# WT310/WT310HC/WT332/WT333
## Digital Power Meter
## Communication Interface
# USER'S MANUAL

YOKOGAWA
Yokogawa Meters & Instruments Corporation

IM WT310-17EN
3rd Edition

Thank you for purchasing the WT310, WT310HC, WT332, or WT333 Digital Power Meter. This Communication Interface User's Manual explains the following interface features and commands.

- USB interface
- GP-IB interface
- RS-232 interface
- Ethernet interface

To ensure correct use, please read this manual thoroughly before beginning operation.
After reading this manual, keep it in a safe place.

## List of Manuals

The following manuals, including this one, are provided as manuals for this instrument. Please read all manuals.

| Manual Title | Manual No. | Description |
|---|---|---|
| WT310/WT310HC/WT332/WT333 Digital Power Meter User's Manual | IM WT310-01EN | The manual explains all features of this instrument, except for the communication interface features, and how to use them. |
| WT310/WT310HC/WT332/WT333 Digital Power Meter Getting Started Guide | IM WT310-02EN | Provided as a printed manual. The manual explains the handling precautions and basic operations of this instrument and provides an overview of its features. |
| WT310/WT310HC/WT332/WT333 Digital Power Meter Communication Interface User's Manual | IM WT310-17EN | This guide. This manual explains the communication interface features of this instrument and how to use them. |
| WT310/WT310HC/WT332/WT333 Digital Power Meter | IM WT310-92Z1 | Document for China |

PDF files of all the manuals above are included in the accompanying CD.
The "EN" and "Z1" in the manual numbers are the language codes.

Contact information of Yokogawa offices worldwide is provided on the following sheet.

| Document No. | Description |
|---|---|
| PIM 113-01Z2 | List of worldwide contacts |

## Notes

- The contents of this manual are subject to change without prior notice as a result of continuing improvements to the instrument's performance and functionality. The figures given in this manual may differ from those that actually appear on your screen.
- Every effort has been made in the preparation of this manual to ensure the accuracy of its contents. However, should you have any questions or find any errors, please contact your nearest YOKOGAWA dealer.
- Copying or reproducing all or any part of the contents of this manual without the permission of YOKOGAWA is strictly prohibited.
- Safety precautions are provided in the Getting Started Guide, IM WT310-02EN. Be sure to observe the safety precautions.
- The TCP/IP software of this product and the documents concerning it have been developed/created by YOKOGAWA based on the BSD Networking Software, Release 1 that has been licensed from the Regents of the University of California.

## Trademarks

- Microsoft, Internet Explorer, MS-DOS, Windows, Windows Vista, Windows 7, Windows 8, and Windows 8.1 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Adobe and Acrobat are either registered trademarks or trademarks of Adobe Systems Incorporated.
- In this manual, the ® and TM symbols do not accompany their respective registered trademark or trademark names.
- Other company and product names are registered trademarks or trademarks of their respective holders.

## Revisions

- 1st Edition: January 2013
- 2nd Edition: June 2013
- 3rd Edition: January 2016

3rd Edition: January 2016 (YMI)

# About the USB Interface and Ethernet Interface

- To use the USB communication features, your PC must have the following:
  - Library  of this instrument (TMCTL)
  - USB device driver for connecting this instrument to the PC
- To use the Ethernet communication features, your PC must have the following:
  - Library of this instrument (TMCTL)

  You can download the library and driver from the following web page.
  http://tmi.yokogawa.com/

  If you install WTViewerFreePlus in your PC, the above library and driver will be installed automatically.

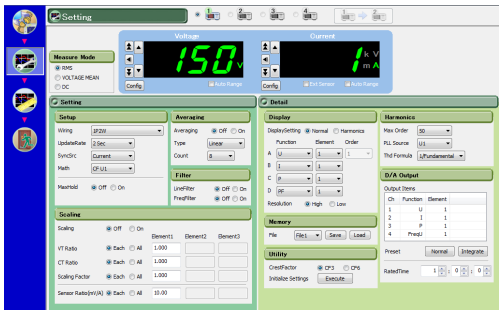
# Sample Programs

You can download sample programs for this instrument from the following web page:
http://tmi.yokogawa.com/

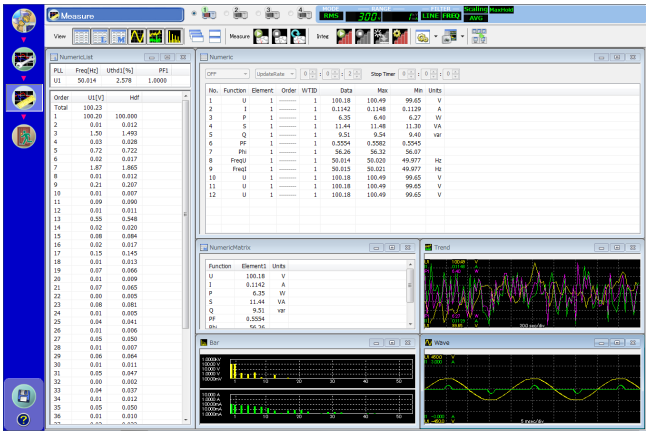
# WTViewerFreePlus

WTViewerFreePlus is a dedicated software application for this instrument. It is included in the accompanying CD.
By using WTViewerFreePlus, you can display measured data on a dedicated window, save measured data to the PC, and change the settings of this instrument from the PC.
For information on how to install and use WTViewerFreePlus, see the WTViewerFreePlus User's Manual, IM 760121-02E.

**Example of a Window for Configuring the WT**



**Example of a Window Showing the Measured Data**



# Conventions Used in This Manual

## Notes

The notes and cautions in this manual are categorized using the following symbols.

**WARNING** **Calls attention to actions or conditions that could cause serious or fatal injury to the user, and precautions that can be taken to prevent such occurrences.**

**CAUTION** **Calls attention to actions or conditions that could cause light injury to the user or cause damage to the instrument or user's data, and precautions that can be taken to prevent such occurrences.**

***Note*** Calls attention to information that is important for proper operation of the instrument.

## Units

| | |
|---|---|
| k | Denotes 1000. Example: 100 kHz (frequency) |
| K | Denotes 1024. Example: 720 KB (file size) |

## Characters That Appear on the 7-Segment LED

Because this instrument uses a 7-segment LED display, numbers, letters, and mathematical symbols are displayed using special characters in the manner shown below. Some of the characters shown below are not used by this instrument.

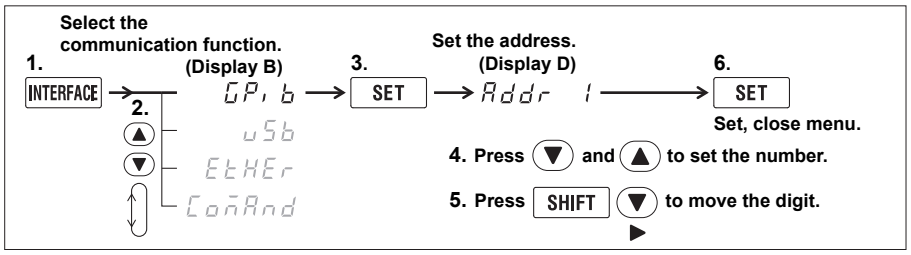
| | | | | |
|---|---|---|---|---|
| 0 → 0 | A → A | K → K | U → U | ^ (exponentiation) → ^ |
| 1 → 1 | B → b | L → L | V → V | |
| 2 → 2 | C → C **Lowercase c → c** | M → M | W → W | |
| 3 → 3 | D → d | N → n | X → X | |
| 4 → 4 | E → E | O → o | Y → Y | |
| 5 → 5 | F → F | P → P | Z → Z | |
| 6 → 6 | G → G | Q → q | + → + | |
| 7 → 7 | H → H **Lowercase h → h** | R → r | – → - | |
| 8 → 8 | I → I | S → S | × → * | |
| 9 → 9 | J → J | T → t | ÷ → / | |

## Symbols and Conventions Used in Procedural Explanations

In chapters 1 to 4, the contents of the procedural explanations are indicated using the following symbols.

**Procedure** Operations are explained using flowcharts. See the example below for an explanation of how various operations are indicated. All procedures are written under the assumption that you are starting operation at the beginning of the procedure, so you may not need to carry out all the steps in a procedure when you are changing the settings.

Example: Operations for setting the GP-IB address




The above flow chart indicates the following operations.
You can configure items that are blinking.

1. Press **INTERFACE**.
   A menu appears in display B.
2. Use ▲ or ▼ to select GPib.
   Pressing either key cycles through 4 menu items.
3. Press **SET** to confirm the selection of GPib.
   The GPib function menu that you selected in step 2 appears in display D.
4. Use ▲ or ▼ to select the GP-IB address.
5. If necessary, press the **SHIFT** key so that it illuminates, and then press ▼ to move the input digit.
6. Press **SET** to confirm the setting and return the menu display to the measurement data display.
   The selected or set item is confirmed when you press SET.

- When you are making a number positive (no sign) or negative (–) or setting a number, when the digit in the display that the input will be added to is blank, an underscore flashes at the position of the digit.
- While you are performing menu operations, to leave the menu display, press HOLD (ESC). All setting changes that you have confirmed by pressing the SET key will be reflected in the settings.

**Explanation** This section describes the setup items and the limitations regarding the procedures.

## Symbols Used in the Syntax

The following table contains the symbols that are used in the syntax discussed mainly in chapters 5 and 6. These symbols are referred to as BNF (Backus-Naur Form) symbols. For details on how to write data using these symbols, see pages 5-6 and 5-7.

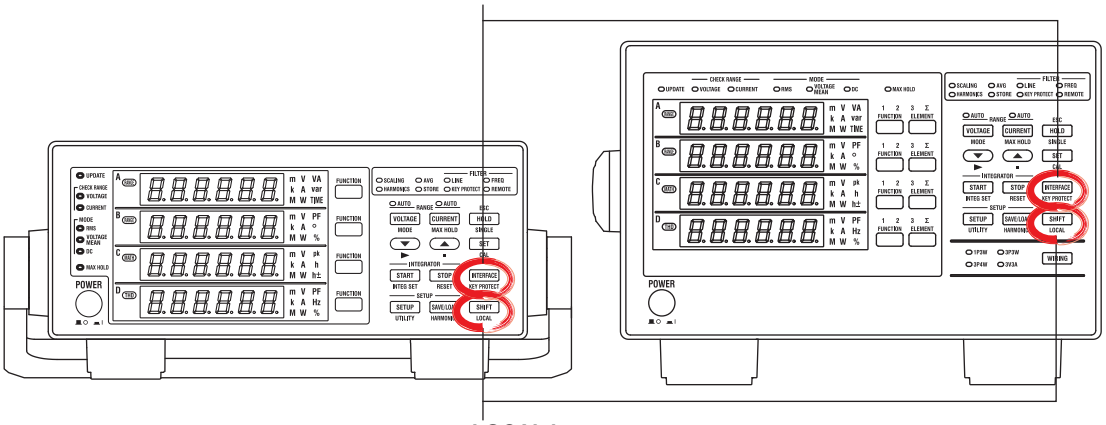
| Symbol | Meaning | Example | Example of Input |
|---|---|---|---|
| <> | A defined value | ELEMent<x> <x> = 1 to 3 | ELEMENT2 |
| {}<br>\| | Select an option in { }<br>Exclusive OR | MODE {RMS\|VMEan\|DC} | MODE RMS |
| [] | Can be omitted | NUMeric[:NORMal]:VALue? | NUMERIC:VALUE? |

# Contents

# 1.1 Component Names and Functions

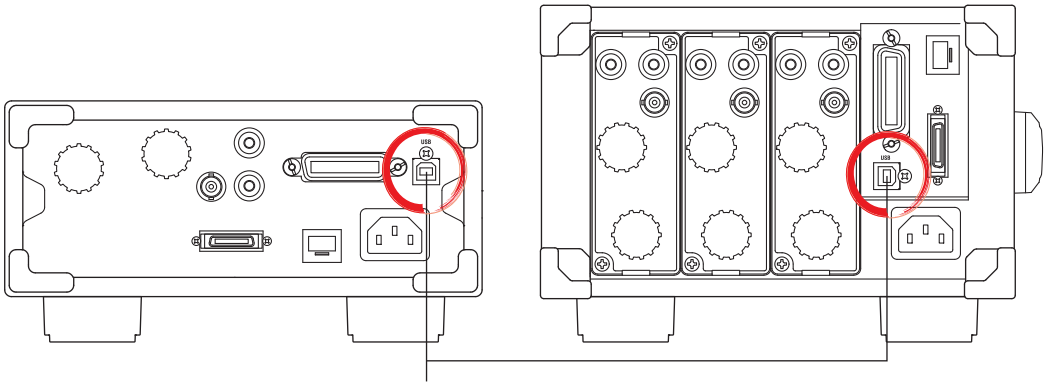
## Front Panel

**INTERFACE key (page 1-4)**
Press this key to view the serial number that is used in USB TMC communication.



**LOCAL key**
Press this key to switch to local mode. In local mode, remote mode (remote control using communication commands) is cleared, and key operation becomes possible.
This key is disabled when local lockout (see page 1-2) has been activated by a controller.

## Rear Panel



**USB port**
This port is for connecting this instrument to a controller (such as a PC) using a USB cable. For details on how to connect, see page 1-3.

# 1.2 USB Interface Features and Specifications

## USB Interface Features

### Reception Feature

- You can use the reception feature to specify the same settings that you specify by using the front panel keys.
- Output requests for measured and computed data, panel setup parameters, and error codes can be received.

### Transmission Feature

- This instrument can transmit measured and computed data.
- This instrument can transmit panel setup parameters and the status byte.
- This instrument can transmit error codes when errors occur.

## USB Interface Specifications

| Item | Specifications |
|---|---|
| Number of ports | 1 |
| Connector | Type B connector (receptacle) |
| Electrical and mechanical | Complies with USB Rev. 2.0 |
| Supported transfer modes | HS (High Speed; 480 Mbps) and FS (Full Speed; 12 Mbps) |
| Supported protocols | USBTMC-USB488 (USB Test and Measurement Class Ver. 1.0) |
| PC system requirements | PC running Windows 7 (32 bit/64 bit), Vista (32 bit), XP (SP2 and later; 32bit) English and Japanese and with a USB port |

## Switching between Remote and Local Modes

### When Switching from Local to Remote Mode

This instrument switches to remote mode when it is in local mode and it receives a :COMMunicate:REMote ON command from the PC.

- The REMOTE indicator illuminates.
- All keys except the **SHIFT (LOCAL)** key are disabled.
- Settings entered in local mode are retained even when this instrument switches to remote mode.

### When Switching from Remote to Local Mode

When this instrument is in remote mode and you press SHIFT (LOCAL), this instrument switches to local mode. However, this does not work if this instrument has received a `:COMMunicate:LOCKout ON` command from the PC. This instrument switches to local mode when it receives a `:COMMunicate:REMote OFF` command from the PC, regardless of the local lockout state.

- The REMOTE indicator turns off.
- Key operations are enabled.
- Settings entered in remote mode are retained even when this instrument switches to local mode.

***Note***

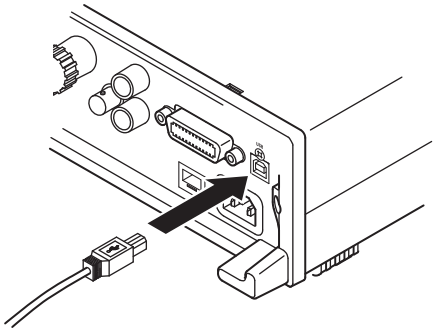
You cannot use the USB interface simultaneously with other interfaces (GP-IB, RS-232, and Ethernet interfaces).

# 1.3 Connecting to the USB Interface

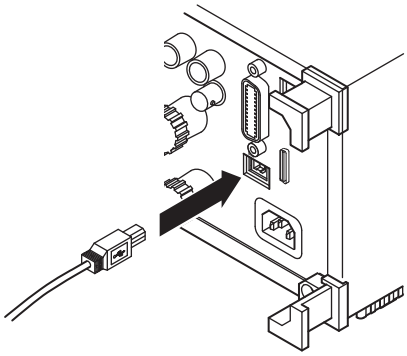
## Connection Procedure

Connect the cable as shown below.

WT310/WT310HC



WT332/WT333



## Notes on Connection

- Be sure to insert the USB cable connectors firmly into the USB ports.
- If you are connecting multiple devices by using a USB hub, connect this instrument to the USB hub port that is closest to the port that the controller is connected to.
- Do not connect or remove USB cables from the time when this instrument is turned on until operation becomes available (approximately 20 to 30 seconds). Doing so may damage this instrument.
- On the WT310 and WT310HC, it is physically impossible to connect a GP-IB cable and a USB cable at the same time.

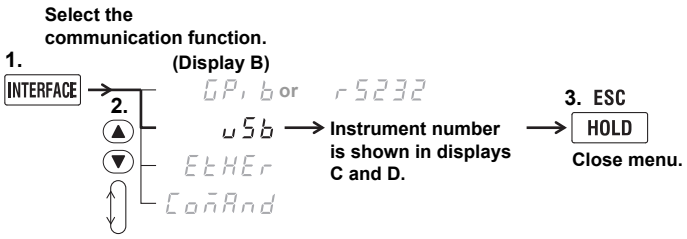
# 1.4 Configuring the USB Settings of This Instrument

This section explains the following setting for controlling this instrument remotely through a USB interface:

## Procedure

Follow the procedure indicated by the thick lines in the following menu.

- **Viewing the serial number that is used in USB TMC communications**




### *Note*

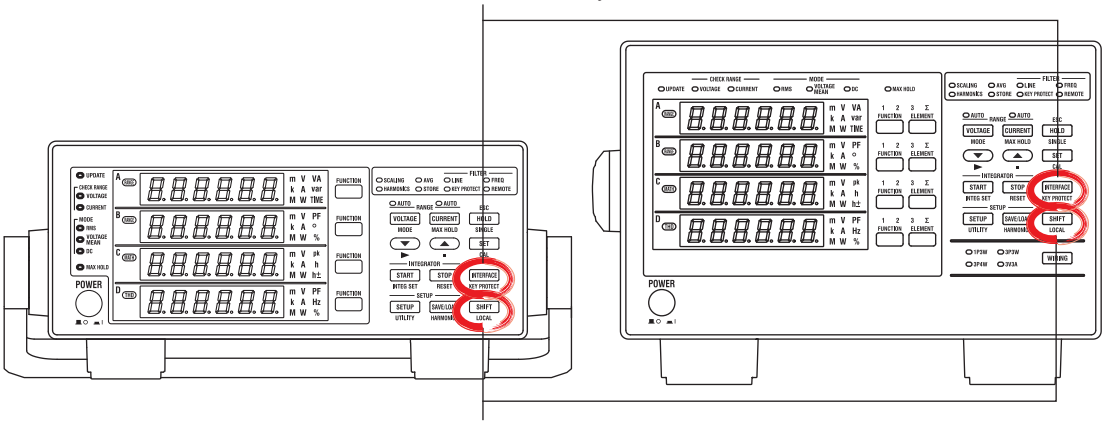
- Only use one communication interface: USB, GP-IB, RS-232, or Ethernet. If you send commands simultaneously from more than one communication interface, this instrument will not execute the commands properly.
- Install the YOKOGAWA USB TMC (Test and Measurement Class) driver on your PC. For information about how to obtain the YOKOGAWA USB TMC driver, contact your nearest YOKOGAWA dealer. You can also access the YOKOGAWA USB driver download web page and download the driver.
  http://tmi.yokogawa.com/
- Do not use USB TMC drivers (or software) supplied by other companies.

# 2.1 Component Names and Functions



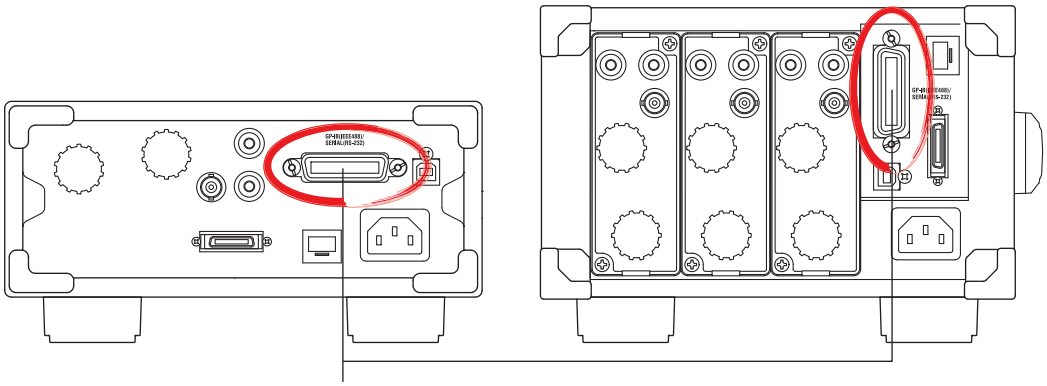
## Front Panel

**INTERFACE key (page 2-6)**
Press this key to set the GP-IB address.



**LOCAL key**
Press this key to switch to local mode. In local mode, remote mode (remote control using communication commands) is cleared, and key operation becomes possible.
This key is disabled when local lockout (see page 2-3) has been activated by a controller.

## Rear Panel



**GP-IB port**
This port is for connecting this instrument to a controller (such as a PC) using a GP-IB cable. For details on how to connect, see page 2-4.

# 2.2 GP-IB Interface Features and Specifications

## GP-IB Interface Features

### Reception Feature

- You can use the reception feature to specify the same settings that you specify by using the front panel keys.
- Output requests for measured and computed data, panel setup parameters, and error codes can be received.

### Transmission Feature

- This instrument can transmit measured and computed data.
- This instrument can transmit panel setup parameters and the status byte.
- This instrument can transmit error codes when errors occur.

***Note***

Talk-only, listen-only, and controller capabilities are not available.

## GP-IB Interface Specifications

| Item | Specifications |
|---|---|
| Supported devices | National Instruments Corporation<br>• PCI-GPIB or PCI-GPIB+<br>• PCIe-GPIB or PCIe-GPIB+<br>• PCMCIA-GPIB or PCMCIA-GPIB+<br>(not supported on Windows Vista or Windows 7.)<br>• GPIB-USB-HS<br>Driver NI-488.2M Version 2.8.1 and later |
| Electrical and mechanical | Conforms to IEEE St'd 488-1978 |
| Functional specifications | See the table below. |
| Protocol | Conforms to IEEE St'd 488.2-1992 |
| Code | ISO (ASCII) codes |
| Mode | Addressable mode |
| Address setting | Press **INTERFACE**, and then select the **GPIB** menu. Set the address to a value between 0 and 30. |
| Clear remote mode | Press **SHIFT (LOCAL)** to clear remote mode.<br>This is not possible when local lockout has been activated by the controller. |

### Functional Specifications

| Function | Subset Name | Description |
|---|---|---|
| Source handshaking | SH1 | Full source handshaking capability |
| Acceptor handshaking | AH1 | Full acceptor handshaking capability |
| Talker | T6 | Basic talker capability, serial polling, and untalk on MLA (My Listen Address). No talk-only capability. |
| Listener | L4 | Basic listener capability and unlisten on MTA (My Talk Address). No listen-only capability |
| Service request | SR1 | Full service request capability |
| Remote local | RL1 | Full remote/local capability |
| Parallel polling | PP0 | No parallel polling capability |
| Device clear | DC1 | Full device clear capability |
| Device trigger | DT1 | Device trigger capability |
| Controller | C0 | No controller capability |
| Electric characteristics | E1 | Open collector |

## Switching between Remote and Local Modes

### When Switching from Local to Remote Mode

This instrument switches to remote mode when it is in local mode and it receives a REN (Remote Enable) message from the PC.

- The REMOTE indicator illuminates.
- All keys except the **SHIFT (LOCAL)** key are disabled.
- Settings entered in local mode are retained even when this instrument switches to remote mode.

### When Switching from Remote to Local Mode

When this instrument is in remote mode and you press **SHIFT (LOCAL)**, this instrument switches to local mode. This key is disabled when local lockout (see page 2-7) has been activated by a controller.

- The REMOTE indicator turns off.
- Key operations are enabled.
- Settings entered in remote mode are retained even when this instrument switches to local mode.

***Note***

You cannot use the GP-IB interface simultaneously with other interfaces (USB and Ethernet interfaces).

# 2.3 Connecting to the GP-IB Interface

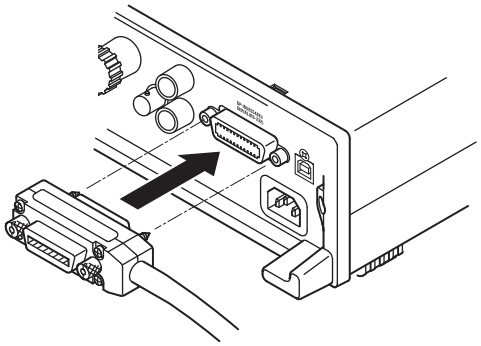
## GP-IB Cable

This instrument is equipped with an IEEE St'd 488-1978 24-pin GP-IB connector. Use a GP-IB cable that conforms to this standard.

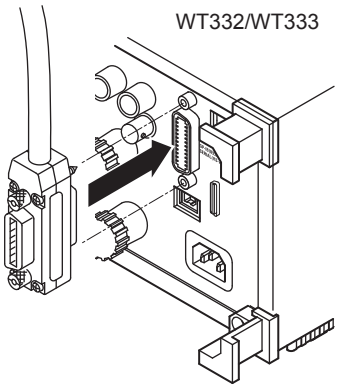
## Connection Procedure

Connect the cable as shown below.

WT310/WT310HC

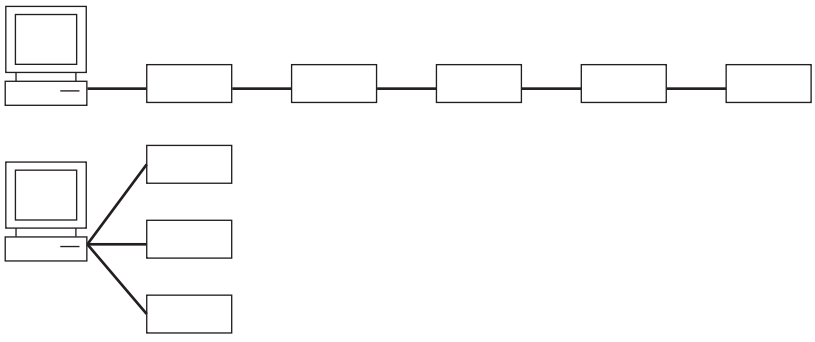


WT332/WT333



## Notes on Connection

- Firmly tighten the screws on the GP-IB cable connector.
- On the PC end, use a GP-IB board (or card) made by National Instruments. For details, see section 2.2.
- This instrument may not operate properly if this instrument is connected to the PC through converters (such as a GP-IB to USB converter). For more details, contact your nearest YOKOGAWA dealer.
- Several cables can be used to connect multiple devices. However, no more than 15 devices, including the controller, can be connected on a single bus.
- When connecting multiple devices, you must assign a unique address to each device.
- Use cables that are 2 m or shorter in length to connect devices.
- Make sure the total length of all cables does not exceed 20 m.
- When devices are communicating, have at least two-thirds of the devices on the bus turned on.
- To connect multiple devices, wire them in a daisy-chain or star configuration as shown below. You can also mix these configurations. Loop configuration is not allowed.



- On the WT310 and WT310HC, it is physically impossible to connect a GP-IB cable and a USB cable at the same time.

**CAUTION**

Be sure to turn off the PC and this instrument before you connect or remove communication cables. Otherwise, erroneous operation may result, or the internal circuitry may break.

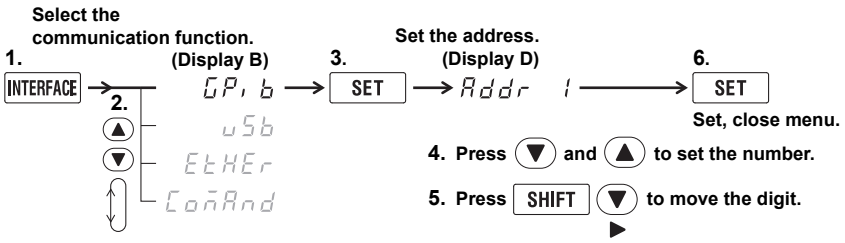
# 2.4 Configuring the GP-IB Settings of This Instrument

This section explains the following setting for controlling this instrument remotely through a GP-IB interface:

## Procedure

Follow the procedure indicated by the thick lines in the following menu.

### Setting the GP-IB Address




***Note***

- Only use one communication interface: USB, GP-IB, or Ethernet. If you send commands simultaneously from more than one communication interface, this instrument will not execute the commands properly.
- When the controller is communicating with this instrument or with other devices through GP-IB, do not change the address.
- Each device that is connected by GP-IB has its own unique address in the GP-IB system. This address is used to distinguish between different devices. Therefore, you must assign a unique address to this instrument when connecting it to a PC or other device.

# 2.5 Responses to Interface Messages



## Responses to Interface Messages

### Responses to Uni-Line Messages

- **IFC (Interface Clear)**
  Clears the talker and listener functions. Stops data transmission if it is in progress.
- **REN (Remote Enable)**
  Switches between the remote and local modes.

  IDY (Identify) is not supported.

### Responses to Multi-Line Messages (Address commands)

- **GTL (Go To Local)**
  Switches the instrument to local mode.
- **SDC (Selected Device Clear)**
  - Clears the program message (command) being received and the output queue (see page 7-6 for details).
  - Discards *OPC and *OPC? commands that are being executed.
  - Immediately aborts *WAI and COMMunicate:WAIT commands.
- **GET (Group Execute Trigger)**
  The same operation as the *TRG command.

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

### Responses to Multi-Line Messages (Universal commands)

- **LLO (Local Lockout)**
  Prohibits switching to local mode by disabling the LOCAL key on the front panel.
- **DCL (Device Clear)**
  The same operation as the SDC message.
- **SPE (Serial Poll Enable)**
  Sets the talker function on all devices on the bus to serial polling mode. The controller will poll each device one by one.
- **SPD (Serial Poll Disable)**
  Clears the serial polling mode of the talker function on all devices on the bus.

  PPU (Parallel Poll Unconfigure) is not supported.

## What Are Interface Messages?

Interface messages are also referred to as interface commands or bus commands. They are commands that are issued by the controller. They are classified as follows:

### Uni-line Messages

A single control line is used to transmit uni-line messages. The following three types are available.

- IFC (Interface Clear)
- REN (Remote Enable)
- IDY (Identify)

## Multi-line Messages

Eight data lines are used to transmit multi-line messages. The messages are classified as follows:

- **Address Commands**

  Some address commands are valid when a device is designated as a listener, and some are valid when it is designated as a talker. The following five commands are available.

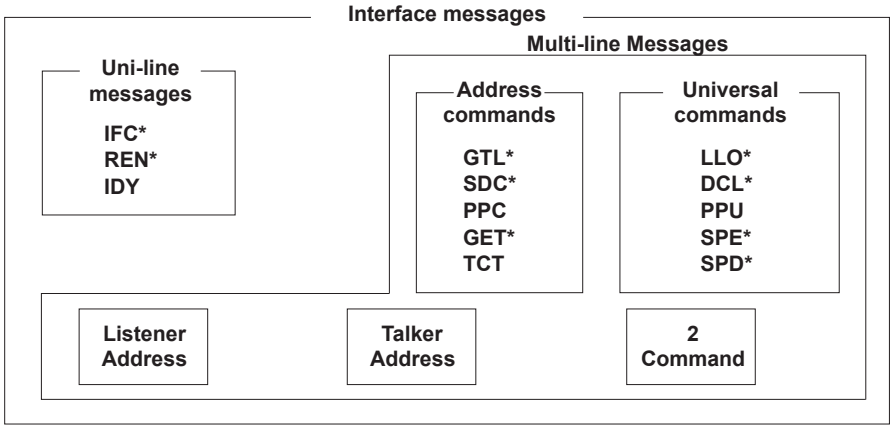
  Commands available to a device designated as a listener
  - GTL (Go To Local)
  - SDC (Selected Device Clear)
  - PPC (Parallel Poll Configure)
  - GET (Group Execute Trigger)

  Commands available to a device designated as a talker
  - TCT (Take Control)

- **Universal Commands**

  Universal commands are available to all devices regardless of their listener or talker designation. The following five commands are available.
  - LLO (Local Lockout)
  - DCL (Device Clear)
  - PPU (Parallel Poll Unconfigure)
  - SPE (Serial Poll Enable)
  - SPD (Serial Poll Disable)

There are other interface messages: listener-address, talk-address, and secondary commands.




This instrument supports interface messages marked with an asterisk.

### *Note*

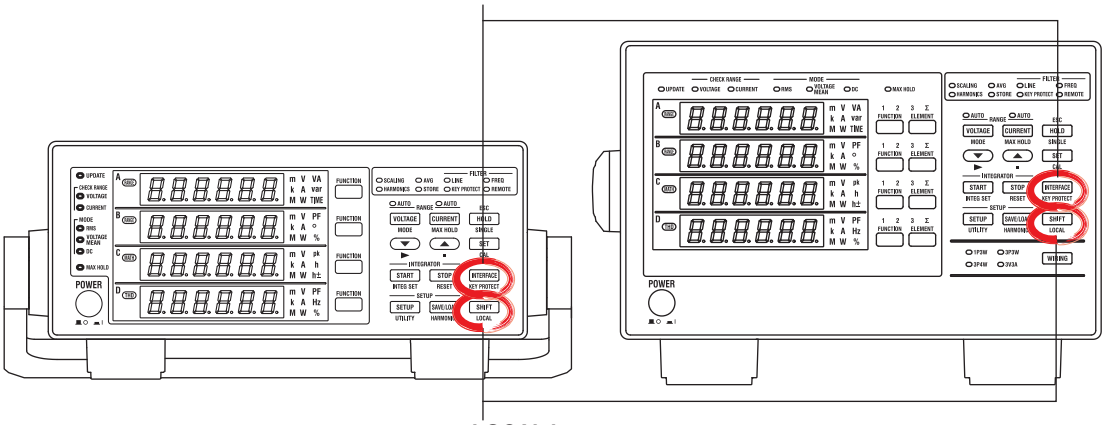
**Difference between SDC and DCL**

In multi-line messages, SDC messages are those that require talker or listener designation and DCL messages are those that do not require a designation. Therefore, SDC messages are directed at a particular instrument while DCL messages are directed at all instruments on the bus.

# 3.1 Component Names and Functions

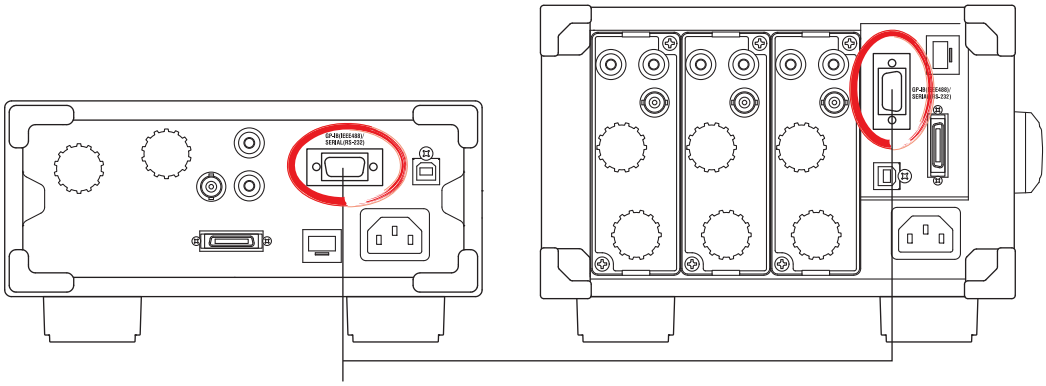
## Front Panel

**INTERFACE key (page 3-5)**
Press this key to set the handshaking, data format, baud rate, or terminator.



**LOCAL key**
Press this key to switch to local mode. In local mode, remote mode (remote control using communication commands) is cleared, and key operation becomes possible.
This key is disabled when local lockout (see page 3-2) has been activated by a controller.

## Rear Panel



**RS-232 connector**
This port is for connecting this instrument to a controller (such as a PC) using an RS-232 cable. For details on how to connect, see page 3-4.

# 3.2 RS-232 Interface Features and Specifications

## RS-232 Interface Features

### Reception Feature

- You can use the reception feature to specify the same settings that you specify by using the front panel keys.
- Output requests for measured and computed data, panel setup parameters, and error codes can be received.

### Transmission Feature

- This instrument can transmit measured and computed data.
- This instrument can transmit panel setup parameters and the status byte.
- This instrument can transmit error codes when errors occur.

## RS-232 Interface Specifications

| Item | Specifications |
|---|---|
| Electrical specifications | Complies with EIA-574 (EIA-232 (RS-232) standard for 9-pin) |
| Connection | Point to point |
| Transmission mode | Full duplex |
| Synchronization | Start-stop synchronization |
| Baud rate | 1200, 2400, 4800, 9600, 19200, 38400, 57600 |
| Start bit | 1 bit (fixed) |
| Data length | 7 or 8 bits |
| Parity | Even, odd, no parity |
| Stop bits | 1 or 2 bits |
| Connector | DELC-J9PAF-13L6 (JAE or equivalent) |
| Hardware handshaking | Select whether to use the CA and CB signals as controller lines or assume that they are always true. |
| Software handshaking | Transmission and reception can be controlled with X-ON and X-OFF signals.<br>X-ON (ASCII 11H)<br>X-OFF (ASCII 13H) |
| Receive buffer size | 256 bytes. |

## Switching between Remote and Local Modes

### When Switching from Local to Remote Mode

This instrument switches to remote mode when it is in local mode and it receives a `:COMMunicate:REMote ON` command from the PC.

- The REMOTE indicator illuminates.
- All keys except the **SHIFT (LOCAL)** key are disabled.
- Settings entered in local mode are retained even when this instrument switches to remote mode.

### When Switching from Remote to Local Mode

When this instrument is in remote mode and you press SHIFT (LOCAL), this instrument switches to local mode. However, this does not work if this instrument has received a :COMMunicate:LOCKout ON command from the PC. This instrument switches to local mode when it receives a :COMMunicate:REMote OFF command from the PC, regardless of the local lockout state.

- The REMOTE indicator turns off.
- Key operations are enabled.
- Settings entered in remote mode are retained even when this instrument switches to local mode.

***Note***

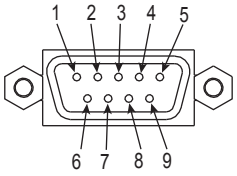
You cannot use the RS-232 interface simultaneously with other communication interfaces (USB and Ethernet interfaces).

# 3.3 Connecting to the RS-232 Interface

To connect this instrument to a PC, use an interface cable that is compatible with this instrument specifications. Be sure to align the handshaking, data transfer rate, data format, and so on with the PC. For the settings, see section 3.4.

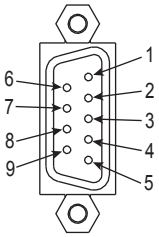
## Connector and Signal Names

WT310/WT310HC




**DELC-J9PAF-13L6 or equivalent**

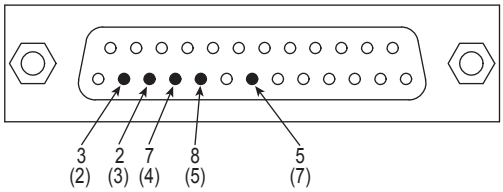
WT332/WT333




| Pin No. | Signal Name | Input or Output | Function |
|---|---|---|---|
| 2 | RD (Received Data) | Input | Data received from the PC |
| 3 | SD (Send Data) | Output | Data sent to the PC |
| 5 | SG (Signal Ground) | ---- | Signal ground |
| 7 | RS (Request to Send) | Output | Handshaking signal for receiving data from the PC |
| 8 | CS (Clear to Send) | Input | Handshaking signal for sending data to the PC |

* Pins 1, 4, 6, and 9 are not used.

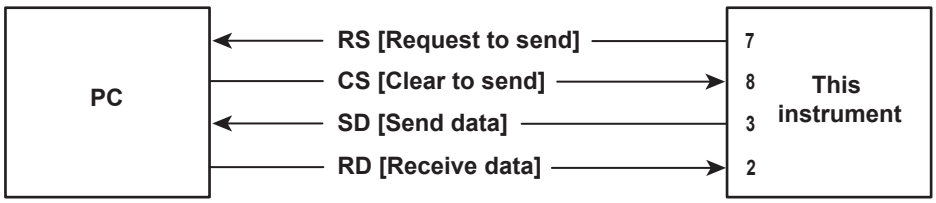
## 9-pin to 25-pin Adapter and Signal Names




Numbers in parentheses are pin numbers for the 25-pin connector.

## Signal Direction

The following figure shows the directions of the signals of this instrument RS-232 interface.



## RS-232 Standard Signals and Their JIS and CCITT Abbreviations

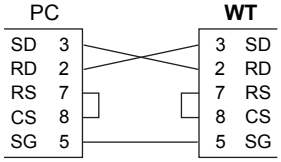
| Pin No. (9-pin connector) | Abbreviation | | | Name |
|---|---|---|---|---|
| | RS-232 | CCITT | JIS | |
| 5 | AB (GND) | 102 | SG | Signal ground |
| 3 | BA (TXD) | 103 | SD | Transmitted data |
| 2 | BB (RXD) | 104 | RD | Received data |
| 7 | CB (CTS) | 105 | RS | Request to send |
| 8 | CA (RTS) | 106 | CS | Clear to send |

## Signal Wiring Example

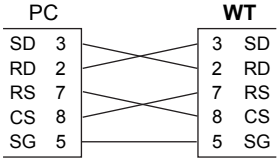
The pin numbers are for the 9-pin connector.

In general, use a crossover cable.

OFF-OFF / XON-XON



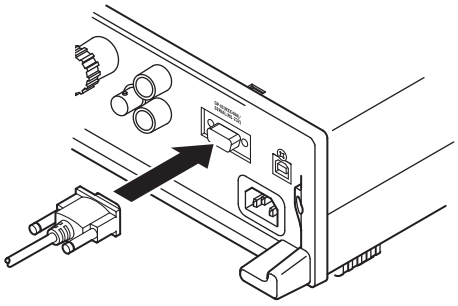

CTS-RTS (CS-RS)



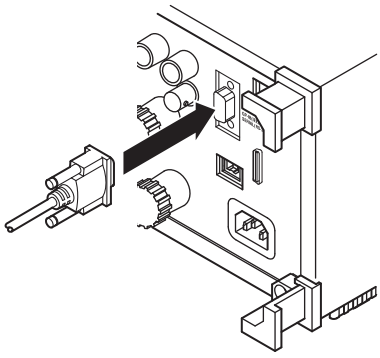

## Connection Procedure

Connect the cable as shown below.

WT310/WT310HC



WT332/WT333

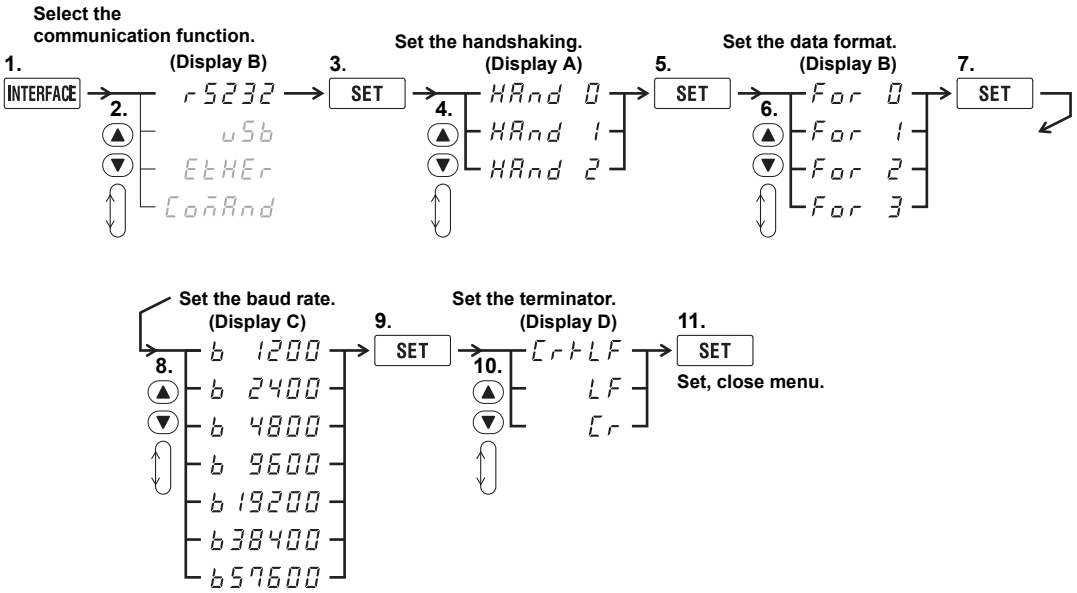


# 3.4 Configuring the RS-232 Settings of This Instrument

This section explains the following setting for controlling this instrument remotely through a RS-232 interface:

## Procedure

Follow the procedure indicated by the thick lines in the following menu.




### *Note*

Only use one communication interface: USB, RS-232, or Ethernet. If you send commands simultaneously from more than one communication interface, this instrument will not execute the commands properly.

## Explanation

# Handshaking

To use the RS-232 interface to communicate with a PC, the devices on both sides must negotiate a set of rules to ensure the proper transfer of data. This negotiation is called handshaking. Because there are many handshaking methods that can be used between this instrument and the PC, you must make sure that the same method is chosen by both this instrument and the PC.

You can choose any of handshaking methods shown below.
NO-NO, XON-XON, CS-RS

| Handshaking | | Data Transmission Control (Sending data to the PC) | | | Data Reception Control (Receiving data from the PC) | | |
|---|---|---|---|---|---|---|---|
| | | Software Handshaking | Hardware Handshaking | No handshaking | Software Handshaking | Hardware Handshaking | No handshaking |
| | WT Menu | Stop sending when X-OFF is received; resume when X-ON is received. | Stop sending when CB (CTS) is false; resume when it is true. | | Send X-OFF when the receive buffer is 3/4 full; send X-ON when it is 1/4 full. | Set CA (RTS) to false when the receive buffer is 3/4 full; set CA (RTS) to it is 1/4 full. | |
| OFF-OFF | HAnd 0 | | | Yes | | | Yes |
| XON-XON | HAnd 1 | Yes | | | Yes | | |
| CS-RS | HAnd 2 | | Yes | | | Yes | |

## OFF-OFF

### Data Transmission Control

There is no handshaking between this instrument and the PC. The “X-OFF” and “X-ON” signals are treated as data, and the CS signal is ignored.

### Data Reception Control

There is no handshaking between this instrument and the PC. When the receive buffer is full, excessive data is discarded.
RS is fixed to true.

## XON-XON

### Data Transmission Control

Software handshaking is performed between this instrument and the PC. When this instrument receives an “X-OFF” code from the PC while it is sending data, it stops the data transmission. It resumes the operation when it receives a “X-ON” code. The CS signal received from the PC is ignored.

### Data Reception Control

Software handshaking is performed between this instrument and the PC. When the free area of the receive buffer falls to 64 bytes, this instrument sends an “X-OFF” code to the PC. When the free area reaches 192 bytes, this instrument sends an “X-ON” code.
RS is fixed to true.

## CS-RS

### Data Transmission Control

Hardware handshaking is performed between this instrument and the PC. If CS is set to false while this instrument is sending data, it stops the data transmission. Then, if CS is set to true, it resumes the operation. The “X-OFF” and “X-ON” signals are treated as data.

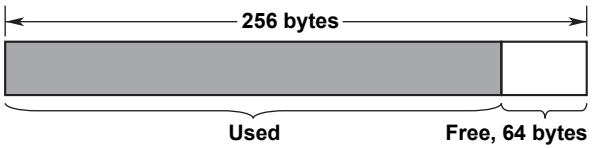
### Data Reception Control

Hardware handshaking is performed between this instrument and the PC. When the free area of the receive buffer falls to 64 bytes, this instrument sets RS to false. When the free area reaches 192 bytes, this instrument sets RS to true.

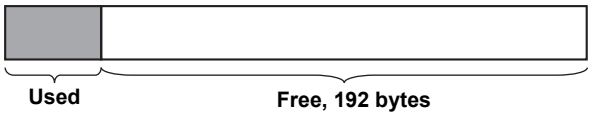
## Notes on Data Reception Control

When handshaking is used to control the reception of data, data may still be received from the PC even when the free space in the receive buffer drops below 64 bytes. In such cases, if the receive buffer becomes full, excessive data will be discarded, regardless of the handshaking method. When more space becomes available, data storage resumes.

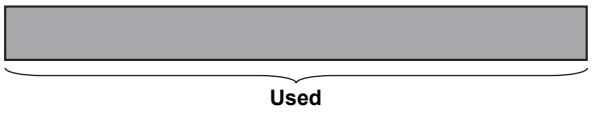
**Data reception control using handshaking**




**When handshaking is in use and the free space in the buffer drops to 64 bytes (because the data in the buffer cannot be passed to the internal program fast enough), data reception stops.**




**Data continues to be passed to the internal program. If the free space in the buffer reaches 192 bytes, data reception resumes.**



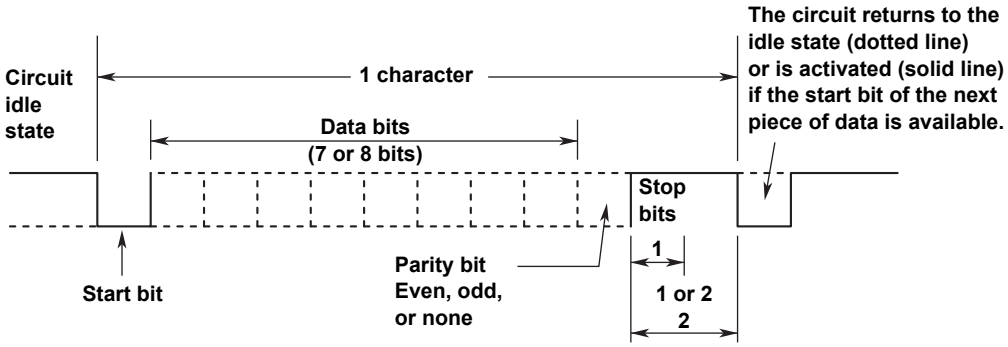

**Regardless of the handshaking method, if the buffer becomes full, excessive data will be discarded.**

***Note***

You must design PC programs so that the receive buffers of both this instrument and PC do not become full.

## Data Format

The RS-232 interface of this instrument performs communication using start-stop synchronization. In start-stop synchronization, characters are transmitted one at a time. Each character consists of a start bit, data bits, a parity bit, and a stop bit (see the following figure).




Select the data length and parity stop bit from the following options.
8-NO-1, 7-EVEN-1, 7-ODD-1, 7-NO-2

| WT Menu | Start bit | Data length | Parity | Stop bits |
|---|---|---|---|---|
| For 0 | 1 | 8 | None | 1 |
| For 1 | 1 | 7 | Odd | 1 |
| For 2 | 1 | 7 | Even | 1 |
| For 3 | 1 | 7 | None | 2 |

## Baud rate

Select from the following baud rates.
1200, 2400, 4800, 9600, 19200, 38400, 57600

## Terminator

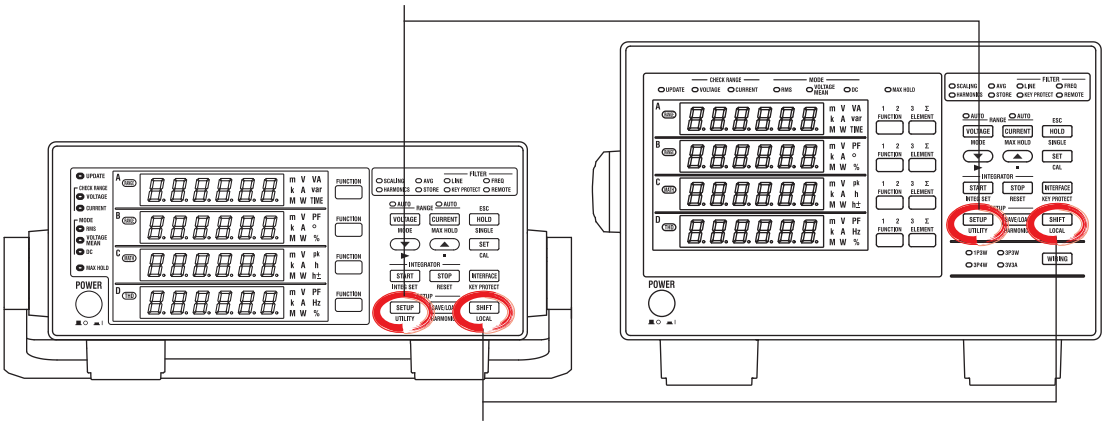
Select the terminator to use when sending data from this instrument.
Cr, Lf, Cr+Lf

Use “LF” or “Cr+Lf” for the terminator when sending data to this instrument.

# 4.1 Component Names and Functions

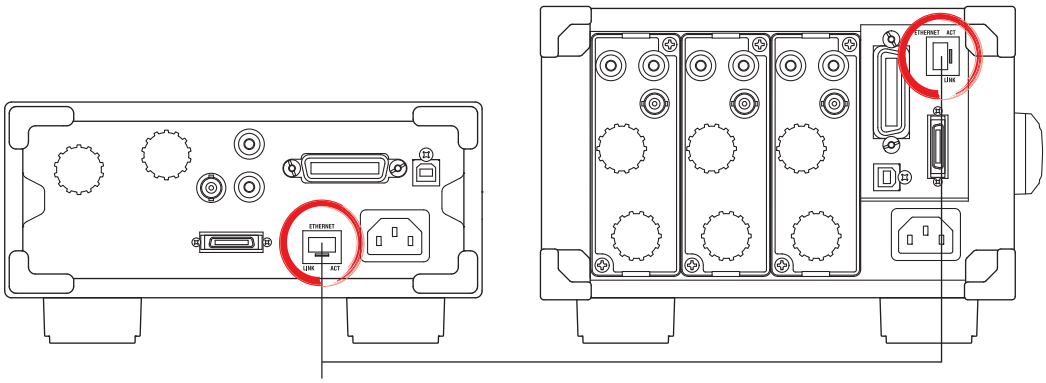
## Front Panel

**UTILITY key (page 4-4)**
Press this key to configure TCP/IP settings.



**LOCAL key**
Press this key to switch to local mode. In local mode, remote mode (remote control using communication commands) is cleared, and key operation becomes possible.
This key is disabled when local lockout (see page 4-2) has been activated by a controller.

## Rear Panel



**Ethernet port**
This port is for connecting this instrument to a controller (such as a PC) using an Ethernet cable. For details on how to connect, see page 4-3.

# 4.2 Ethernet Interface Features and Specifications

## Ethernet Interface Features

### Reception Feature

- You can use the reception feature to specify the same settings that you specify by using the front panel keys.
- Output requests for measured and computed data, panel setup parameters, and error codes can be received.

### Transmission Feature

- This instrument can transmit measured and computed data.
- This instrument can transmit panel setup parameters and the status byte.
- This instrument can transmit error codes when errors occur.

## Ethernet Interface Specifications

| Item | Specifications |
|---|---|
| Electrical and mechanical | IEEE802.3 |
| Simultaneous connections | 1 |
| Communication protocol | TCP/IP (VXI-11) |
| Connector type | RJ-45 |

## Switching between Remote and Local Modes

### When Switching from Local to Remote Mode

This instrument switches to remote mode when it is in local mode and it receives a `:COMMunicate:REMote ON` command from the PC.

- The REMOTE indicator illuminates.
- All keys except the **SHIFT (LOCAL)** key are disabled.
- Settings entered in local mode are retained even when this instrument switches to remote mode.

### When Switching from Remote to Local Mode

When this instrument is in remote mode and you press SHIFT (LOCAL), this instrument switches to local mode. However, this does not work if this instrument has received a `:COMMunicate:LOCKout ON` command from the PC. This instrument switches to local mode when it receives a `:COMMunicate:REMote OFF` command from the PC, regardless of the local lockout state.

- The REMOTE indicator turns off.
- Key operations are enabled.
- Settings entered in remote mode are retained even when this instrument switches to local mode.

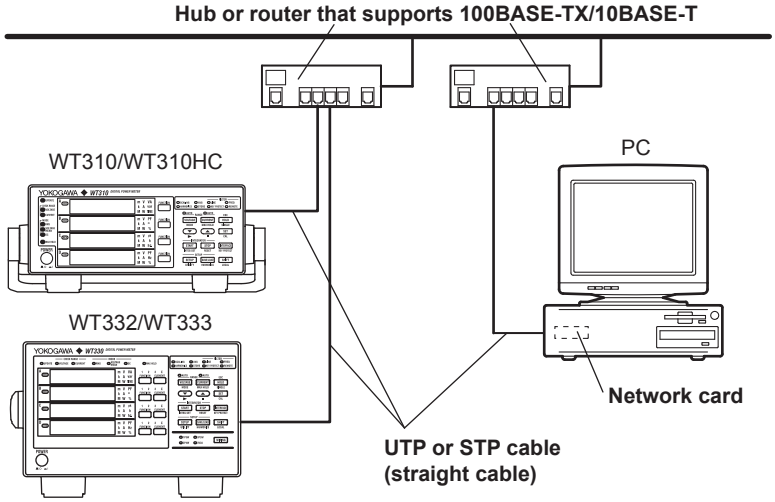
***Note***

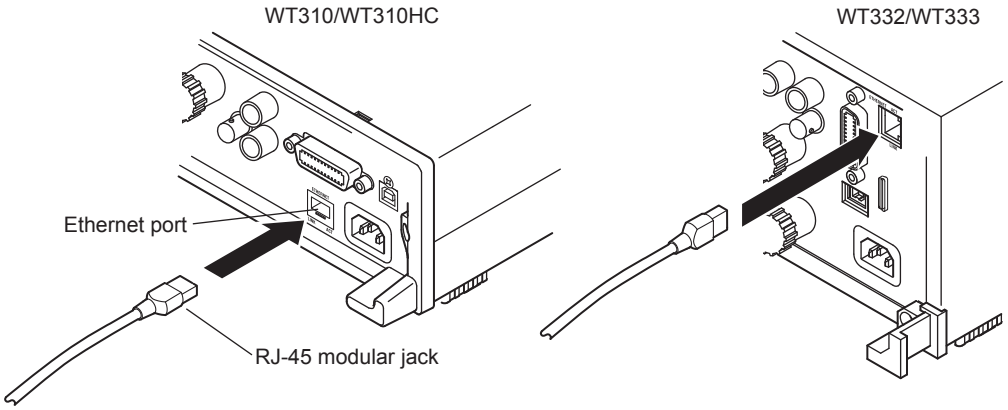
You cannot use the Ethernet interface simultaneously with other interfaces (GP-IB, RS-232, and USB).

# 4.3 Connecting to the Ethernet Interface

## Connection Procedure

Connect a UTP (Unshielded Twisted-Pair) or STP (Shielded Twisted-Pair) cable that is connected to a hub or other network device to the Ethernet port on the rear panel of this instrument.




## Notes on Connection

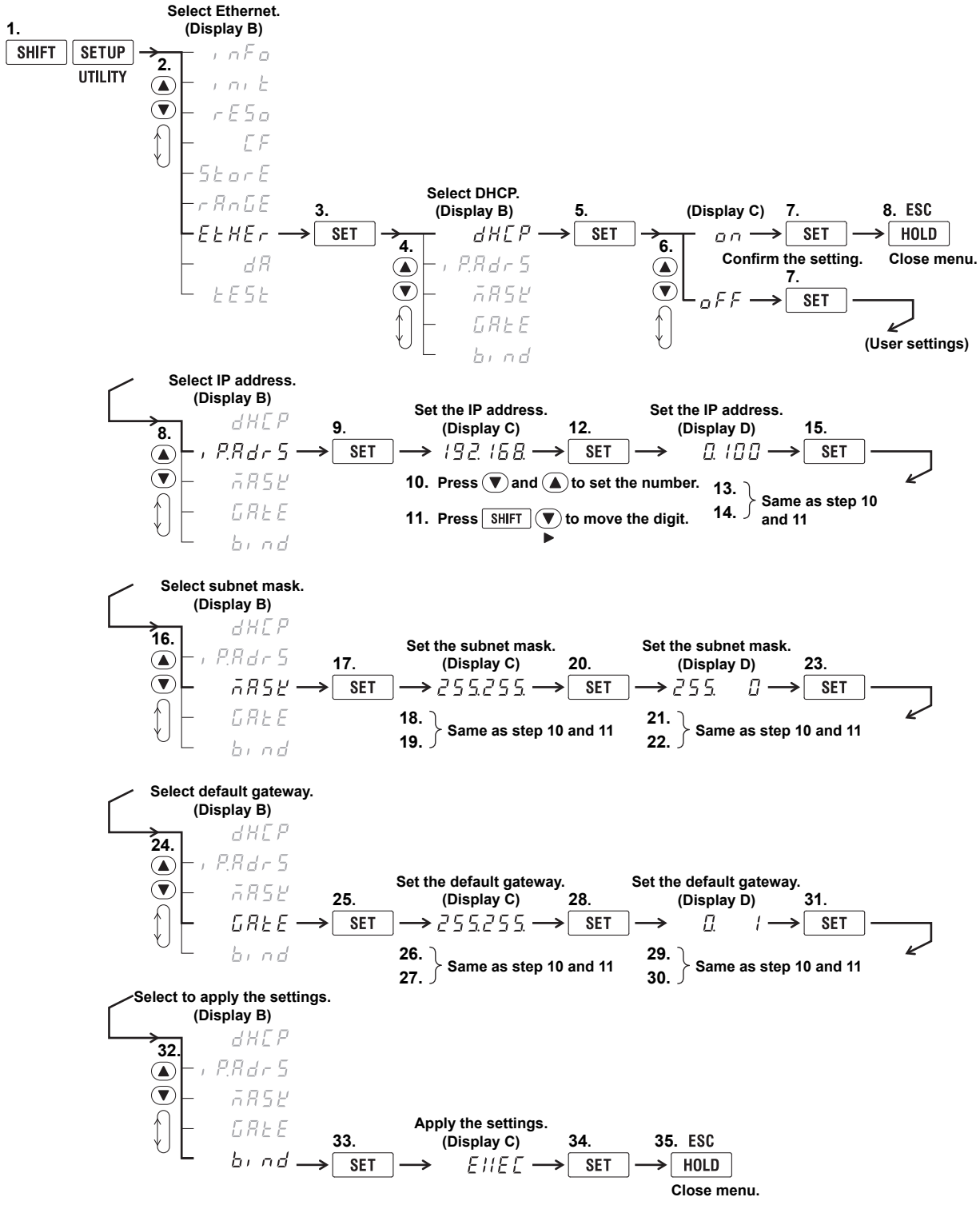
- To connect this instrument to a PC, be sure to use straight cables and to connect through a hub or router. Proper operation is not guaranteed for a one-to-one connection using a crossover cable.
- Use a network cable that supports the data rate of your network.

# 4.4 Configuring the Ethernet Settings of This Instrument

This section explains the following setting for remotely controlling this instrument via the Ethernet interface:

## Procedure

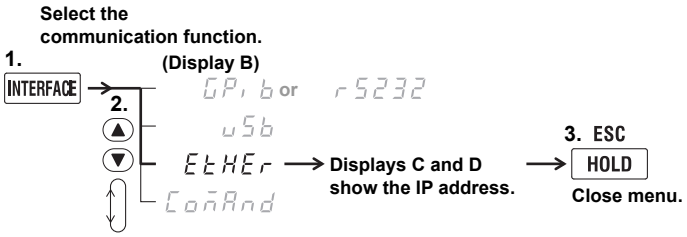
Follow the procedure indicated by the thick lines in the following menu.



***Note***

Only use one communication interface: USB, GP-IB, RS-232, or Ethernet. If you send commands simultaneously from more than one communication interface, this instrument will not execute the commands properly.

You can view the IP address that has been assigned by the DHCP server or the IP address that you specified by following the procedure below.




## Explanation

## Configuring the TCP/IP Settings

To use the Ethernet interface, you must specify TCP/IP settings.

### DHCP

DHCP is a protocol that temporarily allocates necessary information to a device so that it can connect to the Internet.

**ON**

If you are connecting this instrument to a network with a DHCP server, you can turn on the DHCP setting. If you do, the IP address will be automatically assigned to this instrument when it is connected to the network, so you do not have to set the address.

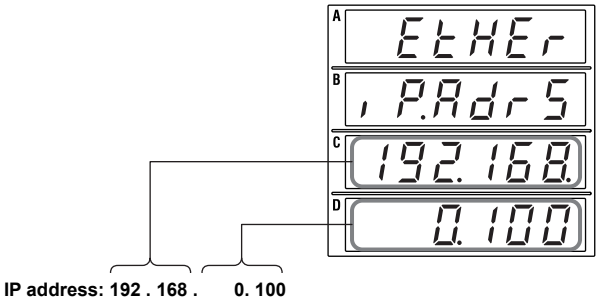
**OFF**

If you set DHCP to OFF, set the appropriate IP address, subnet mask, and default gateway for your network.

### IP Address, Subnet Mask, and Default Gateway

The IP address, subnet mask, and default gateway appear in the following positions on the displays of this instrument.

**IP address display example**




### Applying the Settings

The TCP/IP settings are applied when:

- You select EXEC on the Bind menu and press SET.
- You restart this instrument.

# 5.1 Messages

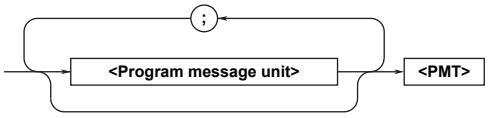
## Messages

Messages are used to exchange information between the controller and this instrument. Messages that are sent from the controller to this instrument are called program messages, and messages that are sent from this instrument back to the controller are called response messages.

If a program message contains a command that requests a response (a query), this instrument returns a response message upon receiving the program message. This instrument returns a single response message in response to a single program message.

## Program Messages

The program message format is shown below.




### <Program Message Unit>

A program message consists of one or more program message units. Each unit corresponds to one command. This instrument executes the commands in the order that they are received.

Separate each program message unit with a semicolon.

For details on the program message syntax, see the next section.

Example `:INPut:MODE RMS;CFACtor 3<PMT>`

**Unit** **Unit**

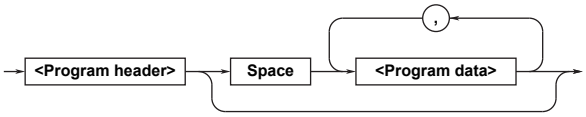
### <PMT>

This is a program message terminator. The following three types are available.

| | |
|---|---|
| `NL` (new line): | Same as LF (line feed). ASCII code "0AH." |
| `^END`: | The END message as defined by IEEE 488.1 (The data byte that is sent with the END message is the last data byte of the program message.) |
| `NL^END`: | NL with an END message attached. (NL is not included in the program message.) |

## Program Message Unit Syntax

The program message unit syntax is shown below.




### <Program Header>

The program header indicates the command type. For details, see page 5-3.

### (Program Data>

Attach program data if there are conditions that are required to execute a command. Separate the program data from the header with a space (ASCII code 20H). If there are multiple sets of program data, separate each set with a comma.

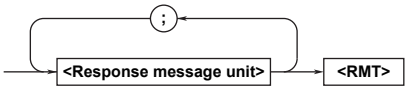
For details, see page 5-6.

Example `:INPut:CFACtor 3<PMT>`

Header Data

## Response Messages

The response message syntax is as follows:




### <Response Message Unit>

A response message consists of one or more response message units. Each unit corresponds to one response.

Separate each response message unit with a semicolon.

For details on the response message syntax, see the next page.

Example

`:INPUT:FILTER:LINE 0;FREQUENCY 0<RMT>`

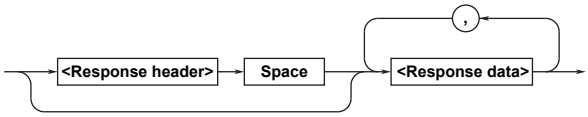
**Unit** **Unit**

### <RMT>

RMT is a response message terminator. It is `NL^END`.

## Response Message Unit Syntax

The response message unit syntax is as follows:



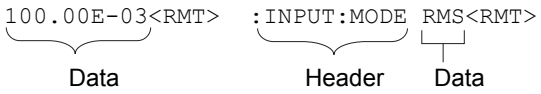

**<Response Header>**

A response header sometimes precedes the response data. Separate the data from the header with a space. For details, see page 5-5.

**<Response Data>**

Response data contains the content of the response. If there are multiple sets of response data, separate each set with a comma. For details, see page 5-5.

Example




If there are multiple queries in a program message, responses are returned in the same order that the queries were received in. In most cases, a single query returns a single response message unit, but there are a few queries that return multiple units. The first response message unit always corresponds to the first query, but the nth response unit may not necessarily correspond to the nth query. Therefore, if you want to make sure that every response is retrieved, divide the program messages into individual messages.

## Precautions to Be Taken when Sending and Receiving Messages

- If the controller sends a program message that does not contain a query, the controller can send the next program message at any time.
- If the controller sends a program message that contains a query, the controller must finish receiving the response message before it can send the next program message. If the controller sends the next program message before receiving the response message in its entirety, an error will occur. A response message that is not received in its entirety will be discarded.
- If the controller tries to receive a response message when there is none, an error will occur. If the controller tries to receive a response message before the transmission of the program message is complete, an error will occur.
- If the controller sends a program message containing multiple message units, but the message contains incomplete units, this instrument will try to execute the ones that are believed to be complete. However, these attempts may not always be successful. In addition, if such a message contains queries, this instrument may not necessary return responses.

## Deadlock

This instrument can store at least 1024 bytes of messages in its transmit and receive buffers (the number of available bytes varies depending on the operating conditions). If both the transmit and receive buffers become full at the same time, this instrument will no longer be able to operate. This condition is called a deadlock. If this happens, you can resume operation by discarding response messages. Deadlock will not occur if the program message (including the <PMT>) is kept below 1024 bytes. Program messages that do not contain queries never cause deadlocks.

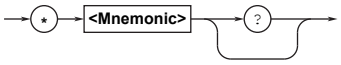
# 5.2 Commands

### Commands

There are three types of commands (program headers) that a controller may send to this instrument. The commands differ in their program header formats.

### Common Command Header

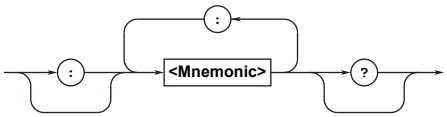
Commands that are defined in IEEE 488.2-1992 are called common commands. The common command header syntax is shown below. Be sure to include an asterisk (*) at the beginning of a common command.




Common command example: `*CLS`

### Compound Header

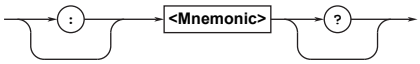
Commands, other than common commands, that are specific to this instrument are classified and arranged in a hierarchy according to their functions. The compound header syntax is shown below. Be sure to use a colon to specify a lower hierarchical level.




Compound header example: `:INPut:MODE`

### Simple Header

These commands are functionally independent and are not contained within a hierarchy. The format of a simple header is shown below.




Simple header example: `:HOLD`

***Note***

A <mnemonic> is an alphanumeric character string.

### When Concatenating Commands

- **Command Groups**
  A command group is a group of commands that have common compound headers arranged in a hierarchy. A command group may contain sub-groups.

  Example A portion of the commands from the integration command group

```
:INTEGrate?
:INTEGrate:MODE
:INTEGrate:TIMer
:INTEGrate:STARt
:INTEGrate:STOP
:INTEGrate:RESet
```

- **When Concatenating Commands of the Same Group**
  This instrument stores the hierarchical level of the command that is currently being executed and processes the next command on the assumption that it belongs to the same level. Therefore, the common header section can be omitted for commands that belong to the same group.

```
Example :INTEGrate:MODE NORMal;
        TIMer 1,0,0<PMT>
```

- **When Concatenating Commands of Different Groups**
  If the subsequent command does not belong to the same group, place a colon in front of the header (this colon cannot be omitted).

```
Example :INTEGrate:MODE NORMal;:INPut:
        MODE RMS<PMT>
```

- **When Concatenating Simple Headers**
  If a simple header follows another command, place a colon in front of the simple header (this colon cannot be omitted).

```
Example :INTEGrate:MODE NORMal;:HOLD
        ON<PMT>
```

- **When Concatenating Common Commands**
  Common commands that are defined in IEEE 488.2-1992 are independent of hierarchy. A colon is not needed before a common command.

```
Example :INTEGrate:MODE NORMal;*CLS;:
        INTEGrate:TIMer 1,0,0<PMT>
```

- **When Separating Commands with <PMT>**
  If you separate two commands with a terminator, two program messages will be sent. Therefore, the common header must be specified for each command even when commands belonging to the same command group are being concatenated.

```
Example :INTEGrate:MODE NORMal<PMT>:
        INTEGrate:TIMer 1,0,0<PMT>
```

### Upper-level Query

An upper-level query is a query that is made by appending a question mark to a command higher in the group. The controller can receive all of the settings in a group collectively by executing a highest-level query. Some query groups which are comprised of more than three hierarchical levels can output all the lower level settings.

Example : `INTEGrate?<PMT> -> :INTEGRATE :MODE NORMAL;TIMER 0,0,0<RMT>`

The response to an upper-level query can be sent back to this instrument as a program message. This enables the settings that were present when the upper-level query was made to be reproduced later on. However, some upper-level queries do not return setup parameters that are not currently in use. Exercise caution because not all of a group's information is necessarily returned in a response.

### Header Interpretation Rules

This instrument interprets the header that it receives according to the rules below.

- Mnemonics are not case sensitive.
  Example "`INPut`" can also be written as "`input`" or "`INPUT`."
- The lower-case characters can be omitted.
  Example "`INPut`" can also be written as "`INPu`" or "`INP`."
- The question mark at the end of a header indicates that it is a query. You cannot omit the question mark.
  Example The shortest abbreviation for "`INPut?`" is "`INP?`."
- If the <x> (value) at the end of a mnemonic is omitted, it is interpreted as a 1.
  Example If "`ELEMent`" is written as "`ELEM`," it means "`ELEMent1`."
- Parts of commands and parameters enclosed in square brackets ([ ]) can be omitted.
  Example "`[:INPut]:SCALing[:STATe] ON`" can be written as "`SCAL ON`."
  However, the last section enclosed in square brackets cannot be omitted in an upper-level query.
  Example "`SCALing?`" and "`SCALing:STATe?`" are different queries.

# 5.3 Responses

### Response

When the controller sends a message unit that has a question mark in its program header (a query), this instrument returns a response message to the query. This instrument returns response messages in one of the following two forms.

- **Response Consisting of a Header and Data**
  Responses that can be used as program messages without any changes are returned with command headers attached.
  Example `:INTEGrate:MODE?<PMT> ->`
  `:INTEGRATE:MODE NORMAL<RMT>`

- **Response Consisting Only of Data**
  Responses that cannot be used as program messages unless changes are made (query-only commands) are returned without headers. However, there are query-only commands whose responses this instrument will attach headers to.
  Example `INTEGrate:STATe?<PMT> ->`
  `RESET<RMT>`

### If You Want this instrument to Return Responses without Headers

You can configure this instrument so that even responses that have both headers and data are returned without headers. Use the `COMMunicate:HEADer` command for this purpose.

### Abbreviated Form

This instrument normally returns response headers with the lower-case section removed. You can configure this instrument so that full headers are returned. Use the `COMMunicate:VERBose` command for this purpose. The sections enclosed in square brackets ([ ]) are also omitted in the abbreviated form.

# 5.4 Data

## Data

Data contains conditions and values that are written after the header. A space separates the data from the header. Data is classified as follows:

| Data | Description |
|---|---|
| <Decimal> | A value expressed in decimal notation<br>(Example: VT ratio setting<br>`->[:INPut]:SCALing:VT 100`) |
| <Voltage><Current><br><Time> | A physical value<br>(Example: Voltage range setting<br>`->[:INPut]:VOLTage:RANge 150V`) |
| <Register> | A register value expressed as binary, octal, decimal, or hexadecimal<br>(Example: Extended event register value<br>`->:STATUS:EESE #HFE`) |
| <Character data> | Predefined character string (mnemonic). Select from the available strings in braces.<br>(Example: Measurement mode selection<br>`->[:INPut]:MODE {RMS\|VMEan\|DC}`) |
| <Boolean> | Indicates on and off. Specify ON, OFF, or a value.<br>(Example: Turning data hold on<br>`->:HOLD ON`) |
| <String data> | An arbitrary character string<br>(Example: Model code response<br>`->:SYSTEM:MODEL "WT310"`) |
| <Block data> | Data that contains 8-bit values<br>(Example: Measured data (binary format) response<br>`-> #40012ABCDEFGHIJKL`) |

### <Decimal>

<Decimal> indicates a value expressed as a decimal number, as shown in the table below. Decimal values are written in the NR form as specified in ANSI X3.42-1975.

| Symbol | Meaning | Example | | |
|---|---|---|---|---|
| <NR1> | Integer | `125` | `-1` | `+1000` |
| <NR2> | Fixed-point number | `125.0` | `-.90` | `+001.` |
| <NR3> | Floating-point number | `125.0E+0` | `-9E-1` | `+.1E4` |
| <NRf> | Any of the forms <NR1> to <NR3> | | | |

- This instrument can receive decimal values that are sent from the controller in any of the forms <NR1> to <NR3>. This is expressed as <NRf>.
- This instrument returns a response to the controller in one of the forms from <NR1> to <NR3> depending on the query. The same form is used regardless of the size of the value.
- For the <NR3> form, the plus sign after the “E” can be omitted. You cannot omit the minus sign.
- If a value outside the range is entered, the value is adjusted to the closest value within the range.
- If a value has more significant digits than are available, the value will be rounded.

### <Voltage>, <Current>, and <Time>

<Voltage>, <Current>, and <Time> indicate decimal values that have physical significance. A <Multiplier> or <Unit> can be attached to the  form that was described earlier. The following types of expressions are possible.

| Form | Example |
|---|---|
| <NRf><Multiplier><Unit> | `5MV` |
| <NRf><Unit> | `5E-3V` |
| <NRf><Multiplier> | `5M` |
| <NRf> | `5E-3` |

### <Multiplier>

Multipliers that you can use are indicated in the following table.

| Symbol | Word | Multiplier |
|---|---|---|
| EX | Exa | $10^{18}$ |
| PE | Peta | $10^{15}$ |
| T | Tera | $10^{12}$ |
| G | Giga | $10^{9}$ |
| MA | Mega | $10^{6}$ |
| K | Kilo | $10^{3}$ |
| M | Milli | $10^{-3}$ |
| U | Micro | $10^{-6}$ |
| N | Nano | $10^{-9}$ |
| P | Pico | $10^{-12}$ |
| F | Femto | $10^{-15}$ |

### <Unit>

Units that you can use are indicated in the following table.

| Symbol | Word | Meaning |
|---|---|---|
| V | Volt | Voltage |
| A | Ampere | Current |
| S | Second | Time |

- <Multiplier> and <Unit> are not case sensitive.
- “U” is used to indicate micro (”μ”).
- “MA” is used for Mega to distinguish it from Milli. However, “MA” is interpreted as milliampere for current.
- If both <Multiplier> and <Unit> are omitted, the basic unit (V, A, or S) is used.
- Response messages are always expressed in the <NR3> form. Additionally, they are returned in the basic form, without a multiplier or unit attached.

**<Register>**

<Register> indicates an integer, and can be expressed in hexadecimal, octal, or binary as well as a decimal number. This is used when each bit of the value has a particular meaning. The following types of expressions are possible.

| Form | Example |
|---|---|
| <NRf> | 1 |
| #H<Hexadecimal value made up of 0 to 9 and A to F> | #H0F |
| #Q<Octal value made up of 0 to 7> | #Q777 |
| #B<Binary value made up of 0 and 1> | #B001100 |

- <Register> is not case sensitive.
- Response messages are always expressed in the <NR1> form.

**<Character Data>**

<Character Data> is a specified string of character data (a mnemonic). It is mainly used to indicate options and is chosen from the character strings given in { }. The data interpretation rules are the same as those described in “Header Interpretation Rules” on page 5-4.

| Form | Example |
|---|---|
| {RMS\|VMEan\|DC} | RMS |

- As with the header, the COMMunicate:VERBose command can be used to select whether to return the response in the full form or in the abbreviated form.
- The COMMunicate:HEADer setting does not affect <Character data>.

**<Boolean>**

<Boolean> is data that indicates ON or OFF. The following types of expressions are possible.

| Form | Example |
|---|---|
| {ON\|OFF\|<NRf>} | ON OFF 1 0 |

- When <NRf> is expressed in the form, “OFF” is selected if the rounded integer value is 0, and “ON” is selected for all other cases.
- A response message is always returned with a 1 if the value is ON and with a 0 if the value is OFF.

**<String Data>**

<String data> is not a specified character string like <Character data>. It is an arbitrary character string. The character string must be enclosed in single quotation marks (') or double quotation marks (").

| Form | Example |
|---|---|
| <String data> | 'ABC' "IEEE488.2-1992" |

- If a character string contains a double quotation mark ("), the double quotation mark is expressed as two consecutive quotation marks (""). This rule also applies to single quotation marks.
- A response message is always enclosed in double quotation marks (").
- <String data> is any character string. Therefore, the instrument assumes that the remaining program message units are part of the character string if no closing single (') or double quotation mark (") is encountered. As a result, no error is detected if a quotation mark is omitted.

**<Block Data>**

<Block data> contains 8-bit values. It is only used in response messages on this instrument. The syntax is as follows:

| Form | Example |
|---|---|
| #N<N-digit decimal number><br><Data byte sequence> | #800000010ABCDEFGHIJ |

- #N
  Indicates that the data is <Block data>. N indicates the number of succeeding data bytes (digits) in ASCII code.
- <N-digit decimal number>
  Indicates the number of bytes of data (example: 00000010 = 10 bytes).
- <Data byte sequence>
  Expresses the actual data (example: ABCDEFGHIJ).
- Data is comprised of 8-bit values (0 to 255). This means that the ASCII code “0AH,” which stands for “NL,” can also be included in the data. Hence, care must be taken when programming the controller.

# 5.5 Synchronization with the Controller

### Overlap and Sequential Commands

There are two types of commands: overlap and sequential. With overlap commands, the execution of the next command may start before the execution of the previous command is finished. With sequential commands, the execution of the next command is held until the execution of the previous command is finished (even if multiple commands are sent consecutively). All commands of this instrument are sequential commands. Even when only sequential commands are available, there are times when it is necessary to achieve synchronization to properly query the measured data. For example, if you want to query the most recent numeric data each time that the measured data is updated, you can attempt to do this by sending the `:NUMeric[:NORMal]:VALue?` command with some arbitrary timing. However, because this instrument returns the current measured data regardless of whether the measured data has been updated since the previous query, this method may return data that is the same as the previous data. If this happens, you must use the following method to synchronize with the end of measured data updating.

- **Using the STATus:CONDition? Query**
  `STATus:CONDition?` is used to query the contents of the condition register (see page 7-5). You can determine whether the measured data is being updated by reading bit 0 of the condition register. If bit 0 of the condition register is 1, the measured data is being updated. If it is 0, the measured data can be queried.
  However, in the case of this instrument, it is difficult to determine the updating of measured data with `STATus:CONDition?` because the period during which bit 0 of the condition register remains at 1 is very short.

- **Using the Extended Event Register**
  The changes in the condition register can be reflected in the extended event register (see page 6-5 for details).

  Example
  ```
  :STATus:FILTer1 FALL;:STATus:EESE 1;EESR?;*SRE 8<PMT>
  (Read the response to STATus:EESR?)
  Loop
  (Wait for a service request)
  :NUMeric[:NORMal]:VALue?<PMT>
  (Read the response to :NUMeric[]NORMal]:VALue?)
  :STATus:EESR?<PMT>
  (Read the response to STATus:EESR?)
  (Return to Loop)
  ```

  The `STATus:FILTer1 FALL` command sets the transition filter so that bit 0 in the extended event (FILTer1) is set to 1 when bit 0 in the condition register changes from 1 to 0, in other words when the updating of measured data is finished.
  The `STATus:EESE 1` command is used to only change the status byte based on bit 0 in the extended event register.
  The `STATus:EESR?` command is used to clear the extended event register.
  The `*SRE 8` command is used to generate service requests based only on the changes in the extended event register bits.
  The `:NUMeric[:NORMal]:VALue?` command is not executed until a service request is generated.

- **Using the COMMunicate:WAIT Command**
  The `COMMunicate:WAIT` command is used to wait for a specific event to occur.

  Example
  ```
  :STATus:FILTer1 FALL;:STATus:EESR?<PMT>
  (Read the response to STATus:EESR?)
  Loop
  COMMunicate:WAIT 1<PMT>
  :NUMeric[:NORMal]:VALue?<PMT>
  (Read the response to :NUMeric[]NORMal]:VALue?)
  :STATus:EESR?<PMT>
  (Read the response to STATus:EESR?)
  (Return to Loop)
  ```

  For a description of `STATus:FILTer1 FALL` and `STATus:EESR?`, see the previous section about the extended event register.
  The `COMMunicate:WAIT 1` command specifies that the program will wait for bit 0 in the extended event register to be set to 1.
  `:NUMeric[:NORMal]:VALue?` is not executed until bit 0 in the extended event register becomes 1.

# 6.1 List of Commands

| Command | Function | Page |
|---|---|---|
| **AOUTput Group** | | |
| :AOUTput? | Queries all D/A output settings. | 6-5 |
| :AOUTput[:NORMal]:PRESet | Sets the D/A output items to their default values. | 6-5 |
| :AOUTput[:NORMal]:CHANnel<x> | Sets or queries a D/A output item (function or element). | 6-5 |
| :AOUTput[:NORMal]:IRTime | Sets or queries the rated integration time that is used in the D/A output of the integrated value. | 6-5 |
| :AOUTput[:NORMal]:MODE<x> | Sets or queries a D/A range mode. | 6-5 |
| :AOUTput[:NORMal]:RATE<x> | Sets or queries the maximum and minimum values for when the D/A output is in manual range mode. Sets or queries the comparison upper and lower limits for when the D/A output is in comparator mode. | 6-6 |
| **COMMunicate group** | | |
| :COMMunicate? | Queries all communication settings. | 6-7 |
| :COMMunicate:HEADer | Sets or queries whether headers are attached to query responses. | 6-7 |
| :COMMunicate:LOCKout | Sets/clears local lockout. | 6-7 |
| :COMMunicate:REMote | Sets this instrument to remote or local mode. ON is remote mode. | 6-7 |
| :COMMunicate:STATus? | Queries the line-specific status. | 6-7 |
| :COMMunicate:VERBose | Sets or queries whether the response to a query is returned fully spelled out or in its abbreviated form. | 6-7 |
| :COMMunicate:WAIT | Waits for a specified extended event to occur. | 6-7 |
| :COMMunicate:WAIT? | Creates the response that is returned when a specified extended event occurs. | 6-7 |
| **DISPlay group** | | |
| :DISPlay? | Queries all display settings. | 6-8 |
| :DISPlay:NORMal? | Queries all normal measurement data display settings. | 6-8 |
| :DISPlay[:NORMal]:ITEM<x> | Sets or queries a normal measurement data display item. | 6-8 |
| :DISPlay:HARMonics? | Queries all harmonic measurement data display settings. | 6-8 |
| :DISPlay:HARMonics:ITEM<x> | Sets or queries a harmonic measurement data display item. | 6-8 |
| **HARMonics Group** | | |
| :HARMonics? | Queries all harmonic measurement settings. | 6-10 |
| :HARMonics:PLLSource | Sets or queries the PLL source. | 6-10 |
| :HARMonics:ORDer | Sets or queries the maximum and minimum harmonic orders that are analyzed. | 6-10 |
| :HARMonics:THD | Sets or queries the equation used to compute the THD (total harmonic distortion). | 6-10 |
| :HARMonics:DISPlay? | Queries all harmonic measurement display settings. | 6-10 |
| :HARMonics:DISPlay[:STATe] | Sets or queries the on/off state of harmonic measurement data display. | 6-10 |
| :HARMonics:DISPlay:ORDer | Sets or queries the harmonic order of the harmonic component that is shown in display B for the harmonic measurement data display. | 6-10 |
| **HOLD Group** | | |
| :HOLD | Sets or queries the on/off state of the output hold feature for display, communication, and other types of data. | 6-11 |
| **INPut Group** | | |
| :INPut? | Queries all input settings. | 6-12 |
| [:INPut]:CFACtor | Sets or queries the crest factor. | 6-12 |
| [:INPut]:WIRing | Sets or queries the wiring system. | 6-12 |
| [:INPut]:MODE | Sets or queries the voltage and current measurement mode. | 6-12 |
| [:INPut]:VOLTage? | Queries all voltage measurement settings. | 6-12 |
| [:INPut]:VOLTage:RANGe | Sets or queries the voltage range. | 6-12 |
| [:INPut]:VOLTage:AUTO | Sets or queries the voltage auto range on/off state. | 6-12 |
| [:INPut]:VOLTage:CONFig | Sets or queries the valid voltage range. | 6-12 |

| Command | Function | Page |
|---|---|---|
| [:INPut]:VOLTage:POJump | Sets or queries the jump destination range that is used when a voltage peak over-range occurs. | 6-13 |
| [:INPut]:CURRent? | Queries all electric current measurement settings. | 6-13 |
| [:INPut]:CURRent:RANGe | Sets or queries the current range. | 6-13 |
| [:INPut]:CURRent:AUTO | Sets or queries the current auto range on/off state. | 6-13 |
| [:INPut]:CURRent:CONFig | Sets or queries the valid current range. | 6-13 |
| [:INPut]:CURRent:POJump | Sets or queries the jump destination range that is used when a current peak over-range occurs. | 6-13 |
| [:INPut]:CURRent:EXTSensor:CONFig | Sets or queries the valid external current sensor range. | 6-13 |
| [:INPut]:CURRent:EXTSensor:POJump | Sets or queries the jump destination range that is used when a current peak over-range occurs. | 6-14 |
| [:INPut]:CURRent:SRATio? | Queries the external current sensor conversion ratios of all elements. | 6-14 |
| [:INPut]:CURRent:SRATio[:ALL] | Collectively sets the external current sensor conversion ratios of all elements. | 6-14 |
| [:INPut]:CURRent:SRATio:ELEMent<x> | Sets or queries the external current sensor conversion ratio of the specified element. | 6-14 |
| [:INPut]:RCONfig | Sets or queries the on/off state of the range configuration (valid range selection) feature. | 6-14 |
| [:INPut]:SCALing? | Queries all scaling settings. | 6-14 |
| [:INPut]:SCALing[:STATe] | Sets or queries the scaling on/off state. | 6-14 |
| [:INPut]:SCALing:{VT\|CT\|SFACtor}? | Queries the VT ratios, CT ratios, or power coefficients of all elements. | 6-14 |
| [:INPut]:SCALing:{VT\|CT\|SFACtor}[:ALL] | Collectively sets the VT ratio, CT ratio, or power coefficient of all elements. | 6-14 |
| [:INPut]:SCALing:{VT\|CT\|SFACtor}:ELEMent<x> | Sets or queries the VT ratio, CT ratio, or power coefficient of the specified element. | 6-14 |
| [:INPut]:SYNChronize | Sets or queries the synchronization source. | 6-14 |
| [:INPut]:FILTer? | Queries all input filter settings. | 6-15 |
| [:INPut]:FILTer:LINE | Sets or queries the line filter. | 6-15 |
| [:INPut]:FILTer:FREQuency | Sets or queries the frequency filter. | 6-15 |
| [:INPut]:POVer? | Queries the peak over-range information. | 6-15 |
| [:INPut]:CRANge? | Sets or queries the check range status. | 6-15 |

## INTEGrate Group

| | | |
|---|---|---|
| :INTEGrate? | Queries all integration settings. | 6-16 |
| :INTEGrate:MODE | Sets or queries the integration mode. | 6-16 |
| :INTEGrate:TIMer | Sets or queries the integration timer value. | 6-16 |
| :INTEGrate:STARt | Starts integration. | 6-16 |
| :INTEGrate:STOP | Stops integration. | 6-16 |
| :INTEGrate:RESet | Resets the integrated value. | 6-16 |
| :INTEGrate:STATe? | Queries the integration status. | 6-16 |

## MATH Group

| | | |
|---|---|---|
| :MATH | Sets or queries the MATH equation. | 6-17 |

## MEASure Group

| | | |
|---|---|---|
| :MEASure? | Queries all measured and computed data output settings. | 6-18 |
| :MEASure:AVERaging? | Queries all averaging settings. | 6-18 |
| :MEASure:AVERaging[:STATe] | Sets or queries the on/off state of averaging. | 6-18 |
| :MEASure:AVERaging:TYPE | Sets or queries the averaging type. | 6-18 |
| :MEASure:AVERaging:COUNt | Sets or queries the averaging coefficient. | 6-18 |
| :MEASure:MHOLd | Sets the MAX hold on/off state. | 6-18 |

| Command | Function | Page |
|---|---|---|
| **NUMeric Group** | | |
| :NUMeric? | Queries all numeric data output settings. | 6-19 |
| :NUMeric:FORMat | Sets or queries the numeric data format. | 6-19 |
| :NUMeric:NORMal? | Queries all numeric data output settings. | 6-19 |
| :NUMeric[:NORMal]:VALue? | Queries the numeric data. | 6-19 |
| :NUMeric[:NORMal]:NUMber | Sets or queries the number of numeric data items that are transmitted by the :NUMeric[:NORMal]:VALue? command. | 6-19 |
| :NUMeric[:NORMal]:ITEM<x> | Sets or queries the specified numeric data output item (function, element, and harmonic order). | 6-20 |
| :NUMeric[:NORMal]:PRESet | Presets the numeric data output item pattern. | 6-20 |
| :NUMeric[:NORMal]:CLEar | Clears numeric data output items (sets the items to NONE). | 6-20 |
| :NUMeric[:NORMal]:DELete | Deletes numeric data output items. | 6-20 |
| :NUMeric[:NORMal]:HEADer? | Queries the numeric data header. | 6-20 |
| :NUMeric:LIST? | Queries all harmonic measurement numeric list data output settings. | 6-21 |
| :NUMeric:LIST:VALue? | Queries the harmonic measurement numeric list data. | 6-21 |
| :NUMeric:LIST:NUMber | Sets or queries the number of numeric list data items that are transmitted by :NUMeric:LIST:VALue?. | 6-21 |
| :NUMeric:LIST:ORDer | Sets or queries the maximum output harmonic order of the harmonic measurement numeric list data. | 6-21 |
| :NUMeric:LIST:SELect | Sets or queries the output components of the harmonic measurement numeric list data. | 6-21 |
| :NUMeric:LIST:ITEM<x> | Sets or queries the output item (function and element) of the specified harmonic measurement numeric list data item. | 6-22 |
| :NUMeric:LIST:PRESet | Presets the harmonic measurement numeric list data output item pattern. | 6-22 |
| :NUMeric:LIST:CLEar | Clears harmonic measurement numeric list data output items (sets the items to NONE). | 6-22 |
| :NUMeric:LIST:DELete | Deletes harmonic measurement numeric list data output items. | 6-22 |
| :NUMeric:HOLD | Sets or queries the on/off (hold/release) status of the numeric data hold feature. | 6-23 |
| **RATE Group** | | |
| :RATE | Sets or queries the data update interval. | 6-29 |
| **RECall Group** | | |
| :RECall:NUMber? | Queries the number of blocks of measured data that is stored. | 6-30 |
| :RECall[:NORMal]:VALue? | Queries the numeric data at the specified block number. | 6-30 |
| :RECall:LIST:VALue? | Queries the numeric list data of harmonic measurement at the specified block number. | 6-30 |
| :RECall:PANel | Loads a setup parameter file. | 6-30 |
| **STATus group** | | |
| :STATus? | Queries all the settings for the communication status feature. | 6-31 |
| :STATus:CONDition? | Queries the contents of the condition register. | 6-31 |
| :STATus:EESE | Sets or queries the extended event enable register. | 6-31 |
| :STATus:EESR? | Queries the contents of the extended event register and clears the register. | 6-31 |
| :STATus:ERRor? | Queries the error code and message of the last error that has occurred (top of the error queue). | 6-31 |
| :STATus:FILTer<x> | Sets or queries the transition filter. | 6-31 |
| :STATus:QENable | Sets or queries whether messages other than errors will be stored to the error queue (ON) or not (OFF). | 6-31 |
| :STATus:QMESsage | Sets or queries whether message information will be attached to the response to the STATus:ERRor? query (ON/OFF). | 6-31 |
| :STATus:SPOLl? | Executes serial polling. | 6-31 |

| Command | Function | Page |
|---|---|---|
| **STORe Group** | | |
| :STORe? | Queries all storage settings. | 6-32 |
| :STORe[:STATe] | Sets or queries the storage on/off state. | 6-32 |
| :STORe:INTerval | Sets or queries the storage interval. | 6-32 |
| :STORe:PANel | Saves setup parameters to a file. | 6-32 |
| **SYSTem Group** | | |
| :SYSTem? | Queries all system settings. | 6-33 |
| :SYSTem:MODel? | Queries the model code. | 6-33 |
| :SYSTem:SUFFix? | Queries the suffix code. | 6-33 |
| :SYSTem:SERial? | Queries the serial number. | 6-33 |
| :SYSTem:VERsion[:FIRMware]? | Queries the firmware version. | 6-33 |
| :SYSTem:KLOCk | Sets or queries the on/off state of the key protection. | 6-33 |
| :SYSTem:RESolution | Sets or queries the numeric data display resolution. | 6-33 |
| :SYSTem:COMMunicate:COMMand | Sets or queries the command type. | 6-33 |
| :SYSTem:COMMunicate:ETHernet:MACaddress? | Sets or queries the Ethernet MAC address. | 6-33 |
| **Common Command Group** | | |
| *CAL? | Executes zero calibration (zero-level compensation, the same operation as pressing CAL (SHIFT+SET)) and queries the result. | 6-34 |
| *CLS | Clears the standard event register, extended event register, and error queue. | 6-34 |
| *ESE | Sets or queries the standard event enable register. | 6-34 |
| *ESR? | Queries and clears the standard event register. | 6-34 |
| *IDN? | Queries the instrument model. | 6-34 |
| *OPC | Sets bit 0 (the OPC bit) of the standard event register to 1 upon the completion of the specified overlap command. | 6-34 |
| *OPC? | Returns ASCII code 1 if the specified overlap command has finished. | 6-35 |
| *OPT? | Queries the installed options. | 6-35 |
| *RST | Initializes the settings. | 6-35 |
| *SRE | Sets or queries the service request enable register value. | 6-35 |
| *STB? | Queries the Status Byte Register value. | 6-35 |
| *TRG | Executes single measurement (the same operation as when SINGLE (SHIFT+HOLD) is pressed). | 6-35 |
| *TST? | Executes a self-test and queries the result. | 6-36 |
| *WAI | Holds the execution of the subsequent command until the completion of the specified overlap command. | 6-36 |

# 6.2 AOUTput Group

The commands in this group deal with D/A output.
You can make the same settings and queries that you can by pressing the UTILITY key on the front panel and then using the dA menu or by pressing the INTEG SET key and then using the dAtimE menu. The commands in this group are only valid on models with the D/A output (/DA4 or /DA12) option.

### :AOUTput?

Function Queries all D/A output settings.
Syntax `:AOUTput?`

### :AOUTput[:NORMal]:PRESet

Function Sets the D/A output items to their default values.
Syntax `:AOUTput[:NORMal]:PRESet {NORMal|INTEGrate}`
Example `:AOUTPUT:NORMAL:PRESET NORMAL`

### :AOUTput[:NORMal]:CHANnel<x>

Function Sets or queries a D/A output item (function or element).
Syntax `:AOUTput[:NORMal]:CHANnel<x> {NONE|<Function>[,<Element>]}`
`:AOUTput[:NORMal]:CHANnel<x>?`
`<x>` = 1 to 12 (output channel)
`NONE` = no output item
`<Function>` = {U|I|P|S|Q|LAMBda|PHI|FU|FI|WH|WHP|WHM|AH|AHP|AHM|MATH|UPeak|IPeak}
`<Element>` = {<NRf>|SIGMa}
(`<NRf>` = 1 to 3)
Example
```
:AOUTPUT:NORMAL:CHANNEL1 U,1
:AOUTPUT:NORMAL:CHANNEL1?
-> :AOUTPUT:NORMAL:CHANNEL1 U,1
```
Description
- For details on <Function>, see “Function Option List (1)” on page 6-24.
- If <Element> is omitted, the element is set to 1.
- For {MATH}, <Element> does not need to be specified. In responses, <Element> is omitted.

### :AOUTput[:NORMal]:IRTime

Function Sets or queries the rated integration time that is used in the D/A output of the integrated value.
Syntax `:AOUTput[:NORMal]:IRTime {<NRf>,<NRf>,<NRf>}`
`:AOUTput[:NORMal]:IRTime?`
{<NRf>,<NRf>,<NRf>} = 0,0,0 to 10000,0,0
First <NRf> = 0 to 10000 (hour)
Second <NRf> = 0 to 59 (minute)
Third <NRf> = 0 to 59 (second)
Example
```
:AOUTPUT:NORMAL:IRTIME 1,0,0
:AOUTPUT:NORMAL:IRTIME?
-> :AOUTPUT:IRTIME 1,0,0
```

### :AOUTput[:NORMal]:MODE<x>

Function Sets or queries a D/A range mode.
Syntax `:AOUTput[:NORMal]:MODE<x> {FIXed|MANual|COMPare}`
`:AOUTput[:NORMal]:MODE<x>?`
`<x>` = 1 to 12 (output channel)
Example
```
:AOUTPUT:NORMAL:MODE1 FIXED
:AOUTPUT:NORMAL:MODE1?
-> :AOUTPUT:NORMAL:MODE1 FIXED
```
Description
- This command is supported by firmware version 1.04 or later.
- FIXed = Fixed range mode (default value)
  Outputs +5 V when the rated value of each measurement function is received.
- MANual = Manual range mode
  The displayed values of the measurement function when +5 V and –5 V are output as D/A output can be set to any values of your choice. This enables the D/A output to be expanded or reduced for each channel (D/A zoom).
- COMPare = Comparator mode
  By comparing with the comparator limits, this instrument outputs +5 V, 0 V, or –5 V. Relay contact output, like that of the WT210/WT230, can be emulated by driving a relay with the output voltage.

### :AOUTput[:NORMal]:RATE<x>

Function Sets or queries the maximum and minimum values for when the D/A output is in manual range mode. Sets or queries the comparison upper and lower limits for when the D/A output is in comparator mode.

Syntax
```
:AOUTput[:NORMal]:RATE<x>
{<NRf>,<NRf>}
:AOUTput[:NORMal]:RATE<x>?
```
<x> = 1 to 12 (output channel)
<Element> = {<NRf>|SIGMA}
<NRf> = -9.999E+12 to 9.999E+12

Example
```
:AOUTPUT:NORMAL:RATE1 100,-100
:AOUTPUT:NORMAL:RATE1?
-> :AOUTPUT:NORMAL:RATE1 100.0E+00,
-100.0E+00
```

Description
- This command is supported by firmware version 1.04 or later.
- When the D/A output is in manual range mode (:AOUTput[:NORMal]:MODE<x> MANual) Set the rated value for +5 V output and then that for –5 V output.
- When the D/A output is in comparator mode (:AOUTput[:NORMal]:MODE<x> COMPare) Set the upper limit and then the lower limit.
- When the D/A output is in fixed range mode (:AOUTput[:NORMal]:MODE<x> FIXed) There is no need to set these values. (The values do not affect the output operation.)

# 6.3 COMMunicate group

The commands in this group deal with communications.
There are no front panel keys that correspond to the commands in this group.

### :COMMunicate?

Function Queries all communication settings.
Syntax `:COMMunicate?`

### :COMMunicate:HEADer

Function Sets or queries whether headers are attached to query responses.
Syntax `:COMMunicate:HEADer {<Boolean>}`
`:COMMunicate:HEADer?`
Example `:COMMUNICATE:HEADER ON`
`:COMMUNICATE:HEADER?`
`-> :COMMUNICATE:HEADER 1`
Description Example of a response with a header
:INPUT:VOLTAGE:RANGE 150.0E+00
Example of a response without a header
150.0E+00

### :COMMunicate:LOCKout

Function Sets/clears local lockout.
Syntax `:COMMunicate:LOCKout {<Boolean>}`
`:COMMunicate:LOCKout?`
Example `:COMMUNICATE:LOCKOUT ON`
`:COMMUNICATE:LOCKOUT?`
`-> :COMMUNICATE:LOCKOUT 1`

### :COMMunicate:REMote

Function Sets this instrument to remote or local mode. ON is remote mode.
Syntax `:COMMunicate:REMote {<Boolean>}`
`:COMMunicate:REMote?`
Example `:COMMUNICATE:REMOTE ON`
`:COMMUNICATE:REMOTE?`
`-> :COMMUNICATE:REMOTE 1`

### :COMMunicate:STATus?

Function Queries the line-specific status.
Syntax `:COMMunicate:STATus?`
Example `:COMMUNICATE:STATUS?`
`-> 0`
Description The meaning of each status bit is as follows:

| Bit | RS-232 |
|---|---|
| 0 | Parity error |
| 1 | Framing error |
| 2 | Break character detection |
| 3 and higher | Always zero |

When an event occurs, the corresponding bit is set in the status. When the bit is read, it is cleared.
Zero is returned for interfaces other than RS-232.

### :COMMunicate:VERBose

Function Sets or queries whether the response to a query is returned fully spelled out or in its abbreviated form.
Syntax `:COMMunicate:VERBose {<Boolean>}`
`:COMMunicate:VERBose?`
Example `:COMMUNICATE:VERBOSE ON`
`:COMMUNICATE:VERBOSE?`
`-> :COMMUNICATE:VERBOSE 1`
Description Example of a response fully spelled out
:INPUT:VOLTAGE:RANGE 150.0E+00
Example of a response in abbreviated form
:VOLT:RANG 150.0E+00

### :COMMunicate:WAIT

Function Waits for a specified extended event to occur.
Syntax `:COMMunicate:WAIT <Register>`
<Register> = 0 to 65535 (extended event register, see page 7-5)
Example `:COMMUNICATE:WAIT 1`
Description For information about how to synchronize a program using COMMunicate:WAIT, see page 5-8.

### :COMMunicate:WAIT?

Function Creates the response that is returned when a specified extended event occurs.
Syntax `:COMMunicate:WAIT? <Register>`
<Register> = 0 to 65535 (extended event register; see page 7-5.)
Example `:COMMUNICATE:WAIT? 65535 -> 1`

# 6.4 DISPlay group

The commands in this group deal with the display.
You can make the same settings and queries that you can by using keys such as the FUNCTION and ELEMENT keys on the front panel.

**:DISPlay?**

Function Queries all display settings.
Syntax :DISPlay?

**:DISPlay:NORMal?**

Function Queries all normal measurement data display settings.
Syntax :DISPlay:NORMal?

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

Function Sets or queries a normal measurement data display item.
Syntax :DISPlay[:NORMal]:ITEM<x> <Function>[,<Element>]}
:DISPlay[:NORMal]:ITEM<x>?
<x> = 1 to 4 (display)
Function of display A (<x>=1)
<Function> = {U|I|P|S|Q|TIME}
Function of display B (<x>=2)
<Function> = {U|I|P|LAMBda|PHI}
Function of display C (<x>=3)
<Function> = {U|I|P|UPPeak|UMPeak|IPPeak|IMPeak|PPPeak|PMPeak|WH|WHP|WHM|AH|AHP|AHM|MATH}
Function of display D (<x>=4)
<Function> = {U|I|P|LAMBda|FU|FI|UTHD|ITHD}
<Element> = {<NRf>|SIGMa}
(<NRf> = 1 to 3)
Example :DISPLAY:NORMAL:ITEM1 U,1
:DISPLAY:NORMAL:ITEM1?
-> :DISPLAY:NORMAL:ITEM1 U,1
Description
- For details on <Function>, see “Numeric Data Display Functions” on the next page.
- If <Element> is omitted, the element is set to 1.
- For {TIME|MATH}, <Element> does not need to be specified. In responses, <Element> is omitted.
- {UTHD|ITHD} can be selected only on models with the harmonic measurement (/G5) option.

**:DISPlay:HARMonics?**

Function Queries all harmonic measurement data display settings.
Syntax :DISPlay:HARMonics?

**:DISPlay:HARMonics:ITEM<x>**

Function Sets or queries a harmonic measurement data display item.
Syntax :DISPlay:HARMonics:ITEM<x>{<Function>[,<Element>]}
:DISPlay:HARMonics:ITEM<x>?
<x> = 1 to 4 (display)
Function of display A (<x>=1)
<Function> = {U|I|P|ORDer}
Function of display B (<x>=2)
<Function> = {U|I|P|UHDF|IHDF|PHDF|PHIU|PHII}
Function of display C (<x>=3)
<Function> = {U|I|P}
Function of display D (<x>=4)
<Function> = {U|I|P|LAMBda|FU|FI|UTHD|ITHD}
<Element> = {<NRf>} (<NRf> = 1 to 3)
Example :DISPLAY:HARMONICS:ITEM2 I,1
:DISPLAY:HARMONICS:ITEM2?
-> :DISPLAY:HARMONICS:ITEM2 I,1
Description
- For details on <Function>, see “Numeric Data Display Functions” on the next page.
- If <Element> is omitted, the element is set to 1.
- For {ORDer}, <Element> does not need to be specified. In responses, <Element> is omitted.

### Numeric Data Display Functions

Applicable command

```
:DISPlay[:NORMal]:ITEM<x> {<Function>[,<Element>]}
```

| <Function> | Function | WT Indicator | <Element> | WT Displays | | | |
|---|---|---|---|---|---|---|---|
| | | | | A | B | C | D |
| | | | | 1 | 2 | 3 | 4 |
| U | Voltage U | [V] | Yes | Yes | Yes | Yes | Yes |
| I | Current I | [A] | Yes | Yes | Yes | Yes | Yes |
| P | Active power P | [W] | Yes | Yes | Yes | Yes | Yes |
| S | Apparent power S | [VA] | Yes | Yes | | | |
| Q | Reactive power Q | [var] | Yes | Yes | | | |
| LAMBda | Power factor λ | [PF] | Yes | | Yes | | Yes |
| PHI | Phase difference Φ | [°] | Yes | | Yes | | |
| FU | Voltage frequency fU | [V Hz] | Yes | | | | Yes |
| FI | Current frequency fI | [A Hz] | Yes | | | | Yes |
| UPPeak | Maximum voltage: U+pk | [V pk] | Yes | | | Yes | |
| UMPeak | Minimum voltage: U–pk | [V pk] | Yes | | | Yes | |
| IPPeak | Maximum current: I+pk | [A pk] | Yes | | | Yes | |
| IMPeak | Minimum current: I–pk | [A pk] | Yes | | | Yes | |
| PPPeak | Maximum power: P+pk | [W pk] | Yes | | | Yes | |
| PMPeak | Minimum power: P–pk | [W pk] | Yes | | | Yes | |
| TIME | Integration time | [TIME] | No | Yes | | | |
| WH | Watt hour WP | [W h] | Yes | | | Yes | |
| WHP | Positive watt hour WP+ | [W h±] | Yes | | | Yes | |
| WHM | Negative watt hour WP- | [W h±] | Yes | | | Yes | |
| AH | Ampere hour q | [A h] | Yes | | | Yes | |
| AHP | Positive ampere hour q+ | [A h±] | Yes | | | Yes | |
| AHM | Negative ampere hour q- | [A h±] | Yes | | | Yes | |
| MATH | Computed value, such as efficiency | [MATH] | No | | | Yes | |
| UTHD | Total harmonic distortion of voltage Uthd | [THD V %] | Yes | | | | Yes |
| ITHD | Total harmonic distortion of current Ithd | [THD A %] | Yes | | | | Yes |

Yes: Required. No: Not required.

Applicable command

```
:DISPlay:HARMonics:ITEM<x> {<Function>[,<Element>]}
```

| <Function> | Function | WT Indicator | <Element> | WT Displays | | | |
|---|---|---|---|---|---|---|---|
| | | | | A | B | C | D |
| | | | | 1 | 2 | 3 | 4 |
| ORDer | Harmonic order | “or. 01” | No | Yes | | | |
| U | Voltage U | [V] | Yes | Yes | Yes | Yes | Yes |
| I | Current I | [A] | Yes | Yes | Yes | Yes | Yes |
| P | Active power P | [W] | Yes | Yes | Yes | Yes | Yes |
| PHIU | Phase difference between harmonic voltage U(k) and the fundamental wave U(1) ΦU( ) | [V °] | Yes | | Yes | | |
| PHII | Phase difference between harmonic current I(k) and the fundamental wave I(1) ΦI( ) | [A °] | Yes | | Yes | | |
| UHDF | Harmonic voltage distortion factor Uhdf( ) | [V %] | Yes | | Yes | | |
| IHDF | Harmonic current distortion factor Ihdf( ) | [A %] | Yes | | Yes | | |
| PHDF | Harmonic active power distortion factor Phdf( ) | [W %] | Yes | | Yes | | |
| LAMBda | Power factor of fundamental signal λ(1) | [PF] | Yes | | | | Yes |
| FU | Voltage frequency fU | [V Hz] | Yes | | | | Yes |
| FI | Current frequency fI | [A Hz] | Yes | | | | Yes |
| UTHD | Total harmonic distortion of voltage Uthd | [THD V %] | Yes | | | | Yes |
| ITHD | Total harmonic distortion of current Ithd | [THD A %] | Yes | | | | Yes |

Yes: Required. No: Not required.

# 6.5 HARMonics Group

The commands in this group deal with harmonic measurement.
You can make the same settings and queries that you can make by pressing HARMONICS on the front panel.
The commands in this group are valid only on models with the harmonic measurement (/G5) option.

### :HARMonics?

Function Queries all harmonic measurement settings.
Syntax `:HARMonics?`

### :HARMonics:PLLSource

Function Sets or queries the PLL source.
Syntax `:HARMonics:PLLSource {U<x>|I<x>}`
`:HARMonics:PLLSource?`
`<x>` = 1 to 3 (element)
Example `:HARMONICS:PLLSOURCE U1`
`:HARMONICS:PLLSOURCE?`
`-> :HARMONICS:PLLSOURCE U1`

### :HARMonics:ORDer

Function Sets or queries the maximum and minimum harmonic orders that are analyzed.
Syntax `:HARMonics:ORDer {<NRf>,<NRf>}`
`:HARMonics:ORDer?`
First `<NRf>` = 1 (minimum harmonic order that is analyzed, fixed at 1)
Second `<NRf>` = 1 to 50 (maximum harmonic order that is analyzed)
Example `:HARMONICS:ORDER 1,50`
`:HARMONICS:ORDER?`
`-> :HARMONICS1:ORDER 1,50`

### :HARMonics:THD

Function Sets or queries the equation used to compute the THD (total harmonic distortion).
Syntax `:HARMonics:THD {TOTal|FUNDamental}`
`:HARMonics:THD?`
Example `:HARMONICS:THD FUNDAMENTAL`
`:HARMONICS:THD?`
`-> :HARMONICS:THD FUNDAMENTAL`

### :HARMonics:DISPlay?

Function Queries all harmonic measurement display settings.
Syntax `:HARMonics:DISPlay?`

### :HARMonics:DISPlay[:STATe]

Function Sets or queries the on/off state of harmonic measurement data display.
Syntax `:HARMonics:DISPlay[:STATe]`
`{<Boolean>}`
`:HARMonics:DISPlay:STATe?`
Example `:HARMONICS:DISPLAY:STATE ON`
`:HARMONICS:DISPLAY:STATE?`
`-> :HARMONICS:DISPLAY:STATE 1`

### :HARMonics:DISPlay:ORDer

Function Sets or queries the harmonic order of the harmonic component that is shown in display B for the harmonic measurement data display.
Syntax `:HARMonics:DISPlay:ORDer {<NRf>}`
`:HARMonics:DISPlay:ORDer?`
`<NRf>` = 1 to 50 (harmonic order)
Example `:HARMONICS:DISPLAY:ORDER 1`
`:HARMONICS:DISPLAY:ORDER?`
`-> :HARMONICS:DISPLAY:ORDER 1`

# 6.6 HOLD Group

The command in this group deals with the output data hold feature.
You can make the same settings and queries that you can make by pressing HOLD on the front panel.

**:HOLD**

Function Sets or queries the on/off state of the output hold feature for display, communication, and other types of data.

Syntax
```
:HOLD {<Boolean>}
:HOLD?
```

Example
```
:HOLD OFF
:HOLD? -> :HOLD 0
```

# 6.7 INPut Group

The commands in this group deal with the measurement conditions of the input elements.
You can make the same settings and queries that you can by pressing the WIRING, MODE, VOLTAGE, CURRENT, or UTILITY key on the front panel and then using the CF or rAnGE menu or by pressing the SETUP key and then using the SCALE, rAtio, SYnC, L.FiLt, or F.FiLt menu.

### :INPut?

Function Queries all input settings.
Syntax `:INPut?`

### [:INPut]:CFACtor

Function Sets or queries the crest factor.
Syntax `[:INPut]:CFACtor {<NRf>}`
`[:INPut]:CFACtor?`
`<NRf>` = 3, 6
Example `:INPUT:CFACTOR 3`
`:INPUT:CFACTOR?`
`-> :INPUT:CFACTOR 3`

### [:INPut]:WIRing

Function Sets or queries the wiring system.
Syntax `[:INPut]:WIRing {(P1W2|P1W3|P3W3|P3W4|V3A3)}`
`[:INPut]:WIRing?`
`P1W2` = Single-phase, two-wire system `[1P2W]`
`P1W3` = Single-phase, three-wire system `[1P3W]`
`P3W3` = Three-phase, three-wire system `[3P3W]`
`P3W4` = Three-phase, four-wire system `[3P4W]`
`V3A3` = Three-phase, three-wire system with a three-voltage, three-current method `[3V3A]`
Example `:INPUT:WIRING P1W3`
`:INPUT:WIRING?`
`-> :INPUT:WIRING P1W3`
Description
- For the WT310 and WT310HC, the wiring system is fixed to P1W2. No other setting is allowed.
- For the WT332 and WT333, the wiring system cannot be set to P1W2.

### [:INPut]:MODE

Function Sets or queries the voltage and current measurement mode.
Syntax `[:INPut]:MODE {RMS|VMEan|DC}`
`[:INPut]:MODE?`
Example `:INPUT:MODE RMS`
`:INPUT:MODE? -> :INPUT:MODE RMS`

### [:INPut]:VOLTage?

Function Queries all voltage measurement settings.
Syntax `[:INPut]:VOLTage?`

### [:INPut]:VOLTage:RANGe

Function Sets or queries the voltage range.
Syntax `[:INPut]:VOLTage:RANGe {<Voltage>}`
`[:INPut]:VOLTage:RANGe?`
- When the crest factor is set to 3
<Voltage> = 15, 30, 60, 150, 300, 600(V)
- When the crest factor is set to 6
<Voltage> = 7.5, 15, 30, 75, 150, 300(V)

Example `:INPUT:VOLTAGE:RANGE 600V`
`:INPUT:VOLTAGE:RANGE?`
`-> :INPUT:VOLTAGE:RANGE 600.0E+00`

### [:INPut]:VOLTage:AUTO

Function Sets or queries the voltage auto range on/off state.
Syntax `[:INPut]:VOLTage:AUTO {<Boolean>}`
`[:INPut]:VOLTage:AUTO?`
Example `:INPUT:VOLTAGE:AUTO ON`
`:INPUT:VOLTAGE:AUTO?`
`-> :INPUT:VOLTAGE:AUTO 1`

### [:INPut]:VOLTage:CONFig

Function Sets or queries the valid voltage range.
Syntax `[:INPut]:VOLTage:CONFig {ALL|<Voltage>[,<Voltage>][,<Voltage>]...}`
`[:INPut]:VOLTage:CONFig?`
`ALL` = All ranges are valid.
`<Voltage>` = See (`:INPut:VOLTage:RANGe`).
Example `:INPUT:VOLTAGE:CONFIG ALL`
`:INPUT:VOLTAGE:CONFIG?`
`-> :INPUT:VOLTAGE:CONFIG ALL`
`:INPUT:VOLTAGE:CONFIG 600,150,15`
`:INPUT:VOLTAGE:CONFIG?`
`-> :INPUT:VOLTAGE:CONFIG 600.0E+00, 150.0E+00,15.0E+00`
Description In the parameters, list the voltage ranges that you want to enable. To enable all the ranges, specify the parameter “ALL.”

### [:INPut]:VOLTage:POJump

Function Sets or queries the jump destination range that is used when a voltage peak over-range occurs.

Syntax
```
[:INPut]:VOLTage:POJump
{OFF|<Voltage>}
[:INPut]:VOLTage:POJump?
```
OFF = No jump destination voltage range
<Voltage> = See (:INPut:VOLTage:RANGe).

Example
```
:INPUT:VOLTAGE:POJUMP 600V
:INPUT:VOLTAGE:POJUMP?
-> :INPUT:VOLTAGE:POJUMP 600.0E+00
```

### [:INPut]:CURRent?

Function Queries all electric current measurement settings.

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

### [:INPut]:CURRent:RANGe

Function Sets or queries the current range.

Syntax
```
[:INPut]:CURRent:RANGe
{<Current>|(EXTernal,<Voltage>)}
[:INPut]:CURRent:RANGe?
```
- For direct current input
- When the crest factor is set to 3
  <Current> = 5, 10, 20, 50, 100, 200, 500(mA), 1, 2, 5, 10, 20(A) (WT310)
  <Current> = 1, 2, 5, 10, 20, 40(A) (WT310HC)
  <Current> = 0.5, 1, 2, 5, 10, 20(A) (WT332, WT333)
- When the crest factor is set to 6
  <Current> = 2.5, 5, 10, 25, 50, 100, 250(mA), 0.5, 1, 2.5, 5, 10(A) (WT310)
  <Current> = 0.5, 1, 2.5, 5, 10, 20(A) (WT310HC)
  <Current> = 0.25, 0.5, 1, 2.5, 5, 10(A) (WT332, WT333)
- For external current sensor input
- When the crest factor is set to 3
  <Voltage> = 2.5, 5, 10(V) (/EX1)
  <Voltage> = 50, 100, 200, 500(mV), 1, 2(V) (/EX2)
- When the crest factor is set to 6
  <Voltage> = 1.25, 2.5, 5(V) (/EX1)
  <Voltage> = 25, 50, 100, 250(mV), 0.5, 1(V) (/EX2)

Example
```
:INPUT:CURRENT:RANGE 20A
:INPUT:CURRENT:RANGE?
-> :INPUT:CURRENT:RANGE 20.0E+00
:INPUT:CURRENT:RANGE EXTERNAL,10V
:INPUT:CURRENT:RANGE?
-> :INPUT:CURRENT:RANGE
EXTERNAL,10.0E+00
```

Description EXTernal and <Voltage> can only be selected on models with the external current sensor input (/EX1 or /EX2) option.

### [:INPut]:CURRent:AUTO

Function Sets or queries the current auto range on/off state.

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

Example
```
:INPUT:CURRENT:AUTO ON
:INPUT:CURRENT:AUTO?
-> :INPUT:CURRENT:AUTO 1
```

### [:INPut]:CURRent:CONFig

Function Sets or queries the valid current range.

Syntax
```
[:INPut]:CURRent:CONFig {ALL|<Current>
[,<Current>][,<Current>]...}
[:INPut]:CURRent:CONFig?
```
ALL = All ranges are valid.
<Current> = See (:INPut:CURRent:RANGe).

Example
```
:INPUT:CURRENT:CONFIG ALL
:INPUT:CURRENT:CONFIG?
-> :INPUT:CURRENT:CONFIG ALL
:INPUT:CURRENT:CONFIG 20,5,1
:INPUT:CURRENT:CONFIG?
-> :INPUT:CURRENT:CONFIG
20.0E+00,5.0E+00,1.0E+00
```

Description In the parameters, list the current ranges that you want to enable. To enable all the ranges, specify the parameter “ALL.”

### [:INPut]:CURRent:POJump

Function Sets or queries the jump destination range that is used when a current peak over-range occurs.

Syntax
```
[:INPut]:CURRent:POJump
{OFF|<Current>}
[:INPut]:CURRent:POJump?
```
OFF = No jump destination current range
<Current> = See (:INPut:CURRent:RANGe).

Example
```
:INPUT:CURRENT:POJUMP 20A
:INPUT:CURRENT:POJUMP?
-> :INPUT:CURRENT:POJUMP 20.0E+00
```

### [:INPut]:CURRent:EXTSensor:CONFig

Function Sets or queries the valid external current sensor range.

Syntax
```
[:INPut]:CURRent:EXTSensor:CONFig
{ALL|<Voltage>[,<Voltage>]
[,<Voltage>]...}
[:INPut]:CURRent:EXTSensor:CONFig?
```
ALL = All ranges are valid.
<Voltage> = See (:INPut:CURRent:RANGe).

Example
```
:INPUT:CURRENT:EXTSENSOR:CONFIG ALL
:INPUT:CURRENT:EXTSENSOR:CONFIG? ->
:INPUT:CURRENT:EXTSENSOR:CONFIG ALL
:INPUT:CURRENT:EXTSENSOR:CONFIG
2,0.5,0.1
:INPUT:CURRENT:EXTSENSOR:CONFIG?
-> :INPUT:CURRENT:EXTSENSOR:CONFIG
2.00E+00,500.0E-03,100.0E-03
```

Description In the parameters, list the external current sensor ranges that you want to enable. To enable all the ranges, specify the parameter “ALL.”

### [:INPut]:CURRent:EXTSensor:POJump

Function Sets or queries the jump destination range that is used when a current peak over-range occurs.

Syntax
```
[:INPut]:CURRent:EXTSensor:POJump
{OFF|<Voltage>}
[:INPut]:CURRent:EXTSensor:POJump?
```
OFF = No jump destination current range
<Voltage> = See (:INPut:CURRent:RANGe).

Example
```
:INPUT:CURRENT:EXTSENSOR:POJUMP 2V
:INPUT:CURRENT:EXTSENSOR:POJUMP?
-> :INPUT:CURRENT:EXTSENSOR:POJUMP
2.00E+00
```

### [:INPut]:CURRent:SRATio?

Function Queries the external current sensor conversion ratios of all elements.

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

Description This command is only valid on models with the external current sensor input (/EX1 or /EX2) option.

### [:INPut]:CURRent:SRATio[:ALL]

Function Collectively sets the external current sensor conversion ratios of all elements.

Syntax
```
[:INPut]:CURRent:SRATio[:ALL]
{<NRf>}
```
<NRf> = 0.001 to 9999.

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

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

Function Sets or queries the external current sensor conversion ratio of the specified element.

Syntax
```
[:INPut]:CURRent:SRATio:ELEMent<x>
{<NRf>}
[:INPut]:CURRent:SRATio:ELEMent<x>?
```
<x> = 1 to 3 (element)
<NRf> = 0.001 to 9999.

Example
```
:INPUT:CURRENT:SRATIO:ELEMENT1 10
:INPUT:CURRENT:SRATIO:ELEMENT1?
-> :INPUT:CURRENT:SRATIO:ELEMENT1
10.00
```

### [:INPut]:RCONfig

Function Sets or queries the on/off state of the range configuration (valid range selection) feature.

Syntax
```
[:INPut]:RCONfig {<Boolean>}
[:INPut]:RCONfig?
```

Example
```
:INPUT:RCONFIG OFF
:INPUT:RCONFIG? -> :INPUT:RCONFIG 0
```

Description The following commands are enabled only when this command is set to ON. Measurement range can be skipped.
```
:INPut:VOLTage:CONFig
:INPut:VOLTage:POJump
:INPut:CURRent:CONFig
:INPut:CURRent:POJump
:INPut:CURRent:EXTSensor:CONFig
:INPut:CURRent:EXTSensor:POJump
```

### [:INPut]:SCALing?

Function Queries all scaling settings.

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

### [:INPut]:SCALing[:STATe]

Function Sets or queries the scaling on/off state.

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

Example
```
:INPUT:SCALING:STATE OFF
:INPUT:SCALING:STATE?
-> :INPUT:SCALING:STATE 0
```

### [:INPut]:SCALing:{VT|CT|SFACtor}?

Function Queries the VT ratios, CT ratios, or power coefficients of all elements.

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

### [:INPut]:SCALing:{VT|CT|SFACtor}[:ALL]

Function Collectively sets the VT ratio, CT ratio, or power coefficient of all elements.

Syntax
```
[:INPut]:SCALing:{VT|CT|SFACtor}
[:ALL] {<NRf>}
```
<NRf> = 0.001 to 9999.

Example
```
:INPUT:SCALING:VT:ALL 1
```

### [:INPut]:SCALing:{VT|CT|SFACtor}:ELEMent<x>

Function Sets or queries the VT ratio, CT ratio, or power coefficient of the specified element.

Syntax
```
[:INPut]:SCALing:{VT|CT|SFACtor}
:ELEMent<x> {<NRf>}
[:INPut]:SCALing:{VT|CT|SFACtor}
:ELEMent<x>?
```
<x> = 1 to 3 (element)
<NRf> = 0.001 to 9999.

Example
```
:INPUT:SCALING:VT:ELEMENT1 1
:INPUT:SCALING:VT:ELEMENT1?
-> :INPUT:SCALING:VT:ELEMENT1 1.000
```

### [:INPut]:SYNChronize

Function Sets or queries the synchronization source.

Syntax
```
[:INPut]:SYNChronize
{VOLTage|CURRent|OFF}
[:INPut]:SYNChronize?
```

Example
```
:INPUT:SYNCHRONIZE VOLTAGE
:INPUT:SYNCHRONIZE?
-> :INPUT:SYNCHRONIZE VOLTAGE
```

### [:INPut]:FILTer?

Function Queries all input filter settings.

Syntax `[:INPut]:FILTer?`

### [:INPut]:FILTer:LINE

Function Sets or queries the line filter.

Syntax
```
[:INPut]:FILTer:LINE {<Boolean>}
[:INPut]:FILTer:LINE?
```

Example
```
:INPUT:FILTER:LINE OFF
:INPUT:FILTER:LINE?
-> :INPUT:FILTER:LINE 0
```

### [:INPut]:FILTer:FREQuency

Function Sets or queries the frequency filter.

Syntax
```
[:INPut]:FILTer:FREQuency
{<Boolean>}
[:INPut]:FILTer:FREQuency?
```

Example
```
:INPUT:FILTER:FREQUENCY OFF
:INPUT:FILTER:FREQUENCY?
-> :INPUT:FILTER:FREQUENCY 0
```

### [:INPut]:POVer?

Function Queries the peak over-range information.

Syntax `[:INPut]:POVer?`

Example `:INPUT:POVER? -> 0`

Description
- The peak over-range information of each element is mapped as shown below. For the response, the sum of the values of each bit is returned in decimal format.
- For example, a response of 16 indicates that a peak over-range is occurring at U3.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | I3 | U3 | I2 | U2 | I1 | U1 |

### [:INPut]:CRANge?

Function Sets or queries the check range status.

Syntax `[:INPut]:CRANge?`

Example `:INPUT:CRANGE? -> 0`

Description
- The CHECK RANGE LED status is mapped as shown below. For the response, the sum of the values of each bit is returned in decimal format.
- For example, a response of 0 indicates appropriate range. A response of 64 indicates that a current over-range is occurring.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AP | AO | AH | AL | VP | VO | VH | VL |

VL : The voltage is at the condition for reducing the auto range or less.
VH : The voltage exceeds the condition for raising the auto range.
VO : The voltage is over-range.
VP : The voltage is peak over-range.
AL : The current is at the condition for reducing the auto range or less.
AH : The current exceeds the condition for raising the auto range.
AO : The current is over-range.
AP : The current is peak over-range.

- For information on the color of the LED and the condition to light, see “Auto Range Monitor Indications” in section 1.5 in the Getting Started Guide, IM WT310-02EN.

# 6.8 INTEGrate Group

The commands in this group deal with integration.
You can make the same settings and queries that you can make by pressing INTEG SET, START, STOP, and RESET on the front panel.

**:INTEGrate?**
Function Queries all integration settings.
Syntax :INTEGrate?

**:INTEGrate:MODE**
Function Sets or queries the integration mode.
Syntax :INTEGrate:MODE {NORMal|CONTinuous}
:INTEGrate:MODE?
NORMal = Standard integration mode
CONTinuous = Continuous integration mode
Example :INTEGRATE:MODE NORMAL
:INTEGRATE:MODE?
-> :INTEGRATE:MODE NORMAL

**:INTEGrate:TIMer**
Function Sets or queries the integration timer value.
Syntax :INTEGrate:TIMer {<NRf>,<NRf>,<NRf>}
:INTEGrate:TIMer?
{<NRf>,<NRf>,<NRf>} = 0,0,0 to 10000,0,0
First <NRf> = 0 to 10000 (hours)
Second <NRf> = 0 to 59 (minutes)
Third <NRf> = 0 to 59 (seconds)
Example :INTEGRATE:TIMER 1,0,0
:INTEGRATE:TIMER?
-> :INTEGRATE:TIMER 1,0,0

**:INTEGrate:STARt**
Function Starts integration.
Syntax :INTEGrate:STARt
Example :INTEGRATE:START

**:INTEGrate:STOP**
Function Stops integration.
Syntax :INTEGrate:STOP
Example :INTEGRATE:STOP

**:INTEGrate:RESet**
Function Resets the integrated value.
Syntax :INTEGrate:RESet
Example :INTEGRATE:RESET

**:INTEGrate:STATe?**
Function Queries the integration status.
Syntax :INTEGrate:STATe?
Example :INTEGRATE:STATE? -> RESET
Description • The response is as follows:
RESet = Integration reset
STARt = Integration in progress
STOP = Integration stop
ERRor = Abnormal integration termination (integration overflow, power failure)
TIMeup = Integration stop due to integration timeout

# 6.9 MATH Group

The commands in this group deal with computations. You can make the same settings and queries that you can by pressing SETUP on the front panel and using the MAtH menu.

**:MATH**

Function Sets or queries the MATH equation.

Syntax `:MATH {EFFiciency|CFU<x>|CFI<x>|ADD|SUB|MUL|DIV|DIVA|DIVB|AVW<x>}`
`:MATH?`
`<x>` of `{CFU|CFI} = 1 to 3` (element)
`<x>` of `{AVW} = 1 to 3` (element), `4 (Σ)`

Example `:MATH CFU1`
`:MATH? -> :MATH CFU1`

Description The equations that correspond to each option are as follows:
EFFiciency : Efficiency (valid only on the WT332/WT333)
CFU : Voltage crest factor
CFI : Current crest factor
ADD : A+B
SUB : A-B
MUL : A×B
DIV : A/B
DIVA: A/B^2
DIVB: A^2/B
AVW : Average active power during integration

# 6.10 MEASure Group

The commands in this group deal with computation.
You can make the same settings and queries that you can by pressing SETUP on the front panel and using the AVG menu or by pressing MAX HOLD on the front panel.

### :MEASure?

Function Queries all measured and computed data output settings.
Syntax `:MEASure?`

### :MEASure:AVERaging?

Function Queries all averaging settings.
Syntax `:MEASure:AVERaging?`

### :MEASure:AVERaging[:STATe]

Function Sets or queries the on/off state of averaging.
Syntax `:MEASure:AVERaging[:STATe] {<Boolean>}`
`:MEASure:AVERaging:STATe?`
Example `:MEASURE:AVERAGING:STATE ON`
`:MEASURE:AVERAGING:STATE?`
`-> :MEASURE:AVERAGING:STATE 1`

### :MEASure:AVERaging:TYPE

Function Sets or queries the averaging type.
Syntax `:MEASure:AVERaging:TYPE {LINear|EXPonent}`
`:MEASure:AVERaging:TYPE?`
Example `:MEASURE:AVERAGING:TYPE LINEAR`
`:MEASURE:AVERAGING:TYPE?`
`-> :MEASURE:AVERAGING:TYPE LINEAR`
Description The averaging of harmonic measurement functions (option) is only valid when the type is set to EXPonent. For details, see the User's Manual, IM WT310-01EN.

### :MEASure:AVERaging:COUNt

Function Sets or queries the averaging coefficient.
Syntax `:MEASure:AVERaging:COUNt {<NRf>}`
`:MEASure:AVERaging:COUNt?`
`<NRf>` = `8`, `16`, `32`, `64` (moving average count or attenuation constant)
Example `:MEASURE:AVERAGING:COUNT 8`
`:MEASURE:AVERAGING:COUNT?`
`-> :MEASURE:AVERAGING:COUNT 8`
Description The averaging of harmonic measurement functions (option) is only valid when TYPE is set to EXPonent (attenuation constant). For details, see the User's Manual, IM WT310-01EN.

### :MEASure:MHOLd

Function Sets the MAX hold on/off state.
Syntax `:MEASure:MHOLd {<Boolean>}`
`:MEASure:MHOLd?`
Example `:MEASURE:MHOLD ON`
`:MEASURE:MHOLD? -> :MEASURE:MHOLD 1`

# 6.11 NUMeric Group

The command in this group deal with numeric data output.
There are no front panel keys that correspond to the commands in this group.
The commands in the DISPlay group are used to make the same settings and queries as the FUNCTION and ELEMENT keys on the front panel.

### :NUMeric?

Function Queries all numeric data output settings.
Syntax `:NUMeric?`

### :NUMeric:FORMat

Function Sets or queries the numeric data format.
Syntax `:NUMeric:FORMat {ASCii|FLOat}`
`:NUMeric:FORMat?`
Example `:NUMERIC:FORMAT ASCII`
`:NUMERIC:FORMAT?`
`-> :NUMERIC:FORMAT ASCII`
Description
- The format of the numeric data that is output varies depending on how this command is set. The different formats are explained below.

(1) ASCii
Physical values are output in the <NR3> format.(Only the elapsed integration time—TIME—is output in <NR1> format.)
The data items are separated by commas.

(2) FLOat
A header (for example, "#240" or "#3208") is added in front of each numeric data block.
A physical value in IEEE single-precision floating point (4-byte) format follows the header.
The byte order of the data of each item is MSB First.

- For the formats of each individual numeric data item, see "Numeric Data Formats" at the end of this group of commands (page 6-25).

### :NUMeric:NORMal?

Function Queries all numeric data output settings.
Syntax `:NUMeric:NORMal?`
Description The number of numeric data items output by :NUMeric[:NORMal]:ITEM<x> is determined by :NUMeric[:NORMal]NUMber.

### :NUMeric[:NORMal]:VALue?

Function Queries the numeric data.
Syntax `:NUMeric[:NORMal]:VALue? {<NRf>}`
`<NRf>` = 1 to 255 (item number)
Example
- When `<NRf>` is specified
  `:NUMERIC:NORMAL:VALUE? 1`
  `-> 103.79E+00`
- When `<NRf>` is omitted
  `:NUMERIC:NORMAL:VALUE?`
  `-> 103.79E+00,1.0143E+00,105.27E+00,..(omitted)..,50.001E+00`
- When `:NUMeric:FORMat` is set to `{FLOat}`
  `:NUMERIC:NORMAL:VALUE?`
  `-> #N` (N-digit byte number)(data byte sequence)

Description
- If <NRf> is specified, only the numeric data for the specified item is output.
- If <NRf> is omitted, the numeric data items from 1 to the number specified by the : NUMeric[:NORMal]:NUMber command are output in order.
- For the formats of each individual numeric data output item, see "Numeric Data Formats" at the end of this group of commands (page 6-25).

### :NUMeric[:NORMal]:NUMber

Function Sets or queries the number of numeric data items that are transmitted by the :NUMeric[:NORMal]:VALue? command.
Syntax `:NUMeric[:NORMal]:NUMber {<NRf>|ALL}`
`:NUMeric[:NORMal]:NUMber?`
`<NRf>` = 1 to 255(ALL)
Example `:NUMERIC:NORMAL:NUMBER 10`
`:NUMERIC:NORMAL:NUMBER`
`-> :NUMERIC:NORMAL:NUMBER 10`
Description
- If the parameter is omitted from the :NUMeric[:NORMal]:VALue? command, the numeric data items from 1 to the specified value are output in order.
- By default, the number of numeric data items is set to 10.

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

Function Sets or queries the specified numeric data output item (function, element, and harmonic order).

Syntax
```
:NUMeric[:NORMal]:ITEM<x> {NONE|
<Function>[,<Element>][,<Order>]}
:NUMeric[:NORMal]:ITEM<x>?
```
<x> = 1 to 255 (item number)
NONE = No output item
<Function> = {U|I|P|S|Q|...}
<Element> = {<NRf>|SIGMa}
(<NRf> = 1 to 3)
<Order> = {TOTal|DC|<NRf>}
(<NRf> = 1 to 50)

Example
```
:NUMERIC:NORMAL:ITEM1 U,1
:NUMERIC:NORMAL:ITEM1?
-> :NUMERIC:NORMAL:ITEM1 U,1
:NUMERIC:NORMAL:ITEM1 UK,1,1
:NUMERIC:NORMAL:ITEM1?
-> :NUMERIC:NORMAL:ITEM1 UK,1,1
```

Description
- For details on <Function> options, see "Function Option List (1)" at the end of this group of commands (page 6-24).
- If <Element> is omitted, the element is set to 1.
- If <Order> is omitted, the order is set to TOTal.
- <Element> and <Order> are omitted from responses to functions that do not need them.
- This instrument does not measure data for <Order> = DC.

### :NUMeric[:NORMal]:PRESet

Function Presets the numeric data output item pattern.

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

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

Description
- For information about the output items that are preset, see "(1) Preset Patterns for Numeric Data Items" on page 6-26 at the end of the commands for this group.
- By default, the output items of Pattern 2 are selected.

### :NUMeric[:NORMal]:CLEar

Function Clears numeric data output items (sets the items to NONE).

Syntax
```
:NUMeric[:NORMal]:CLEar
{ALL|<NRf>[,<NRf>]}
```
ALL = Clear all items
First <NRf> = 1 to 255 (the number of the first item to clear)
Second <NRf> = 1 to 255 (the number of the last item to clear)

Example
```
:NUMERIC:NORMAL:CLEAR ALL
```

Description If the 2nd <NRf> is omitted, the output item specified by the first and all following output items (up to number 255) are cleared.

### :NUMeric[:NORMal]:DELete

Function Deletes numeric data output items.

Syntax
```
:NUMeric[:NORMal]:DELete
{<NRf>[,<NRf>]}
```
1st <NRf> = 1 to 255 (the number of the first item to delete)
Second <NRf> = 1 to 255 (the number of the last item to delete)

Example
```
:NUMERIC:NORMAL:DELETE 1
```
(Deletes ITEM1 and shifts ITEM2 and subsequent items forward)
```
:NUMERIC:NORMAL:DELETE 1,3
```
(Deletes ITEM1 to ITEM3 and shifts ITEM4 and subsequent items forward)

Description
- When output items are deleted, subsequent items shift forward to fill the empty positions. Empty positions at the end are set to NONE.
- If the second <NRf> is omitted, only the output item specified by the first number is deleted.

### :NUMeric[:NORMal]:HEADer?

Function Queries the numeric data header.

Syntax
```
:NUMeric[:NORMal]:HEADer? {<NRf>}
```
<NRf> = 1 to 255 (item number)

Example
- When <NRf> is specified
```
:NUMERIC:NORMAL:HEADER? 1
-> U-E1
```
- When <NRf> is omitted (when :NUMeric[:NORMal]:NUMber is set to 3)
```
:NUMERIC:NORMAL:HEADER?
-> U-E1,I-E1,P-E1
```

Description
- The data name (header) of the output item is generated.
- If <NRf> is specified, only the data name for the specified item number is output.
- If <NRf> is omitted, the data names of the items from 1 to the number specified by the : NUMeric[:NORMal]:NUMber command are output in order.

### :NUMeric:LIST?

Function Queries all harmonic measurement numeric list data output settings.

Syntax `:NUMeric:LIST?`

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- The number of numeric list data items output by :NUMeric:LIST:ITEM<x> is determined by :NUMeric:LIST:NUMber.

### :NUMeric:LIST:VALue?

Function Queries the harmonic measurement numeric list data.

Syntax `:NUMeric:LIST:VALue? {<NRf>}`
`<NRf>` = 1 to 32 (item number)

Example
- When <NRf> is specified
  `:NUMERIC:LIST:VALUE? 1`
  `-> 103.58E+00,NAN,103.53E+00`
  `,0.09E+00,2.07E+00,0.04E+00,..`
  (omitted)`..,0.01E+00,0.01E+00`
  (up to 52 data values)
- When `<NRf>` is omitted (when `:NUMeric:LIST:NUMber` is set to 5)
  `:NUMERIC:LIST:VALUE?`
  `-> 103.58E+00,NAN,103.53E+00`
  `,0.09E+00,2.07E+00,0.04E+00,..`
  (omitted)`..,0.00E+00,0.00E+00`
  (up to 52*5 = 260 data values)
- When `:NUMeric:FORMat` is set to `{FLOat}`
  `:NUMERIC:LIST:VALUE?`
  `-> #N` (N-digit byte number)(data byte sequence)

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- A single numeric list data item consists of up to 52 items of numeric data in the following order: TOTal, DC, 1st harmonic, ..., :NUMeric:LIST:ORDer.
- If <NRf> is specified, only the numeric list data of the specified item number is output (up to 52 items of data).
- If <NRf> is omitted, the numeric list data of item numbers from 1 to :NUMeric:LIST:NUMber is output in order (up to 52 times the number specified by :NUMeric:LIST:NUMber).
- For the formats of each individual numeric data output item, see “Numeric Data Formats” at the end of this group of commands (page 6-25).
- This instrument does not measure data for the DC component. It is always “NAN.”

### :NUMeric:LIST:NUMber

Function Sets or queries the number of numeric list data items that are transmitted by :NUMeric:LIST:VALue?.

Syntax `:NUMeric:LIST:NUMber {<NRf>|ALL}`
`:NUMeric:LIST:NUMber?`
`<NRf>` = 1 to 32 (ALL)

Example `:NUMERIC:LIST:NUMBER 5`
`:NUMERIC:LIST:NUMBER`
`-> :NUMERIC:LIST:NUMBER 5`

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- If the parameter is omitted from the :NUMeric:LIST:VALue? command, the numeric list data items from 1 to the specified value are output in order.
- By default, the number of numeric list data items is set to 1.

### :NUMeric:LIST:ORDer

Function Sets or queries the maximum output harmonic order of the harmonic measurement numeric list data.

Syntax `:NUMeric:LIST:ORDer {<NRf>|ALL}`
`:NUMeric:LIST:ORDer?`
`<NRf>` = 1 to 50 (ALL)

Example `:NUMERIC:LIST:ORDER 50`
`:NUMERIC:LIST:ORDER?`
`-> :NUMERIC:LIST:ORDER 50`

Description This is only valid on models with the harmonic measurement (/G5) option.

### :NUMeric:LIST:SELect

Function Sets or queries the output components of the harmonic measurement numeric list data.

Syntax `:NUMeric:LIST:SELect {EVEN|ODD|ALL}`
`:NUMeric:LIST:SELect?`

Example `:NUMERIC:LIST:SELECT ALL`
`:NUMERIC:LIST:SELECT?`
`-> :NUMERIC:LIST:SELECT ALL`

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- The available options are explained below.
  EVEN = Outputs the components of TOTal, DC, and even-order harmonics
  ODD = Outputs the components of TOTal, DC, and odd-order harmonics
  ALL = Outputs all components

### :NUMeric:LIST:ITEM<x>

Function Sets or queries the output item (function and element) of the specified harmonic measurement numeric list data item.

Syntax `:NUMeric:LIST:ITEM<x> {NONE|<Function>,<Element>}`
`:NUMeric:LIST:ITEM<x>?`
`<x> = 1 to 32` (item number)
`NONE` = No output item
`<Function>` = `{U|I|P|PHIU|PHII|UHDF|IHDF|PHDF}`
`<Element>` = `{<NRf>}(<NRf>` = `1` to `3)`

Example `:NUMERIC:LIST:ITEM1 U,1`
`:NUMERIC:LIST:ITEM1?`
`-> :NUMERIC:LIST:ITEM1 U,1`

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- For details on <Function> options, see "Function Option List (2)" at the end of this group of commands (page 6-24).

### :NUMeric:LIST:PRESet

Function Presets the harmonic measurement numeric list data output item pattern.

Syntax `:NUMeric:LIST:PRESet {<NRf>}`
`<NRf>` = `1` to `4`

Example `:NUMERIC:LIST:PRESET 1`

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- For information about the output items that are preset, see "(2) Preset Patterns for Harmonic Measurement Numeric List Data Output Items" on page 6-28 at the end of the commands for this group.
- By default, the output items of Pattern 2 are selected.

### :NUMeric:LIST:CLEar

Function Clears harmonic measurement numeric list data output items (sets the items to NONE).

Syntax `:NUMeric:LIST:CLEar {ALL|<NRf>[,<NRf>]}`
`ALL` = Clear all items
First `<NRf>` = `1` to `32` (the number of the first item to clear)
Second `<NRf>` = `1` to `32` (the number of the last item to clear)

Example `:NUMERIC:LIST:CLEAR ALL`

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- If the 2nd <NRf> is omitted, the output item specified by the first number and all following output items (up to number 32) are cleared.

### :NUMeric:LIST:DELete

Function Deletes harmonic measurement numeric list data output items.

Syntax `:NUMeric:LIST:DELete {<NRf>[,<NRf>]}`
1st `<NRf>` = `1` to `32` (the number of the first item to delete)
Second `<NRf>` = `1` to `32` (the number of the last item to delete)

Example `:NUMERIC:LIST:DELETE 1` (Deletes ITEM1 and shifts ITEM2 and subsequent items forward)
`:NUMERIC:LIST:DELETE 1,3` (Deletes ITEM1 to ITEM3 and shifts ITEM4 and subsequent items forward)

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- When output items are deleted, subsequent items shift forward to fill the empty positions. Empty positions at the end are set to NONE.
- If the second <NRf> is omitted, only the output item specified by the first number is deleted.

**:NUMeric:HOLD**

Function Sets or queries the on/off (hold/release) status of the numeric data hold feature.

Syntax
```
:NUMeric:HOLD {<Boolean>}
:NUMeric:HOLD?
```

Example
```
:NUMERIC:HOLD ON
:NUMERIC:HOLD? -> :NUMERIC:HOLD 1
```

Description
- If :NUMeric:HOLD is set to ON before :NUMeric[:NORMal]:VALue? or :NUMeric:LIST:VALue? is executed, all the numeric data at that point in time can be held internally.
- As long as :NUMeric:HOLD is set to ON, numeric data is held even when the numeric data on the screen is updated.
- For example, if you wish to retrieve various types of numeric data from each element at the same point in time, use the following commands:
  :NUMeric:HOLD ON
  :NUMeric[:NORMal]:ITEM1 U,1;ITEM2 I,1;...
  (Set the numeric data items of element 1.)
  :NUMeric[:NORMal]:VALue?
  (Receive the numeric data of element 1.)
  :NUMeric[:NORMal]:ITEM1 U,2;ITEM2 I,2;...
  (Set the numeric data items of element 2.)
  :NUMeric[:NORMal]:VALue?
  (Receive the numeric data of element 2.)
  :NUMeric[:NORMal]:ITEM1 U,3;ITEM2 I,3;...
  (Set the numeric data items of element 3.)
  :NUMeric[:NORMal]:VALue?
  (Receive the numeric data of element 3.)
  :NUMeric:HOLD OFF
- If :NUMeric:HOLD is set to ON after having already been set to ON before, the numeric data is cleared, and the most recent numeric data is held internally. When retrieving numeric data continuously, this method can be used to circumvent the need to repeatedly set :NUMeric:HOLD to OFF.

### Function Option List (Settings That Can Be Used for <Function>)

#### (1) Numeric data functions

Applicable commands

```
:NUMeric[:NORMal]:ITEM<x> {NONE|<Function>[,<Element>][,<Order>]}
:AOUTput[:NORMal]:CHANnel<x> {NONE|<Function>[,<Element>]}
```

| <Function> | Function | WT Indicator | <Element> | <Order> |
|---|---|---|---|---|
| U | Voltage U | [V] | Yes | No |
| I | Current I | [A] | Yes | No |
| P | Active power P | [W] | Yes | No |
| S | Apparent power S | [VA] | Yes | No |
| Q | Reactive power Q | [var] | Yes | No |
| LAMBda | Power factor λ | [PF] | Yes | No |
| PHI | Phase difference Φ | [°] | Yes | No |
| FU | Voltage frequency fU | [V Hz] | Yes | No |
| FI | Current frequency fI | [A Hz] | Yes | No |
| UPPeak | Maximum voltage: U+pk | [V pk] | Yes | No |
| UMPeak | Minimum voltage: U–pk | [V pk] | Yes | No |
| IPPeak | Maximum current: I+pk | [A pk] | Yes | No |
| IMPeak | Minimum current: I–pk | [A pk] | Yes | No |
| PPPeak | Maximum power: P+pk | [W pk] | Yes | No |
| PMPeak | Minimum power: P–pk | [W pk] | Yes | No |
| TIME | Integration time | [TIME] | No | No |
| WH | Watt hour WP | [W h] | Yes | No |
| WHP | Positive watt hour WP+ | [W h±] | Yes | No |
| WHM | Negative watt hour WP- | [W h±] | Yes | No |
| AH | Ampere hour q | [A h] | Yes | No |
| AHP | Positive ampere hour q+ | [A h±] | Yes | No |
| AHM | Negative ampere hour q- | [A h±] | Yes | No |
| MATH | Computed value, such as efficiency | [MATH] | No | No |
| URANge | Voltage range | | No | No |
| IRANge | Current range | | No | No |
| URMS | True rms voltage Urms | | Yes | No |
| UMN | Rectified mean voltage calibrated to the rms value Umn | | Yes | No |
| UDC | Simple voltage average Udc | | Yes | No |
| URMN | Rectified mean voltage Urmn | | Yes | No |
| UAC | AC voltage component Uac | | Yes | No |
| IRMS | True rms current Irms | | Yes | No |
| IMN | Rectified mean current calibrated to the rms value Imn | | Yes | No |
| IDC | Simple current average Idc | | Yes | No |
| IRMN | Rectified mean current Irmn | | Yes | No |
| IAC | AC current component Iac | | Yes | No |
| **Functions used in AOUTput[:NORMal]:CHANnel<x>** | | | | |
| UPeak | Voltage peak Upk | [V pk] | Yes | No |
| IPeak | Current peak Ipk | [A pk] | Yes | No |
| **Functions that require the harmonic measurement (/G5) option** | | | | |
| UK | Rms voltage of harmonic order k U(k) | [V] | Yes | Yes |
| IK | Rms current of harmonic order k I(k) | [A] | Yes | Yes |
| PK | Active power of harmonic order k P(k) | [W] | Yes | Yes |
| LAMBDAK | Power factor of harmonic order k λ(k) | [PF] | Yes | Yes (k=1 only) |
| PHIK | Phase difference between the voltage and current of harmonic order k φ(k) | [V °] or [A °] | Yes | Yes (k=1 only) |
| PHIUk | Phase difference between harmonic voltage U(k) and the fundamental wave U(1) φU(k) | [V °] | Yes | Yes (k=2 and higher) |
| PHIIk | Phase difference between harmonic current I(k) and the fundamental wave I(1) φI(k) | [A °] | Yes | Yes (k=2 and higher) |
| UHDFk | Harmonic distortion factor of voltage Uhdf(k) | [V %] | Yes | Yes |
| IHDFk | Harmonic distortion factor of current Ihdf(k) | [A %] | Yes | Yes |
| PHDFk | Harmonic distortion factor of active power Phdf(k) | [W %] | Yes | Yes |
| UTHD | Total harmonic distortion of voltage Uthd | [THD V %] | Yes | No |
| ITHD | Total harmonic distortion of current Ithd | [THD A %] | Yes | No |

| <Function> | Function | WT Indicator | <Element> | <Order> |
|---|---|---|---|---|
| FPLL | PLL source frequency fPLL | [V Hz] or [A Hz] | No | No |

Yes: Required. No: Not required.

**(2) Numeric List Data Output Functions (These functions require the harmonic measurement (/G5) option)**

Applicable command

`:NUMeric:LIST:ITEM<x> {NONE|`**`<Function>`**`,<Element>}`

| <Function> | Function |
|---|---|
| U | Voltage U( ) |
| I | Current I( ) |
| P | Active power P( ) |
| PHIU | Phase difference between harmonic voltage U(k) and the fundamental wave U(1) φU( ) |
| PHII | Phase difference between harmonic current I(k) and the fundamental wave I(1) φI(k) |
| UHDF | Harmonic distortion factor of voltage Uhdf( ) |
| IHDF | Harmonic distortion factor of current Ihdf( ) |
| PHDF | Harmonic distortion factor of active power Phdf( ) |

**Numeric Data Format**

(1)Normal Data

- Integrated values WH, WHP, WHM, AH, AHP, and AHM
  ASCII: <NR3> format (mantissa: up to 6 digits, exponent: 2 digits. Example: [-]123.456E+00)
  FLOAT: IEEE single-precision floating point (4-byte) format
- Elapsed integration time (TIME)
  ASCII: <NR1> format in units of seconds. Example: 3600 for 1 hour (1:00:00).
  FLOAT: IEEE single-precision floating point (4-byte) format in units of seconds. Example: 0x45610000 for 1 hour (1:00:00).
- No items (NONE)
  ASCII: NAN (Not A Number)
  FLOAT: 0x7E951BEE (9.91E+37)
- Other
  ASCII: <NR3> format (mantissa: up to 5 digits, exponent: 2 digits. Example: [-]123.45.456E+00)
  FLOAT: IEEE single-precision floating point (4-byte) format

(2)Error Data

- Data does not exist (the display shows “---------”)
  ASCII: NAN (Not A Number)
  FLOAT: 0x7E951BEE (9.91E+37)
- Over-range (the display shows “---O L---”)
- Overflow (the display shows “---O F---”)
- Data over (the display shows “ Error “)
  ASCII: INF (INFinity)
  FLOAT: 0x7E94F56A (9.9E+37)

***Note***

In 180° (Lead/Lag) display, the phase differences Φ (PHI) of elements 1 to 3 are output in the range between –180.0 to 180.0 with lead (D) and lag (G) set to negative and positive values, respectively.

*** Preset Patterns for Numeric Data Items**

The Function Option List contains a list of the function names used in commands and their corresponding functions and panel LED indicators.

## *Note*

This list indicates the measurement function and element that are assigned to each item number (ITEM <x>). Items that are not set to be measured are displayed or output in the same fashion as when the data does not exist. For example, if frequency FI of the current of element 2 is not set to be measured, the output of ITEM19 in pattern 2 is the same as the output when the data does not exist (NAN if the data format is ASCII).

(1)Preset Patterns for Numeric Data Items

These patterns apply to the :NUMeric[:NORMal]:PRESet command.

Pattern 1

| ITEM<x> | <Function> | <Element> |
|---|---|---|
| 1 | U | 1 |
| 2 | I | 1 |
| 3 | P | 1 |
| 4 to 6 | U to P | 2 |
| 7 to 9 | U to P | 3 |
| 10 to 12 | U to P | SIGMA |
| 13 to 255 | NONE | |

Pattern 2

| ITEM<x> | <Function> | <Element> |
|---|---|---|
| 1 | U | 1 |
| 2 | I | 1 |
| 3 | P | 1 |
| 4 | S | 1 |
| 5 | Q | 1 |
| 6 | LAMBda | 1 |
| 7 | PHI | 1 |
| 8 | FU | 1 |
| 9 | FI | 1 |
| 10 | NONE | |
| 11 to 19 | U to FI | 2 |
| 20 | NONE | |
| 21 to 29 | U to FI | 3 |
| 30 | NONE | |
| 31 to 39 | U to FI | SIGMA |
| 40 | NONE | |
| 41 to 255 | NONE | |

Pattern 3

| ITEM<x> | <Function> | <Element> |
|---|---|---|
| 1 | U | 1 |
| 2 | I | 1 |
| 3 | P | 1 |
| 4 | S | 1 |
| 5 | Q | 1 |
| 6 | LAMBda | 1 |
| 7 | PHI | 1 |
| 8 | FU | 1 |
| 9 | FI | 1 |
| 10 | UPPeak | 1 |
| 11 | UMPeak | 1 |
| 12 | IPPeak | 1 |
| 13 | IMPeak | 1 |
| 14 | PPPeak | 1 |
| 15 | PMPeak | 1 |
| 16 to 30 | U to PMPeak | 2 |
| 31 to 45 | U to PMPeak | 3 |
| 46 to 60 | U to PMPeak | SIGMA |
| 61 to 255 | NONE | |

Pattern 4

| ITEM<x> | <Function> | <Element> |
|---|---|---|
| 1 | U | 1 |
| 2 | I | 1 |
| 3 | P | 1 |
| 4 | S | 1 |
| 5 | Q | 1 |
| 6 | LAMBda | 1 |
| 7 | PHI | 1 |
| 8 | FU | 1 |
| 9 | FI | 1 |
| 10 | UPPeak | 1 |
| 11 | UMPeak | 1 |
| 12 | IPPeak | 1 |
| 13 | IMPeak | 1 |
| 14 | TIME | 1 |
| 15 | WH | 1 |
| 16 | WHP | 1 |
| 17 | WHM | 1 |
| 18 | AH | 1 |
| 19 | AHP | 1 |
| 20 | AHM | 1 |
| 21 to 40 | U to AHM | 2 |
| 41 to 60 | U to AHM | 3 |
| 61 to 80 | U to AHM | SIGMA |
| 81 to 255 | NONE | |

(2)Preset Patterns for Harmonic Measurement Numeric List Data Output Items

These patterns apply to the `:NUMeric:LIST:PRESet` command.

Pattern 1

| ITEM<x> | <Function> | <Element> |
|---|---|---|
| 1 | U | 1 |
| 2 | I | 1 |
| 3 | P | 1 |
| 4 to 6 | U to P | 2 |
| 7 to 9 | U to P | 3 |
| 10 to 32 | NONE | |

Pattern 2

| ITEM<x> | <Function> | <Element> |
|---|---|---|
| 1 | U | 1 |
| 2 | I | 1 |
| 3 | P | 1 |
| 4 | PHIU | 1 |
| 5 | PHII | 1 |
| 6 to 10 | U to PHII | 2 |
| 11 to 15 | U to PHII | 3 |
| 16 to 32 | NONE | |

Pattern 3

| ITEM<x> | <Function> | <Element> |
|---|---|---|
| 1 | U | 1 |
| 2 | I | 1 |
| 3 | P | 1 |
| 4 | UHDF | 1 |
| 5 | IHDF | 1 |
| 6 | PHDF | 1 |
| 7 to 12 | U to PHDF | 2 |
| 13 to 18 | U to PHDF | 3 |
| 19 to 32 | NONE | |

Pattern 4

| ITEM<x> | <Function> | <Element> |
|---|---|---|
| 1 | U | 1 |
| 2 | I | 1 |
| 3 | P | 1 |
| 4 | PHIU | 1 |
| 5 | PHII | 1 |
| 6 | UHDF | 1 |
| 7 | IHDF | 1 |
| 8 | PHDF | 1 |
| 9 to 16 | U to PHDF | 2 |
| 17 to 24 | U to PHDF | 3 |
| 25 to 32 | NONE | |

# 6.12 RATE Group

The command in this group deals with the data update interval. You can make the same settings and queries that you can by pressing SETUP on the front panel and using the u.rAtE menu.

**:RATE**

Function Sets or queries the data update interval.

Syntax `:RATE {<Time>}`
`:RATE?`
<Time> = `100`, `250`, `500(ms)`, `1`, `2`, `5(s)`

Example `:RATE 250MS`
`:RATE? -> :RATE 250.0E-03`

# 6.13 RECall Group

The commands in this group deal with outputting stored measured data and loading setup parameters. You can make the same settings that you can by pressing SAVE/LOAD on the front panel and using the LoAd menu. There are no front panel keys that output measured data that is stored.

**:RECall:NUMber?**

Function Queries the number of blocks of measured data that is stored.

Syntax `:RECall:NUMber?`

Example `:RECALL:NUMBER? -> 600`

**:RECall[:NORMal]:VALue?**

Function Queries the numeric data at the specified block number.

Syntax `:RECall[:NORMal]:VALue? {<NRf>}`

<NRf> = 1 to 9000 (block number)

Description
- Always specify <NRf>. The numeric data at the specified block number will be returned.
- If you omit <NRf> or specify a number greater than the number of blocks that contain stored measured data (the number returned by : RECall:NUMber?), the entire returned numeric data will be “NAN” (no data).
- The output items and format are the same as those of “:NUMeric[:NORMal]:VALue? (when the item number is not specified).” To set the output items and format, use the NUMeric group commands.

**:RECall:LIST:VALue?**

Function Queries the numeric list data of harmonic measurement at the specified block number.

Syntax `:RECall:LIST:VALue? {<NRf>}`

<NRf> = 1 to 600 (block number)

Description
- This is only valid on models with the harmonic measurement (/G5) option.
- Always specify <NRf>. The numeric list data at the specified block number will be returned.
- If you omit <NRf> or specify a number greater than the number of blocks that contain stored measured data (the number returned by : RECall:NUMber?), the entire returned numeric list data will be “NAN” (no data).
- The output items and format are the same as those of “:NUMeric:LIST:VALue? (when the item number is not specified).” To set the output items and format, use the NUMeric group commands.

**:RECall:PANel**

Function Loads a setup parameter file.

Syntax `:RECall:PANel {<NRf>}`

`<NRf>` = `1` to `4` (file number)

# 6.14 STATus group

The commands in this group are used to make settings and queries related to the status report. There are no front panel keys that correspond to the commands in this group. For information about status reports, see chapter 7.

### :STATus?

Function Queries all the settings for the communication status feature.

Syntax `:STATus?`

### :STATus:CONDition?

Function Queries the contents of the condition register.

Syntax `:STATus:CONDition?`

Example `:STATUS:CONDITION? -> 16`

Description For information about the condition register, see chapter 7, “Status Reports.”

### :STATus:EESE

Function Sets or queries the extended event enable register.

Syntax `:STATus:EESE <Register>`
`:STATus:EESE?`
`<Register>` = 0 to 65535

Example `:STATUS:EESE #B0000000000000000`
`:STATus:EESE?`
`-> :STATUS:EESE 0`

Description For information about the extended event enable register, see chapter 7, “Status Reports.”

### :STATus:EESR?

Function Queries the contents of the extended event register and clears the register.

Syntax `:STATus:EESR?`

Example `:STATUS:EESR? -> 0`

Description For information about the extended event register, see chapter 7, “Status Reports.”

### :STATus:ERRor?

Function Queries the error code and message of the last error that has occurred (top of the error queue).

Syntax `:STATus:ERRor?`

Example `:STATUS:ERROR?`
`-> 113,"Underfined Header"`

Description
- If no errors have occurred, 0,”No error” is returned.
- You can use the :STATus:QMESsage command to specify whether the message is included.

### :STATus:FILTer<x>

Function Sets or queries the transition filter.

Syntax `:STATus:FILTer<x> {RISE|FALL|BOTH|NEVer}`
`:STATus:FILTer<x>?`
<x> = 1 to 16

Example `:STATUS:FILTER2 RISE`
`:STATus:FILTER2?`
`-> :STATUS:FILTER2 RISE`

Description
- Set how each bit in the condition register must change to trigger the setting of an event. If a bit is set to RISE, an event is set when the bit changes from 0 to 1.
- For information about the transition filter, see chapter 7, “Status Reports.”

### :STATus:QENable

Function Sets or queries whether messages other than errors will be stored to the error queue (ON) or not (OFF).

Syntax `:STATus:QENable {<Boolean>}`
`:STATus:QENable?`

Example `:STATUS:QENABLE ON`
`:STATus:QENABLE?`
`-> :STATus:QENABLE 1`

### :STATus:QMESsage

Function Sets or queries whether message information will be attached to the response to the STATus:ERRor? query (ON/OFF).

Syntax `:STATus:QMESsage {<Boolean>}`
`:STATus:QMESsage?`

Example `:STATUS:QMESSAGE ON`
`:STATus:QMESSAGE?`
`-> :STATus:QMESSAGE 1`

### :STATus:SPOLl?

Function Executes serial polling.

Syntax `:STATus:SPOLl?`

Example `:STATUS:SPOLL? -> :STATUS:SPOLL 0`

# 6.15 STORe Group

The commands in this group deal with storing measured data and saving setup parameters. You can make the same settings and queries that you can by pressing the UTILITY key on the front panel and then using the StorE menu or by pressing the SAVE/LOAD key and then using the SAVE menu.

### :STORe?

Function Queries all storage settings.
Syntax :STORe?

### :STORe[:STATe]

Function Sets or queries the storage on/off state.
Syntax :STORe[:STATe] {<Boolean>}
:STORe:STATe?
Example :STORE:STATE ON
:STORE:STATE? -> :STORE:STATE 1

### :STORe:INTerval

Function Sets or queries the storage interval.
Syntax :STORe:INTerval {<NRf>,<NRf>,<NRf>}
:STORe:INTerval?
{<NRf>,<NRf>,<NRf>} = 0,0,0 to 99,59,59
First <NRf> = 0 to 99 (hours)
Second <NRf> = 0 to 59 (minutes)
Third <NRf> = 0 to 59 (seconds)
Example :STORE:INTERVAL 0,0,0
:STORE:INTERVAL?
-> :STORE:INTERVAL 0,0,0

### :STORe:PANel

Function Saves setup parameters to a file.
Syntax :STORe:PANel {<NRf>}
<NRf> = 1 to 4 (file number)
Example :STORE:PANEL 1

# 6.16 SYSTem Group

The commands in this group deal with the system. You can make the same settings and queries that you can by pressing the UTILITY key on the front panel and then using the inFo or rESo menu or by pressing the KEY PROTECT or INTERFACE key.

### :SYSTem?

Function Queries all system settings.
Syntax :SYSTem?

### :SYSTem:MODel?

Function Queries the model code.
Syntax :SYSTem:MODel?
Example :SYSTEM:MODEL?
-> :SYSTEM:MODEL "WT310"
Description Returns the Model item of the Utility -> Info menu.

### :SYSTem:SUFFix?

Function Queries the suffix code.
Syntax :SYSTem:SUFFix?
Example :SYSTEM:SUFFIX?
-> :SYSTEM:SUFFIX "-D-C1/C7/EX1/G5/DA4"
Description Returns the Suffix item string of the Utility -> Info menu.

### :SYSTem:SERial?

Function Queries the serial number.
Syntax :SYSTem:SERial?
Example :SYSTEM:SERIAL?
-> :SYSTEM:SERIAL "123456789A"
Description Returns the No. item string of the Utility -> Info menu.

### :SYSTem:VERsion[:FIRMware]?

Function Queries the firmware version.
Syntax :SYSTem:VERsion[:FIRMware]?
Example :SYSTEM:VERSION:FIRMWARE? -> "1.01"
Description Returns the Ver. item string of the Utility -> Info menu.

### :SYSTem:KLOCk

Function Sets or queries the on/off state of the key protection.
Syntax :SYSTem:KLOCk {<Boolean>}
:SYSTem:KLOCk?
Example :SYSTEM:KLOCK OFF
:SYSTEM:KLOCK? -> :SYSTEM:KLOCK 0

### :SYSTem:RESolution

Function Sets or queries the numeric data display resolution.
Syntax :SYSTem:RESolution {<NRf>}
:SYSTem:RESolution?
<NRf> = 4, 5 (digit)
Example :SYSTEM:RESOLUTION 5
:SYSTEM:RESOLUTION?
-> :SYSTEM:RESOLUTION 5

### :SYSTem:COMMunicate:COMMand

Function Sets or queries the command type.
Syntax :SYSTem:COMMunicate:COMMand {WT300|WT200}
:SYSTem:COMMunicate:COMMand?
Example :SYSTEM:COMMUNICATE:COMMAND WT300
:SYSTEM:COMMUNICATE:COMMAND?
-> :SYSTEM:COMMUNICATE:COMMAND WT300

### :SYSTem:COMMunicate:ETHernet:MACaddress?

Function Sets or queries the Ethernet MAC address.
Syntax :SYSTem:COMMunicate:ETHernet:MACaddress?
Example :SYSTEM:COMMUNICATE:ETHERNET:MACADDRESS?
-> :SYSTEM:COMMUNICATE:ETHERNET:MACADDRESS "000064_809_413"
Description This command is only valid on models with the Ethernet interface (/C7) option.

# 6.17 Common Command Group

The commands in this group are defined in IEEE 488.2-1992 and are independent from the instrument's individual functions. There are no front panel keys that correspond to the commands in this group.

***CAL?**

Function Executes zero calibration (zero-level compensation, the same operation as pressing CAL (SHIFT+SET)) and queries the result.
Syntax `*CAL?`
Example `*CAL? -> 0`
Description If the zero-level compensation ends normally, 0 is returned. If an error is detected, 1 is returned.

***CLS**

Function Clears the standard event register, extended event register, and error queue.
Syntax `*CLS`
Example `*CLS`
Description
- If the *CLS command is located immediately after the program message terminator, the output queue is also cleared.
- For information about each register and queue, see chapter 7.

***ESE**

Function Sets or queries the standard event enable register.
Syntax `*ESE {<NRf>}`
`*ESE?`
`<NRf>` = `0` to `255`
Example `*ESE 251`
`*ESE? -> 251`
Description
- Specify the value as a sum of the values of each bit in decimal format.
- For example, specifying *ESE 251 will cause the standard enable register to be set to 11111011. In this case, bit 2 of the standard event register is disabled. This means that bit 5 (ESB) of the status byte register is not set to 1, even if a query error occurs.
- The default value is *ESE 0 (all bits disabled).
- A query using *ESE? will not clear the contents of the standard event enable register.
- For information about the standard event enable register, see page 7-4.

***ESR?**

Function Queries and clears the standard event register.
Syntax `*ESR?`
Example `*ESR? -> 32`
Description
- A sum of the values of each bit is returned in decimal format.
- When an SRQ is sent, you can check what types of events have occurred.
- For example, if a value of 32 is returned, this indicates that the standard event register is set to 00100000. This means that the SRQ occurred due to a command syntax error.
- A query using *ESR? will clear the contents of the Standard Event Register.
- For information about the standard event register, see page 7-4.

***IDN?**

Function Queries the instrument model.
Syntax `*IDN?`
Example `*IDN? ->`
`YOKOGAWA,WT310,123456789A,F1.01`
Description
- The information is returned in this form: <manufacture>, <model>, <serial number>, <firmware version>.
- For details on the model, see "Checking the Package Contents" in the Getting Started Guide, IM WT310-02EN.

***OPC**

Function Sets bit 0 (the OPC bit) of the standard event register to 1 upon the completion of the specified overlap command.
Syntax `*OPC`
Example `*OPC`
Description
- This instrument does not have overlap commands. The OPC bit is always set to 1.

***OPC?**

Function Returns ASCII code 1 if the specified overlap command has finished.
Syntax `*OPC?`
Example `*OPC? -> 1`
Description This instrument does not have overlap commands. 1 is always returned.

***OPT?**

Function Queries the installed options.
Syntax `*OPT?`
Example `*OPT? -> C1,C7,EX1,G5,DA4`
Description
- This command returns whether the GP-IB (C1), RS-232 (C2), Ethernet communication (C7), external current sensor input (EX1, 2.5 V/5 V/10 V), external current sensor input (EX2, 50 mV/100 mV/200 mV/500 mV/1 V/2 V), harmonic measurement (G5), 4-channel D/A output (DA4, for the WT310 and WT310HC), and 12-channel D/A output (DA12, for the WT332 and WT333) are available.
- The *OPT? query must be the last query of a program message.
  An error occurs if there is a query after the *OPT query.

***RST**

Function Initializes the settings.
Syntax `*RST`
Example `*RST`
Description All settings except communication settings are reset to their factory default values. For details on initialization, see section 8.2 in the User's Manual, IM WT310-01EN.

***SRE**

Function Sets or queries the service request enable register value.
Syntax `*SRE {<NRf>}`
`*SRE?`
`<NRf>` = `0` to `255`
Example `*SRE 239`
`*SRE?`
`-> 175` (because the bit 6, MSS, setting is ignored)
Description
- Specify the value as a sum of the values of each bit in decimal format.
- For example, specifying *SRE 239 will cause the standard enable register to be set to 11101111. In this case, bit 4 of the service request enable register is disabled. This means that bit 4 (MAV) of the status byte register is not set to 1, even if the output queue is not empty.
- Bit 6 (MSS) of the status byte register is the MSS bit itself and is therefore ignored.
- The default value is *SRE 0 (all bits disabled).
- A query using *SRE? will not clear the contents of the service request enable register.
- For information about the service request enable register, see page 7-3.

***STB?**

Function Queries the Status Byte Register value.
Syntax `*STB?`
Example `*STB? -> 4`
Description
- A sum of the values of each bit is returned as a decimal value.
- Because the register is read without executing serial polling, bit 6 is an MSS bit, not an RQS bit.
- For example, if a value of 4 is returned, this indicates that the status byte register is set to 00000100. This means that the error queue is not empty (in other words, an error occurred).
- A query using *STB? will not clear the contents of the status byte register.
- For information about the status byte register, see page 7-3.

***TRG**

Function Executes single measurement (the same operation as when SINGLE (SHIFT+HOLD) is pressed).
Syntax `*TRG`
Example `*TRG`
Description A multi-line message GET (Group Execute Trigger) will perform the same operation as this command.

***TST?**

Function Executes a self-test and queries the result.

Syntax `*TST?`

Example `*TST? -> 0`

Description
- The self-test consists of tests of each kind of internal memory.
- This command returns 0 if the self-test is successful and 1 if it is not.
- It takes approximately 6 seconds for the test to complete. When receiving a response from this instrument, set the timeout to a relatively large value.

***WAI**

Function Holds the execution of the subsequent command until the completion of the specified overlap command.

Syntax `*WAI`

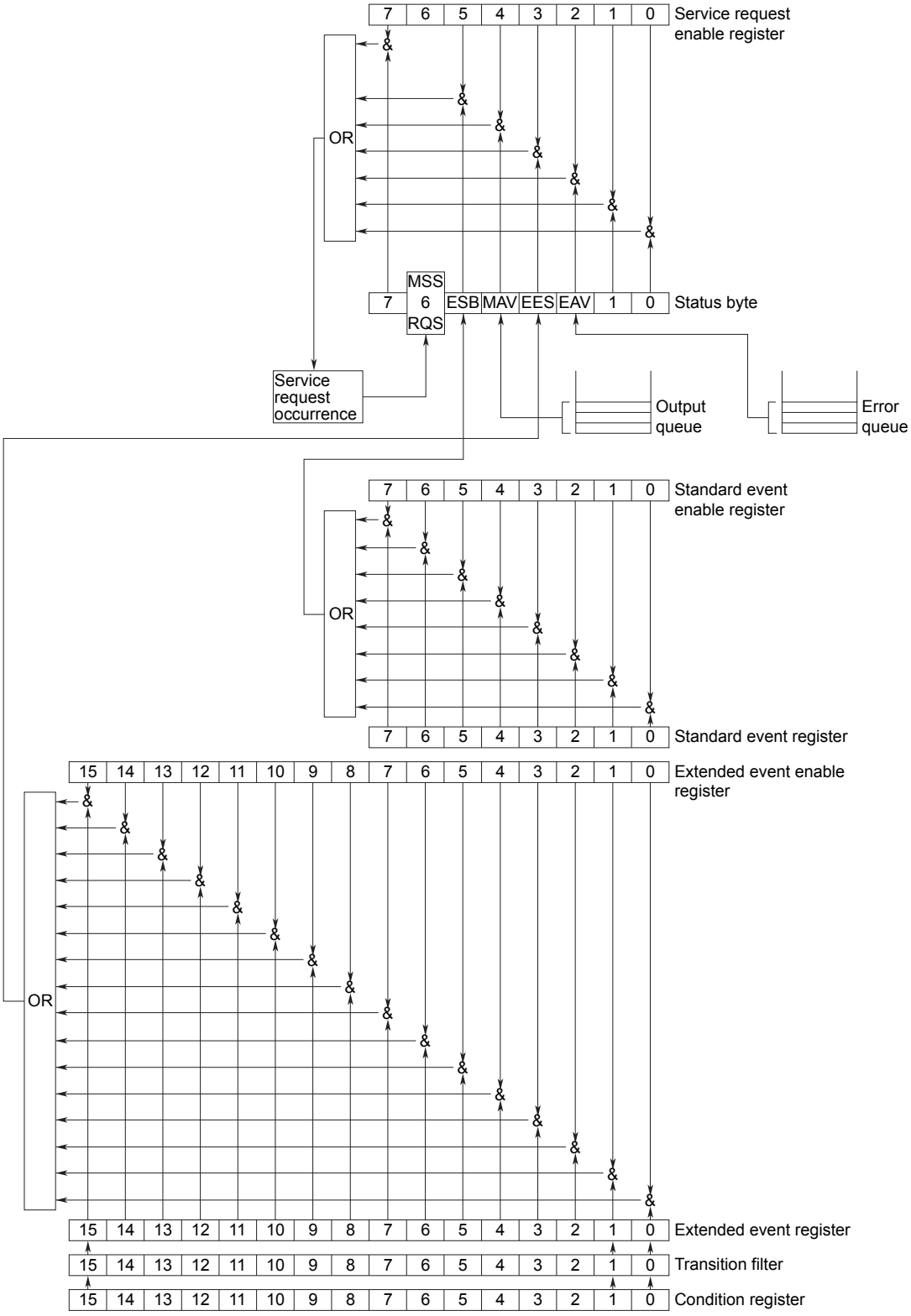
Example `*WAI`

Description This instrument does not have overlap commands. This command will be ignored.

# 7.1 About Status Reports

## Status Reports

The figure below shows the format of status reports that are read by serial polling. This status report format is an extended version of the status report format defined in IEEE 488.2-1992.



## Overview of Registers and Queues

| Name | Function | Write | Read |
|---|---|---|---|
| Status byte | – | – | Serial polling (RQS), *STB?(MSS) |
| Service request enable register | Status byte mask | *SRE | *SRE? |
| Standard event register | Indicates device status changes | – | *ESR? |
| Standard event enable register | Standard event register mask | *ESE | *ESE? |
| Extended event register | Indicates device status changes | – | STATus:EESR? |
| Extended event enable register | Extended event register mask | STATus:EESE | STATus:EESE? |
| Condition register | Current device status | – | STATus:CONDition? |
| Transition filter | Conditions that change the extended event register | STATus:FILTer<x> | STATus:FILTer<x>? |
| Output queue | Stores response messages for queries | Query commands | |
| Error queue | Stores error numbers and messages | – | STATus:ERRor? |

## Registers and Queues That Affect the Status Byte

The following registers affect the status byte bits.

| Register | Affected Status Byte Bit |
|---|---|
| Standard event register | Sets bit 5 (ESB) to 1 or 0 |
| Output queue | Sets bit 4 (MAV) to 1 or 0 |
| Extended event register | Sets bit 3 (EES) to 1 or 0 |
| Error queue | Sets bit 2 (EAV) to 1 or 0 |

## Enable Registers

The following registers are used to mask a bit so that the bit will not affect the status byte even when it is set to 1.

| Masked Register | Mask Register |
|---|---|
| Status byte | Service request enable register |
| Standard event register | Standard event enable register |
| Extended event register | Extended event enable register |

## Reading and Writing to Registers

For example, use the *ESE command to set the standard event enable register bits to 1 and 0. You can use the *ESE? command to query whether the standard event enable register bits are ones or zeros. For details on these commands, see chapter 6.

# 7.2 Status Byte

### Status Byte

| 7 | RQS<br>6<br>MSS | ESB | MAV | EES | EAV | 1 | 0 |
|---|---|---|---|---|---|---|---|

- **Bits 0, 1, and 7**
  Not used (always 0)
- **Bit 2 EAV (Error Available)**
  This bit is 1 when the error queue is not empty. In other words, this bit is set to 1 when an error occurs. For details, see page 7-6.
- **Bit 3 EES (Extend Event Summary Bit)**
  This bit is set to 1 when the logical AND of the extended event register and the extended event enable register is 1. In other words, this bit is set to 1 when a certain event takes place inside the instrument. For details, see page 7-5.
- **Bit 4 MAV (Message Available)**
  This bit is 1 when the output queue is not empty. In other words, this bit is set to 1 when there is data to be transmitted in response to a query. For details, see page 7-6.
- **Bit 5 ESB (Event Summary Bit)**
  This bit is set to 1 when the logical AND of the standard event register and the standard event enable register is 1. In other words, this bit is set to 1 when a certain event takes place inside the instrument. For details, see page 7-4.
- **Bit 6 RQS (Request Service)/MSS (Master Status Summary)**
  This bit is 1 when the logical AND of the status byte excluding bit 6 and the service request enable register is 1. In other words, this bit is set to 1 when the instrument requests service from the controller. RQS is set to 1 when the MSS bit changes from 0 to 1 and is cleared when serial polling is carried out or when the MSS bit changes to 0.

### Bit Masking

To mask a bit in the status byte so that it does not cause an SRQ, set the corresponding bit of the service request enable register to 0.
For example, to mask bit 2 (EAV) so that service is not requested when an error occurs, set bit 2 of the service request enable register to 0. Do this using the `*SRE` command. To query whether each bit of the service request enable register is 1 or 0, use `*SRE?`. For details on the `*SRE` command, see chapter 6.

### Status Byte Operation

A service request is issued when bit 6 in the status byte becomes 1. Bit 6 is set to 1 when any other bit becomes 1 (when the corresponding bit of the service request enable register is also set to 1). For example, if an event occurs and the logical OR of a standard event register bit and its corresponding enable register bit is 1, then bit 5 (ESB) is set to 1. At this point, if bit 5 of the service request enable register is 1, bit 6 (MSS) is set to 1, and this instrument requests service from the controller.
You can check what type of event occurred by reading the contents of the status byte.

### Reading the Status Byte

There are two ways to read the contents of the status byte.

- ***STB? query**
  Bit 6 functions as MSS when a query is made using `*STB?`. This causes the MSS to be read. This query does not cause any of the status byte bits to be cleared after the status byte is read.
- **Serial polling**
  Serial polling causes bit 6 to function as an RQS bit. This causes the RQS to be read. After the status byte is read, only the RQS bit is cleared. You cannot read the MSS bit when serial polling is used.

### Clearing the Status Byte

There is no way to clear all the bits in the status byte. The bits that are cleared for each operation are shown below.

- ***STB? query**
  None of the bits are cleared.
- **Serial polling**
  Only the RQS bit is cleared.
- **When a *CLS command is received**
  When a `*CLS` command is received, the status byte itself is not cleared, but the contents of the standard event register, which affects the bits in the status byte, are cleared. As a result, the corresponding status byte bits are cleared. Because the output queue is not cleared with a `*CLS` command, bit 4 (MAV) in the status byte is not affected. However, the output queue will be cleared if the `*CLS` command is received just after a program message terminator.

# 7.3 Standard Event Register

### Standard Event Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PON | URQ | CME | EXE | DDE | QYE | RQC | OPC |

- **Bit 7 PON (Power ON)**
  This bit is set to 1 when the instrument is turned on.
- **Bit 6 URQ (User Request)**
  Not used (always 0)
- **Bit 5 CME (Command Error)**
  This bit is set to 1 when there is a command syntax error.
  Example Command names are misspelled, or character data that is not one of the available options has been received.
- **Bit 4 EXE (Execution Error)**
  This bit is set to 1 when the command syntax is correct, but the command cannot be executed in the current state.
  Example Parameters are out of range, or a command has been received for an option that is not installed.
- **Bit 3 DDE (Device Error)**
  This bit is set to 1 when a command cannot be executed for internal reasons other than a command syntax error or command execution error.
- **Bit 2 QYE (Query Error)**
  This bit is set to 1 when a query command is received, but the output queue is empty or the data is lost.
  Example There is no response data, or data is lost due to an overflow in the output queue.
- **Bit 1 RQC (Request Control)**
  Not used (always 0)
- **Bit 0 OPC (Operation Complete)**
  This bit is set to 1 upon the completion of the operation designated by the `*OPC` command (see chapter 6 for details).

### Bit Masking

To mask a certain bit of the standard event register so that it does not cause bit 5 (ESB) in the status byte to change, set the corresponding bit of the standard event enable register to 0.
For example, to mask bit 2 (QYE) so that ESB will not be set to 1 even if a query error occurs, set bit 2 of the standard event enable register to 0. Do this using the `*ESE` command. To query whether each bit of the standard event enable register is 1 or 0, use `*ESE?`.
For details on the `*ESE` command, see chapter 6.

### Standard Event Register Operation

The standard event register indicates eight types of events that occur inside the instrument. When one of the bits in this register becomes 1 (and the corresponding bit of the standard event enable register is also 1), bit 5 (ESB) in the status byte is set to 1.
Example
1. A query error occurs.
2. Bit 2 (QYE) is set to 1.
3. When bit 2 of the standard event enable register is 1, bit 5 (ESB) in the status byte is set to 1.

You can also check what type of event occurred in this instrument by reading the contents of the standard event register.

### Reading the Standard Event Register

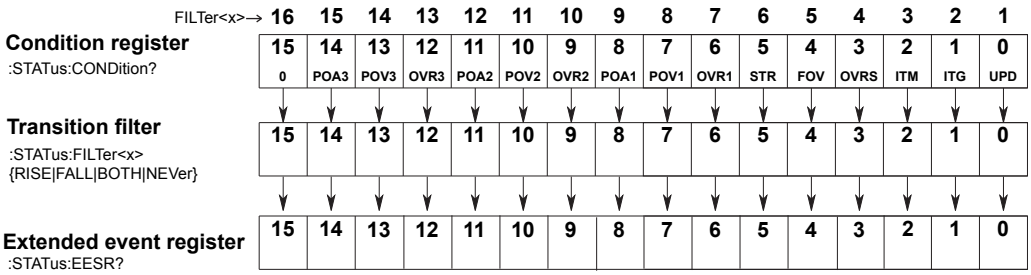
You can use the `*ESR?` command to read the contents of the standard event register. The register is cleared after it is read.

### Clearing the Standard Event Register

The standard event register is cleared in the following three cases.
- When the contents of the standard event register are read using the `*ESR` command.
- When a `*CLS` command is received.
- When this instrument is restarted.

# 7.4 Extended Event Register

The extended event register receives information about changes in the condition register, which indicates the instrument's internal condition. The information is the result of edge detection performed by the transition filter.




The condition register bits are described below.

| | | |
|---|---|---|
| Bit 0 | UPD<br>(Updating) | Set to 1 when the measured data is being updated.<br>UPD changing from 1 to 0 indicates that updating has been completed. |
| Bit 1 | ITG<br>(Integrate Busy) | Set to 1 during integration. |
| Bit 2 | ITM<br>(Integrate Timer Busy) | Set to 1 when the integration timer is operating. |
| Bit 3 | OVRS<br>(Σ results overflow) | Set to 1 when the computed result of Σ overflows. |
| Bit 4 | FOV<br>(Frequency Over) | Set to 1 when the frequency is outside the measurement range. |
| Bit 5 | STR<br>(Store busy) | Set to 1 during storage. |
| Bit 6 | OVR1<br>(Element1 mesured data over) | Set to 1 when the voltage or current of element 1 exceeds its range. |
| Bit 7 | POV1<br>(Element1 voltage peak over) | Set to 1 when a peak over-range is detected in the element 1 voltage. |
| Bit 8 | POA1<br>(Element1 current peak over) | Set to 1 when a peak over-range is detected in the element 1 current. |
| Bit 9 | OVR2<br>(Element2 mesured data over) | Set to 1 when the voltage or current of element 2 exceeds its range. |
| Bit 10 | POV2<br>(Element2 voltage peak over) | Set to 1 when a peak over-range is detected in the element 2 voltage. |
| Bit 11 | POA2<br>(Element2 current peak over) | Set to 1 when a peak over-range is detected in the element 2 current. |
| Bit 12 | OVR3<br>(Element3 mesured data over) | Set to 1 when the voltage or current of element 3 exceeds its range. |
| Bit 13 | POV3<br>(Element3 voltage peak over) | Set to 1 when a peak over-range is detected in the element 3 voltage. |
| Bit 14 | POA3<br>(Element3 current peak over) | Set to 1 when a peak over-range is detected in the element 3 current. |

The transition filter parameters detect changes in the specified condition register bits (numeric suffixes 1 to 16) and overwrite the extended event register in the following ways.

| | |
|---|---|
| RISE | The specified extended event register bit is set to 0 when the corresponding condition register bit changes from 1 to 1. |
| FALL | The specified extended event register bit is set to 1 when the corresponding condition register bit changes from 0 to 1. |
| BOTH | The specified extended event register bit is set to 1 when the corresponding condition register bit changes from 0 to 1 or from 1 to 0. |
| NEVer | Always zero. |

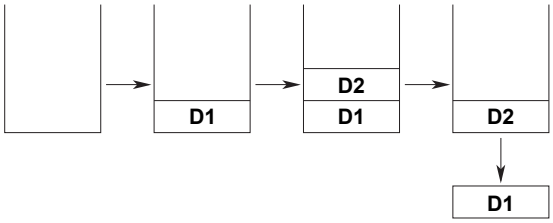
# 7.5 Output and Error Queues

## Output Queue

The output queue stores query response messages. For example, if you send a `:NUMeric[:NORMal]:VALue?` command, which requests for the transmission of measured data, the data is stored in the output queue until it is read.
As shown below, data is stored in order and read from the oldest message first. The output queue is cleared in the following cases.

- When a new message is received from the controller.
- When a deadlock occurs (see page 5-2).
- When a device clear command (DCL or SDC) is received.
- When this instrument is restarted.

The `*CLS` command does not clear the output queue. You can determine whether or not the output queue is empty by checking bit 4 (MAV) in the status byte.




## Error Queue

When an error occurs, the error queue stores the error number and message. For example, if the instrument receives an incorrect program message from the controller, the error number (113) and the error message (“Undefined header”) are stored in the error queue when the instrument displays the error message.
You can use the `:STATus:ERRor?` query to read the contents of the error queue. Like the output queue, the messages in the error queue are read from the oldest one first.
If the error queue overflows, the last message is replaced with the following message: 350, “Queue overflow.”
The error queue is cleared in the following cases.

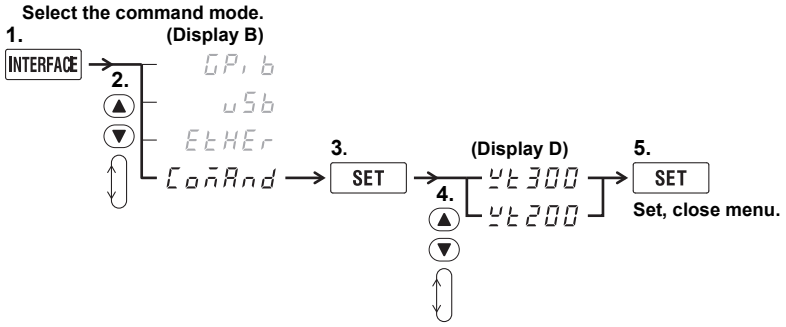
- When a `*CLS` command is received.
- When this instrument is restarted.

You can determine whether or not the error queue is empty by checking bit 2 (EAV) in the status byte.

# 8.1 WT210/WT230 Compatible Command Mode

## Procedure

Follow the procedure indicated by the thick lines in the following menu.




## Explanation

Many of the functions of this instrument can be controlled with the legacy model WT210/WT230 communication commands. For these functions, WT210/WT230 communication programs can be used on this instrument.

### Command Mode

- WT300: Command mode in which communication commands of this instrument are used These commands are not compatible with WT210/WT230 communication commands.
- WT200: Command mode in which the commands are compatible with WT210/WT230 communication commands.

Compatibility with WT210/WT230 communication commands is as follows:

Symbols used in tables

- A: Compatible
- B: Partial limitation
- C: Command accepted but does not work
- D: Command not accepted

***Note***

For details on WT210/WT230 communication commands, see the WT210/WT230 User's Manual.

### AOUTput Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| D/A output item (during normal measurement) | AOUTput:CHANnel<x><br><x> =<br>1 to 12 (for /DA12)<br>1 to 4 (for /DA4, /CMP) | A | | B | NONE = No output item<br><Function> = {U\|I\|P\|S\|Q\|LAMBda\|PHI\|FU\|FI\|WH\|WHP\|WHM\|AH\|AHP\|AHM\|MATH\|UPeak\|IPeak}<br><Element> = {<NRf>\|SIGMa}<br>(<NRf> = 1 to 3) |
| Sets or queries the rated integration time. | :AOUTput:IRTime | B | Cannot be set with a string. | B | Cannot be set with a string. |
| Resets settings to their defaults. | :AOUTput:PRESet | A | | A | |

## COMMunicate group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries whether headers are attached to response data. | :COMMunicate:HEADer | A | | A | |
| Sets/clears local lockout. | :COMMunicate:LOCKout | A | | A | |
| Sets remote or local mode. | :COMMunicate:REMote | A | | A | |
| Queries the line-specific status. | :COMMunicate:STATus? | A | | A | |
| Sets or queries whether query responses are returned in full or abbreviated form. | :COMMunicate:VERBose | A | | A | |
| Waits for the specified extended event to occur. | :COMMunicate:WAIT | A | | A | |
| Creates a response for the specified extended event. | :COMMunicate:WAIT? | A | | A | |

## CONFigure Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Queries all averaging settings. | [:CONFigure]:AVERaging? | A | | D | :MEASure:AVERaging? |
| Sets or queries the on/off state of averaging. | [:CONFigure]:AVERaging[:STATe] | A | | D | :MEASure:AVERaging[:STATe] |
| Sets or queries the averaging type. | [:CONFigure]:AVERaging:TYPE | A | | D | :MEASure:AVERaging:TYPE<br>:MEASure:AVERaging:COUNt |
| Sets or queries the crest factor. | [:CONFigure]:CFACtor | A | | D | [:INPut]:CFACtor |
| Queries all current range settings. | [:CONFigure]:CURRent? | A | | D | [:INPut]:CURRent? |
| Sets or queries the current auto range on/off state. | [:CONFigure]:CURRent:AUTO | A | | D | [:INPut]:CURRent:AUTO |
| Queries all external current sensor scaling constant settings. | [:CONFigure]:CURRent:ESCaling? | A | | D | [:INPut]:CURRent:SRATio? |
| Sets the external current sensor scaling constant on all elements at once. | [:CONFigure]:CURRent:ESCaling[:ALL] | A | You can set the external current sensor scaling constant using the “A/FS(mV)” form, just like the WT210/WT230 computation method.<br><br>FS = External current sensor range rating | D | [:INPut]:CURRent:SRATio[:ALL] |
| Sets or queries the external current sensor scaling constant of an element. | [:CONFigure]:CURRent:ESCaling:ELEMent<x> | A | | D | [:INPut]:CURRent:SRATio:ELEMent<x> |
| Sets or queries the current range. | [:CONFigure]:CURRent:RANGe | A | | D | [:INPut]:CURRent:RANGe |
| Sets or queries the frequency filter on/off state. | [:CONFigure]:FILTer | A | | D | [:INPut]:FILTer:FREQuency |
| Sets or queries the line filter on/off state. | [:CONFigure]:LFILter | A | | D | [:INPut]:FILTer:LINE |
| Sets or queries the MAX hold on/off state. | [:CONFigure]:MHOLd[:STATe] | A | | D | :MEASure:MHOLd |
| Sets or queries the measurement mode. | [:CONFigure]:MODE | A | | D | [:INPut]:MODE |
| Queries all scaling settings. | [:CONFigure]:SCALing? | A | | D | [:INPut]:SCALing? |
| Queries the {voltage\|current\|power} scaling constant. | [:CONFigure]:SCALing:{PT\|CT\|SFACtor}? | A | | D | [:INPut]:SCALing:{VT\|CT\|SFACtor}? |
| Sets the {voltage\|current\|power} scaling constant on all elements at once. | [:CONFigure]:SCALing:{PT\|CT\|SFACtor}[:ALL] | A | | D | [:INPut]:SCALing:{VT\|CT\|SFACtor}[:ALL] |
| Sets or queries the {voltage\|current\|power} scaling constant of an element. | [:CONFigure]:SCALing:{PT\|CT\|SFACtor}:ELEMent<x> | A | | D | [:INPut]:SCALing:{VT\|CT\|SFACtor}:ELEMent<x> |
| Sets or queries the scaling on/off state. | [:CONFigure]:SCALing[:STATe] | A | | D | [:INPut]:SCALing[:STATe] |

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the measurement synchronization source. | [:CONFigure]:SYNChronize | A | | D | [:INPut]:SYNChronize |
| Queries all voltage range settings. | [:CONFigure]:VOLTage? | A | | D | [:INPut]:VOLTage? |
| Sets or queries the voltage auto range on/off state. | [:CONFigure]:VOLTage:AUTO | A | | D | [:INPut]:VOLTage:AUTO |
| Sets or queries the voltage range. | [:CONFigure]:VOLTage:RANGe | A | | D | [:INPut]:VOLTage:RANGe |
| Sets or queries the wiring system. | [:CONFigure]:WIRing | A | | D | [:INPut]:WIRing |

## DISPlay Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the element to be displayed. | :DISPlay<x>:ELEMent<br><x> = 1 to 3<br>1: Display A<br>2: Display B<br>3: Display C | A | | D | Normal measurement data:<br>:DISPlay[:NORMal]:ITEM<x><br>{<Function>[,<Element>]}<br><br>Harmonic measurement data:<br>:DISPlay:HARMonics:ITEM<x><br>{<Function>[,<Element>]} |
| Sets or queries the function to be displayed. | :DISPlay<x>:FUNCtion<br><x> = 1 to 3<br>1: Display A<br>2: Display B<br>3: Display C | A | When you set the following functions, the content of display D changes.<br>• During normal measurement<br>•VHZ<br>•AHZ<br>• During harmonic measurement<br>•VTHD<br>•ATHD<br>•PF | D | |
| Sets or queries the content to be displayed. | :DISPlay<x>:MODE | B | Supports {VALue\|RANGe}. | B | |
| Sets or queries the number of displayed digits. | :DISPlay<x>:RESolution | A | | D | :SYSTem:RESolution {<NRf>} |

## HARMonics Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the harmonic order of the harmonic component that is shown in display B for the harmonic measurement data display. | :HARMonics:DISPlay:ORDer | A | | A | |
| Sets or queries the harmonic measurement source element. | :HARMonics:ELEMent | A | | D | All elements are subject to harmonic measurement, so setting this is not necessary. |
| Sets or queries the on/off state of harmonic measurement mode. | :HARMonics[:STATe] | A | | D | :HARMonics:DISPlay[:STATe] |
| Sets or queries the PLL source. | :HARMonics:SYNChronize | A | | D | :HARMonics:PLLSource |
| Sets or queries the equation used to compute the THD (total harmonic distortion). | :HARMonics:THD | A | | D | :HARMonics:THD<br>{FUNDamental\|TOTal} |

## INTEGrate Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the integration mode. | :INTEGrate:MODE | A | | A | |
| Resets the integrated value. | :INTEGrate:RESet | A | | A | |
| Starts integration. | :INTEGrate:STARt | A | | A | |
| Stops integration. | :INTEGrate:STOP | A | | A | |
| Sets or queries the integration timer value. | :INTEGrate:TIMer | B | Cannot be set with a string. | B | Cannot be set with a string. |

## MATH Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the equation of four arithmetic operations. | :MATH:ARIThmetic | A | | D | :MATH {EFFiciency\|CFU<x>\|CFI <x>\|ADD\|SUB\|MUL\|DIV\|DIVA\| DIVB\|AVW<x>} EFFiciency: Efficiency CFU<x>,CFI<x> : Voltage and current crest factor<x> = 1 to 3 (element) AVW<x> : Average active power during integration <x> = 1 to 3 (element), 4 (Σ) |
| Sets or queries the average active power computation while integration is in progress. | :MATH:AVERage | A | | D | |
| Sets or queries the crest factor equation. | :MATH:CFACtor | A | | D | |
| Sets or queries the computation type. | :MATH:TYPE | A | | D | |

## MEASure Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Queries all harmonic measurement data settings. | :MEASure:HARMonics? | A | | D | :NUMeric:LIST? |
| Queries all settings related to the communication output items of harmonic measurement data. | :MEASure:HARMonics:ITEM? | A | | D | :NUMeric:LIST:ITEM<x> |
| Sets the communication output on/off states of all harmonic measurement functions to a preset pattern at once. | :MEASure:HARMonics:ITEM: PRESet | A | | D | :NUMeric:LIST:PRESet |
| Sets or queries the communication output on/off state of a harmonic measurement function. | :MEASure:HARMonics: ITEM:{<harmonic measurement function>\|SYNChronize} | A | | D | :NUMeric:LIST:ITEM<x> |
| Queries the harmonic measurement data that has been set with commands that start with “MEASure: HARMonics:ITEM.” | :MEASure:HARMonics:VALue? :MEASure:HARMonics:BINary? | A | | D | (1) :NUMeric:FORMat {ASCii\|FLOat} (2) :NUMeric:LIST:VALue? |
| Sets or queries the additional information output on/off state for when outputting measured/ computed data in binary format. | :MEASure:HEADer | C | Setting and querying are possible, but setting this command to ON will not cause additional information to be output. | D | |
| Queries all normal measurement data settings. | :MEASure:NORMal? | A | | D | :NUMeric:NORMal? |
| Queries all settings related to the communication output items of normal measurement data. | :MEASure[:NORMal]:ITEM? | A | | D | :NUMeric:NORMal? |
| Sets the communication output on/off states of all normal measurement functions to a preset pattern at once. | :MEASure[:NORMal]:ITEM: PRESet | A | | D | :NUMeric[:NORMal]:PRESet |

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Queries the communication output setting of the specified normal measurement function. | :MEASure[:NORMal]:ITEM:<normal measurement function>? | A | | D | :NUMeric:NORMal? |
| Sets the communication output on/off state of the specified normal measurement function on all valid elements or Σ at once. | :MEASure[:NORMal]:ITEM:<normal measurement function>[:ALL] | A | | D | :NUMeric[:NORMal]:ITEM<x> |
| Sets or queries the communication output on/off state of the specified normal measurement function on the specified element. | :MEASure[:NORMal]:ITEM:<normal measurement function>:ELEMent<x> | A | | D | |
| Sets or queries the communication output on/off state of the specified normal measurement function on Σ. | :MEASure[:NORMal]:ITEM:<normal measurement function>:SIGMa | A | | D | |
| Sets or queries the communication output on/off state of {elapsed integration time\|MATH}. | :MEASure[:NORMal]:ITEM:{TIME\|MATH} | A | | D | |
| Queries the normal measurement data that has been set with commands that start with “MEASure[:NORMal]:ITEM.” | :MEASure[:NORMal]:VALue?<br>:MEASure[:NORMal]:BINary? | A | | D | (1) :NUMeric:FORMat {ASCii\|FLOat}<br>(2) :NUMeric[:NORMal]:VALue? |

## RECall Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the recall interval. | :RECall:INTerval | D | The function for recalling to the screen of this instrument is not available. | D | |
| Loads a setup parameter file. | :RECall:PANel | A | | A | |
| Sets or queries the recall on/off state. | :RECall[:STATe] | D | The function for recalling to the screen of this instrument is not available. | D | |

## RELay Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the comparator function. | :RELay:… | D | No comparator function | D | |

## SAMPle Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the output data (display, communication, etc.) hold state. | :SAMPle:HOLD | A | | A | :HOLD |
| Sets or queries the data update interval. | :SAMPle:RATE | A | | A | :RATE |

## STATus Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Queries the contents of the condition register. | :STATus:CONDition? | A | | A | |
| Sets or queries the extended event enable register. | :STATus:EESE | A | | A | |
| Queries the contents of the extended event register and clears the register. | :STATus:EESR? | A | | A | |
| Queries the error code and message of the last error that has occurred (top of the error queue). | :STATus:ERRor? | A | | A | |
| Sets or queries the transition filter. | :STATus:FILTer<x><br><x> = 1 to 16 | A | | A | |
| Sets or queries whether message information will be attached to the response to the STATus:ERRor? query. | :STATus:QMESsage | A | | A | |
| Executes serial polling. | :STATus:SPOLl? | A | | A | |

## STORe Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Sets or queries the storage interval. | :STORe:INTerval | B | Cannot be set with a string. | B | Cannot be set with a string. |
| Save setup parameters to a file. | :STORe:PANel | A | | A | |
| Sets or queries the storage on/off state. | :STORe[:STATe] | A | | A | |

## Common Command Group

| Function | WT210/WT230 Command | Command of This Instrument | | | |
|---|---|---|---|---|---|
| | | Command Mode: WT200 | | Command Mode: WT300 | |
| Performs zero-level compensation and queries the result. | *CAL? | A | | A | |
| Clears the standard event register, extended event register, and error queue. | *CLS | A | | A | |
| Sets or queries the standard event enable register. | *ESE | A | | A | |
| Queries and clears the standard event register. | *ESR? | A | | A | |
| Queries the model information. | *IDN? | A | | A | |
| Queries the option information. | *OPT? | A | | A | |
| Sets or queries whether registers will be cleared at power-on. | *PSC | D | | D | |
| Initializes setup parameters. | *RST | A | | A | |
| Sets or queries the service request enable register. | *SRE | A | | A | |
| Queries the Status Byte Register value. | *STB? | A | | A | |
| Executes the same operation as the TRIG (SHIFT+HOLD) key on the front panel. | *TRG | A | | A | |
| Performs a self-test and queries the result. | *TST? | A | | A | |

## Correspondence Table of WT210/WT230 Functions and Functions of This Instrument

Function expressions are shown below. For the WT210/WT230 compatible command mode, see the WT210/WT230 column.

| WT210/ WT230 | This Instrument | Notes |
|---|---|---|
| **Normal measurement** | | |
| V | U | |
| A | I | |
| W | P | |
| VA | S | |
| VAR | Q | |
| PF | LAMBda | |
| DEGRee | PHI | |
| VHZ | FU | |
| AHZ | FI | |
| WH | WH | |
| WHP | WHP | |
| WHM | WHM | |
| AH | AH | |
| AHP | AHP | |
| AHM | AHM | |
| VPK | UPeak | The larger of the absolute values UPPeak and UMPeak |
| APK | IPeak | The larger of the absolute values IPPeak and IMPeak |
| TIME {} | TIME | |
| MATH | MATH | |
| | UPPeak | 1 |
| | UMPeak | 1 |
| | IPPeak | 1 |
| | IMPeak | 1 |
| | PPPeak | 1 |
| | PMPeak | 1 |
| **Harmonic measurement** | | |
| V | U | |
| A | I | |
| W | P | |
| PF | LAMBda | |
| VHZ | FU | |
| AHZ | FI | |
| VTHD | UTHD | |
| ATHD | ITHD | |
| VCON | UHDF | |
| ACON | IHDF | |
| WCON | PHDF | |
| VDEG | PHIU | |
| ADEG | PHII | |
| ORDer | ORDer | |

1 New functions on this instrument

# Appendix 1 Error Messages

This section explains communication error messages.

- Error messages that are read from a PC such as through the :STATus:ERRor? command are output in English.
- If servicing is necessary to solve the problem indicated by a message, contact your nearest YOKOGAWA dealer.
- Only communication error messages are listed here. For details on other error messages, see the Getting Started Guide, IM WT310-02EN.

| | | |
|---|---|---|
| • Communication syntax errors | 100 to 199 | Listed below |
| • Communication execution errors | 200 to 299 | |
| • Device-specific and other errors | 300 to 399 | |
| • Communication query errors | 400 to 499 | |
| • System errors (communication) | 300, 399 | |
| • Information (1 to 99) | 1 to 99 | Listed in section 6.2 of the Getting Started Guide, IM WT310-02EN |
| • Execution Errors (600 to 899) | 600 to 899 | |
| • System Errors | 900 to 999 | |

## Communication Syntax Errors (100 to 199)
## Error in communication command

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 102 | Syntax error. | A syntax error not covered by error codes 100 to 199. | Chapters 5 and 6 |
| 103 | Invalid separator. | Separate data values with a comma. | 5-1 |
| 104 | Data type error. | See page 5-6 and 5-7 and use the correct data type for each parameter. | 5-6 and 5-7 |
| 108 | Parameter not allowed. | Check the number of data values. | 5-6 and chapter 6 |
| 109 | Missing parameter. | Be sure to include all necessary data values. | 5-6 and chapter 6 |
| 111 | Header separator error. | Use a comma to separate each header from its data. | 5-1 |
| 112 | Program mnemonic too long. | Check the command length. | Chapter 6 |
| 113 | Undefined header. | Check the header. | Chapter 6 |
| 114 | Header suffix out of range. | Check the header. | Chapter 6 |
| 120 | Numeric data error. | A value must be specified where the syntax contains <NRf>. | 5-6 |
| 123 | Exponent too large. | Where the syntax contains <NR3>, make the exponent that follows E smaller. | 5-6 and Chapter 6 |
| 124 | Too many digits. | Limit numeric values to 255 digits or less. | 5-6 and Chapter 6 |
| 128 | Numeric data not allowed. | Use a data type other than <NRf>. | 5-6 and Chapter 6 |
| 131 | Invalid suffix. | Check the unit of <Voltage>, <Current>, <Time>, or <Frequency>. | 5-6 |
| 134 | Suffix too long. | Check the unit of <Voltage>, <Current>, <Time>, or <Frequency>. | 5-6 |
| 138 | Suffix not allowed. | Only the following units can be used: <Voltage>, <Current>, <Time>, <Frequency>. | 5-6 |
| 141 | Invalid character data. | Be sure to select one of the listed choices when the syntax contains {...\|...\|...}. | Chapters 5 and 6 |
| 144 | Character data too long. | Check the spelling of the strings when the syntax contains {...\|...\|...}. | Chapter 6 |
| 148 | Character data not allowed. | Use a data type other than {...\|...\|...}. | Chapter 6 |
| 150 | String data error. | Enclose parameters with single or double quotation marks where the syntax contains <String>. | 5-7 |

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 151 | Invalid string data. | The parameter is either too long, or it contains an unusable character. | Chapter 6 |
| 158 | String data not allowed. | Use a data type other than <String>. | Chapter 6 |
| 161 | Invalid block data. | <Block data> cannot be used. | 5-7 and chapter 6 |
| 168 | Block data not allowed. | <Block data> cannot be used. | 5-7 and Chapter 6 |
| 171 | Missing Right | Mathematical operations cannot be used. | — |
| 172 | Invalid expression. | Mathematical operations cannot be used. | Chapter 6 |
| 178 | Expression data not allowed. | Mathematical operations cannot be used. | Chapter 6 |
| 181 | Invalid outside macro definition. | This instrument does not support the IEEE 488.2 macro specifications. | — |

## Communication Execution Errors (200 to 299)
## Error in communication execution

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 221 | Setting conflict. | Check settings that are related to each other. | Chapter 6 |
| 222 | Data out of range. | Check the ranges of the settings. | Chapter 6 |
| 223 | Too much data. | Check data byte lengths. | Chapter 6 |
| 224 | Illegal parameter value. | Check the ranges of the settings. | Chapter 6 |
| 225 | OverFlow. | Keep program messages to 1024 bytes or less in length, including <PMT>. | 5-2 |
| 226 | Out Of Memory. | Keep program messages to 1024 bytes or less in length, including <PMT>. | 5-2 |
| 241 | Hardware missing. | Check that the specified options are all installed. | — |
| 260 | Expression error. | Mathematical operations cannot be used. | — |
| 270 | Macro error. | This instrument does not support the IEEE 488.2 macro specifications. | — |
| 272 | Macro execution error. | This instrument does not support the IEEE 488.2 macro specifications. | — |
| 273 | Illegal macro label. | This instrument does not support the IEEE 488.2 macro specifications. | — |
| 275 | Macro definition too long. | This instrument does not support the IEEE 488.2 macro specifications. | — |
| 276 | Macro recursion error. | This instrument does not support the IEEE 488.2 macro specifications. | — |
| 277 | Macro redefinition not allowed. | This instrument does not support the IEEE 488.2 macro specifications. | — |
| 278 | Macro header not found. | This instrument does not support the IEEE 488.2 macro specifications. | — |

## Communication Query Errors (400 to 499)
## Error in communication Query

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 410 | Query INTERRUPTED. | Check the transmission and reception order. | 5-2 |
| 420 | Query UNTERMINATED. | Check the transmission and reception order. | 5-2 |
| 430 | Query DEADLOCKED. | Keep program messages to 1024 bytes or less in length, including <PMT>. | 5-2 |
| 440 | Query UNTERMINATED after indefinite response. | Do not write a query after *IDN? or *OPT?. | — |

## System Communication Errors (300 and 399)
## Error in System Operation

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 300 | Communication device-specific error. | Servicing is required. | — |
| 399 | Fatal error in the communication driver. | Servicing is required. | — |

## Communication Warning (50)
## Warning

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 50 | *OPC/? exists in message. | Write *OPC or *OPC? at the end of program messages. | — |

## Other Errors (350, 390)

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 350 | Queue overflow. | Read the error queue. | 5-6 |
| 390 | Overrun error (RS-232 only) | Decrease the baud rate. | 3-8 |

***Note***

Code 350 occurs when the error queue overflows. This error is only returned in response to a :STATus:ERRor? query; it is never displayed on the screen.

## Information (1 to 99)

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 3,80,87 | The system has been initialized | For descriptions of errors and their corrective actions, see section 6.2 in the Getting Started Guide, IM WT310-02EN. | — |

## Execution Errors (600 to 899)

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 759 | Failed to initialize network. | Check the network settings. | Chapter 4 |
| 832 | Internal memory access error. | For descriptions of errors and their corrective actions, see section 6.2 in the Getting Started Guide, IM WT310-02EN. | — |
| 841 to 847 | Integrator execute error. | | |
| Other than those above. (812,813,823 ,840,865) | Invalid operation. | | |

## System Errors (900 to 999)

| Code | Message | Corrective Action | Page |
|---|---|---|---|
| 901,915,919 | System error. | For descriptions of errors and their corrective actions, see section 6.2 in the Getting Started Guide, IM WT310-02EN. | — |

# Appendix 2 About the IEEE 488.2-1992 Standard

The GP-IB interface of this instrument conforms to the IEEE 488.2-1992 standard. This standard specifies that the following 23 items be stated in the document. This section describes these items.

**(1) Of the IEEE 488.1 interface functions, the subsets that are supported**
See section 2.2, "GP-IB Interface Features and Specifications."

**(2) The operation of the device when it is assigned an address outside the 0 to 30 range.**
The address of this instrument cannot be set to an address outside the 0 to 30 range.

**(3) Reaction of the device when the user changes the address**
The address change is detected when the user presses INTERFACE and changes the address on the GPib menu. The new address is valid until the next time it is changed.

**(4) Device settings at power-up. The commands that can be used at power-up.**
As a basic rule, the previous settings (the settings that were in use when this instrument was turned off) are used.
There are no limitations on the commands that can be used at power-up.

**(5) Message exchange options**
**(a) Input buffer size**
1024 bytes.
**(b) Queries that return multiple response messages**
See the example of the commands given in chapter 6.
**(c) Queries that create response data when the command syntax is being analyzed**
All queries create response data when the command syntax is analyzed.
**(d) Queries that create response data during reception**
There are no queries of which the response data are created upon receiving a send request from the controller.
**(e) Commands that have parameters that restrict one another**
See the example of the commands given in chapter 6.

**(6) Items that are included in the functional or composite header elements constituting a command**
See chapters 5 and 6.

**(7) Buffer sizes that affect block data transmission**
When block data is being transmitted, the output queue is expanded to match the size of the data that is being transmitted.

**(8) A list of program data elements that can be used in equations and their nesting limitations**
Equations cannot be used.

**(9) Syntax of the responses to queries**
See the example of the commands given in chapter 6.

**(10) Communication between devices that do not follow the response syntax**
Not supported.

**(11) Size of the response data block**
0 to 24576 bytes

**(12) A list of supported common commands**
See section 6.17, "Common Command Group."

**(13) Device condition after a successful calibration**
The device will be performing measurements.

**(14) The maximum length of block data that can be used for the *DDT trigger macro definition**
Not supported.

**(15) The maximum length of the macro label for defining macros, the maximum length of block data that can be used for the macro definition, and the process when recursion is used in macro definitions**
Macro functions are not supported.

**(16) Reply to the *IDN? query**
See section 6.17, "Common Command Group."

**(17) Size of storage area for protected user data for PUD and *PUD?**
*PUD and *PUD? are not supported.

**(18) The length of the *RDT and *RDT? resource names**
*RDT and *RDT? are not supported.

**(19) The change in the status due to *RST, *LRN?, *RCL, *SAV, and *RST**
See section 6.17, "Common Command Group."
***LRN?, *RCL, and *SAV**
These common commands are not supported.

**(20) The extent of the self-test using the *TST? command**
See section 6.17, "Common Command Group."

**(21) The structure of the extended return status**
See chapter 7.

**(22) Whether each command is processed in an overlapped manner or sequentially**
See section 5.5, "Synchronization with the Controller" and chapter 6.

**(23) The description of the execution of each command**
See the explanations of each command's function in chapter 6 and the User's Manual, IM WT310-01EN.

# Index

## P



## Q



## R



## S



## T



## U



## V



## W



## X



## Z