+00; +0
```

(3) When the comparator function is being used for comparison (:COMP:TYPE_2):

If headers are on

```
Transmission        :MEAS?
Response            C +18.19E-06;D +0.0011E+00;    ;-1
```

If headers are off

```
Transmission        :MEAS?
Response            +18.19E-06; +0.0011E+00;    ;-1
```

(4) When the comparator function is being used for comparison (:COMP:TYPE_3):

If headers are on

```
Transmission        :MEAS?
Response            C +18.19E-06;D +0.0011E+00; +0;-1
```

If headers are off

```
Transmission        :MEAS?
Response            +18.19E-06; +0.0011E+00; +0;-1
```

(5) When the comparator function is being used for deviation testing (:COMP:TYPE_3):

If headers are on

```
Transmission        :MEAS?
Response            C -0026.E+00;D +0.0011E+00; -1;-1
```

If headers are off

```
Transmission        :MEAS?
Response            -0026.E+00; +0.0011E+00; -1;-1
```

Processing time

- During normal testing, about 5.8 ms
  However, in the case of averaging with internal triggering, a maximum of 84 ms for 1 kHz testing and of 200 ms for 120 Hz testing is also required in order to produce the result after testing has been completed.
- When the comparator function is in use, about 6.2 ms

:MODE?

■ Queries the equivalent circuit mode.

Syntax :MODE?

Function Returns (in <data>) the currently set equivalent circuit mode as "SERIAL" or "PARALLEL".

SERIAL: serial equivalent circuit mode
PARALLEL: parallel equivalent circuit mode

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:MODE_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:MODE?`	`:MODE?`
Response	`:MODE PARALLEL`	`PARALLEL`

Processing time About 4.7 ms

:MONitor:CURRent?

■ Queries the monitored test signal current value.

Syntax :MONitor:CURRent?

Function
- Returns (in <data>) the monitored current value of test signal as a numerical data value in NR3 format.
- If the test circuit is saturated, all three digits are indicated as "0".

Note
- This query is valid even if the unit is displaying monitored voltage values, or the monitor display is off.
- If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:MONITOR:CURRENT_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:MON:CURR?`	`:MON:CURR?`
Response	`:MONITOR:CURRENT +0.05E+0`	`+0.05E+0`

Processing time About 6.9 ms

:MONitor:VOLTage?

■ Queries the monitored test signal voltage value.

Syntax :MONitor:VOLTage?

Function
- Returns (in <data>) the monitored voltage value of test signal as a numerical data value in NR3 format.
- If the test circuit is saturated, all three digits are indicated as "0".

Note
- This query is valid even if the unit is displaying monitored voltage values, or the monitor display is off.
- If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:MONITOR:VOLTAGE_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:MON:VOLT?`	`:MON:VOLT?`
Response	`:MONITOR:VOLTAGE +0.78E+0`	`+0.78E+0`

Processing time About 6.9 ms

:RANGe

■ Sets the test range.

Syntax :RANGe_<data>

<data> 1 to 22

- Refer to Section 5.6, "Test Range Settings" for the relationship between the range data value and the displayed values.
- The numerical value can be in NRf format, but any digits after the decimal point will be rounded.

Function Sets the test range.

Note When the range is set, the equivalent circuit mode is determined automatically.

Error If <data> is other than NRf format, an execution error occurs.

Example Transmission `:RANG 2`

The test range is set to 2.

Processing time About 5.0 ms

:RANGe?

■ Queries the test range.

Syntax :RANGe?

Function
- Returns the test range setting as numerical value in NR1 format between 1 and 22 (<data>).
- Even if auto-ranging is currently enabled, this query will return the actual current range.
- Refer to Section 5.6, "Test Range Settings" for the relationship between the range data value and the displayed values.

Note If any error occurs, no response message to this query is produced.

Response syntax

If headers are on	:RANGE_<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:RANG?`	`:RANG?`
Response	`:RANGE 1`	`1`

Processing time About 4.7 ms

:TRANsmit:SEParator

■ Sets the data separator for response messages.

Syntax :TRANsmit:SEParator_<data>

<data> 0 or 1 through 255 (numerical data in NR1 format)

Function
- The data separator for MEASure? command is set as follows:
- If <data> = 0, the separator is set to semicolon ";".
- If <data> = any value from 1 to 255, the separator is set to comma ",".
- <data> can be accepted in NRf format, but its effective value will be obtained by rounding off from the decimal point on the basis of 5 and above being rounded up and 4 and below being rounded down.

Note
- Even if you set the data separator to the comma, it will appear as a semicolon when headers are on.
- A semicolon is appeared in the initial state.

Error
- If <data> is other than NRf format, an execution error is generated.
- If the value of data is outside the range 0 to 255, an execution error is generated.

Example

```
Transmission    :HEAD OFF;:MEAS?
Response        +01.23E-06; +0.0045E+00
```
Sets the header to OFF.

```
Transmission    :TRAN:SEP 1;:MEAS?
Response        +01.23E-03, +0.0045E+00
```
Sets the data separator to comma ",".

```
Transmission    :HEAD ON;:MEAS?
Response        C +01.23E-06;D +0.0045E+00
```
When headers are on, data separator appears as a semicolon ";".

```
Transmission    :HEAD OFF;:MEAS?
Response        +01.23E-06, +0.0045E+00
```

Processing time About 5.9 ms

:TRANsmit:SEParator?

■ Queries the data separator for response messages.

Syntax	:TRANsmit:SEParator?
Function	· The data separator for response messages is returned (in <data>) as 0 or 1. · The returned numerical value corresponds to the setting state of the data separator as follows: (1) If <data> = 0, the separator is a semicolon ";". (2) If <data> = 1, the separator is a comma ",".
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :TRANSMIT:SEPARATOR_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	`:TRAN:SEP?`	`:TRAN:SEP?`
Response	`:TRANSMIT:SEPARATOR 1`	`1`

Processing time About 6.8 ms

:TRIGger

■ Sets the type of trigger.

Syntax	:TRIGger_<data>
<data>	INTernal/EXTernal (character data) INTernal Internal trigger mode EXTernal External trigger mode
Function	Sets the type of trigger.
Note	If this command is used while the comparator function is being executed, although the comparator set conditions are changed, they will not be preserved.
Error	If <data> is other than character data described "INTernal" or "EXTernal", a command error occurs.
Example	Transmission `:TRIG EXT` The trigger mode is set to external trigger.
Processing time	About 5.7 ms

:TRIGger?

■ Queries the trigger setting.

Syntax	:TRIGger?
Function	Returns the trigger setting as character data, "INTERNAL" or "EXTERNAL" (<data>).
Note	If any error occurs, no response message to this query is produced.
Response syntax	If headers are on :TRIGGER_<data> If headers are off <data>
Error	If the response message is longer than 800 bytes, a query error is generated.

Example

	If headers are on	If headers are off
Transmission	:TRIG?	:TRIG?
Response	:TRIGGER EXTERNAL	EXTERNAL

Processing time About 5.2 ms

:User:IDENtity

■ Set the user ID

Syntax USER:IDENtity_<data>

<data> For example: ABC9593

Function
- The user can set an identity code for the 3502.
- The ID is backed up in the same way as the main unit settings.
- Enter an ID of exactly seven characters, using capital and lowercase letters, digits 0 to 9, and underscore "_".

Note
- Enter an ID of seven characters.
- If an ID of eight or more characters is entered, the first seven characters are used.
- A zero entered as the first character is ignored.
- Lowercase letters are identified with the corresponding capital letters.

Error If fewer than seven characters are entered, a command error results.

Example

Transmission	`:USER:IDEN ABC9593`

This sets the user ID to "ABC9593"

Processing time About 6.6 ms

:User:IDENtity?

■ Queries the user ID

Syntax :USER:IDENtity?

Function Returns the user ID as seven characters data (<data>).

Response syntax

If headers are on	:USER:IDENTITY<data>
If headers are off	<data>

Error If the response message is longer than 800 bytes, a query error is generated,

Example

	If headers are on	If headers are off
Transmission	`:USER:IDEN?`	`:USER:IDEN?`
Response	`:USER:IDENTITY ABC9593`	`ABC9593`

Processing time About 6.4 ms

ls Specific to the 3502

Chapter 7
Sample Programs

The following sample programs are all written for the Microsoft Quick BASIC. For more details on Quick BASIC, refer to the Quick BASIC documentation.

All commands in the sample programs are used in the short form, and the communication condition setting switches of the 3502 is taken as 00000000.

Program	Function	Page
(1)	Open- and short-circuit compensation	80
(2)	Basic settings and testing	82
(3)	Basic comparator settings	83
(4)	Carrying out comparator testing	84

(1) Open- and short-circuit compensation

Summary This program carries out open- and short-circuit compensation on the 3502.

Program List

```
10   OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1
20   PRINT #1, ":HEAD OFF"
30   PRINT #1, "*CLS"
40   CHECK.OPEN:
50   INPUT "Prepare unit for open circuit compensation,then press Enter", A$
60   CHECK.OPEN1:
70   PRINT "Collecting open circuit compensation data"
80   PRINT #1, ":CORR:OPEN ON"
90   CHECK.OPEN2:
100  PRINT #1, ":ESR0?"
110  INPUT #1, A
120  IF (A AND 128) = 0 THEN GOTO CHECK.OPEN2
130  PRINT #1, "*ESR?"
140  INPUT #1, A
150  IF (A AND 8) = 0 THEN GOTO CHECK.SHORT:
160  PRINT "Open circuit compensation failed"
170  GOTO CHECK.OPEN
180  CHECK.SHORT:
190  INPUT "Prepare unit for short circuit compensation,then press Enter", A$
200  CHECK.SHORT1:
210  PRINT "Collecting short circuit compensation data"
220  PRINT #1, ":CORR:SHOR ON"
230  CHECK.SHORT2:
240  PRINT #1, ":ESR0?"
250  INPUT #1, A
260  IF (A AND 128) = 0 THEN GOTO CHECK.SHORT2
270  PRINT #1, "*ESR?"
280  INPUT #1, A
290  IF (A AND 8) = 0 THEN GOTO EXIT1
300  PRINT "Short circuit compensation failed"
310  GOTO CHECK.SHORT
320  EXIT1:
330  CLOSE #1
340  END
```

Program comments

Line	*Comments*
10	Open the RS-232C circuit file.
20	Response headers off.
30	Clear registers.
80	Execute compensation.
90-120	Wait until compensation ends.
130-150	Check whether compensation completed correctly.
220	Execute compensation.
230-260	Wait until compensation ends.
270-290	Check whether compensation completed correctly.
330	Close the RS-232C circuit file.

(2) Basic settings and testing

Summary

This program selects the test conditions for measurement on the 3502.

- It carries out a single test measurement, and displays the result on the screen.
- It also displays the monitored voltage and current values on the screen.

Program List

```
10   OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1
20   PRINT #1, ":TRIG EXT"
30   PRINT #1, ":AVER ON"
40   PRINT #1, ":FREQ 120"
50   PRINT #1, ":AUTO ON"
60   PRINT #1, ":BIAS ON"
70   PRINT #1, ":DISP:MON VOLT"
80   PRINT #1, ":*TRG"
90   PRINT #1, ":MEAS?"
100  LINE INPUT #1, A$
110  PRINT A$
120  PRINT #1, ":MON:VOLT?"
130  INPUT #1, B$
140  PRINT B$
150  PRINT #1, ":MON:CURR?"
160  INPUT #1, C$
170  PRINT C$
180  CLOSE #1
190  END
```

Program comments

Line	*Comments*
10	Open the RS-232C circuit file.
20	Select external trigger mode.
30	Enable averaging.
40	Test frequency 120 Hz.
50	Enable auto-ranging.
60	Apply bias voltage.
70	Switch monitor display to show voltage.
80	Send trigger.
90	Get measurement value.
100	Read value.
110	Display value.
120	Query monitored voltage.
150	Query monitored current.
180	Close the RS-232C circuit file.

Sample output

```
C +22.24E-06;D +0.0834E+00

:MONITOR:VOLTAGE +0.06E+00
:MONITOR:CURRENT +0.97E-03
```

(3) Basic comparator settings

Summary

This program makes the comparator settings for 3502.

Note

- This program only makes the settings. It does not carry out any testing.
- Before making the comparator settings, read the notes in Section 5.4 "Making Comparator Settings."

Program List

```
10   OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1
20   PRINT #1, ":COMP OFF"
30   PRINT #1, ":COMP:TRIG INT"
40   PRINT #1, ":COMP:AVER ON"
50   PRINT #1, ":COMP:FREQ 120"
60   PRINT #1, ":COMP:RANG 14"
70   PRINT #1, ":COMP:FLIM 20.00,25.00"
80   PRINT #1, ":COMP:SLIM 0.0000,0.0200"
90   PRINT #1, ":COMP:TYPE 3"
100  PRINT #1, ":BIAS ON"
110  PRINT #1, ":COMP ON"
120  CLOSE #1
130  END
```

Program comments

Line	*Comments*
10	Open the RS-232C circuit file.
20	Switch off comparator function.
30	Select internal trigger mode.
40	Enable averaging.
50	Test frequency 120 Hz.
60	Select range 14.
70	Set the lower and upper limits for the first parameter.
80	Set the lower and upper limits for the second parameter.
90	Apply comparison to both capacitance and dissipation.
100	Apply bias voltage.
110	Switch on comparator function.
120	Close the RS-232C circuit file.

(4) Carrying out comparator testing

Summary This program first makes the comparator settings. It then prompts for the number of samples to be tested, and starts testing.

- Its uses queries "MEAS?" for response message to count the occurrences of samples outside the comparator limits (either "Hi" or "Lo").
- At the end of testing, it displays the numbers of the samples which were outside the comparator limits.

Note Before deciding on the comparator settings, read the notes in Section 5.4 "Making Comparator Settings."

Program List

```
10    OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1
20    PRINT #1, ":COMP OFF"
30    PRINT #1, ":COMP:TRIG EXT"
40    PRINT #1, ":COMP:AVER ON"
50    PRINT #1, ":COMP:FREQ 120"
60    PRINT #1, ":COMP:RANG 14"
70    PRINT #1, ":COMP:FLIM 15.00,25.00"
80    PRINT #1, ":COMP:TYPE 1"
90    PRINT #1, ":HEAD OFF"
100   PRINT #1, ":TRANSMIT:SEP 1"
110   C = 1: FH = 0: FL = 0
120   INPUT "Number of samples to measure:", X!
130   OPTION BASE 1
140   DIM C$(X!)
150   DIM D$(X!)
160   DIM FL$(X!)
170   DIM NFH$(X!)
180   DIM NFL$(X!)
190   PRINT #1, ":COMP ON"
200   PRINT #1, ":*CLS"
210   D.MEAS:
220   IF C >= X! + 1 THEN GOTO D.DISP
230   PRINT #1, ":*TRG"
240   PRINT #1, ":MEAS?"
250   INPUT #1, C$, D$, FL$
260   IF VAL(FL$) = 1 THEN FH = FH + 1: NFH(FH) = C
270   IF VAL(FL$) = -1 THEN FL = FL + 1: NFL(FL) = C
280   C$(C) = C$: D$(C) = D$: FL$(C) = FL$
290   C = C + 1
300   GOTO D.MEAS
310   D.DISP:
320   PRINT "Number of samples high:"; FH; "-sample nos.:";
330   FOR IFH = 1 TO FH
340   PRINT NFH(IFH);
350   NEXT IFH
360   PRINT ""
```

```
370  PRINT "Number of samples low:"; FL; "-sample nos.:";
380  FOR IFL = 1 TO FL
390  PRINT NFL(IFL);
400  NEXT IFL
410  PRINT ""
420  FOR K = 1 TO X!
430  PRINT "Sample"; K; ":"; C$(K), D$(K), FL$(K)
440  NEXT K
450  PRINT #1, ":COMP OFF"
460  PRINT #1, ":ERR?"
470  INPUT #1, EE$
480  PRINT "Error:"; EE$
490  EXIT1:
500  CLOSE
510  END
```

Program comments

Line	*Comments*
10	Open the RS-232C circuit file.
20	Switch off comparator function.
30	Select external trigger mode.
40	Enable averaging.
50	Test frequency 120 Hz.
60	Select range 14.
70	Set the lower and upper limits for the first parameter.
80	Apply comparison to capacitance only.
90	Response headers off.
100	Set the data separator.
110	Initialize variables.
130-180	Array for saving comparator failures. (The OPTION statement specifies that arrays start from 1.)
190	Switch on comparator function.
200	Clear registers.
210-300	Measurement loop
310-440	Display comparator results.
450	Switch off comparator function.
460-480	Read out contents of the RS-232C error.
500	Close the RS-232C circuit file.

Sample output

```
Number of samples to measure: 5
Number of samples high: 0 -sample nos.:
Number of samples low: 2 -sample nos.:2 5
Sample 1 :+20.21E-06        +0.0834E+00   +0
Sample 2 :+14.55E-06        +0.0845E+00   -1
Sample 3 :+22.21E-06        +0.0836E+00   +0
Sample 4 :+18.89E-06        +0.0838E+00   +0
Sample 5 :+13.97E-06        +0.0852E+00   -1
Error:0
```

Chapter 8 Troubleshooting

If the RS-232C appears to be malfunctioning, refer to the information below before calling for servicing.

Symptom	Cause / Treatment
The RS-232C has stopped working completely.	Are the cables properly connected?
	Are all the devices powered on?
	Has the communication condition been correctly set?
Transmission on the RS-232C is not taking place properly.	Is the controller delimiter set correctly? (Refer to Section 4.5, "Delimiter".)
When attempting to read data using a BASIC INPUT statement, the RS-232C bus hangs.	Be sure to transmit one query before each INPUT statement.
	Have any of these transmitted queries resulted in as error?
Although a command has been transmitted, nothing has happened.	Using the "*ESR?" query, inspect the standard event status register, and check what type of error has occurred.
	Using the "ERRor?" query, and check whether transmission error occurred on the RS-232C.
The amount of data read in is insufficient.	If the data includes one or more commas, then try using a LINE INPUT statement.
Sending several queries, produces only one response.	Has an error occurred?
	Send the queries one at a time, and read the responses individually. When you want to read them in all at once, try doing so by putting them all on one line separated by the message separator character.

Symptom	Cause / Treatment
Although a response has been read in, the data does not appear.	Have the response messages from 3502 exceeded the buffer capacity of the computer?
	Try dividing up their reading in, by increasing the number of variables of the INPUT statement.
The response message to a query differs from the display on the front panel of the 3502.	Due to the response message being produced at the instant that the 3502 receives the query, there is a possibility that it may not agree with the display at the instant that the controller reads it in.

Service

If you suspect a problem, after confirming above troubleshooting, please contact the dealer from when the equipment was purchased.

Index

January 1996 Edition 1

HIOKI E.E. CORPORATION Printed in Japan

HIOKI

HIOKI E.E. CORPORATION

81 Koizumi, Ueda, Nagano 386-11, Japan
TEL: 0268-28-0562 FAX: 0268-28-0568

9593A981-00 96-01-0001H