Anritsu

MG9637A/MG9638A

Tunable Laser Source

Remote Control

Operation Manual

MG9638A

■ Read this manual before using the equipment. Keep this manual with the equipment.

MG9637A/MG9638A
Tunable Laser Source
Remote Control
Operation Manual

Second Edition

Read this manual before using the equipment.
Keep this manual with the equipment.

Measuring Instruments Division
Measurement Group
ANRITSU CORPORATION

Document No.: M-W1214AE-2.0

MG9637A/MG9638A
Tunable Laser Source Remote Control
Operation Manual

10 February 1997 (First Edition)
24 October 1997 (Second Edition)

Preface

This book explains how to remote-control the MG9637A/MG9638A tunable laser source (TLS). The remote-control function enables to control a computer connected to the GPIB and RS-232C interfaces of the MG9637A/MG9638A tunable laser source and fetch the measurement result into the computer.

Table of Contents

Section 1 Outline

This chapter outlines the remote-control function of the MG9637A/MG9638A tunable laser source.

1.1 Outline

The MG9637A/MG9638A tunable laser source, combined with an external controller (host computer, personal computer, etc.), enables to automatize measurement. This unit provides a GPIB interface bus (IEEE Std 488-2-1987) and RS-232C interface port.

1.2 MG9637A/MG9638A Remote Control

The MG9637A/MG9638A has the following seven functions:

(1) Controlling functions excluding the POWER switch, LOCAL key, etc.

(2) Reading setting conditions

(3) Setting GPIB address using panel

(4) Interrupt function and serial pole operation (GPIB)

(5) Setting RS-232C interface conditions using panel

(6) Selecting interface port use conditions using panel

(7) Configuring automatic measurement system by combination with personal computer and other measuring instruments

1.3 Interface Port Use Selection

The MG9637A/MG9638A provides a GPIB interface bus and RS-232C interface as standard interface ports with an external device. Select the use of these interface ports using the panel.

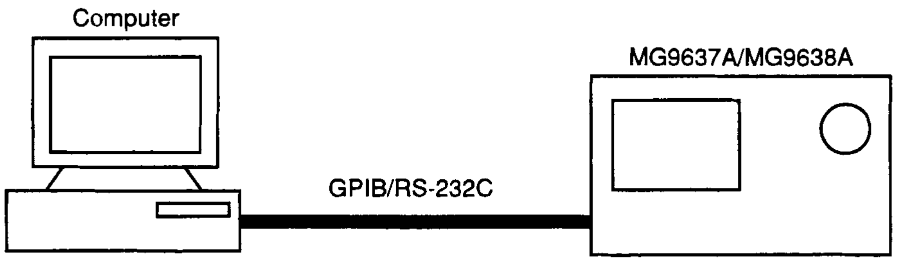
Connection port with external controller	: GPIB or RS-232C
Connection port with MS9710A optical spectrum analyzer	: RS-232C

These connection ports cannot be used at the same time.

1.4 System-Up Examples Using GPIB/RS-232C

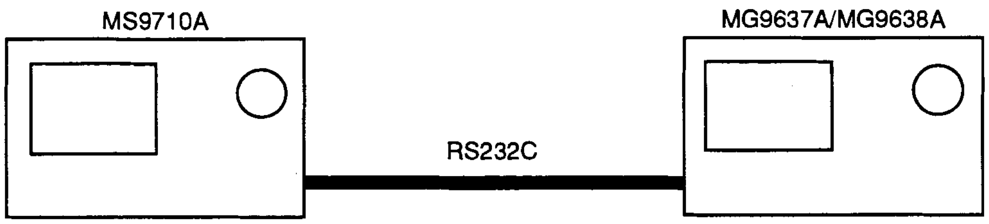
(1) Host computer control

The system is controlled in automatic and remote modes from a computer.



(2) Interlocking with MS9710A optical spectrum analyzer

The MG9637A/MG9638A interlocks with the MS9710A for measurement by MS9710A key operation.



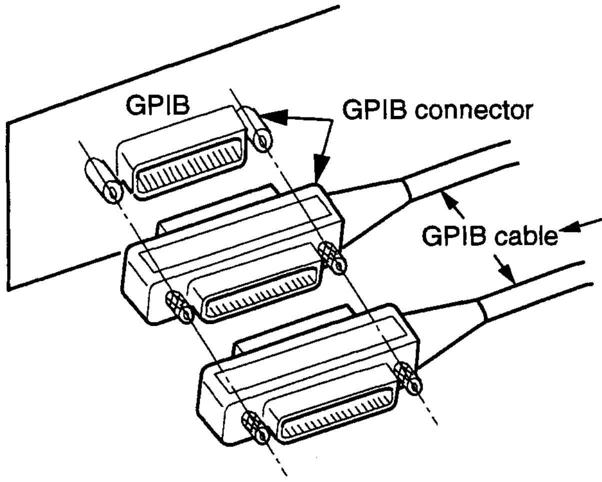
Section 2 Connection

This chapter explains how to connect the GPIB and RS-232C cables with an external device such as a host computer and personal computer and also how to set up those interfaces for this unit.

2.1 Using GPIB Cable for Device Connection

The GPIB cable connector is attached onto the rear panel. Before turning the power on, be sure to connect the GPIB cable.

Up to 15 devices (including a controller) can be connected to one system under the conditions shown on the right of the figure below.



Total cable length	≤20 m
Device-to-device cable length	≤4 m
Number of connectable devices	≤15

2.1.1 Connection port interface setting

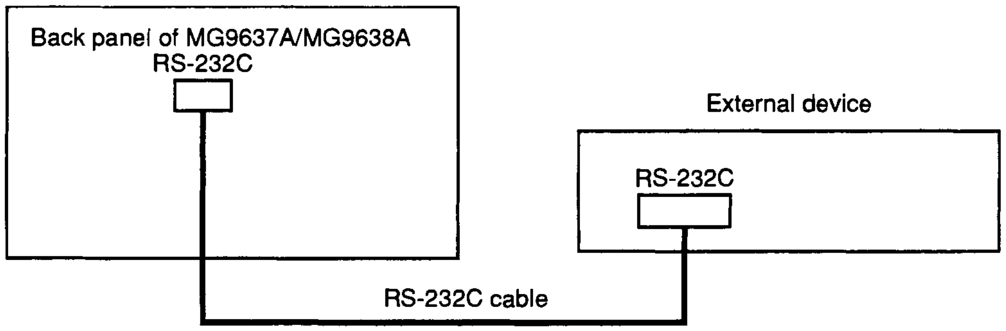
To control the MG9637A/MG9638A in automatic and remote modes, set up a connection port interface. In this case, set the interface GPIB/RS-232C key in advance mode (2/2) to the interface GPIB. After the power is turned on, the GPIB or RS-232C, whichever signal was sent, is set up.

2.1.2 Address confirmation and setting

After the power is turned on, set the GPIB address of the MG9637A/MG9638A. At delivery, GPIB address 24 is already set by battery backup. If address 24 remains unchanged, the GPIB address need not be set. To set a new GPIB address, place the MG9637A/MG9638A into the local state, press the GPIB Address key, and key in a required GPIB address or use the encoder. When the power is turned on, generally, a GPIB device is placed into the local state.

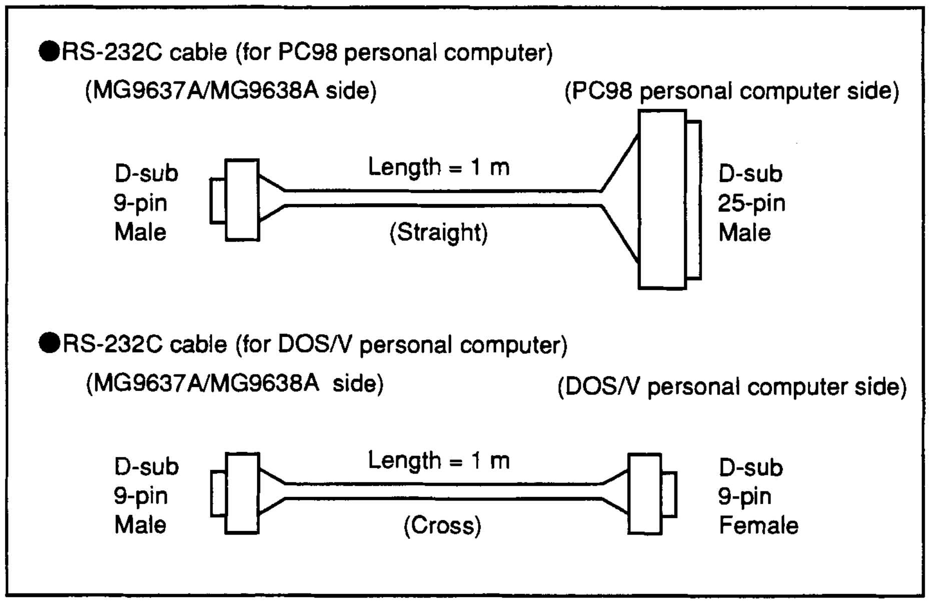
2.2 Using RS-232C Cable for Device Connection

Connect the RS-232C connector (D-sub, 9-pin, female) on the rear of this unit to one of an external device via the RS-232C cable.



Note :

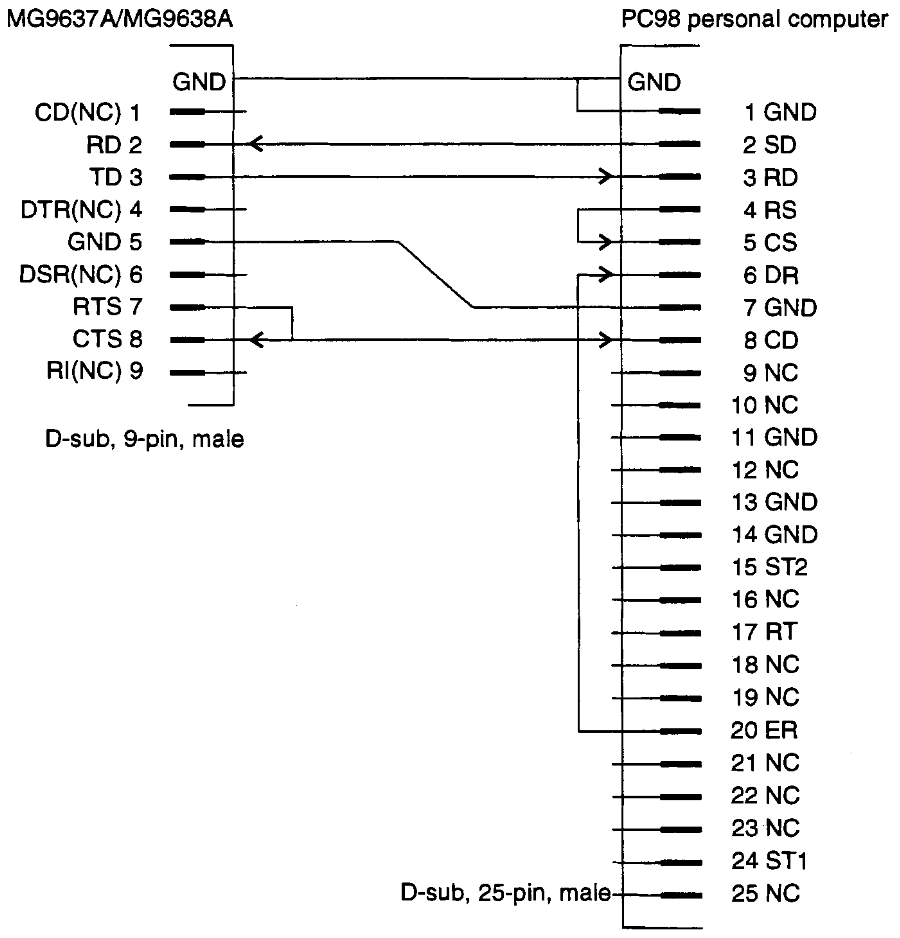
There are two types of RS-232C connectors: 9- and 25-pin connectors. Before purchasing an RS-232C cable, therefore, confirm the number of pins on the RS-232C connector of your external device. This unit has two types of RS-232C cables as application parts.



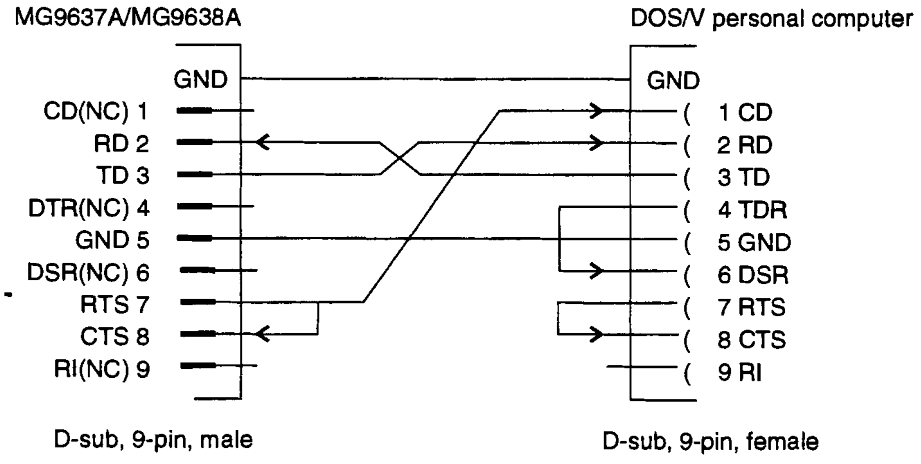
2.2.1 RS-232C interface signal connection diagram

The figures below show a connection diagram of the RS-232C interface signal transferred between the MG9637A/MG9638A and personal computer.

• Connection diagram with PC98 personal computers



• Connection diagram with PC98 personal computers



2.2.2 Connection port interface setting

To control the MG9637A/MG9638A from a computer in automatic and remote modes, set up a connection port interface. In this case, set the interface GPIB/RS-232C key in advance mode (2/2) to the RS-232C interface.

2.2.3 RS-232C interface condition setting

The RS-232C interface conditions of this unit must match those of the connected external device.

Pressing the RS-232C Set Up key in advance mode (2/2) displays the screen below.

Baud Rate	9600
Length of Character	8
Parity	None
Length of Stop Bit	1

Press the Baud Rate, Length of Character, Parity Bit, and Length of Stop Bit keys respectively, then use the < or > key to change each value.

Item	Meaning
Baud Rate	Select the communication speed (baud rate) from 2400, 4800, and 9600 bps.
Parity	Select the parity bit state. None Parity bit not added Even Even parity bit added Odd Odd parity bit added
Length of Stop Bit	Select the stop bit state. 1 One stop bit added 2 Two stop bits added
Length of Character	Select the character length. 7 7 bits 8 8 bits

2.2.4 RS-232C interface specifications

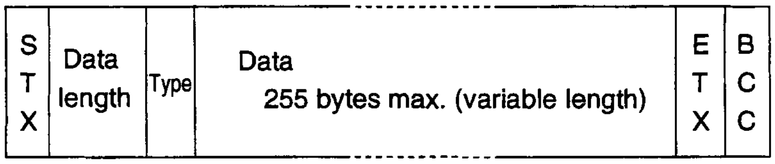
2.2.4.1 Transmission control characters

The table below gives the transmission control characters.

Name	Symbol	Code	Contents
ACKnowledge	ACK	06H	Acknowledgment
NEgative AcKnowlege	NAK	15H	Negative acknowledgment
Start of TeXt	STX	02H	Start of transmitted text
End of TeXt	ETX	03H	End of transmitted text

2.2.4.2 Message format

A message transmitted from a computer, that consists of a maximum 260 bytes, begins with STX and ends with BCC.



Data length :

Indicates a data length with 1-byte binary data.

Type :

Indicates a type of the sent data with one of the codes below.

00H	: Not used.
01H	: Command sending (PC -> TLS Data sending)
02H	: Not used.
03H	: Query command sending (PC -> TLS Data request)
04H	: Not used.
05H	: Not used.
06H	: Not used.
07H	: Response message (TLS -> PC End of request data)
08H	: Format response normal (TLS -> PC Normal response)
09H	: Format response abnormal (TLS -> PC Abnormal response)

Data field :
The contents of the data field vary depending on the type of sent data.

a) Control and query commands
   - Header only
     <Header>
   - Header with one data item
     <Header> <Data>
   - Header with multiple data items
     <Header> <Data>, <Data>, ... <Data>

b) Response
   <Data (binary)>

c) Format response
   The data field is not sent.

Note :

Two or more commands cannot be sent with one message delimited by STX, ... ETX and BCC.

BCC (Block check character) :
One byte for horizontal parity check; indicates an exclusive OR (EXOR) for each bit in bytes ranging from the data length to ETX.

2.2.4.3 Transmission procedure

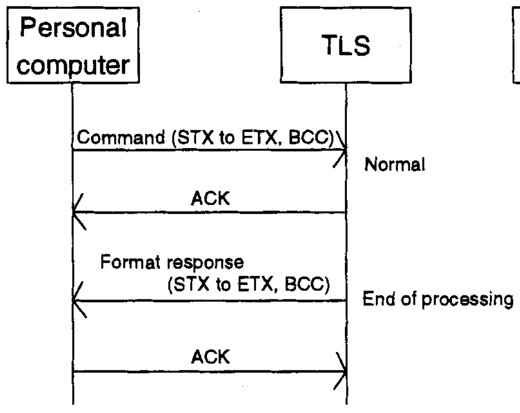
This section explains the command and response sending procedures in the cases below.

- Data transfer in normal state
- Data transfer in abnormal state
- At no response
- Command
- Query command
- Command error
- Query command error

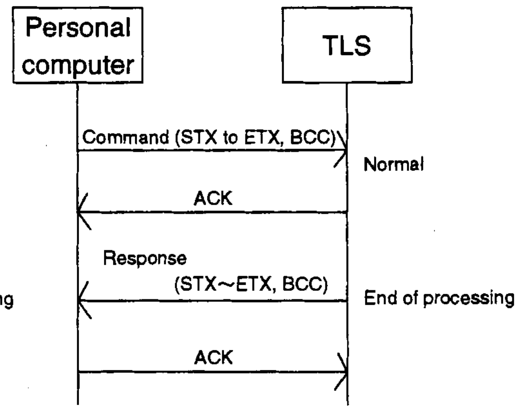
For details, see Appendix D, "RS-232C Use Examples."

(1) Data transfer in normal state

Send an acknowledgment (ACK), then return a format response for control command and response for query command.



Control command

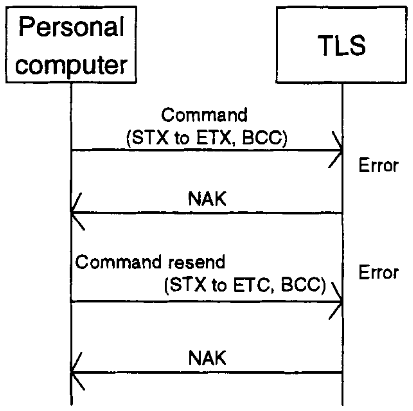


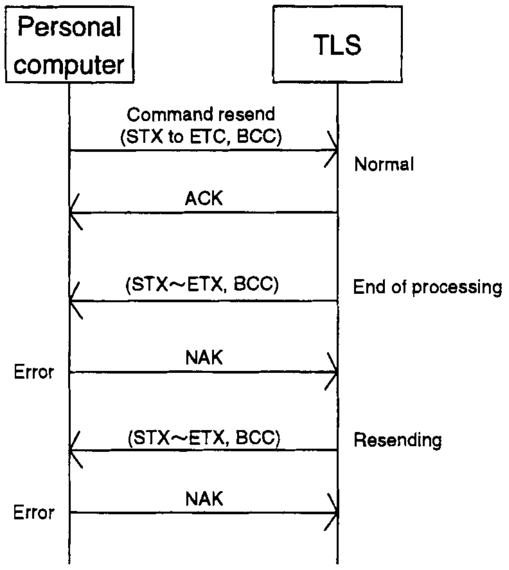
Query command

(2) Data transfer in abnormal state

If the TLS detects a communication error, it sends a negative acknowledgment (NAK). The computer receives the NAK and resends a command.

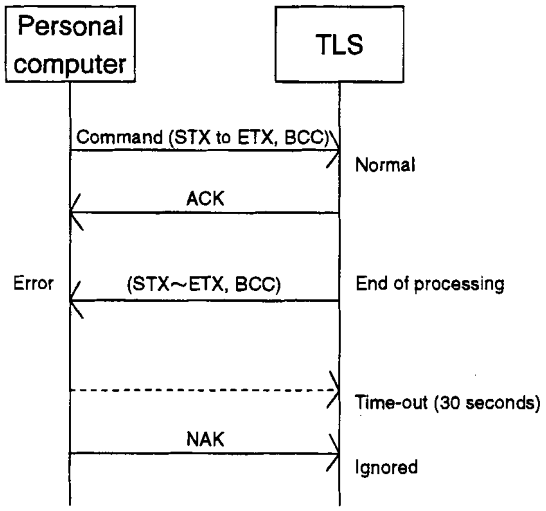
If the computer detects a communication error on a response sent from the TLS, it resends NAK up to two times.





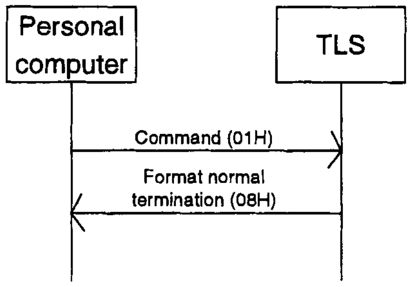
(3) At no response

If no response is returned from the computer within 30 seconds after a message was sent, the TLS stops waiting a response. When NAK is returned after that, the TLS assumes that the message was received normally.



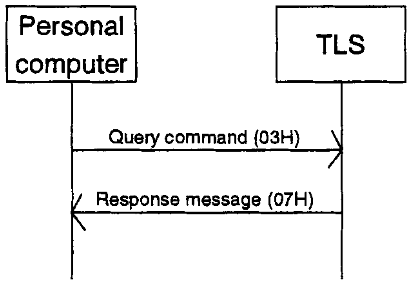
(4) Command

01H is set in the type field of a command that completed sending all data in a message format. After the command is executed, the TLS returns the message "Format Normal Termination."



(5) Query command

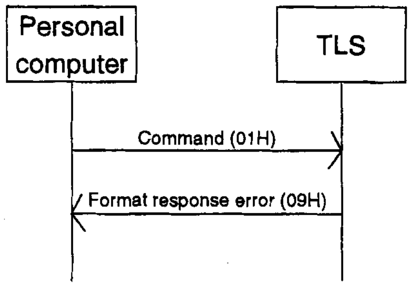
When the computer requests data, it sends a command for which the type field is 03H. The TLS returns a response for which the type field is 07H.



(6) Command error

When the TLS detects an error, it sends the message "Format Response Error."

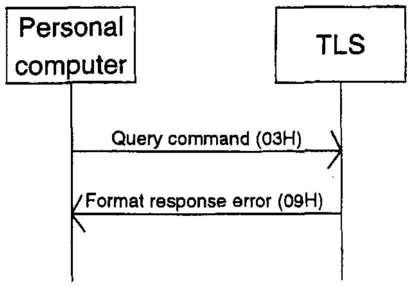
- An undefined command was received.
- A syntax error was detected in the command.
- An execution error occurred.



(7) Query command error

When the TLS detects one of the errors below, it sends the message "Format Response Error."

- An undefined command was received.
- A syntax error was detected in the command.
- An execution error occurred.
- There is no read data.



Section 3 Standard

This chapter explains the MG9637A/MG9638A GPIB standard and the RS-232C standard in addition to the device message list.

3.1 GPIB Standard

The table below gives the MG9637A/MG9638A GPIB standard.

Item	Typical value and supplemental explanation
Function	IEEE 488.2 compatible Controlled from an external controller, using this unit as a device. Controls a printer, using this unit as a controller.
Interface function*	SH1 : Has all source handshake functions; takes a data sending timing. AH1 : Has all acceptor handshake functions; takes a data receiving timing. T5 : Has the basic talker function, serial port function, and MLA talker release function without talk only function. L4 : Has the basic listener function and MTA listener release function without listen only function. SR1 : Has all service request and status byte functions. RL1 : Has all remote and local functions with the local lock-out function. PP0 : Without parallel pole function DC1 : With all device clear functions DT0 : Without device trigger function C0 : Without controller function E2 : Open collector

For details on the interface function subset, refer to Chapter 1, "GPIB Basic Knowledge" in the GPIB Basic Guide (sold separately).

3.2 RS-232C Standard

The table below gives the MG9637A/MG9638A RS-232C standard.

Item	Typical value
Function	Control from external controller
Communication method	Asynchronous (asynchronous method), full-duplex
Communication control method	No flow control
Baud rate	2400, 4800, 9600 bps
Data bit	7 bits, 8 bits
Parity	Odd (ODD), Even (EVEN), None (NON)
Start bit	1 bit
Stop bit	1 bit, 2 bits
Connector	D-sub, 9-pin, female

3.3 Device Message List

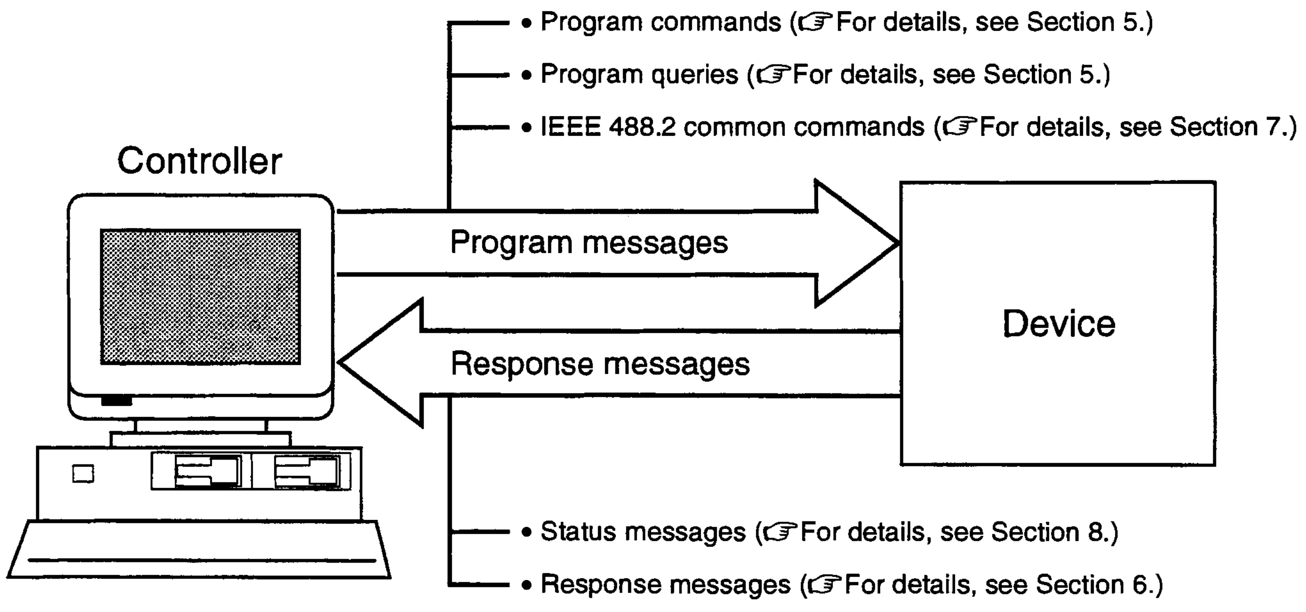
The device message is a data message transferred between the controller and device. There are two types of device messages : program message and response message.

The program message is an ASCII data message transferred from the controller to a device. There are two types of program messages : program command and program query command. For details, see Section 3.3.1 and after.

The program commands are classified into two types: commands particular to a device used only to control the MG9637A/ MG9638A and IEEE 488.2 common commands. The IEEE 488.2 common commands are program commands common to the MG9637A/MG9638A and other IEEE 488.2 compatible measuring instruments connected to the GPIB interface bus.

The program query command is used to obtain a response message from a device. It is transferred from the controller to a device in advance, and the controller accepts a response message from the device.

The response message is an ASCII data message transferred from a device to the controller. The response messages corresponding to status messages and program query commands are listed in Section 3.3.1 and after.


• Program commands (☞For details, see Section 5.)
• Program queries (☞For details, see Section 5.)
• IEEE 488.2 common commands (☞For details, see Section 7.)
Controller
Program messages
Device
Response messages
• Status messages (☞For details, see Section 8.)
• Response messages (☞For details, see Section 6.)

A suffix (unit) must be assigned to numerical data in the program data and response message of the data messages.

These messages are transferred via an input-output buffer of the device. The output buffer is also called an output queue. The table below gives a brief explanation on the input and output buffers.

Input buffer	Output queue
A memory area of the FIFO (first-in first-out) used to temporarily store DAB (program and query messages) before analyzing their syntaxes and executing those commands. The MG9637A/MG9638A input buffer size is 256 bytes.	A FIFO-type queue memory area. All DABs (response messages) output from a device to the controller are stored in this memory until they are read by the controller. The MG9637A/MG9638A output queue size is 256 bytes.

3.3.1 IEEE 488.2 common commands and supported commands

The table below lists 39 types of common commands defined in the IEEE 488.2 standard. Of these common commands, the IEEE 488.2 common commands used by the MG9637A/MG9638A are indicated by the mark ◎.

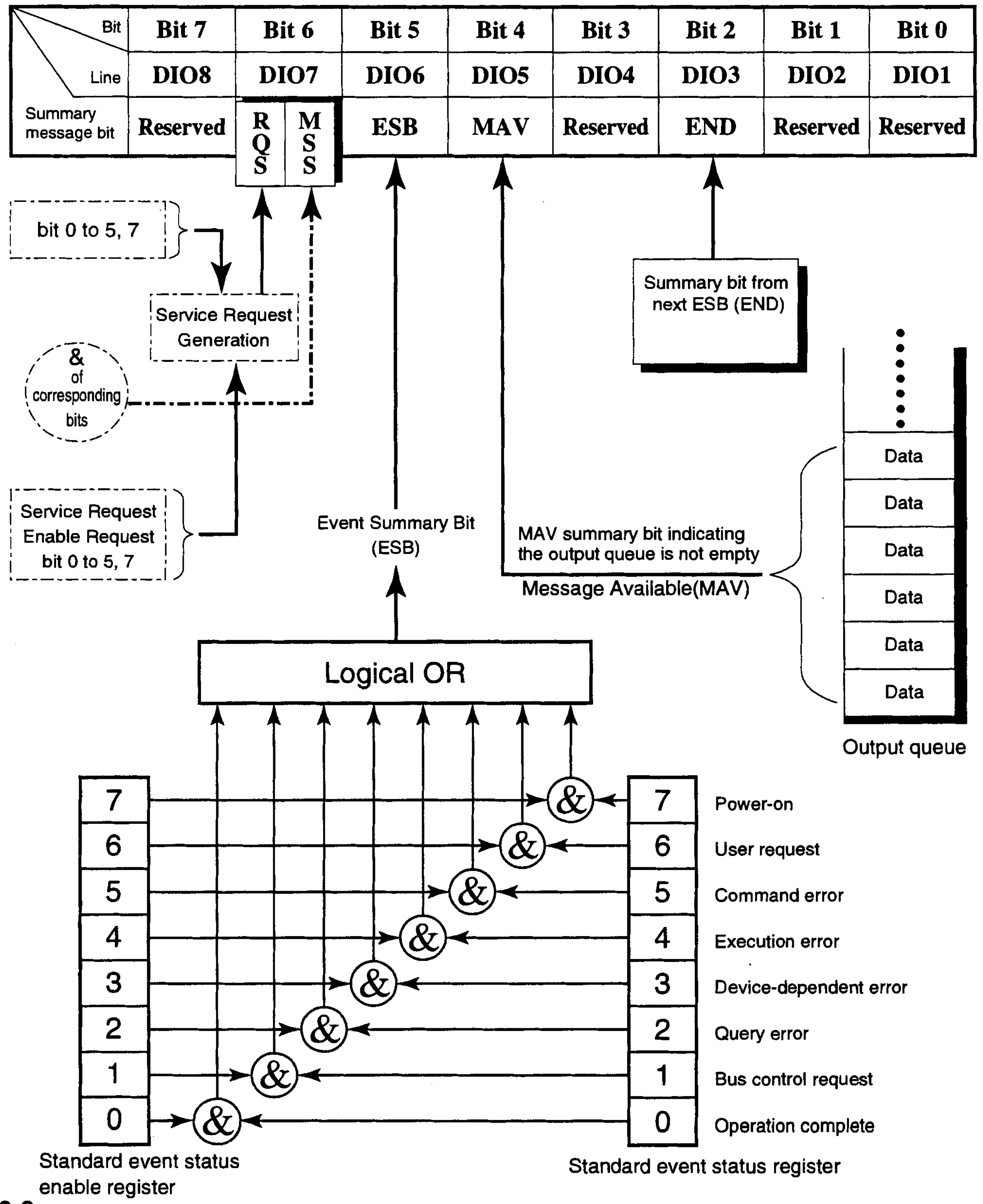
Mnemonic	Full spelled command name	Specification by IEEE 488.2	Support by MG9637A/MG9638
* ADD	Accept Address Command	Optional	
* CAL	Calibration Query	Optional	
* CLS	Clear Status Command	Required	◎
* DDT	Define Device Trigger Command	Optional	
* DDT?	Define Device Trigger Query	Optional	
* DLF	Disable Listener Function Command	Optional	
* DMC	Define Macro Command	Optional	
* EMC	Enable Macro Command	Optional	
* EMC?	Enable Macro Query	Optional	
* ESE	Standard Event Status Enable Command	Required	◎
* ESE?	Standard Event Status Enable Query	Required	◎
* ESR?	Standard Event Status Register Query	Required	◎
* GMC?	Get Macro contents Query	Optional	
* IDN?	Identification Query	Required	◎
* IST?	Individual Status Query	Optional	
* LMC?	Learn Macro Query	Optional	
* LRN?	Learn Device Setup Query	Optional	
* OPC	Operation Complete Command	Required	◎
* OPC?	Operation Complete Query	Required	◎
* OPT?	Option Identification Query	Optional	◎
* PCB	Pass Control Back Command	Other than C0 : Required	
* PMC	Purge Macro Command	Optional	
* PRE	Parallel Poll Register Enable Command	Optional	
* PRE?	Parallel Poll Register Enable Query	Optional	
* PSC	Power On Status Clear Command	Optional	
* PSC?	Power On Status Clear Query	Optional	
* PUD	Protected User Data Command	Optional	
* PUD?	Protected User Data Query	Optional	
* RCL	Recall Command	Optional	◎
* RDT	Resource Description Transfer Command	Optional	
* RDT?	Resource Description Transfer Query	Optional	
* RST	Reset Command	Required	◎
* SAV	Save Command	Optional	◎
* SRE	Service Request Enable Command	Required	◎
* SRE?	Service Request Enable Query	Required	◎
* STB?	Read Status Byte Query	Required	◎
* TRG	Trigger Command	DT1 : Required	
* TST?	Self Test Query	Required	◎
* WAI	Wait to Continue Command	Required	◎

☞ The IEEE 488.2 common commands always begin with an asterisk (*). For details, see Chapter 7.

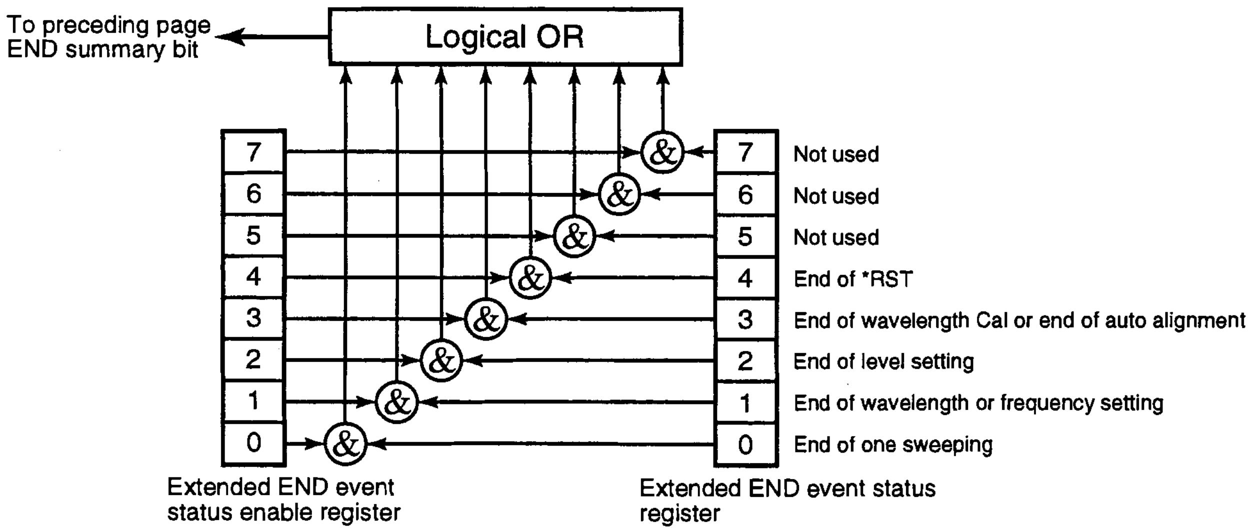
3.3.2 Status message

The figure below shows the structure of a service request summary message in the status byte register used by the MG9637A/MG9638A.

Summary bit configuration in status byte register



Bit / Line / Summary message bit	Bit 7	Bit 6		Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Line	DIO8	DIO7		DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
Summary message bit	Reserved	RQS	MSS	ESB	MAV	Reserved	END	Reserved	Reserved


To preceding page END summary bit
Logical OR
7
6
5
4
3
2
1
0
Not used
Not used
Not used
End of *RST
End of wavelength Cal or end of auto alignment
End of level setting
End of wavelength or frequency setting
End of one sweeping
Extended END event status enable register
Extended END event status register

3.3.3 MG9637A/MG9638A device message list

This section lists the MG9637A/MG9638A program commands, query commands, and response messages.

MG9637A/MG9638A device message list (1/5)

Function	Device message			Remarks
	Command	Data request	Response	
Optical Output				
Laser Output On/Off	OUTP s s = 0 or OFF 1 or ON	OUTP?	n n = 0 : Off 1 : On	
Output Condition		OUTC?	b $b = 2^0+2^1+2^2$ 2^0 : Key On/Off 2^1 : Fiber 2^2 : Interlock connector	b : Total weighted value
Modulation				
Modulation Int Frequency	AMIN p p = 0.2 to 20.0 kHz	AMIN?	n n = 2.00000000E+002 to 2.00000000E+004	Disabled in advance mode With unit for setting Read in Hz units.
Modulation Ext	AMEX			Disabled in advance mode
Modulation Off	AMOF			Disabled in advance mode
Modulation Status		AMST?	n n = 0 : Mod Off 1 : Mod Int 2 : Mod Ext	Disabled in advance mode
Power				
Power	POW p p = –20 dBm to 10 dBm = 10 uW to 10 mW	POW?	n n = -2.00000000E+001 to 1.00000000+001 = 1.00000000E-005 to 1.00000000E-002	With unit for setting Read in dBm or W units. The setting range varies depending on wavelengths.
Max Power	POWM	POWM?	n n = Max. power value	In sweep mode only The response format is the same as for Power.
Power Unit	POWU p p = dbm : dBm mw : mW uw : µW	POWU?	n n = 0 : dBm 1 : mW 2 : µW	Disabled in advance mode

MG9637A/MG9638A device message list (2/5)

Function	Device message			Remarks
	Command	Data request	Response	
Wavelengh				
Start Wavelengh	WSTA p p = 1.5 to 1.58 um = 1500 to 1580 nm	WSTA?	n n = 1.50000000E-006 to 1.58000000E-006	With unit for setting Read in m units. The setting range varies depending on devices. Disabled in CW mode.
Stop Wavelengh	WSTO p p = 1.5 to 1.58 um = 1500 to 1580 nm	WSTO?	n n = 1.50000000E-006 to 1.58000000E-006	With unit for setting Read in m units. The setting range varies depending on devices. Disabled in CW mode.
Center Wavelengh	WCNT p p = 1.5 to 1.58 um = 1500 to 1580 nm	WCNT?	n n = 1.50000000E-006 to 1.58000000E-006	With unit for setting Read in m units. The setting range varies depending on devices. Disabled in CW and sweep modes.
Span Wavelengh	WSPN p p = 0.002 to 80.000 nm = 2 to 80000 pm	WSPN?	n n = 2.00000000E-012 to 8.00000000E-008	With unit for setting Read in m units. The setting range varies depending on devices. Disabled in CW mode.
Step Wavelengh	WSTP p p = 0.001 to 80.000 nm = 1 to 80000 pm	WSTP?	n n = 1.00000000E-012 to 8.00000000E-008	With unit for setting Read in m units. The setting range varies depending on devices. Disabled in CW mode.
Frequency				
Start Frequency	FSTA p p = 189742.0 to 199861.6 GHz	FSTA?	n n = 1.89742000E+014 to 1.99861600E+014	With unit for setting Read in Hz units. The setting range varies depending on devices. Disabled in CW mode.

MG9637A/MG9638A device message list (3/5)

Function	Device message			Remarks
	Command	Data request	Response	
Frequency				
Stop Frequency	FSTO p p = 189742.0 to 199861.6 GHz	FSTO?	n n = 1.89742000E+014 to 1.99861600E+014	With unit for setting Read in Hz units. The setting range varies depending on devices. Disabled in CW mode.
Center Frequency	FCNT p p = 189742.0 to 199861.6 GHz	FCNT?	n n = 1.89742000E+014 to 1.99861600E+014	With unit for setting Read in Hz units. The setting range varies depending on devices. Disabled in CW and sweep modes.
Span Frequency	FSPN p p = 0.2 to 10000.0 GHz	FSPN?	n n = 2.00000000E+008 to 1.00000000E+013	With unit for setting Read in Hz units. The setting range varies depending on devices. Disabled in CW mode.
Step Frequency	FSTP p p = 0.1 to 10000.0 GHz	FSTP?	n n = 1.00000000E+008 to 1.00000000E+013	With unit for setting Read in Hz units. The setting range varies depending on devices. Disabled in CW mode.
Dwell Time	DWEL p p = 0.01 to 100 s	DWEL?	n n = 1.00000000E–002 to 1.00000000E+002	With s unit for setting
Sweep Speed	SWPT n n = 1 to 5	SWPT?	n n = 1 to 5	1 step tuning enabled only.
Sweep				
Single Sweep	SNGL			
Repeat Sweep	RPT			
Sweep Pause	PAUS			
Sweep Continue	CONT			
Sweep Status		SWST?	n n = 0 : Stop 1 : Repetitive sweeping currently 2 : Single sweeping currently	

MG9637A/MG9638A device message list (4/5)

Function	Device message			Remarks
	Command	Data request	Response	
Mode setting				
CW mode	MCW			
Sweep mode	MSWP			
Advance mode	MADV			
1 step tuning mode				
Mode Status	MONE	MST?	n n = 0 : CW mode 1 : Sweep mode 2 : 1 Step Tuning 3 : Advance mode	
Coherecy Control On/Off	COH s s = 0 or OFF 1 or ON		n n = 0 : Off 1 : On	
Wavelength setting confirmation		MOVE?	n n = 0 : End of wave length setting 1 : Wavelength setting currently	
Heat-up status reading		TEMP?	p p = 0 to 100 %	Output with unit
Current output laser wavelength reading		OUTW?	n n = 1.50000000E-006 to 1.58000000E-006	Read in m units.
Current output laser frequency reading		OUTF?	n n = 1.89742000E+014 to 1.99861600E+014	Read in Hz units.
Display Enable On/Off	DENA s s = 0 or OFF 1 or ON	DENA?	n n = 0 : Off 1 : On	
Display Revrece On/Off	DREV s s = 0 or OFF 1 or ON	DREV?	n n = 0 : Off 1 : On	
Wave or Freq mode setting	SETM s s = wave freq	SETM?	n n = 0 : Wave 1 : Freq	

MG9637A/MG9638A device message list (5/5)

Function	Device message			Remarks
	Command	Data request	Response	
Wavelength calibration				
Execution of wavelength Cal	CAL s s = start stop	CAL?	n n= 0 : Normal termination 1 : Currently executed 2 : Abnormal termination	
Cal wavelength setting	CALW p p = 1.5 to 1.58 µm = 1500 to 1580 nm	CALW?	n = 1.50000000E–006 to 1.58000000E–006	Read in m units.
Cal frequency setting	CALF p p = 189742.0 to 199861.6 GHz	CALF?	n n = 1.89742000E+014 to 1.99861600E+014	Read in Hz units.
Frequency Offset	FOFS p p = –50 to 50 GHz	FOFS?	n n = –5.00000000E+010 to 5.00000000E+010	Read in Hz units.
Auto alignment	XALN p p = init start stop	XALN?	n n= 0 : Normal termination 1 : Currently executed 2 : Abnormal termination	
Laser cut-off ON or OFF	: SET : NOP s s = 0 or OFF 1 or ON	: SET : NOP?	n = 0 : Off 1 : On	
Extension event register 2				
Extension event register 2 bit setting	ESE2	ESE2?	b b = 0 to 255	
Extension event register 2 reading		ESR2?	b b = 0 to 255	
Error reading		ERR?	n n = Error code	

Section 4 Initialization

The GPIB interface system is initialized at three levels. At level 1, "bus initialization", a system bus is placed into idle state. At level 2, "message exchange initialization", a device is placed into the program message receivable state. At level 3, "device initialization", device functions are initialized. These initialization levels, 1 to 3, are equivalent to a preparation for starting device operation.

Section 4 Initialization

In the conventional IEEE 488.1, the following two initialization methods are defined for GPIB systems:

- Bus initialization .... Initializes all interface functions connected to the bus with an IFC message sent from the controller.
- Device initialization ...... Returns all GPIB devices with GPIB bus command DCL or only a specified device with GPIB bus command SDC to the initial state defined for each device.

In the IEEE 488.2, the GPIB system initialization is classified into three levels. Level 1, "bus initialization", is located at the highest position. "Device initialization" is divided into two levels: level 2 "message exchange initialization" and level 3 "device initialization." The device at power-on is defined into a specific state.

These contents are summarized in the table below.

Level	Initialization type	Outline	Level combination and order
1	Bus initialization	Initializes all interface functions connected to the bus with an IFC message sent from the controller	Combinable with another level; however, level 1 must be executed before level 2.
2	Message exchange initialization	Initializes message exchange for all GPIB devices with GPIB bus command DCL or a specified device with GPIB bus command SDC ; invalidates a function that reports the end of operation to the controller.	Combinable with another level; however, level 2 must be executed before level 3.
3	Device initialization	Returns only the specified GPIB device to its known state with an *RST command regardless of the past use status.	Combinable with another level; however, level 3 must be executed before levels 1 and 2.

The device initialization function at level 3 is available to control the MG9637A/MG9638A from the controller using the RS-232C interface port. The initialization functions at levels 2 and 3 are not applicable. To control the MG9637A/ MG9638A from the controller using the GPIB interface bus, all the initialization functions at levels 1 to 3 are available.

This chapter explains the instructions for executing levels 1 to 3 and the items to be initialized that are their results. It also describes the known state set at power-on.

4.1 Bus Initialization with IFC Statement

■ Format

IFC Δ @ select-code

■ Example

IFC @ 1

■ Description

This function is available to control the MG9637A/MG9638A from the controller using the GPIB interface bus.

In the GPIB fitting to a specified select code, the IFC line is placed into active state (electrically low level) for about 100 É s. Executing IFC@ initializes the interface function of all devices connected to the GPIB bus line of the specified select code. Only the system controller can send data.

The initialization of the interface function is to release the state (talker, listener, etc.) of the interface function of a device set by the controller and return it to the initial state. The functions marked by o in the table below are initialized. Mark _ indicates that a part is initialized.

No	Function	Symbol	Initialization by IFC
1	Source handshake	SH	○
2	Acceptor handshake	AH	○
3	Talker or extended talker	T or TE	○
4	Listener or extended listener	L or LT	○
5	Service request	SR	△
6	Remote/local	RL	
7	Parallel/poll	PP	
8	Device clear	DC	
9	Device trigger	DT	
10	Controller	C	○

When the IFC statement is true (the IFC line is set to a low level by executing the IFC@ statement), levels 2 and 3 are not initialized; therefore, there is no effect on the device operation state.

In the table above, the device states are initialized with the IFC statement as follows :

1) Talker/listener .... All talkers and listeners enter the idle state (TIDS, LIDS) within 100 μ s.

2) Controller .... The controller enters the idle state (CIDS : controller idle state) if it is not active (SACS : system control active state).

3) Control right return .... If the system controller (device defined first as a controller on the GPIB) transfers its functions to another device at execution of the IFC@ statement, those functions are returned to the system controller. The IFC message is generally displayed from the system controller by pressing the RESET key of the system controller.

4) Service request device .... A state in which the device sends an SRQ message to the controller (state in which the SRQ line is set to a low level by the device) is not released with the IFC statement. However, executing the IFC statement releases a state in which the controller places all devices under the system bus into the serial pole mode.

5) Device in remote mode .... A device placed currently in remote mode is not released by executing the IFC statement.

4.2 Message Exchange Initialization by DCL and SDC Bus Commands

■ Format

DCL Δ @ select-code [primary-address] [secondary-address]

■ Example

DCL@1	Initializes message exchange for all devices under the bus. (DCL sending)
DCL@103	Initializes message exchange only for the device allocated to address 3. (SDC sending)

■ Description

This function is available to control the MG9637A/MG9638A from the controller using the GPIB interface bus.

The DCL@ statement initializes message exchange for all devices on the GPIB having the specified select code or only for the specified device.

The initialization of message exchange is aimed at preparing for sending a new instruction from the controller by initializing message exchange when a part related to message exchange in the device is unavailable to control from the controller because another program runs although the panel setting need not be changed.

■ Select code only specified

The DCL@ statement initializes message exchange for all devices on the GPIB having the specified select code. It outputs the DCL (Device Clear) bus command to the GPIB.

■ Up to addresses specified

The DCL@ statement initializes message exchange for the specified device. It releases the listener in the GPIB having the specified select code, sets only the specified device to the listener, and outputs the SDC (Selected Device Clear) bus command to the GPIB.

■ Items to be initialized for message exchange

1) Input buffer and output queue .... Cleared.
2) Syntax analysis, execution control, and response generation parts ...... Reset.
3) Device commands including *RST .... Clears all commands that obstruct the execution of these commands.

4) Parameter program message	Abandons all commands and queries of which execution is delayed for parameters.
5) *OPC command processing	Places the device into the operation complete command idle state (OCIS). As a result, the operation end bit cannot be set in the standard event status register. (☞ See page 7-7)
6) *OPC? query processing	Places the device into the operation complete query idle state (OQIS). As a result, operation end bit 1 cannot be set in the output queue. The MAV bit is cleared. (☞ See page 7-7)
7) Automatization of system configuration	Invalidates the *ADD and *DLF common commands that execute this function. (This unit does not support these commands.)
8) Device function	Places the part related to the message exchange into the idle state. The device keeps waiting for a message from the controller.

The items below must not be processed by clearing the device.

1) Changing data set and stored in the current device
2) Interrupting input-output devices on front panel
3) Clearing the MAV bit and changing other status byte to clear the output queue
4) Affecting and interrupting the device operation in progress

■ GPIB bus command sending order by DCL@ statement

The table below lists the DCL and SDC (GPIB bus commands) sending order in the DCL@ statement.

Statement	Bus command issue order (ATN line: Low level)	Data (ATN line : High level)
DCL @ select-code	UNL, DCL	————
DCL @ device-number	UNL, LISTEN address, [secondary-address], SDC	————

4.3 Device Initialization by *RST Command

■ Format

```
* RST
```

■ Example

WRITE@103: "*RST" Initializes only the device allocated to address 3 at level 3.

■ Description

The *RST (Reset) command, one of the IEEE488.2 common commands, resets the specified device at level 3.

Devices are generally placed into various states using commands (device messages) particular to the device. The *RST command is used to place the device into a specific known state again. Like level 2, this command invalidates the end of device operation.

■ Device address specification in WRITE@ statement

The *RST command initializes the device allocated to the specified address at level 3.

■ Items to be device-initialized

1) Device function and state	Returns the device to a specific known state regardless of the paste history. (☞ Listed on the next page.)
2) *OPC command processing	Places the device into the operation complete command idle state (OCIS). As a result, the operation end bit cannot be set in the standard event status register. (☞ See page 7-7)
3) *OPC? query processing	Places the device into the operation complete query idle state (OQIS). As a result, operation end bit 1 cannot be set in the output queue. The MAV bit is cleared. (☞ See page 7-7)
4) Macro command	Inhibits the macro operation and sets a macro command unreceivable mode. The macro definition returns to a state specified by the designer.

Note :

The *RST command does not affect the items below.

(1) IEEE 488.1 interface state

(2) Device address

(3) Output queue

(4) Service request enable register

(5) Standard event status enable register

(6) Power-on-status-clear flag setting

(7) Calibration data affecting the device standard

(8) RS-232C interface conditions

Table 4-1 lists the MG9637A/MG9638A initialization items. Each initialization condition indicates a state of device initialization by the *RST command. In the backup status column, mark o indicates items backed up when the power is turned off.

Table 4-1 MG9637A/MG9638A initialization item list

Measurement mode	Item	Set value	Backup status
CW mode	Wavelength	1550.000 nm	○
	Frequency	193414.4 GHz	○
	Power	–10 dBm	○
Sweep mode	Start Wavelength	1530.000 nm	○
	Stop Wavelength	1570.000 nm	○
	Center Wavelength	1550.000 nm	○
	Span Wavelength	40 nm	○
	Start Frequency	195942.7 GHz	○
	Stop Frequency	190950.6 GHz	○
	Center Frequency	193414.4 GHz	○
	Span Frequency	4992.1 GHz	○
	Sweep Step Wavelength	0.100 nm	○
	Sweep Step Frequency	12.8 GHz	○
	Dwell Time	1.00 s	○
	Power	–10 dBm	○
1 Step Tuning	Sweep Speed	1/4	○
Common to each measurement mode	f ←→ λ	λ	○
	Modulation	Off	○
	Modulation Int Freq.	20.0 kHz	○
	Coherency	Off	○
	Reverse On/Off	Off	○
	Calibration Wavelength	1550 nm	○
	Calibration Frequency	193414.4 GHz	○

The Frequency offset value is NOT initialized and backed-up.

Error message list

Error No.	Error message	Status	Output conditions
Key operation errors			
1001	Span or Step Limit		Span is narrower than sweep step.
1002	Power Limit		A higher output level was specified for this unit in wavelength mode.
1003	Power Limit		A higher output level was specified for this unit in frequency mode.
1004	Out of Limit		The entered set value is outside the specified range.
1005	Can't Find Recall Data		Recall data is not found.
Remote control errors			
2001	Invalid Command	ESE-CME	Remote command error
2002	Invalid Parameter of Command	ESE-EXE	Remote command parameter error
2003	Can't Execute Command	ESE-DDE	No request command is accepted in the current mode.
2004	Can't Execute Command	ESE-DDE	No set command is accepted in the current mode.
2005	Out of Cal Temperature	ESE-DDE	This processing cannot be executed currently because the LD module does not reach heat-up 100 %.
System errors			
4001			A motion control origin detection error occurred.
4002			A motion control operation error occurred.
4003			A backup memory checksum error occurred.
4004			An ND filter origin detection error occurred.
4005			An LD (APC) self-check error occurred.

4.4 Device State at Power-On

When the power is turned on :

1) the device returns to the state set when the power was turned off last ;

2) the input buffer and output queue are cleared ;

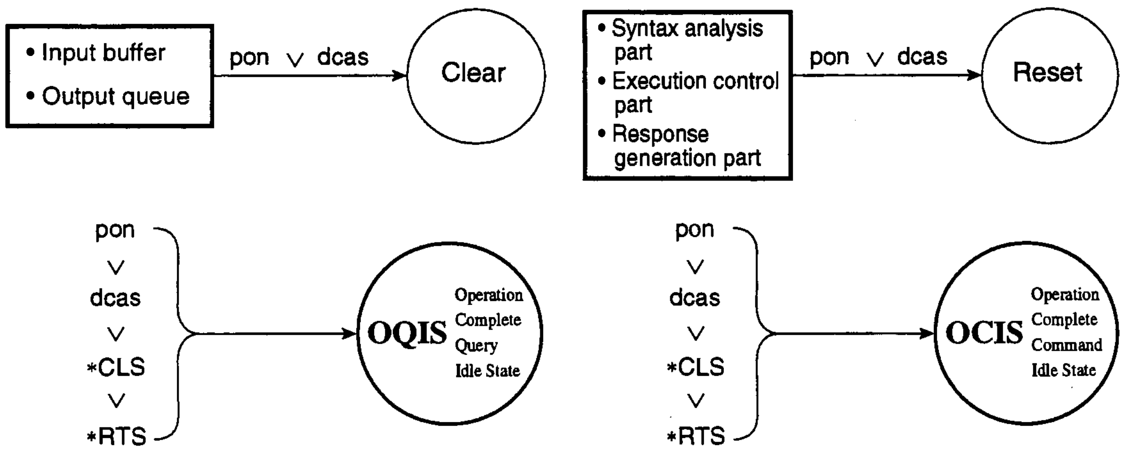
3) the syntax analysis, execution control, and response generation parts are reset ;

4) the device is placed into the operation complete command idle state (OCIS) ;

5) the device is placed into the operation complete query idle state (OQIS) ; and

6) the standard event status register and standard event status enable register are cleared because this unit does not support the *PSC command.

Items 2) to5) are also executed in cases other than power-on. Their status diagram is shown below.



■ Unchanged items at power-on

1) Address

2) Related calibration data

3) Data and state that change by a response returned to the common query commands below

* IDN? (See page 7-6)
* OPT? (See page 7-9)
* PSC? (not supported in this unit)
* PUD? (not supported in this unit)
* RDT? (not supported in this unit)

■ Power on status clear (PSC) flag

When the PSC flag is false, the service request enable register (See page 8-9), standard event status enable register (See page 8-11), and parallel pole enable register are not affected.
If the PSC flag is true or the *PSC command is not executed, these registers are cleared.

(The PSC command is not supported in this unit.)

■ Changed items at power-on

1) Current device function state

2) Status information

3) *SAV/*RCL register

4) Macro definition specified with the *DDT command (not supported in this unit)

5) Macro definition specified with the *DMC command (not supported in this unit)

6) Macro executable with *EMC command (not supported in this unit)

7) Address received with *PCB command (not supported in this unit)

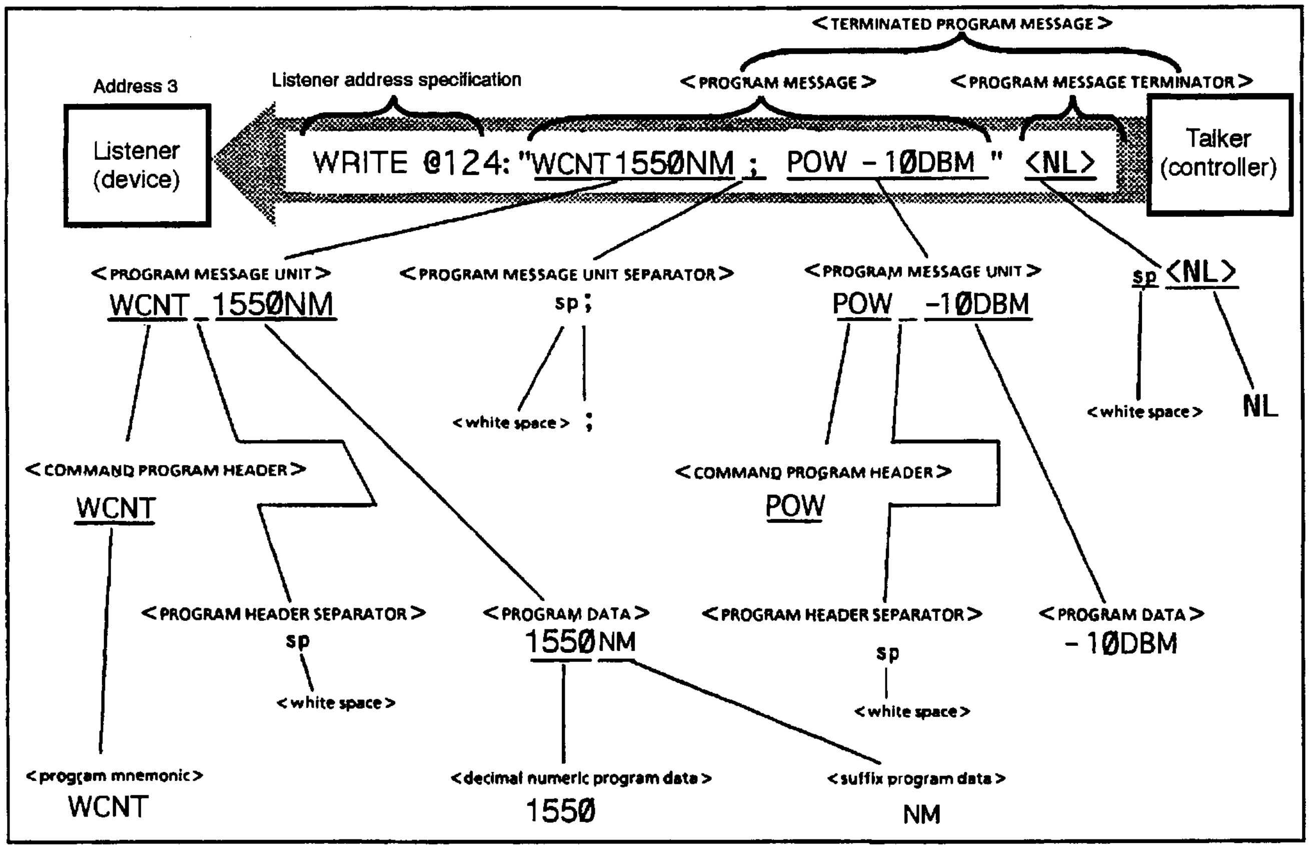
Section 5 Listener Input Format

Device messages, that are data messages transferred between the controller and device, are classified into two types: program and response messages. This chapter explains the formats of program messages a listener receives.

Section 5 Listener Input Format

A program message consists of a sequence of program message units. Each unit is composed of a program instruction or program query.

The figure below shows that two program message units, WCNT 1550NM and POW-10DBM, are connected by a program message unit separator and sent from the controller to a device as a program message to set the center wavelength to 1550 nm and output level to −10 dBm.



The program message format consists of a sequence of function elements each of which is divided into the minimum levels that can indicate each function. In this figure, a function element sample is indicated by uppercase characters enclosed with brackets < >. Each function element is subdivided into encoding elements. In this figure, an encoding element sample is indicated by lowercase characters enclosed with brackets < >.

A diagram indicating how to select a function element in a specific route is called a function syntax diagram. A diagram indicating how to select an encoding element in a specific route is also called an encoding syntax diagram. The next and subsequent pages explain the program message formats using the function syntax and encoding syntax diagrams.

An encoding element indicates an actual bus code required to send function element data bytes to a device. Upon receipt of those function element data bytes, the listener checks whether each function element conforms to the encoding syntax. If it is illegal, the listener reports a command error to the talker without assuming it to be a function element.

5.1 Notation of Listener Input Program Message

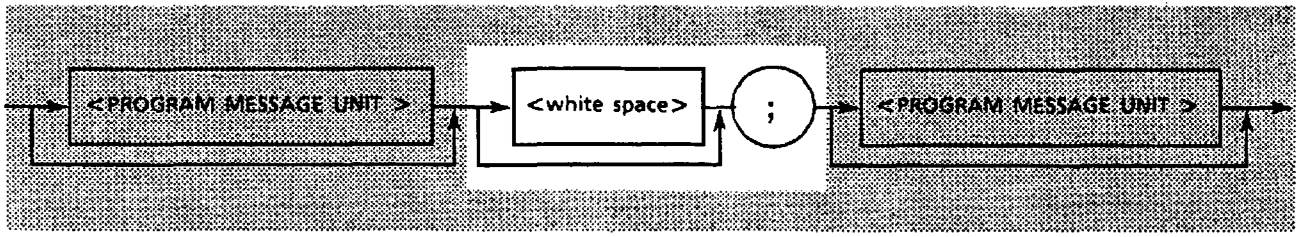
This section outlines the formats of program message function elements (☞ See page 5-8) and program data (☞ See page 5-19). (Compound commands and common commands are omitted here.)

5.1.1 Separator, terminator and header prefix space

(1) PROGRAM MESSAGE UNIT SEPARATOR

Connect two or more program message units by 0 or more spaces + semicolon (;).

<Example 1> General format for connecting two program message units



<Example 2> One space + semicolon

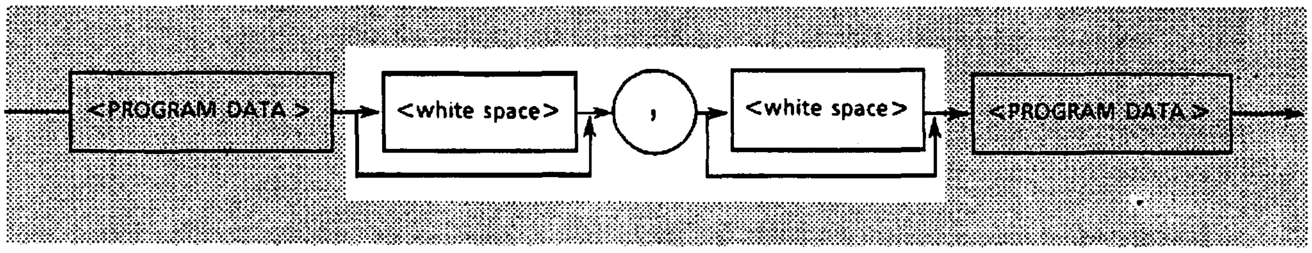
WCNT Δ 1550NM Δ; POWΔ -15DBM

Set the center wavelength to 1550 nm and the output level to –15 dBm.

(2) PROGRAM DATA SEPARATOR

Delimit two or more program data items by 0 or more spaces + comma + 0 or more spaces. This unit supports no corresponding commands.

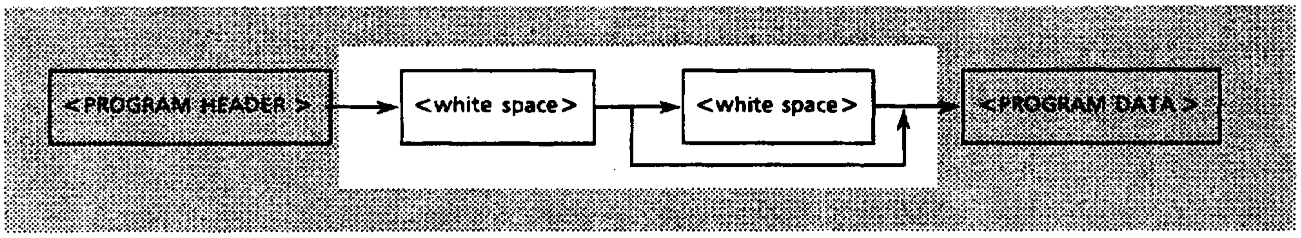
<Example 1> General format for delimiting two program data items



(3) PROGRAM HEADER SEPARATOR

Insert one space + 0 or more spaces between the program header and program data.

<Example 1> General format of single command program header



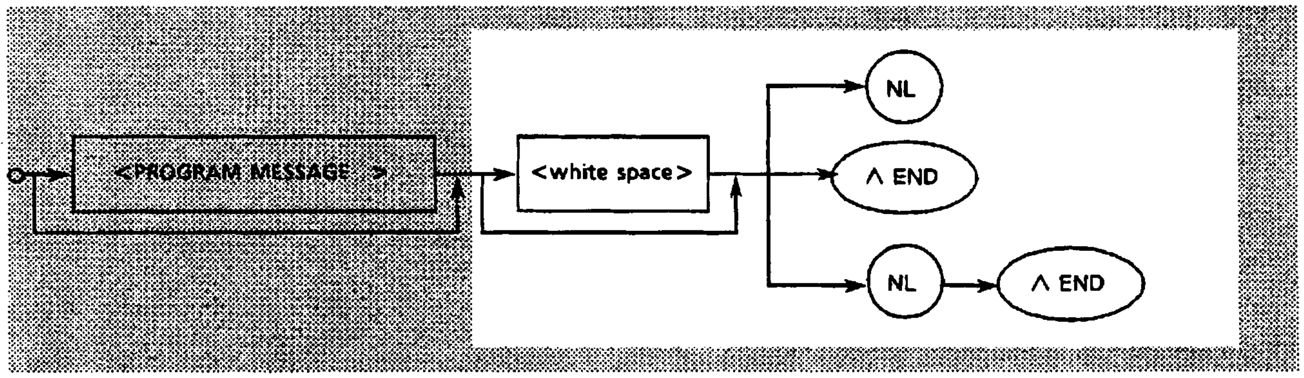
<Example 2> One space

POW Δ–10 DBM

(4) PROGRAM MESSAGE TERMINATOR

Add 0 or more spaces + $\left\{\begin{array}{l}\text{NL}\\\text{EOI}\\\text{NL + EOI}\end{array}\right\}$ at the end of a program message.

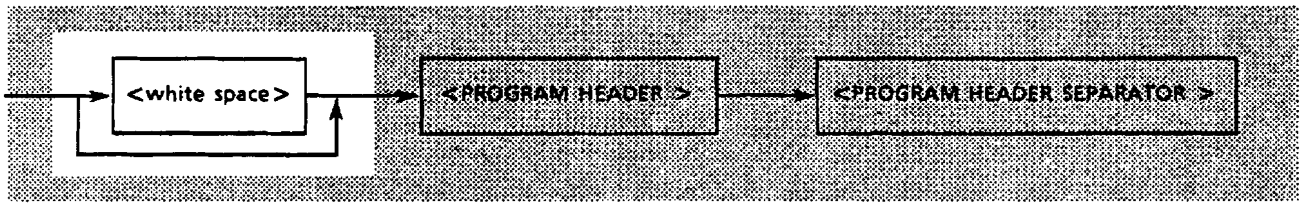
<General format>



(5) Header prefix space

0 or more spaces can be inserted before a program header.

<General format>

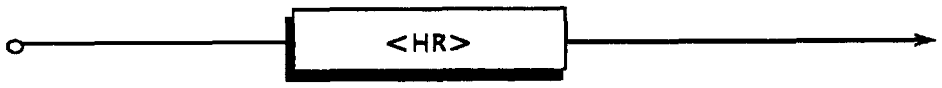


<Example> One space before the second program header, POW

WCNT Δ 1550NM ; Δ POW Δ –15DBM

5.1.2 General format of program command message

(1) Message without data specified

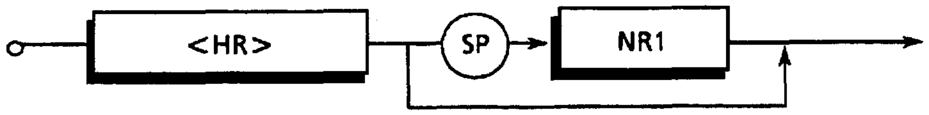


HR : COMMAND PROGRAM HEADER

<Example>

SNGL Starts a single sweeping.

(2) Message with integer data

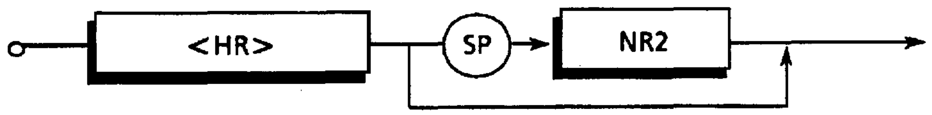


NR1: Integer

<Example>

OUTP Δ 1 Turns the laser signal output switch on.

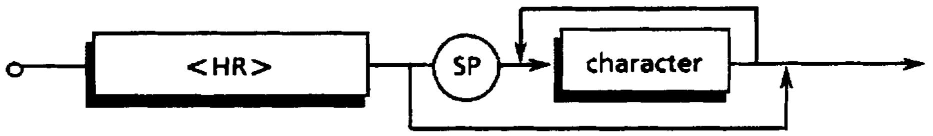
(3) Message with real number



NR2: Real number

This unit supports no corresponding commands.

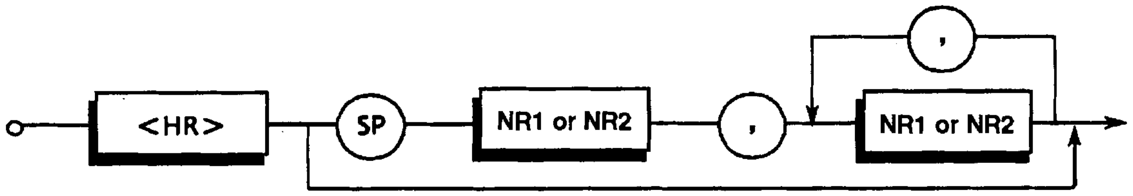
(4) Message with fixed or arbitrary character string data (data length <_ 12 characters)



<Example>

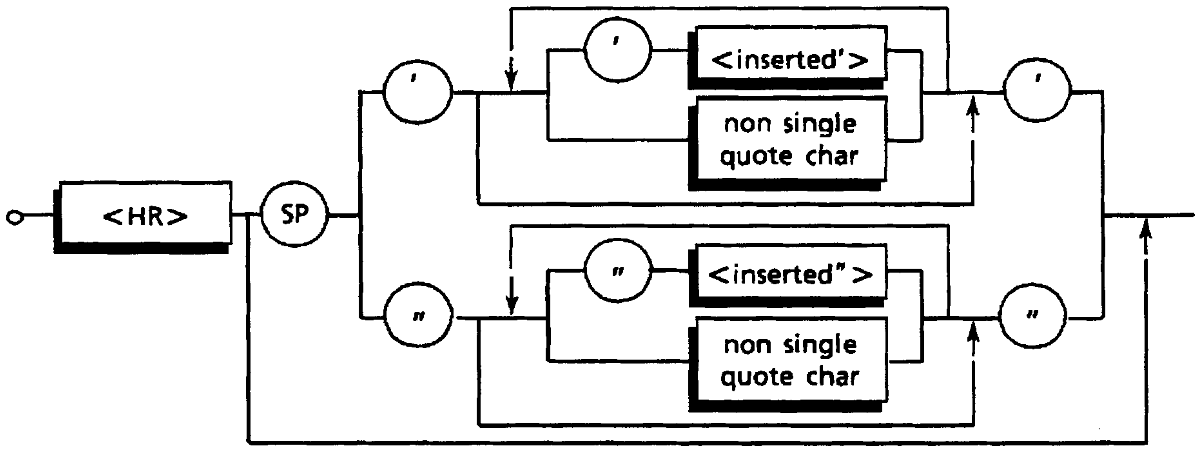
POWU Δ DBM   Sets the power unit to dBm.

(5) Message with program data items (Head NR1)



This unit supports no corresponding commands.

(6) Character string only message available for ASCII seven bits



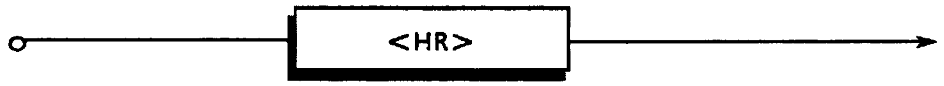
<inserted'>: A single ASCII code representing a value 27
non-single quote char: A single ASCII code representing a value other than 27
<inserted">: A single ASCII code representing a value 22
non-single quote char: A single ASCII code representing a value other than 22

This unit supports no corresponding commands.

5.1.3 General format of query message

For query program header, suffix a question mark (?) to a command program header.

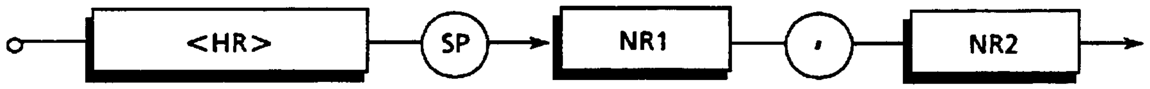
(1) Message without query data specified



<Example>

WCNT? Requests to send a center wavelength value.

(2) Message with query data specified



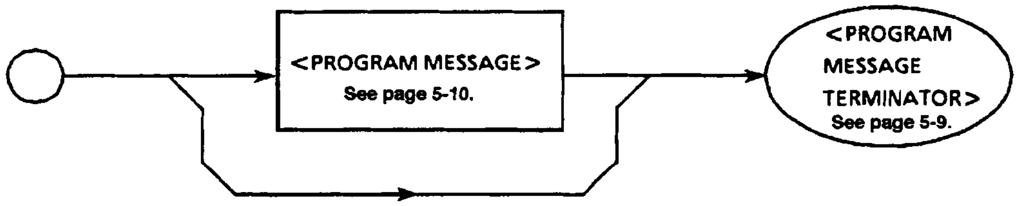
This unit supports no corresponding commands.

5.2 Program Message Function Elements

A device detects a terminator at the end of a program message to accept the program message. This section explains the function elements for each program message.

5.2.1 <TERMINATED PROGRAM MESSAGE>

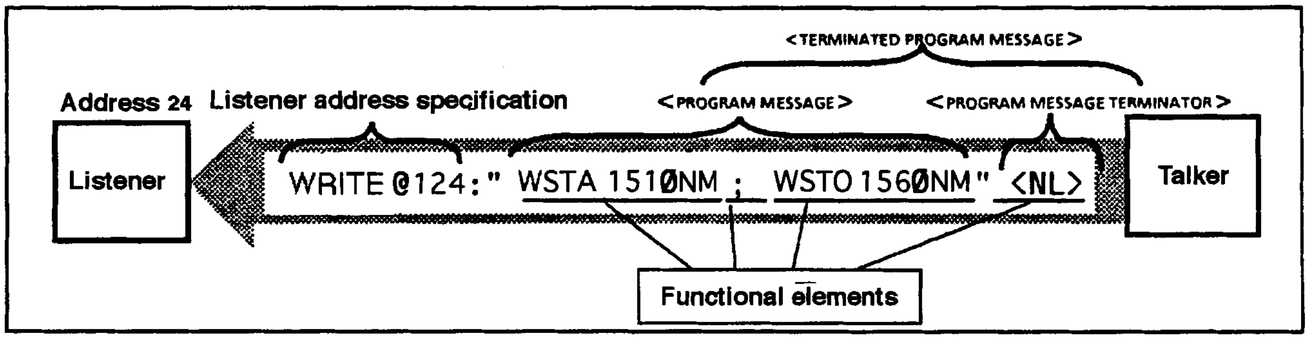
<TERMINATED PROGRAM MESSAGE> is defined as follows:



<TERMINATED PROGRAM MESSAGE> is a data message that satisfies all function elements required to send data from the controller to a listener device.

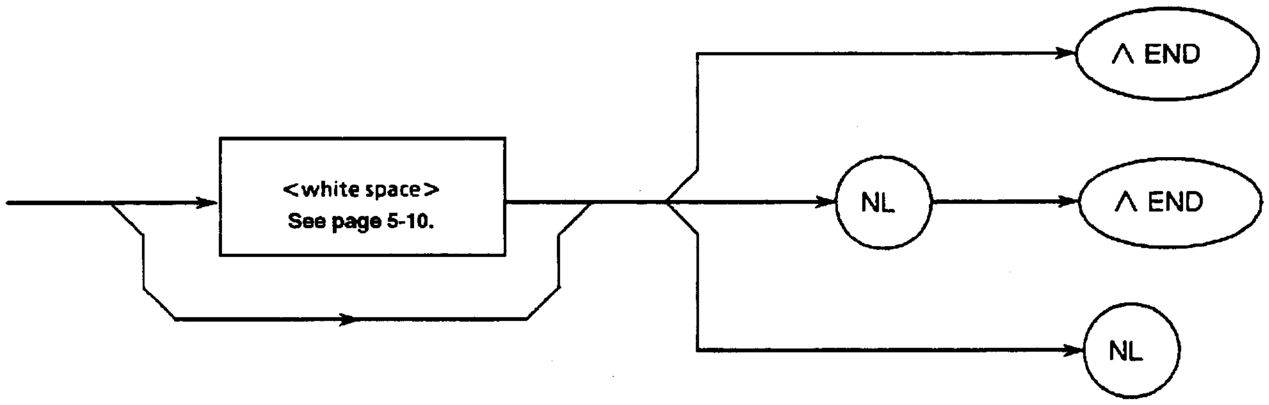
To complete the transfer of <PROGRAM MESSAGE>, <PROGRAM MESSAGE TERMINATOR> is added at the end of <PROGRAM MESSAGE>.

<Example> <TERMINATED PROGRAM MESSAGE> for using a WRITE statement to send two instructions


<TERMINATED PROGRAM MESSAGE>
Address 24
Listener address specification
<PROGRAM MESSAGE>
<PROGRAM MESSAGE TERMINATOR>
Listener
WRITE @124:" WSTA 1510NM ; WSTO 1560NM" <NL>
Talker
Functional elements

5.2.2 <PROGRAM MESSAGE TERMINATOR>

<PROGRAM MESSAGE TERMINATOR> is defined as follows :



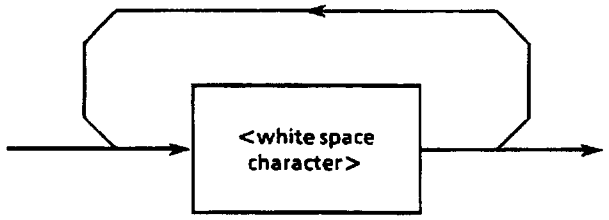
<PROGRAM MESSAGE TERMINATOR> is used to terminate a sequence composed of one or more <PROGRAM MESSAGE UNIT> elements in a fixed length.

NL : Defined as a single ASCII code byte 0A (decimal number 10). In other words, NL, which is equivalent to an ASCII control character LF (line feed), performs a line feed to return the printing position to the next line. By this function, the printing begins with a new line. The line feeding is also called an NL (new line).

END : Sets the EOI line (one of GPIB management buses) to TRUE (LOW level) to generate an EOI signal.

5.2.3 <white space>

<white space> is defined as follows:

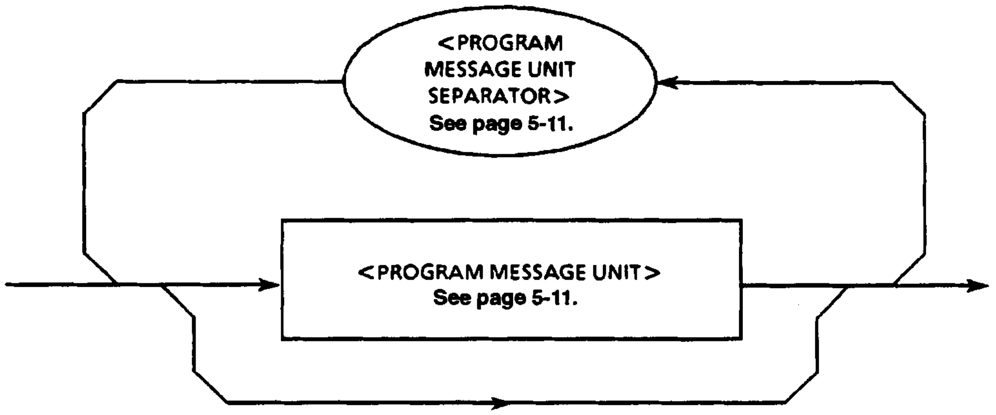


<white space character> is defined as a single ASCII code byte in the range of ASCII code bytes 00 to 09 and 0B to 20 (decimal numbers 0 to 9 and 11 to 32).

This range includes an ASCII control signal and space signal excluding the new line signal. However, the device processes <white space character> as an ordinary space or skips it without interpreting that <white space character> is an ASCII control signal.

5.2.4 <PROGRAM MESSAGE>

<PROGRAM MESSAGE> is defined as follows :

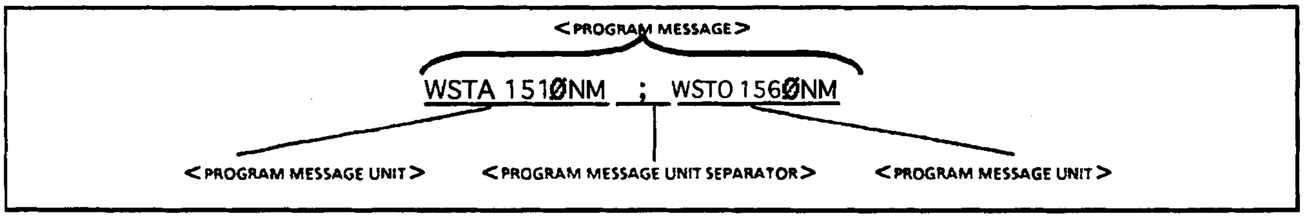


<PROGRAM MESSAGE> is a sequence composed of 0 or one <PROGRAM MESSAGE UNIT> or multiple <PROGRAM MESSAGE UNIT> elements. The <PROGRAM MESSAGE UNIT> element means a programming instruction or data sent from the controller to a device. The <PROGRAM MESSAGE UNIT SEPARATOR> element is used as a separator for delimiting multiple <PROGRAM MESSAGE UNIT> elements.

<Example 1> Program message for setting start wavelength to 1510nm

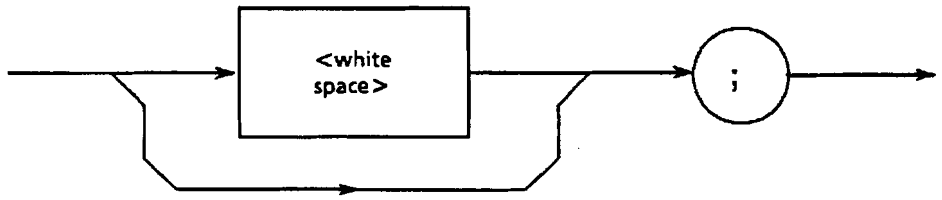
WSTA Δ 1510NM

<Example 2> Program message for also setting stop wavelength to 1560nm

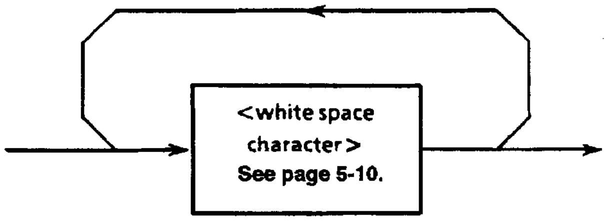


5.2.5 <PROGRAM MESSAGE UNIT SEPARATOR>

<PROGRAM MESSAGE UNIT SEPARATOR> is defined as follows :



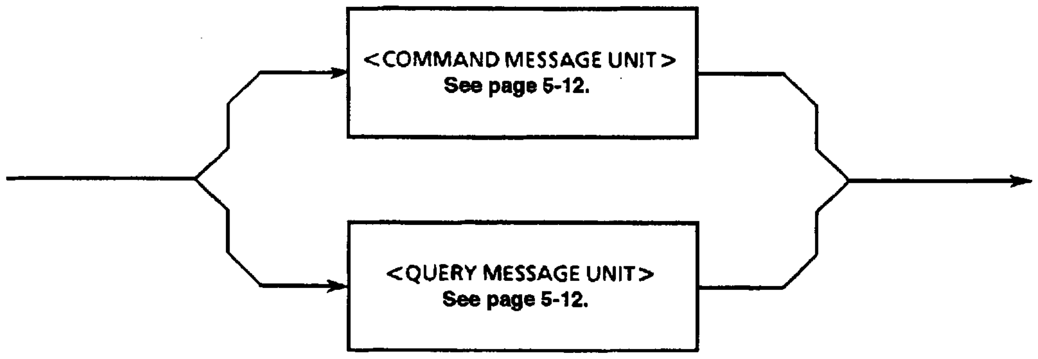
<white space> is defined as follows :



<PROGRAM MESSAGE UNIT SEPARATOR> is used to divide a sequence of <PROGRAM MESSAGE UNIT> elements in the range of <PROGRAM MESSAGE>.
Since a semicolon (;) is assumed to be a separator between <PROGRAM MESSAGE UNIT> elements, the device skips <white space character>'s before and after a semicolon (;). However, <white space character> is available to easily read a program. If <white space> follows a semicolon (;), it is assumed to be one positioned before the next program header.
(☞ <Example 2> in the preceding page or page 5-13.)

5.2.6 <PROGRAM MESSAGE UNIT>

<PROGRAM MESSAGE UNIT> is defined as follows:

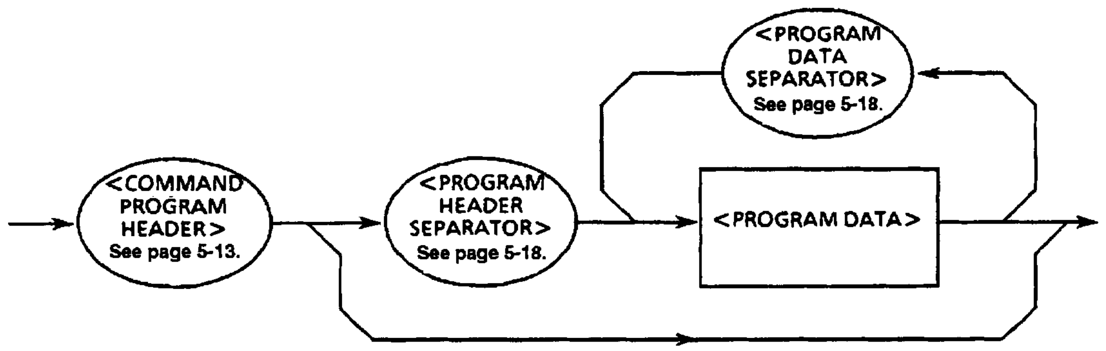


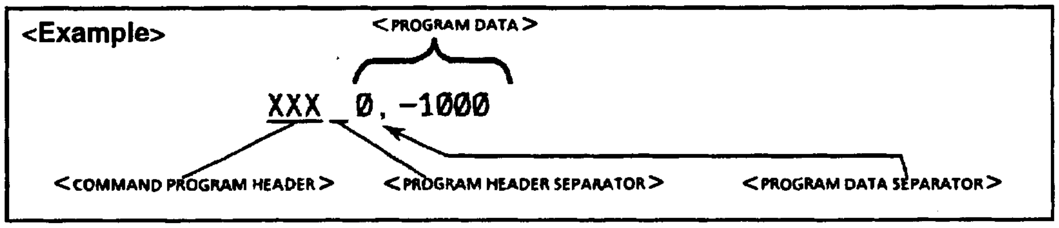
<PROGRAM MESSAGE UNIT>, that is a single command message received by the device, consists of a <COMMAND MESSAGE UNIT> or <QUERY MESSAGE UNIT> that is a single query message.

For details on <COMMAND MESSAGE UNIT> AND <QUERY MESSAGE UNIT>, see the next page.

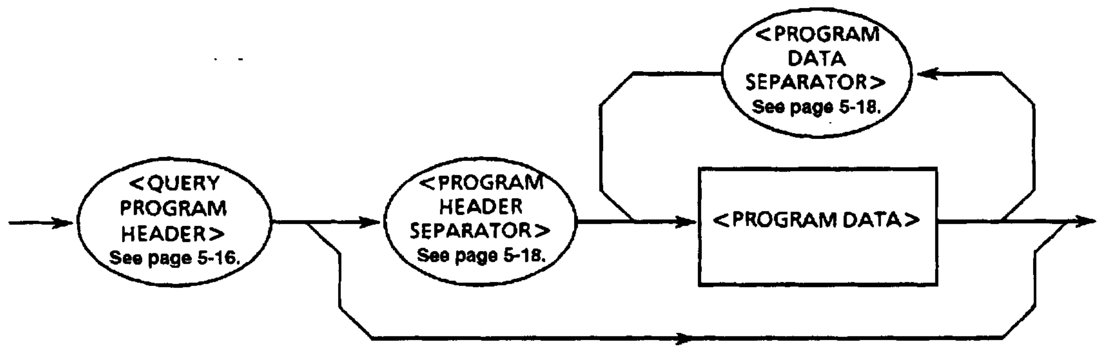
5.2.7 <COMMAND MESSAGE UNIT> and <QUERY MESSAGE UNIT>

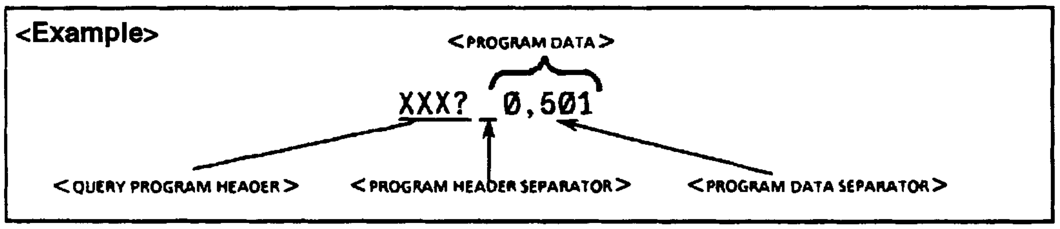
1) <COMMAND MESSAGE UNIT> is defined as follows:


<COMMAND PROGRAM HEADER> See page 5-13.
<PROGRAM HEADER SEPARATOR> See page 5-18.
<PROGRAM DATA SEPARATOR> See page 5-18.
<PROGRAM DATA>



2) <QUERY MESSAGE UNIT> is defined as follows:


<QUERY PROGRAM HEADER> See page 5-16.
<PROGRAM HEADER SEPARATOR> See page 5-18.
<PROGRAM DATA SEPARATOR> See page 5-18.
<PROGRAM DATA>

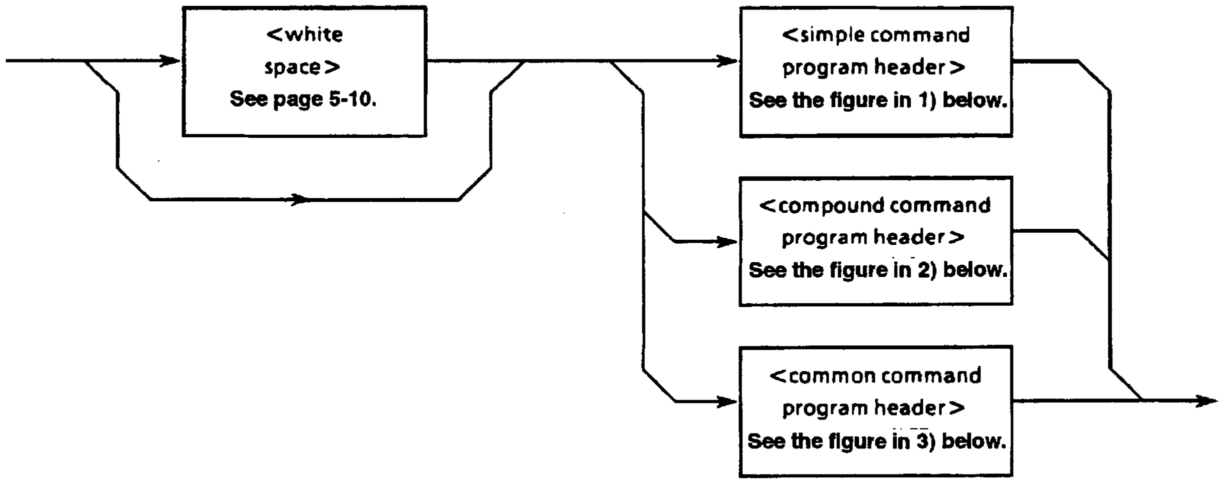


When a program header is followed by program data in <COMMAND MESSAGE UNIT> and <QUERY MESSAGE UNIT>, one space is necessarily inserted between them as a separator. The program header is available to check the use, function, and operation of the program data. If no program data follows a program header, only the program header indicates the use, function, and operation of program data executed by the device.

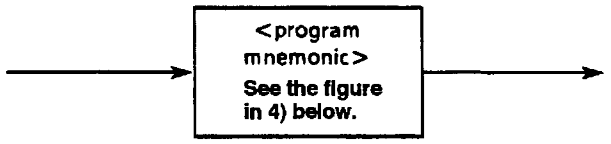
Of the program headers, <COMMAND PROGRAM HEADER> is a command that controls a device from the controller; <QUERY PROGRAM HEADER> is a query command the controller sends to a device in advance to receive a response message from the device. Its header is necessarily suffixed by a query indicator ?.

5.2.8 <COMMAND PROGRAM HEADER>

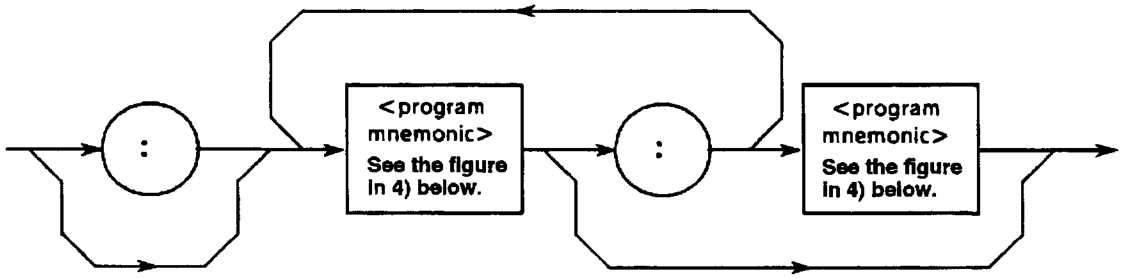
<COMMAND PROGRAM HEADER> is defined as follows. Each header can be prefixed by <white space>.


<white space> See page 5-10.
<simple command program header> See the figure in 1) below.
<compound command program header> See the figure in 2) below.
<common command program header> See the figure in 3) below.

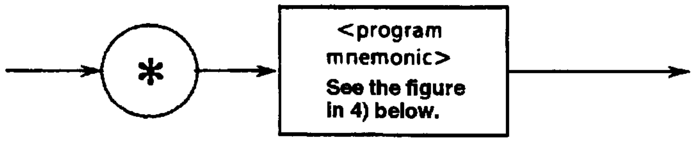
1) <simple command program header> is defined as follows :



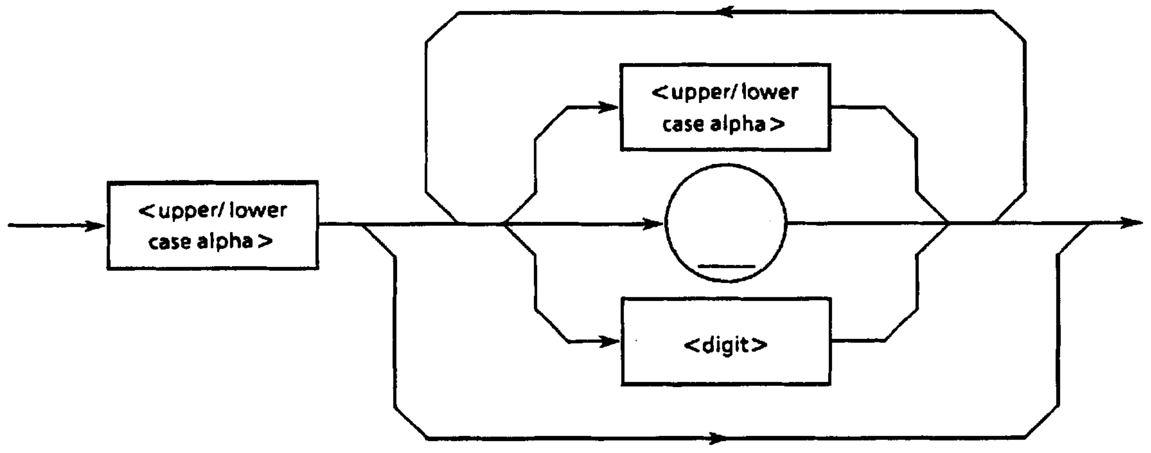
2) <compound command program header> is defined as follows :


:
<program mnemonic> See the figure in 4) below.
:
<program mnemonic> See the figure in 4) below.

3) <common command program header> is defined as follows :



4) <program mnemonic> is defined as follows:



■ <COMMAND PROGRAM HEADER>

<COMMAND PROGRAM HEADER> indicates the use, function, and operation of program data executed by the device. If program data is omitted, only the program header indicates the use, function, and operation of program data executed by the device.

Their meanings are indicated by <program mnemonic> expressed with ASCII code characters. <program mnemonic> is generally called mnemonic. The following explains the standard of mnemonic and items 1) to 3) above.

■ <program mnemonic>

Each mnemonic necessarily begins with an uppercase or lowercase alphabetic character. It is followed by an ordinary combination of uppercase characters A to Z, lowercase characters a to z, underline _, and digits 0 to 9. The maximum length of a mnemonic is 12 characters; generally, three to four alphabetic characters are used. In this case, no space is inserted between these characters.

● <upper/lower case alpha>	Defined as a single ASCII code byte in the range of ASCII code bytes 41 to 5A, 61 to 7A (decimal numbers 65 to 90, 97 to 122 = uppercase alphabetic characters A to Z, lowercase alphabetic characters a to z). The header, whichever is uppercased or lowercased, is accepted by the device.
● <digit>	Defined as a single ASCII code byte in the range of ASCII code bytes 30 to 39 (decimal numbers 48 to 57 = numeric values 0 to 9).
● (_)	Indicates ASCII code byte 5F (decimal number 95 = underline); defined as a single ASCII code byte.

■ <simple command program header>

The standard of <program mnemonic> above applies to <simple command program header>.

■ <compound command program header>

<compound command program header> is a command program header used to execute compound functions. A colon (:) is necessarily inserted as a separator of <compound command program header> before <program mnemonic>. When only one <compound command program header> is used, the colon (:) can be omitted.

■ <common command program header>

In this header, an asterisk (*) is necessarily inserted before <program mnemonic>. Since this command is a program command applied commonly to other IEEE 488.2 measuring instruments connected to the bus, "common" is assigned to this command.

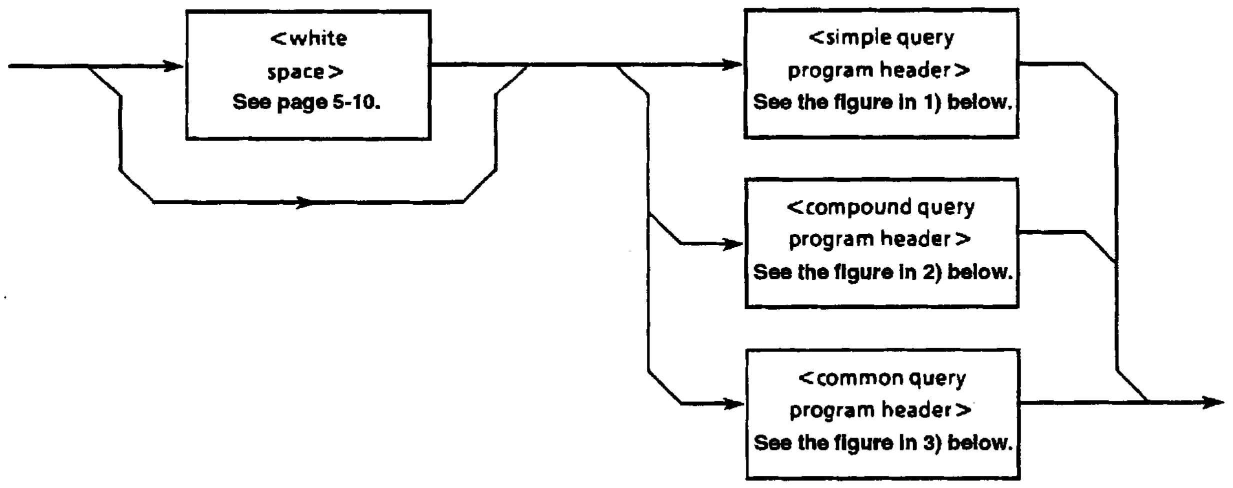
● <Example>

Place the operation termination of the device at address 24 connected to the GPIB interface fitting to select code 1 into idle state and initialize each device to a predetermined state.

WRITE@24:" *RST" ...... An IEEE 244.3 common command, *RST, that is enclosed by double quotation marks ("), executes the processing above.

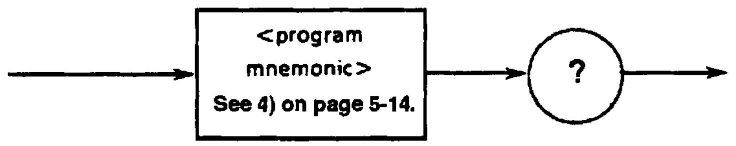
5.2.9 <QUERY PROGRAM HEADER>

<QUERY PROGRAM HEADER> is defined as follows. <white space> can be inserted before each header.

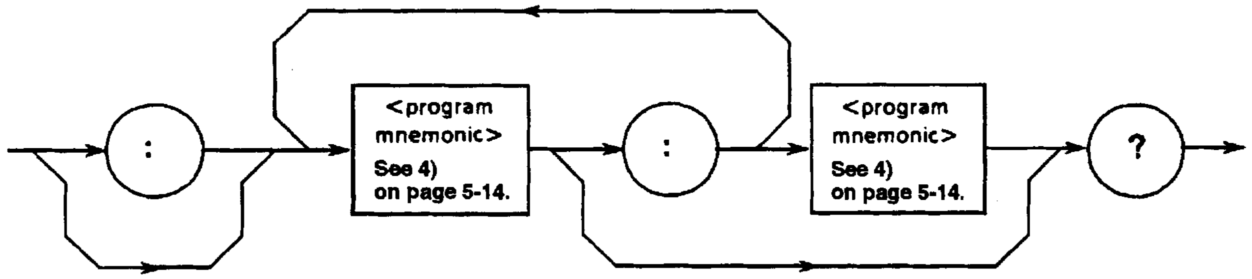


<white space> See page 5-10.
<simple query program header> See the figure in 1) below.
<compound query program header> See the figure in 2) below.
<common query program header> See the figure in 3) below.

1) <simple query program header> is defined as follows:

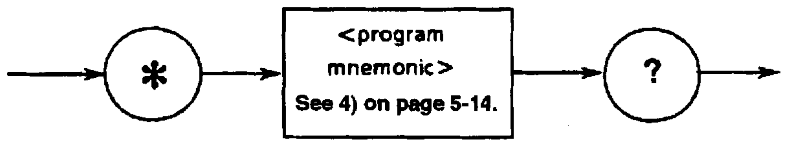


2) <compound query program header> is defined as follows:



:
<program mnemonic> See 4) on page 5-14.
:
<program mnemonic> See 4) on page 5-14.
?

3) <common query program header> is defined as follows:



■ <QUERY PROGRAM HEADER>

<QUERY PROGRAM HEADER> is a query command the controller sends to a device in advance to receive a response message from the device. Its header is necessarily suffixed by a query indicator ?. A program example is shown below.

The format of this <QUERY PROGRAM HEADER> is the same as <COMMAND PROGRAM HEADER> except that a query indicator ? is suffixed to the header. For details, see page 5-13.

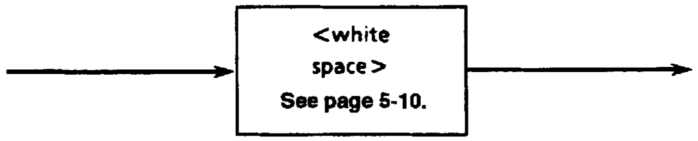
● <Example 1> Specify and read the center wavelength.

```
10 WRITE @108 : "WCNT 1550NM"
20 WRITE @108 : "WCNT?"! ................ Query message WCNT?
30 READ @108 : A
40 PRINT A ; "m"
```

Line 10	A program message that consists of command header WCNT for specifying the center wavelength and program data item 1. Specify 1550NM for the device.
Line 20	A program message that asks the device to send the specified wavelength, 1550 nm, to the controller. Query header WCNT? is used for this purpose.
Line 30	When receiving query header WCNT? from the controller, this unit that is a listener device functions as a talker. The device returns a response message 1.55000000E-006 as a response to WCNT? to the controller that changed to a listener this time. The listener reads the response message to numeric variable A.
Line 40	Displays the center wavelength on the CRT screen.

5.2.10 <PROGRAM HEADER SEPARATOR>

<PROGRAM HEADER SEPARATOR> is defined as follows:



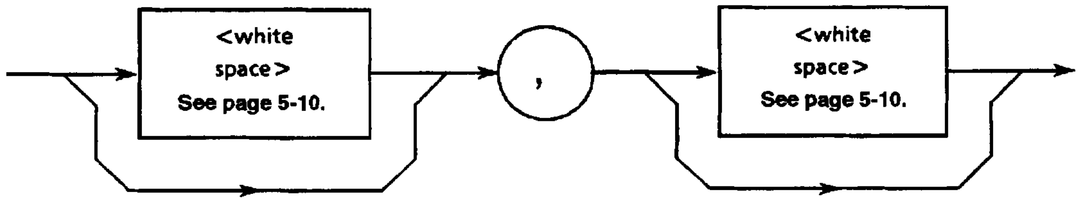
<PROGRAM HEADER SEPARATOR> is used as a separator between <COMMAND PROGRAM HEADER> or <QUERY PROGRAM HEADER> and <PROGRAM DATA>.

When two or more <white space character>'s are inserted between a program header and program data, the first one is assumed to be a separator and the rest is skipped. <white space character> is available to easily read a program.

In other words, only one header separator, that necessarily exists between a program header and data, indicates the end of the program header and the beginning of the program data.

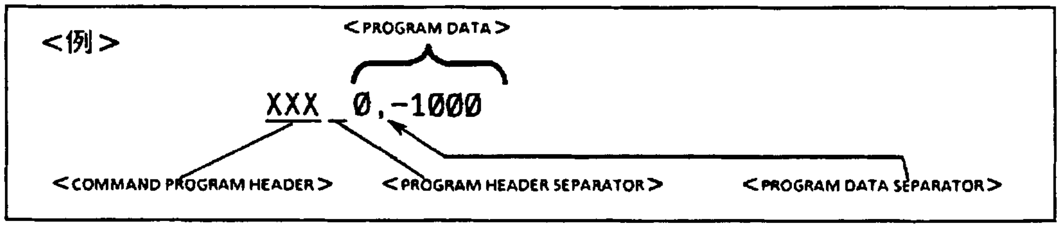
5.2.11 <PROGRAM DATA SEPARATOR>

<PROGRAM DATA SEPARATOR> is defined as follows:



<PROGRAM DATA SEPARATOR> is used to delimit multiple parameters of <COMMAND PROGRAM HEADER> or <QUERY PROGRAM HEADER>.

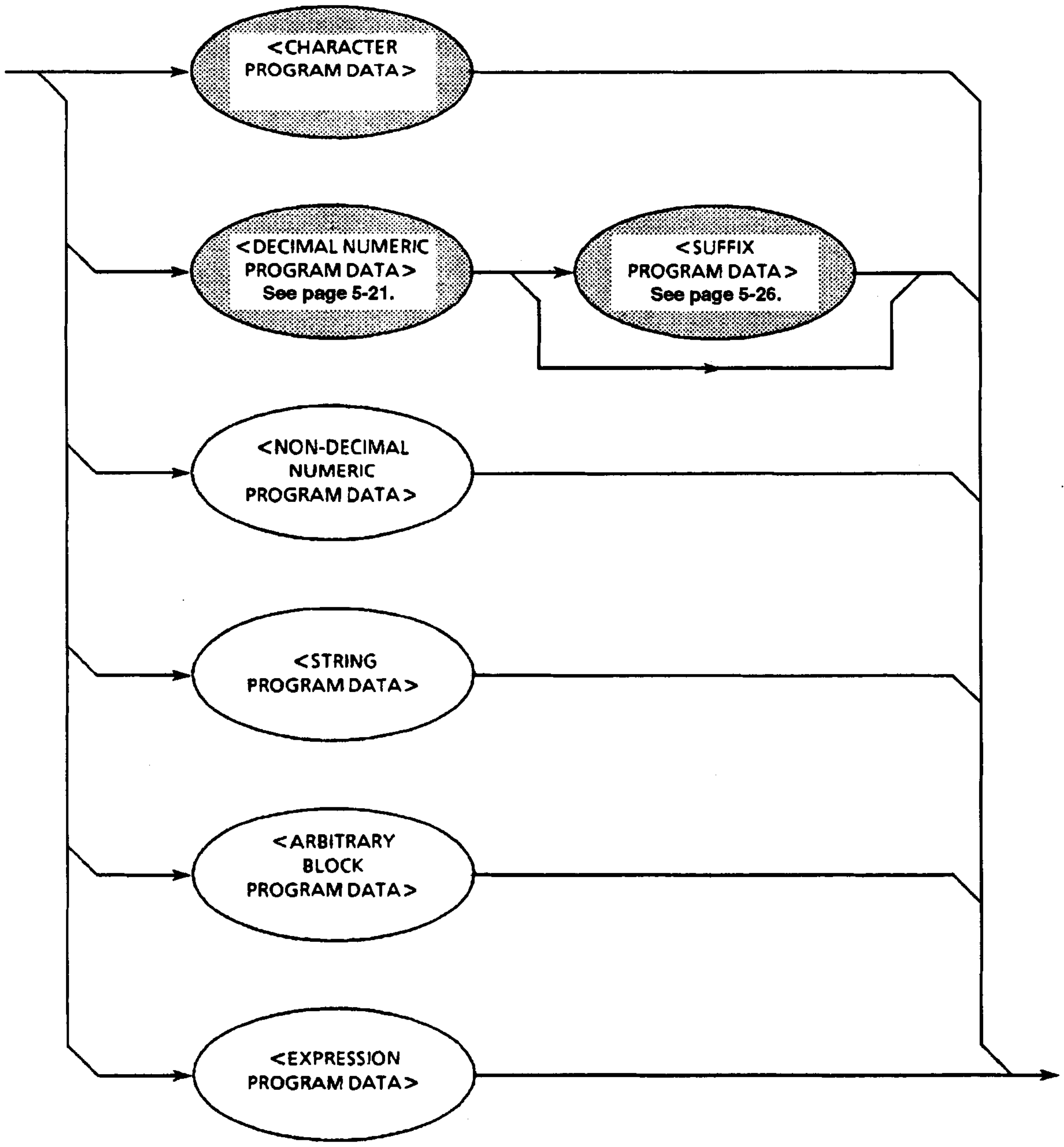
To use the data separator, a comma (:) is required, but <white space character> may be omitted. <white space character> before or after a command is skipped. <white space character> is available to easily read a program.



5.3 Program Data Format

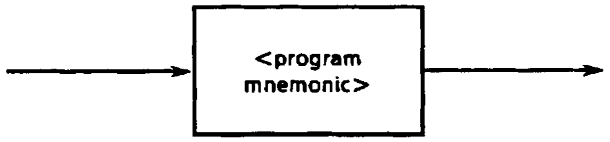
This section explains the format of <PROGRAM DATA> shown in the function syntax diagram (☞ see page 5-12) in the format system of the program message terminated above.

A <PROGRAM DATA> function element is used to transmit various types of parameters related to a program header. The figure below shows these types of program data. This unit accepts the program data enclosed by hatching. Program data not used in this unit is also shown as a reference.

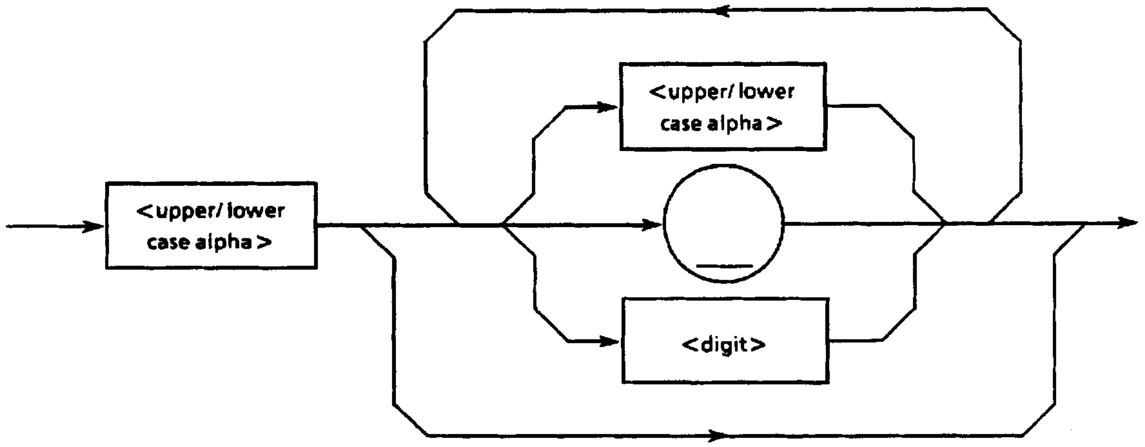

<CHARACTER PROGRAM DATA>
<DECIMAL NUMERIC PROGRAM DATA> See page 5-21.
<SUFFIX PROGRAM DATA> See page 5-26.
<NON-DECIMAL NUMERIC PROGRAM DATA>
<STRING PROGRAM DATA>
<ARBITRARY BLOCK PROGRAM DATA>
<EXPRESSION PROGRAM DATA>

5.3.1 <CHARACTER PROGRAM DATA>

Each <CHARACTER PROGRAM DATA> element is aimed at performing remote control operation by transmitting a short alphabetic text or alphanumeric data. It is defined as follows:



The contents of character data is the same as for program mnemonic. In the conventional system, major control data was numeric data, but this system enables a control using the character program data. A detail of the encoding syntax diagram is shown below.



Program data begins with an uppercase or lowercase alphabetic character, and it is followed by an ordinary combination of uppercase alphabetic characters A to Z, lowercase alphabetic characters a to z, underline, and digits 0 to 9. A combination of these alphanumeric characters is used as a mnemonic symbol. The maximum length of program data is 12 characters. No space exists between characters.

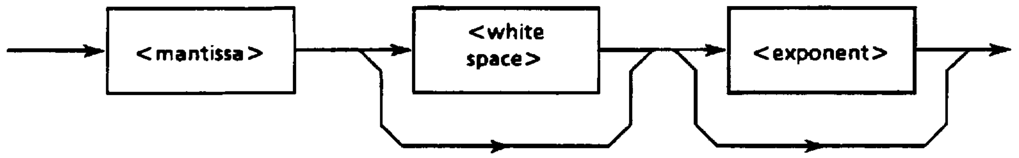
● <upper/lower case alpha>	Defined as a single ASCII code byte in the range of ASCII code bytes 41 to 5A, 61 to 7A (decimal numbers 65 to 90, 97 to 122 = uppercase alphabetic characters A to Z, lowercase alphabetic characters a to z). The header, whichever is uppercased or lowercased, is accepted by the device.
● <digit>	Defined as a single ASCII code byte in the range of ASCII code bytes 30 to 39 (decimal numbers 48 to 57 = numeric values 0 to 9).
● <_>	Indicates ASCII code byte 5F (decimal number 95 = underline); defined as a single ASCII code byte.

<CHARACTER PROGRAM DATA> is program data used to send a shorter mnemonic type of alphabetic character.

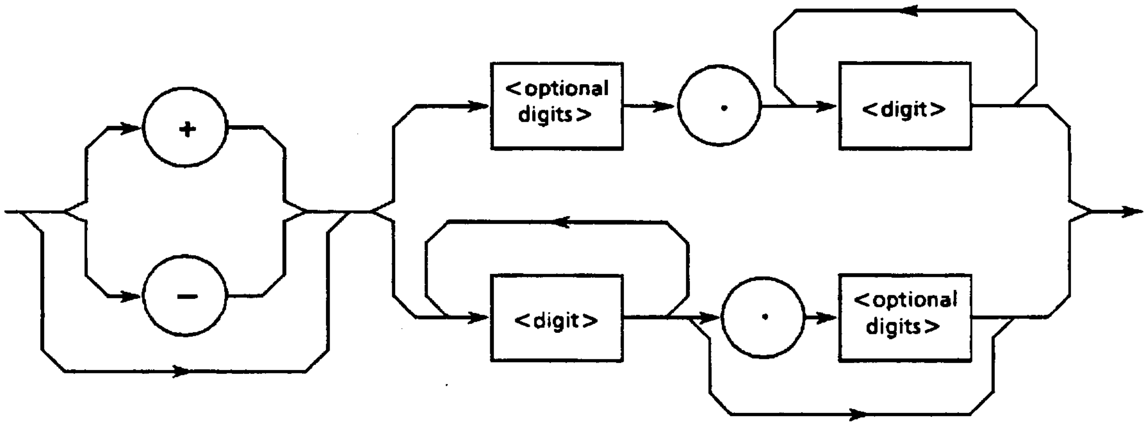
5.3.2 <DECIMAL NUMERIC PROGRAM DATA>

<DECIMAL NUMERIC PROGRAM DATA> is program data used to transmit a numeric constant indicated in decimal notation. There are three types of decimal numeric representations: integral, fixed-point, and floating-point formats.

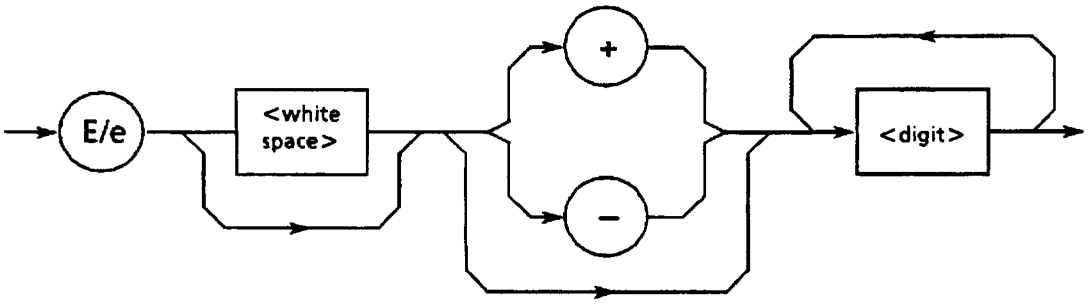
These three types of numeric values are defined as shown in the encoding syntax diagram below in order to change decimal numeric program data to a flexible numeric representation (NRf).



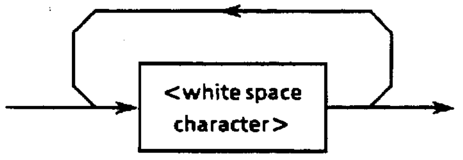
<mantissa> is defined as follows:

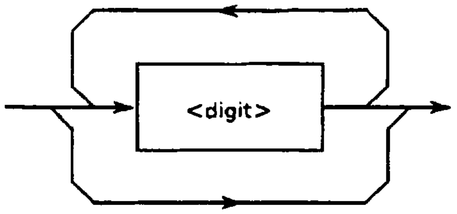


<exponent> is defined as follows:



<white space> and <optional digits> are defined as follows:





☞ <white space> see page 5-11, <digit> see page 5-20.

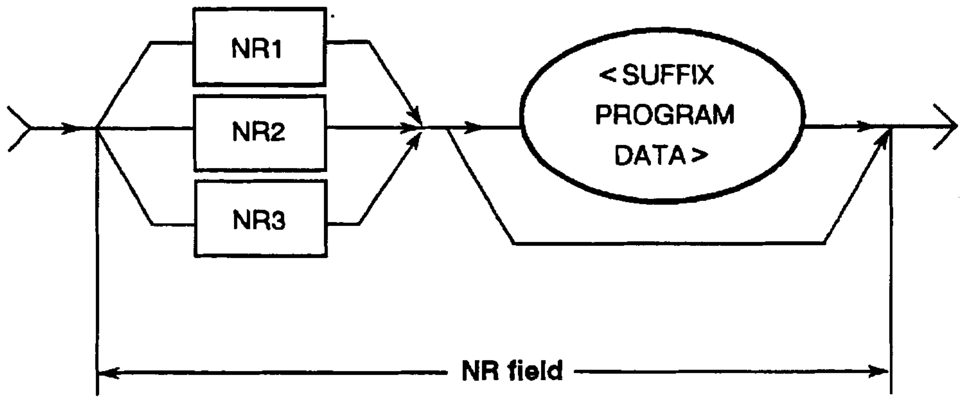
The following paragraphs explain how to transmit decimal numeric program data by dividing the encoding syntax diagram of the decimal numeric program data above into the integral, fixed-point, and floating-point formats.

The processing below is performed for any format.

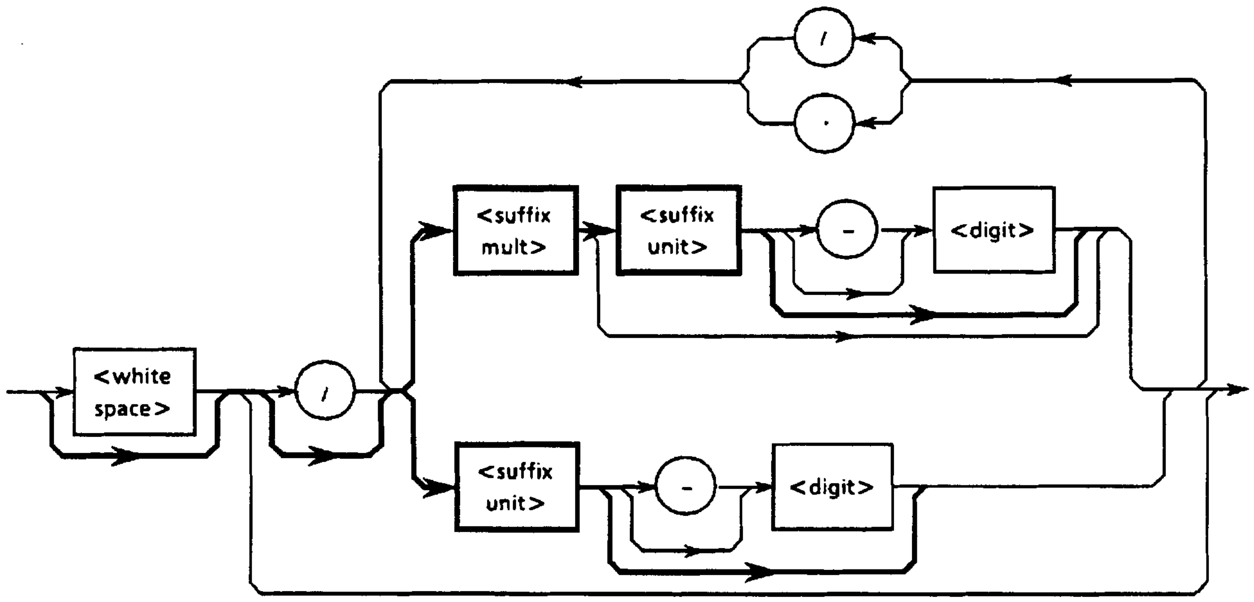
● Data outside range	When the value of a <DECIMAL NUMERIC PROGRAM DATA> element is outside the allowed range on relation with a program header, an execution error is reported.

5.3.3 <SUFFIX PROGRAM DATA>

<SUFFIX PROGRAM DATA> is defined following the <DECIMAL NUMERIC PROGRAM DATA> above (integral format NR1, fixed-point format NR2, or floating-point format NR3). A suffix can be specified at the end of each format.



The suffix is added to the end of decimal numeric program data only when a measurement unit is required for the data. A suffix unit or suffix multipliers and suffix unit are used as a suffix. The syntax diagram is shown below. The routes indicated by bold lines are used generally.



- The suffix multipliers are expressed with uppercase or lowercase characters. For example, 1E3KHz is expressed with 1kHz, assuming 1E3 = k.
- The suffix unit is expressed with uppercase or lowercase characters.
- <SUFFIX PROGRAM DATA> must not be prefixed by character E because it may be confused with character E used in the floating-point format.

The table below lists the suffix multipliers and suffix units.

(1) Suffix multipliers

Table 5-1 Suffix multipliers

Multiplier	Mnemonic	Name
1E18	EX	EXA
1E15	PE	PETA
1E12	T	TERA
1E9	G	GIGA
1E6	MA(NOTE)	MEGA
1E3	K	KILO
1E-3	M(NOTE)	MILLI
1E-6	U	MICRO
1E-9	N	NANO
1E-12	P	PICO
1E-15	F	FEMTO
1E-18	A	ATTO

Note :

Conventionally, 10s̄ of Hz is assumed to be MHz (megahertz) and one of OHM to be MOHM (megohm). These are listed in Table 5-2, "Suffix Unit Table", not in this table.

(2) Relative unit (dB)

• 1 μV decibel	DBUV
• 1 μW decibel	DBUW
• 1 μW decibel	DBMW

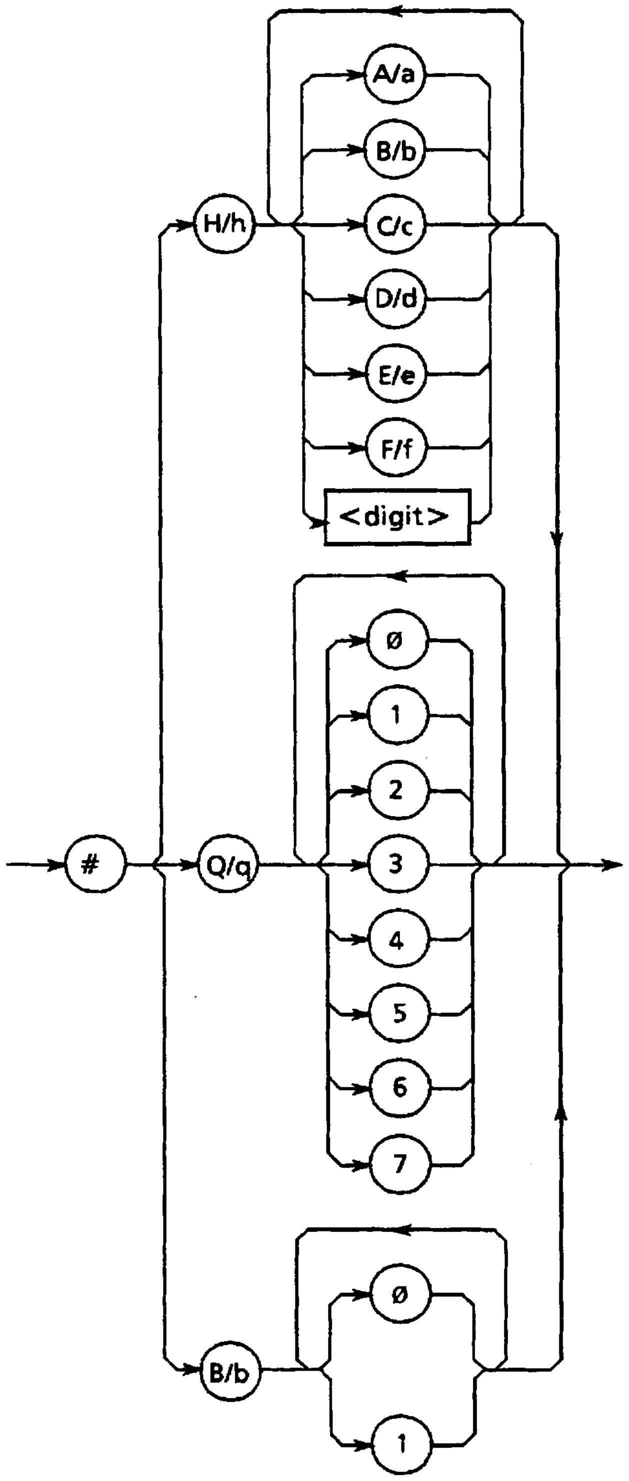
(3) Suffix unit

Table 5-2 Suffix units

Item	Recommended mnemonic of unit	Quasi recommended mnemonic of unit	Name
Current	A		Ampere
Atmospheric pressure	ATM		Atmosphere
Charge	C		Coulomb
Luminance	CD		Candela
Decibel	DB		Decibel
Power	DBM		Decibel milliwatt
Capacitance	F		Farad
Mass		G	Gram
Inductance	H		Henry
Frequency (hertz)	HZ		Hertz
Mercury column	INHG		Inches of mercury
Joule	J		Joule
Temperature	K		Degree Kelvin
		CEL	Degree Celsius
		FAR	Degree Fahrenheit
Volume	L		Liter
Luminance	LM		Lumen
Luminance	LX		Lux
Length (meter)	M		Meter
		FT	Feet
		IN	Inch
Frequency (1E3Hz)		MHZ	Megahertz
Resistance		MOHM	Megaohm
Force	N		Newton
Resistance	OHM		Ohm
Pressure	PAL		Pascal
Ratio (percent)	PCT		Percent
Angle (radian)	RAD		Radian
Angle (degree)		DEG	Degree
		MNT	Minute(of arc)
Time (second)	S	SEC	Second
Conductance	SIE		Siemens
Automatic speed	T		Tesla
Pressure	TORR		Torr
Voltage	V		Volt
Power (watt)	W		Watt
Speed/hour	WB		Weber
Luminance	LM		Lumen

5.3.4 <NON-DECIMAL NUMERIC PROGRAM DATA>

<NON-DECIMAL NUMERIC PROGRAM DATA> is program data used to transmit hexadecimal, octal, and binary numeric data as
non-decimal numeric data. The non-decimal numeric data begins with the mark #. <NON-DECIMAL NUMERIC PROGRAM DATA> is defined as shown in the encoding syntax diagram on the left of the figure below. If an invalid character string is sent, a command error is reported.



The character string following #H or #h is accepted by the device as a hexadecimal number.
The character strings in parentheses are decimal numbers.

#Habc123Ø	(11,256,099D)
#hAbC123	
#H2DC3	(11,715D)
#h2dc3	
#H83Ø1	(33,537D)
#h83Ø1	

The character string following #Q or #q is accepted by the device as an octal number.

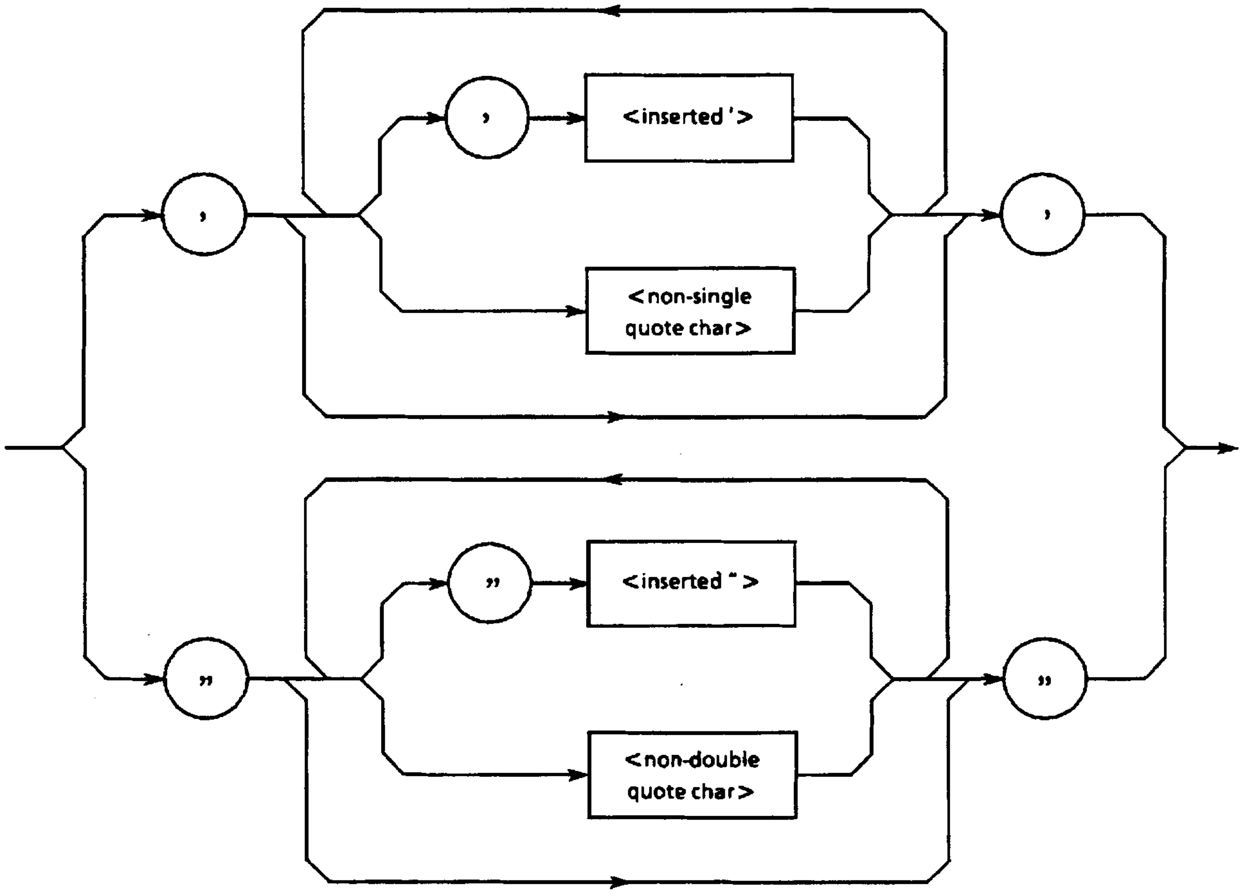
#Q37	(31D)
#q37	
#Q267Ø3	(11,715D)
#q267Ø3	

The character string following #B or #b is accepted by the device as a binary number.

#B1Ø1Ø1Ø1111ØØØØØ1ØØ1ØØØ11	(11,256,099D)
#bØØ1Ø11Ø111ØØØØ11	(11,715D)

5.3.5 <STRING PROGRAM DATA>

<STRING PROGRAM DATA> is program data composed of character strings only. All ASCII 7-bit codes are available for this data. A single or double quotation mark included in a character string must be written twice successively.



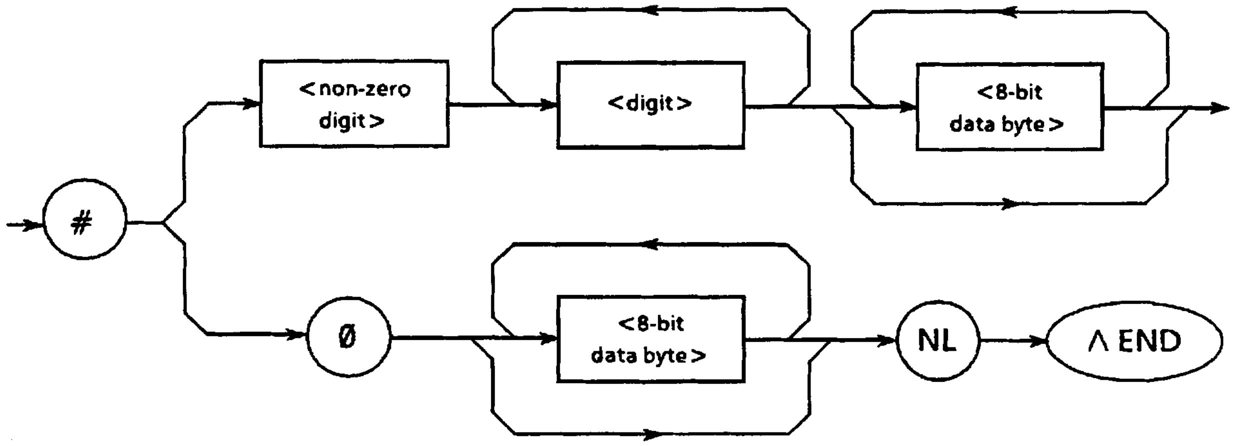
This unit supports no corresponding commands.

5.3.6 <ARBITRARY BLOCK PROGRAM DATA>

<ARBITRARY BLOCK PROGRAM DATA> is non-decimal numeric program data beginning with mark #. It is used to directly transmit binary data, assuming one byte = eight bits to be a minimum block. However, there are the following differences from the non-decimal numeric program data above (page 5-27), <NON-DECIMAL NUMERIC PROGRAM DATA> :

- Character string data and numeric data can be processed regardless of numeric data.
- The number of sent data bytes is written between the mark # and head data.

Like this, the non-decimal numeric data is program data of which data byte to be transferred can be specified arbitrarily. It is defined as follows:



● <digit>	Defined as a single ASCII code byte in the range of ASCII code bytes 30 to 39 (decimal numbers 48 to 57 = numeric values 0 to 9).
● <non-zero digit>	Defined as a single ASCII code byte in the range of ASCII code bytes 31 to 39 (decimal numbers 49 to 57 = numeric values 1 to 9).
● <8-bit data byte>	Defined as 8-bit byte in the range of 00 to FF (decimal numbers 0 to 255).

(1) When the number of data bytes to be sent is known:

The route on the upper right is used in the syntax diagram above. The number of bytes in <8-bit data byte> to be transferred must be specified at the position of <digit> in the figure above, that is, just before data writing. Then, write the number of digits in the specified number of bytes between the mark # and <digit>, that is, at the position of <non-zero digit>. For example, to send 4-byte data byte (DAB), write the following:

To send four bytes, specify 4 at the position of <digit>.

↓

# 14 <DAB> <DAB> <DAB> <DAB>

↑

"4" at the position of <digit> on the right is composed of one digit, and the value of <non-zero digit> is 1.

To send four bytes, specify 4 at the position of <digit>. It may be prefixed by 0(s).

$$\begin{tikzcd}[row sep=small] \arrow[d] \\ \text{\# 3004 <DAB> <DAB> <DAB> <DAB>} \\ \arrow[u] \end{tikzcd}$$

"4" at the position of <digit> on the right is composed of three digits, and the value of <non-zero digit> is 3.

(2) When the number of data bytes to be sent is unknown:

The route shown on the lower right is used in the syntax diagram on page 5-28. Specify #0 before the first data. Since NL_END is specified after the last data, processing terminates without exit.

#0<DAB> <DAB> <DAB> <DAB> <DAB> NL $\wedge$ END

(3) Processing of integer precision binary data

The integer precision binary data is used as transfer data in the <ARBITRARY BLOCK> format for program data and response data. It has the specifications below. A negative number is processed as a twos complement.

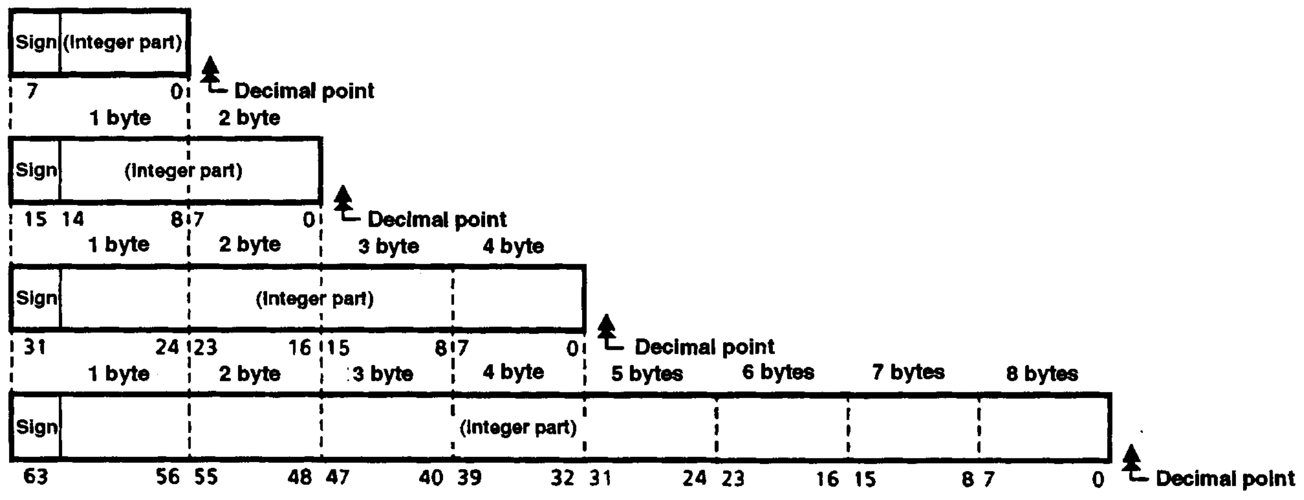
Number of transferred bytes	1, 2, 4 or 8 bytes
Transfer sequence	Transferred in sequence from the highest level.
Signed binary code	● LSD ······ Right-justified from the right end. ● MSB ······ Processed as a sign bit. ● When the data length is shorter than the field length, the excess in the field is padded by MSBs.
Unsigned binary code	● LSD ······ Right-justified from the right end. ● MSB ······ Not processed as a sign bit. ● High-order bits not used are padded by 0s.

The table below gives the signed and unsigned ranges of 1-byte 8-bit and 2-byte 16-bit integral data.

8-Bit Binary	With Sign	No Sign
10000000	-128	128
10000001	-127	129
10000010	-126	130
11111101	-3	253
11111110	-2	254
11111111	-1	255
00000000	0	0
00000001	1	1
00000010	2	2
00000011	3	3
01111101	125	125
01111110	126	126
01111111	127	127

16-Bit Binary	With Sign	No Sign
1000000000000000	-32768	32768
1000000000000001	-32767	32769
1000000000000010	-32766	32770
1111111111111101	-3	65533
1111111111111110	-2	65534
1111111111111111	-1	65535
0000000000000000	0	0
0000000000000001	1	1
0000000000000010	2	2
0000000000000011	3	3
0111111111111101	32765	32765
0111111111111110	32766	32766
0111111111111111	32767	32767

The figure below shows the internal format of integral data composed of one, two, three, four, and eight bytes. Sign bit = 0 indicates positive data; sign bit = 1 indicates negative data.



(4) Processing of floating-point binary data

The floating-point binary data is used as transfer data in the <ARBITRARY BLOCK> format for program data and response data. Its general specifications are as follows. However, our device does not support the floating-point binary data function.

A numeric value in this format must consist of the following three fields:

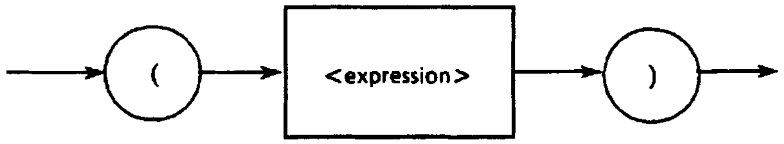
1) Sign field (Sign bit)
2) Exponent field (Exponent bit)
3) Mantissa field (Mantissa bit)

This numeric value is numeric data including a decimal, and its precision is classified into two types: numeric precision and single precision. The table below gives the field configuration and transfer sequence. In this table, a sign bit is indicated by S; the exponent bit in LSB by EL; the exponent bit in MSB by FM; the mantissa bit in LSB by FL.

Precision	Number of transfer bytes	Field structure and transfer order								
		Transfer byte	DIO line							
			8	7	6	5	4	3	2	1
Single precision	4 bytes	1st byte	S	EM	E	E	E	E	E	E
		2nd byte	EL	FM	F	F	F	F	F	F
		3rd byte	F	F	F	F	F	F	F	F
		4th byte	F	F	F	F	F	F	F	FL
		① Sign bit: 1 bit ② Exponent bit: 8 bits (+127 to -126) ③ Mantissa bit: 23 bits								
Double precision	8 bytes	1st byte	S	EM	E	E	E	E	E	E
		2nd byte	E	E	E	EL	FM	F	F	F
		3rd to7th bytes	F	F	F	F	F	F	F	F
		8th byte	F	F	F	F	F	F	F	FL
		① Sign bit: 1 bit ② Exponent bit: 11 bits (+1023 to -1022) ③ Mantissa bit: 52 bits								

5.3.7 <EXPRESSION PROGRAM DATA>

The <EXPRESSION PROGRAM DATA> element is used to send an expression for obtaining a scalar, vector, matrix, or string value to a device so that the device can calculate it instead of the controller. In the encoding syntax diagram, this element is defined as follows:



● <expression>	<expression> uses ASCII characters as a sequence in the range of ASCII code bytes 20 to 7E (decimal numbers 32 to 126). However, the following six characters enclosed by brackets [ ] are excluded : [" (double quotation mark), # (number symbol), ' (single quotation mark), ((left parenthesis),) (right parenthesis), ; (semicolon)]

If a + b + c is specified for <expression>, the following syntax diagram is obtained:

(a + b + c)

Note :

This unit does not support the <expression> function.

Section 6 Talker Output Format

The device message, which is data message transferred between the controller and device, is classified into two types: program and response messages. This chapter explains the format of the response message sent from a talker device to the controller.

6.1 Differences between Listener Input and Talker Output Formats in Syntax

There are the following major differences between listener input format and talker output format in the syntax :

- Listener format — Aims at a flexible program generation so that a device can easily accept program messages from the controller. Therefore, even if there are differences between program message expressions, those program messages function normally. For example, <white space> can be joined to a separator or terminator by an arbitrary number, and the user can generate an easy-to-read program.

- Talker format — An output message is sent according to the strictly defined syntax so that the controller can easily accept a response message output from a device. Oppositely from the listener format, only one notation is assigned to each function in the syntax of the response message.

The table below summarizes differences between listener input format and talker output format. In this table, zero or one or more spaces indicate <white space>.

Item	Listener input program message syntax	Talker output response message syntax
Characteristics	(Flexible)	(Strict)
Alphabetic character	Uppercase and lowercase characters have the same meaning.	Uppercase characters only
Before and after E in NR3 exponent part	0 or more spaces + E/e + 0 or more spaces	Uppercase character E only
+ sign in NR3 exponent part	Can be omitted.	Cannot be omitted.
<white space>	Multiple <white space> 's can be specified before and after a separator or before a terminator.	Not used.
Message unit	(1) Header with program data (2) Header without program data	(1) Data with header (2) Data without header
Unit separator	0 or more spaces + semicolon (;)	Semicolon only
Header prefix space	0 or more spaces + header	Header only
Header separator	Header + one or more spaces	Header + one $20*
Data separator	0 or more spaces + comma + 0 or more spaces	Comma only
Terminator	0 or more spaces+{NL, EOI, or NL + EOI}	NL+EOI

* ASCII code byte 20 (decimal number 32 = ASCII character SP, space)

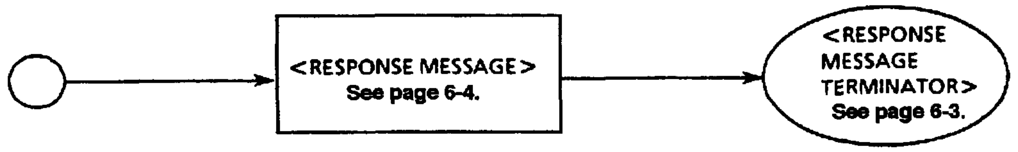
6.2 Response Message Function Elements

A response message output from a talker is terminated by the NL_END signal and accepted by the controller. This section explains each function element of the response message.

The notation in the syntax diagram is the same as for the program message. For details, see Chapter 5. The function elements and encoding elements are omitted if they are duplicated with those of the program message. For details, see Chapter 5.

6.2.1 <TERMINATED RESPONSE MESSAGE>

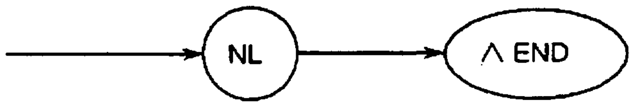
<TERMINATED RESPONSE MESSAGE> is defined as follows:



<TERMINATED RESPONSE MESSAGE> is a data message that satisfies all function elements required to send data from a talker device to the controller.
To complete a transfer of <RESPONSE MESSAGE>, <RESPONSE MESSAGE> is suffixed by <RESPONSE MESSAGE TERMINATOR>.

6.2.2 <RESPONSE MESSAGE TERMINATOR>

<RESPONSE MESSAGE TERMINATOR> is defined as follows:



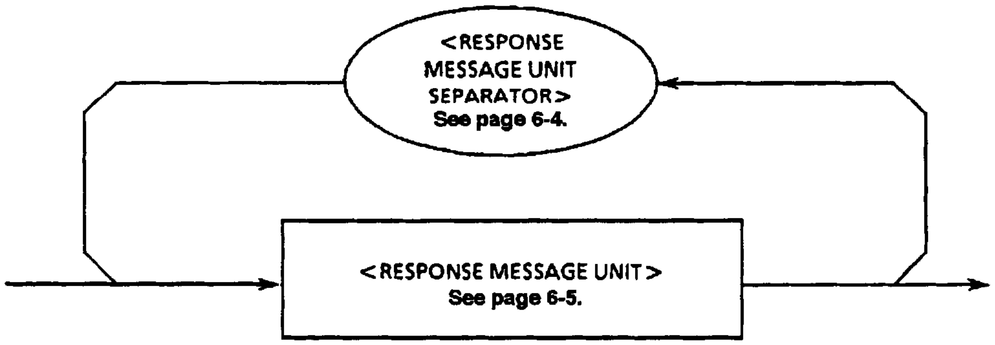
<RESPONSE MESSAGE TERMINATOR>, that is positioned after the last <RESPONSE MESSAGE UNIT>, terminates a sequence composed of one or more <RESPONSE MESSAGE UNIT> elements.
When NL_END is specified, execute the next statement at the start of the program; an EOI signal is sent as an END signal together with terminator LF at sending of the last data byte. (☞ See page 5-9.)

<Example> Read the currently specified center frequency.

```
10 LET   ADR = 101
20 TERM IS   CHR $   (10)   ! .......... Sets the terminator code to LF (new line).
30 EOI   ON ! .................................. Outputs an EOI signal that makes the EOI line true at sending of the last data byte.
40 WRITE   @ ADR : "WCNT ?" ! ..... Center wavelength reading query
50 READ   @ ADR : A$ ! .................. Terminates the response data reading with the EOI signal.
60 PRINT A$
70 END
```

6.2.3 <RESPONSE MESSAGE>

<RESPONSE MESSAGE> is defined as follows:

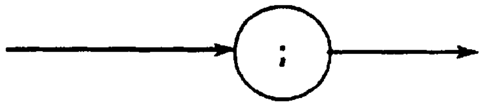


<RESPONSE MESSAGE> is a sequence composed of one or more <RESPONSE MESSAGE UNIT> elements.

The <RESPONSE MESSAGE UNIT> element means a single message sent from a device to the controller. <RESPONSE MESSAGE UNIT SEPARATOR> is used as a separator to delimit two or more <RESPONSE MESSAGE UNIT> elements.

6.2.4 <RESPONSE MESSAGE UNIT SEPARATOR>

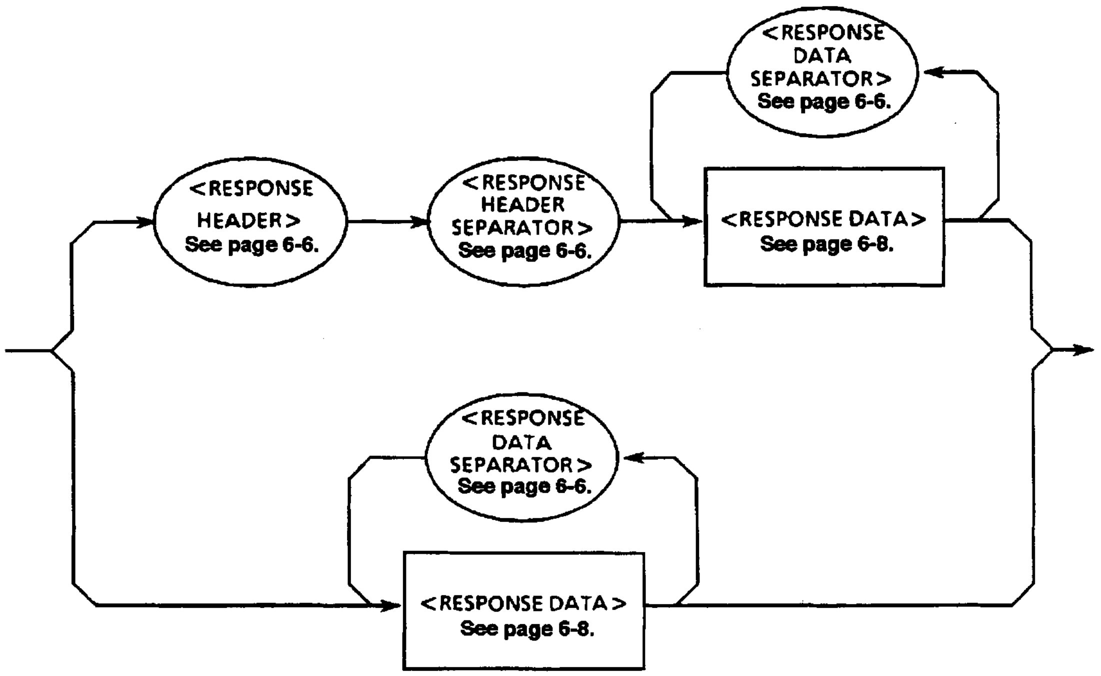
<RESPONSE MESSAGE UNIT SEPARATOR> is defined as follows:



When a sequence composed of two or more <RESPONSE MESSAGE UNIT> elements is output as a <RESPONSE MESSAGE>, <RESPONSE MESSAGE UNIT SEPARATOR> divides those <RESPONSE MESSAGE UNIT> elements by <UNIT SEPARATOR> semicolon (;).

6.2.5 <RESPONSE MESSAGE UNIT>

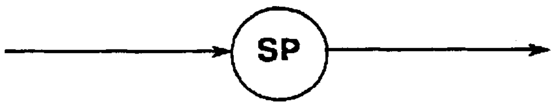
<RESPONSE MESSAGE UNIT> is defined as follows:


<RESPONSE DATA SEPARATOR> See page 6-6.
<RESPONSE HEADER> See page 6-6.
<RESPONSE HEADER SEPARATOR> See page 6-6.
<RESPONSE DATA> See page 6-8.
<RESPONSE DATA SEPARATOR> See page 6-6.
<RESPONSE DATA> See page 6-8.

<RESPONSE MESSAGE UNIT> consists of two basic syntaxes. One is a response message unit with the header used to correctly return the processing result on the information defined with a program message. Another one is a response message unit without header used to effectively return request data only.
This unit supports the latter syntax.

6.2.6 <RESPONSE HEADER SEPARATOR>

<RESPONSE HEADER SEPARATOR> is defined as follows :



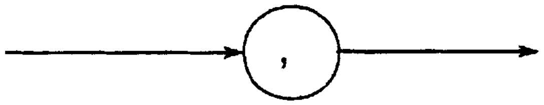
<RESPONSE HEADER SEPARATOR> separates <RESPONSE HEADER> from <RESPONSE DATA> by inserting one space after <RESPONSE HEADER>.

Space SP is indicated by ASCII code byte 20 (decimal number 32).

In other words, a response message with header contains only one space between the header and data as a response header separator. It indicates the end of the response header and the beginning of the response data.

6.2.7 <RESPONSE DATA SEPARATOR>

<RESPONSE DATA SEPARATOR> is defined as follows :



<RESPONSE DATA SEPARATOR> is positioned between <RESPONSE DATA> elements as a separator.

6.2.8 <RESPONSE HEADER>

<RESPONSE HEADER> is the same as <COMMAND PROGRAM HEADER> explained in pages 5-13 to 5-15 in the format, excluding the following three points :

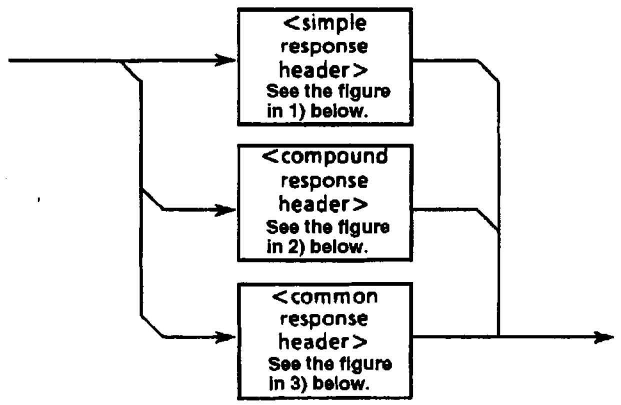
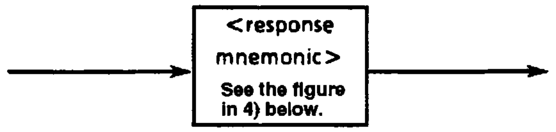
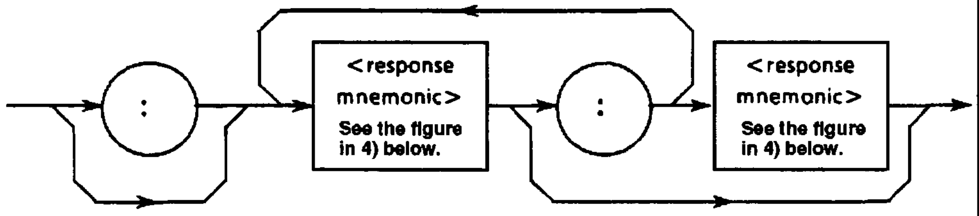
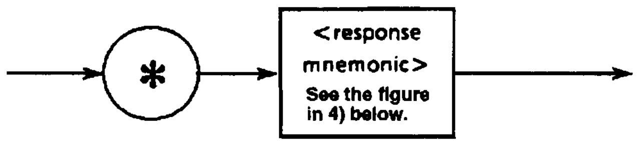
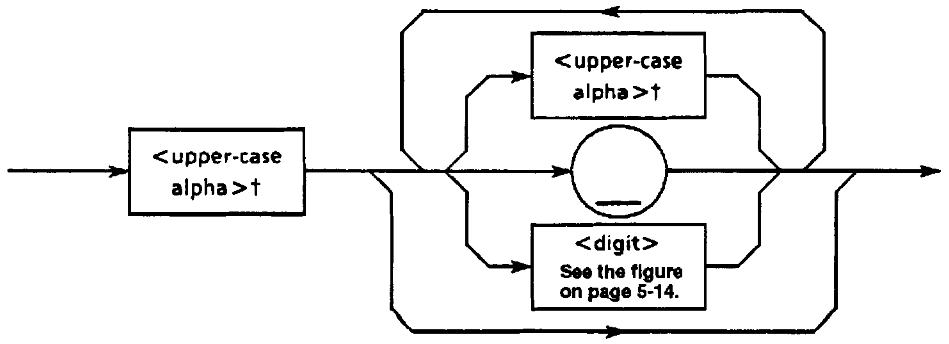
1) Valid characters are defined by <response mnemonic> ; only uppercase characters are allowed. Others are the same as for <program mnemonic>.

2) A space can be positioned before the program header, but cannot be specified before the response header.

3) Two or more spaces can be positioned after the program header, but only one space can be specified after the response header.

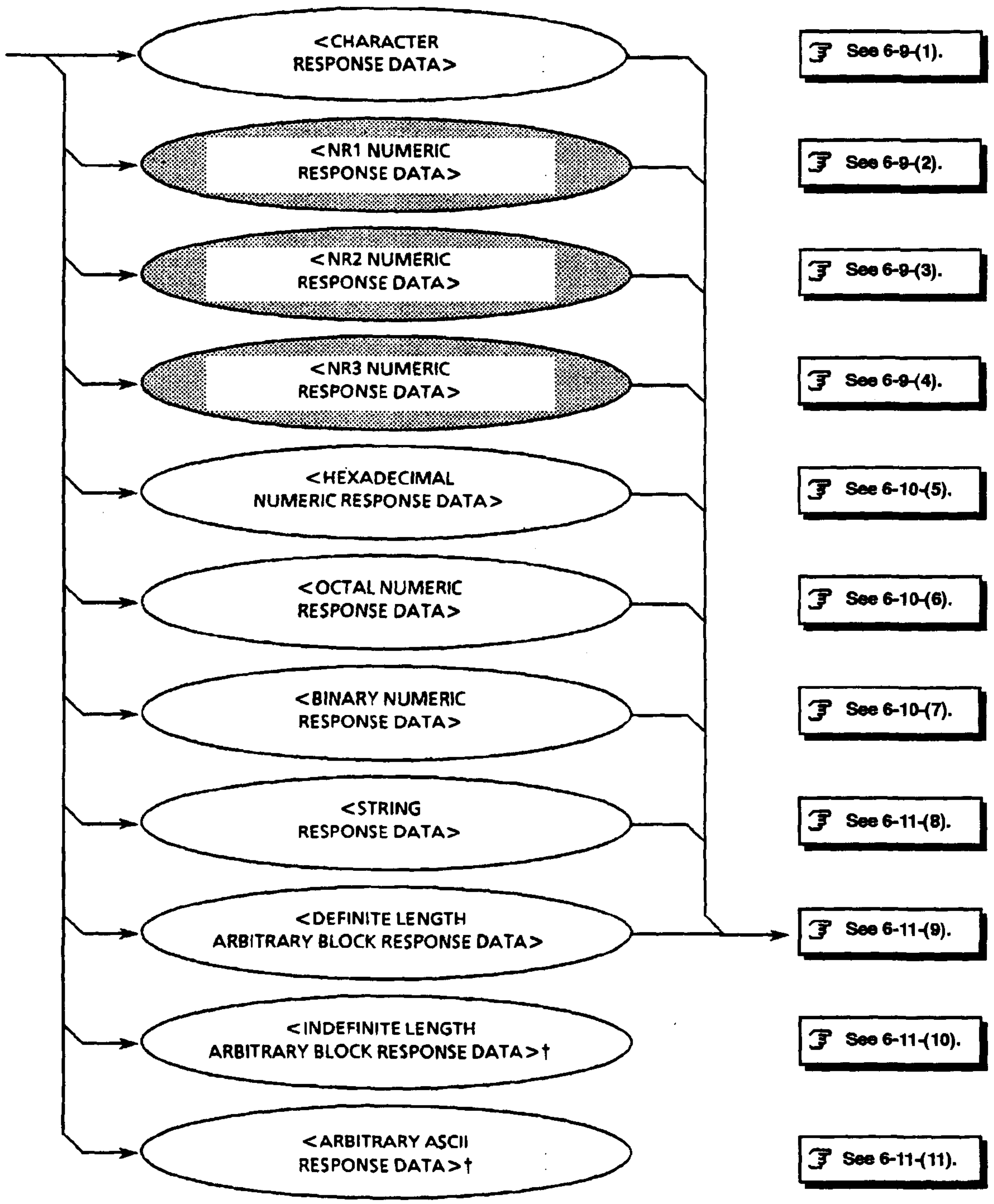
The next page shows the explanation of up to <response mnemonic> in brief.

(☞ <response mnemonic> is the same as <program mnemonic> explained in P.5-14 except that only uppercase characters are allowed for <response mnemonic>.)

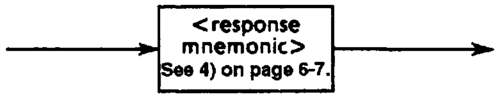
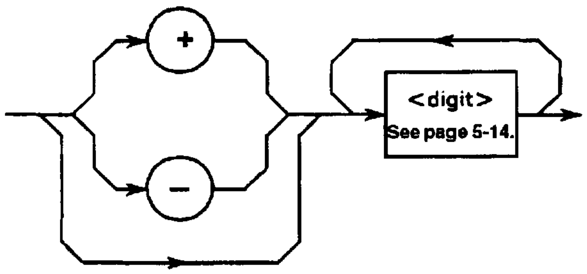
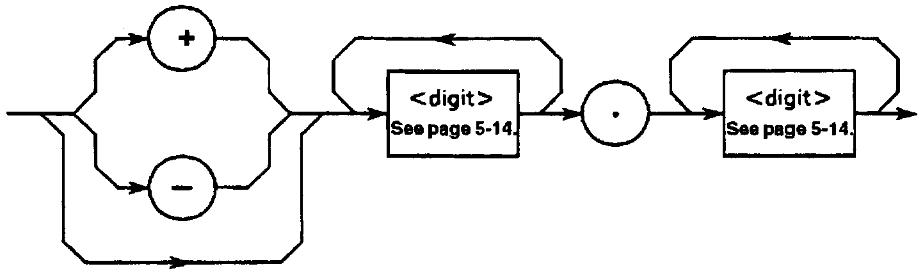
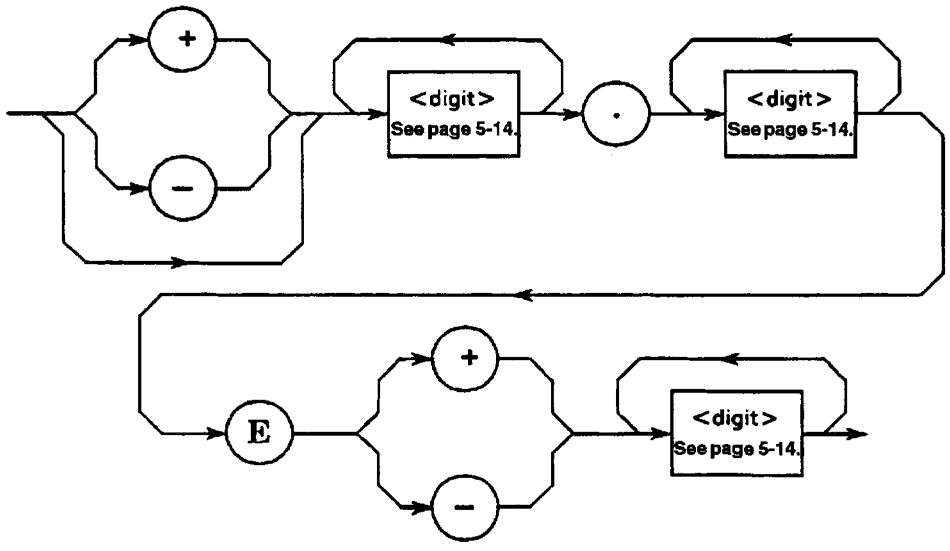
Item	Function
RESPONSE HEADER	**A header indicates a function of response data. It explains the function with a 12-character-long character string or a <response mnemonic> element that consists of uppercase characters, numeric characters, and/or underline.**  <simple response header> See the figure in 1) below. <compound response header> See the figure in 2) below. <common response header> See the figure in 3) below. **1) <simple response header> is defined as follows:**  **2) <compound response header> is defined as follows:**  : <response mnemonic> See the figure in 4) below. : <response mnemonic> See the figure in 4) below. **3) <common response header> is defined as follows:**  **4) <response mnemonic> is defined as follows:**  <upper-case alpha>† <upper-case alpha>† _ <digit> See the figure on page 5-14. † <upper-case alpha>: ASCII code bytes 41 to 5A (decimal values 65-90 = uppercase characters A to Z)

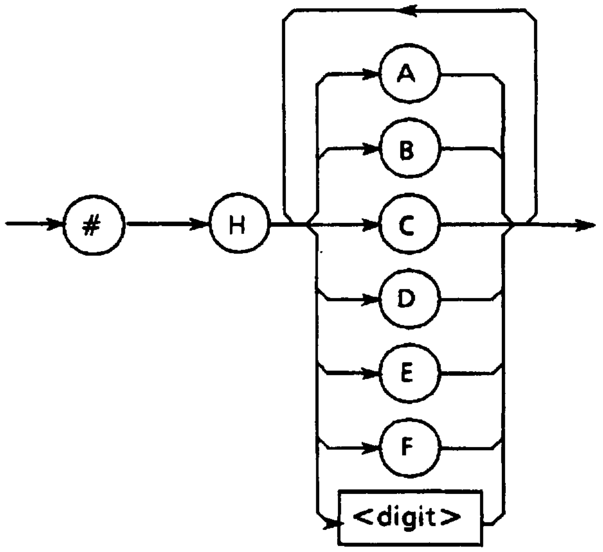
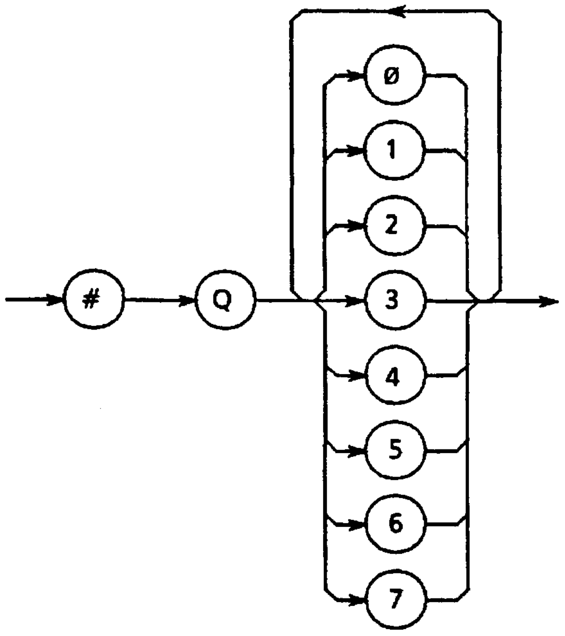
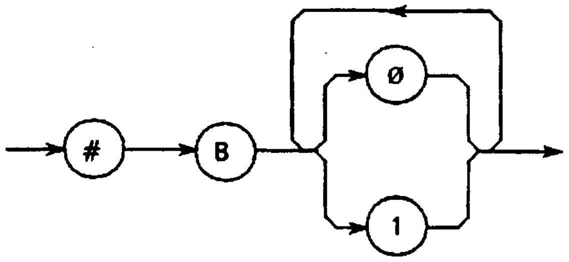
6.2.9 <RESPONSE DATA>

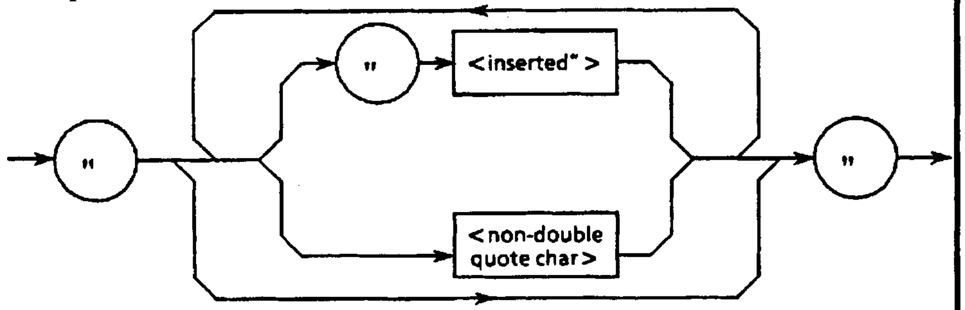
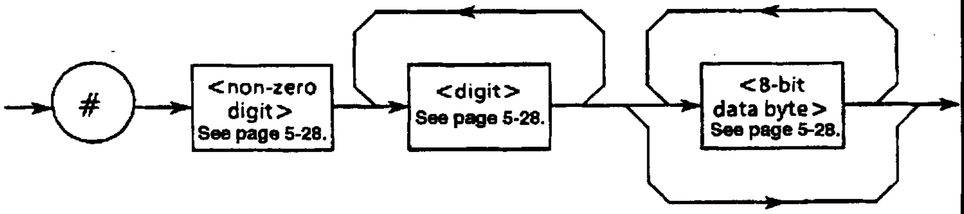
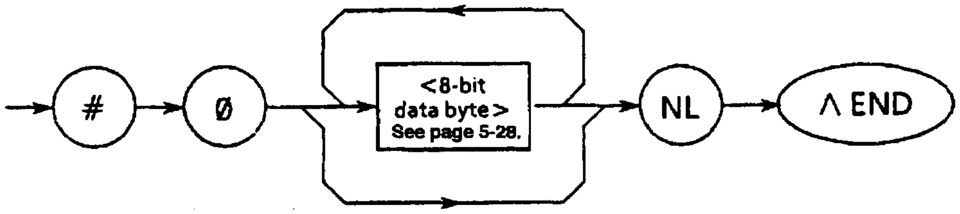
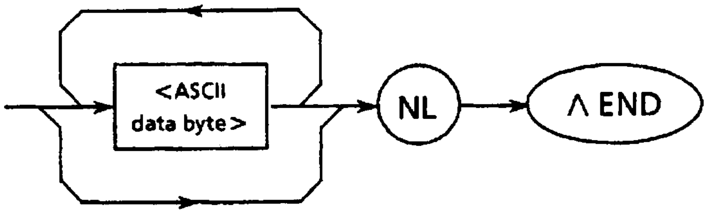
There are 11 types of <RESPONSE DATA> elements. The MG9637A/MG9638A sends the response data enclosed with the hatching of these <RESPONSE DATA> elements to the controller. Which response data is returned is determined according to the query message.


<CHARACTER RESPONSE DATA>
See 6-9-(1).
<NR1 NUMERIC RESPONSE DATA>
See 6-9-(2).
<NR2 NUMERIC RESPONSE DATA>
See 6-9-(3).
<NR3 NUMERIC RESPONSE DATA>
See 6-9-(4).
<HEXADECIMAL NUMERIC RESPONSE DATA>
See 6-10-(5).
<OCTAL NUMERIC RESPONSE DATA>
See 6-10-(6).
<BINARY NUMERIC RESPONSE DATA>
See 6-10-(7).
<STRING RESPONSE DATA>
See 6-11-(8).
<DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>
See 6-11-(9).
<INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>†
See 6-11-(10).
<ARBITRARY ASCII RESPONSE DATA>†
See 6-11-(11).

† <INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> and <ARBITRARY ASCII RESPONSE DATA> are terminated with NL END, following their last data byte.

Item	Function
(1) CHARACTER RESPONSE DATA <Example> ATT2__AUTO ATT3__MANUAL	Data consisting of the same character string as that of <response mnemonic>. Accordingly, the character string always begins with an uppercase character and its length is less than 12 characters. Numeric parameters must not be used. 
(2) NR1 NUMERIC RESPONSE DATA <Example> 123 +123 −1234	Integer data, i.e., a decimal value of an integer that has neither decimal point nor exponent. 
(3) NR2 NUMERIC RESPONSE DATA <Example> 12.3 +12.34 −12.345	Fixed-point data, i.e., a decimal value other than integers or a decimal value having an exponent.  + − <digit> See page 5-14. . <digit> See page 5-14.
(4) NR3 NUMERIC RESPONSE DATA <Example> 1.23E+4 +12.34E−5 −12.345E+6 ● Lowercase characters cannot be used for E. ● E must not be preceded and followed by a space. ● + in the exponent part is mandatory. ● + in the mantissa part is mandatory.	Fixed-point data, i.e., a decimal value having an exponent.  + − <digit> See page 5-14. . <digit> See page 5-14. E + − <digit> See page 5-14.

Item	Function
(5) HEXADECIMAL NUMERIC RESPONSE DATA <Example> #HABC123 #H2DC3 #H8301	Data represented in hexadecimal notation. 
(6) OCTAL NUMERIC RESPONSE DATA <Example> #Q37 #Q26703 #Q30562	Data represented in octal notation. 
(7) BINARY NUMERIC RESPONSE DATA <Example> #B011101 #B1011 #B1011	Data represented in binary notation. 

Item	Function
(8) STRING RESPONSE DATA **<Example>** "This is a text" "Say,""Hello""."	Any ASCII 7-bit code can be used. The character string must be enclosed with double quotation marks. When a character string contains double quotation marks, two identical quotation marks must be written in succession per quotation mark. Since a CR, LF, or space can be used, this element is suitable for outputting a text to the printer or CRT. 
(9) DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA **<Example>** Transferring 11256099D in a 4-byte blocks ↓ #1400ABC123	Fixed-point 8-bit binary block data. It is suitable for transferring large-volume data, 8-bit extended ASCII code, and non-display data. (For details on individual elements, see page 5-28.)  # <non-zero digit> See page 5-28. <digit> See page 5-28. <8-bit data byte> See page 5-28.
(10)INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA **<Example>** Indefinite-length -250, -50, 120, ... are transferred. ↓ #0FF06FFCE0078	Indefinite-length 8-bit binary block data. #0 must be written before the first data. The last data must be followed by NL^END for termination. 
(11)ARBITRARY ASCII RESPONSE DATA **<Example 1>** <ASCII Byte><ASCII Byte>NL∧END **<Example 2>** NL∧END	ASCII data bytes except NL character are transferred in succession. The last data must be followed by NL^END for termination. 

Section 7 Common Commands

This chapter explains common commands and common query commands defined in IEEE488.2. These common commands are not bus commands used for interface messages. Like the device messages, the common commands are a kind of data message used when the bus data mode, that is, ATN line, is false, and they can be used commonly to all measuring instruments including non-ANRITSU products if they are compatible with the IEEE488.2 model. The IEEE488.2 common commands begin with an asterisk (*).

7.1 Functions for each MG9637A/MG9638A Common Command Group

The table below lists the functions for each MG9637A/MG9638A IEEE488.2 common command group. The supported commands are explained in alphabetic order from the next page.

7.2 Classification of Supported Commands and References

The table below gives the functions for each MG9637A/MG9638A-supported command group. The explanation of each command is shown in alphabetic order from the next page.

Group	Function for each group	Mnemonic
System data	Returns the information particular to the device connected to the system, e.g., manufacturer name, model type, and serial number of the device.	*IDN? *OPT?
Internal operation	Device internal control : (1) Device reset at level 3 (2) Device internal self-test and error detection	 *RST *TST?
Synchronization	The controller synchronizes with the device by : (1) waiting for a service request ; (2) waiting for a device output queue response ; or (3) forcibly performing a sequential execution.	 *OPC *OPC? *WAI
Status & event	The status byte consists of a status summary message. Each summary bit in the message is supplied from the standard event register, output queue, and extension event register or extension queue. Three commands and four queries are provided to set, clear, validate, and invalidate data in these registers and queues and check the register setting status by a query.	*CLS *ESE *ESE? *ESR? *SRE *SRE? *STB?

* CLS Clear Status Command

(Clears the status byte register)

■ Format

```
* CLS
```

■ Example

```
WRITE   @124 : "* CLS"
WRITE   @124 : "WCNT   1550NM ; * CLS"
```

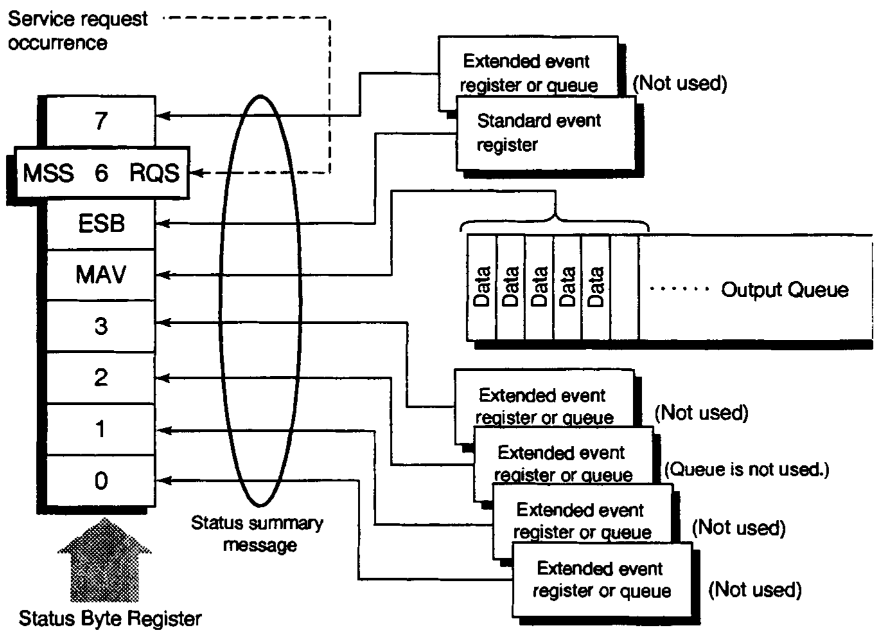
■ Description

The *CLS common command clears all the status data structures (event registers and queues) excluding the output queue and its MAV summary message. It also clears the corresponding summary messages.

In the examples below, the output queue and its MAV summary message are also cleared.

```
WRITE   @124 : "* CLS"
WRITE   @124 : "WCNT   1550NM ; * CLS"
```

When the *CLS command is sent after <PROGRAM MESSAGE TERMINATOR> or before <Query MESSAGE UNIT> element, all the status bytes are cleared. In this method, messages not read in the output queue are also cleared. The set value of each enable register is not affected by the *CLS command.


Service request occurrence
7
MSS 6 RQS
ESB
MAV
3
2
1
0
Status Byte Register
Status summary message
Extended event register or queue
(Not used)
Standard event register
Data
Data
Data
Data
Data
...... Output Queue
Extended event register or queue
(Not used)
Extended event register or queue
(Queue is not used.)
Extended event register or queue
(Not used)
Extended event register or queue
(Not used)

* ESE Standard Event Status Enable Command

(Sets or clears the standard event status enable register)

■ Format

* **ESE** <HEADER SEPARATOR> <DECIMAL NUMERIC PROGRAM DATA>

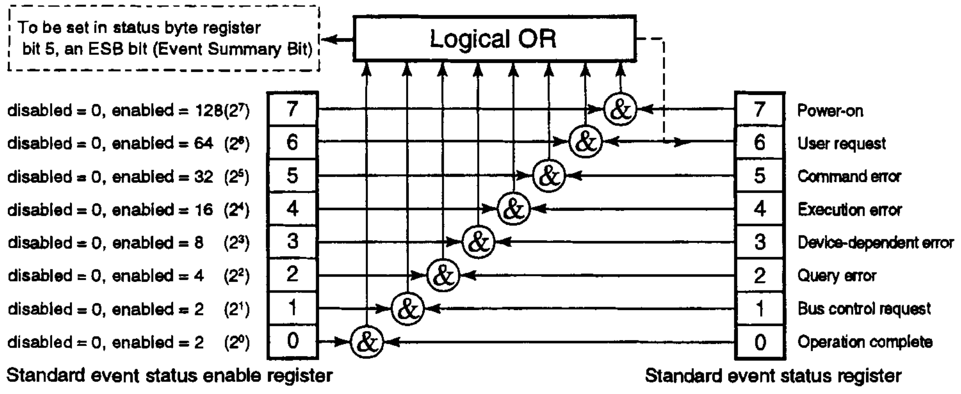
In this format, <DECIMAL NUMERIC PROGRAM DATA> indicates a numeric value rounded to integers 0 to 255. (Must be sperimposed with binary data, assuming 2 to be the base.)

■ Example

WRITE @ 124 : "* ESE 20" ! Set bits 2 and 4 in the enable register.

■ Description

Program data is obtained from the sum of the bit digit values to be enabled of 20 = 1, 21 = 2, 22 = 4, 22 = 8, 22 = 16, 25 = 32, 26 = 64, 27 = 128 corresponding to bits 1 to 7 in the standard event status enable register. The bit digit values to be disabled are set to 0.


To be set in status byte register bit 5, an ESB bit (Event Summary Bit)
Logical OR
disabled = 0, enabled = 128(2^7)
disabled = 0, enabled = 64 (2^6)
disabled = 0, enabled = 32 (2^5)
disabled = 0, enabled = 16 (2^4)
disabled = 0, enabled = 8 (2^3)
disabled = 0, enabled = 4 (2^2)
disabled = 0, enabled = 2 (2^1)
disabled = 0, enabled = 2 (2^0)
7 6 5 4 3 2 1 0
Power-on
User request
Command error
Execution error
Device-dependent error
Query error
Bus control request
Operation complete
Standard event status enable register
Standard event status register

* ESE? Standard Event Status Enable Query

(Returns the current value of the standard event status enable register as a response)

■ Format

* **ESE ?**

■ Example

If *ESE? is sent after *ESE 20, 20 is returned as a response.

■ Description

The *ESE? command returns NR1, which is a value of the standard event status enable register.

■ Response message

NR1 = 0 to 255

* ESR? Standard Event Status Register Query

(Returns the current value of the standard event status register as a response)

■ Format

* ESR ?

■ Example

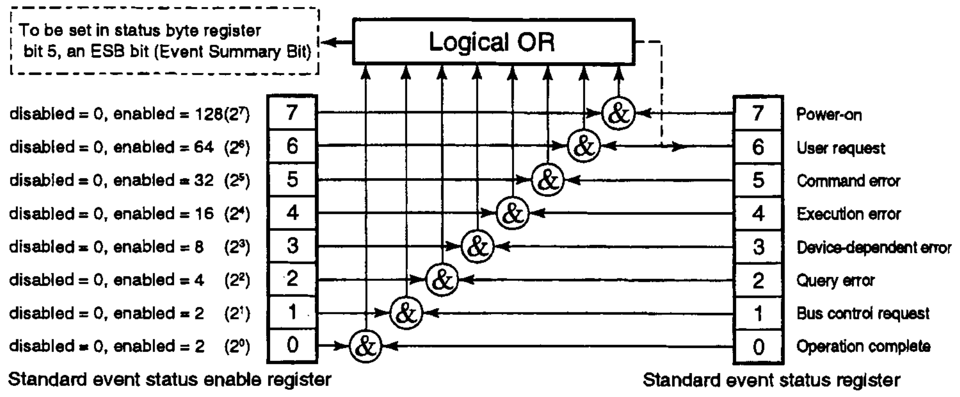
```
30  WRITE   @ 124 : "* ESR ?"
40  READ    @124 : STEVET !  If the contents of this variable is 32, a command error occurs.
50  PRINT   STEV       ET
```

■ Response message NR1

NR1 = 0 to 255

■ Description

The *ESR? command returns NR1 as the current value of the standard event status register. NR1 is obtained from the sum of the bit digit values enabled by the standard event status enable register for 20 = 1, 21 = 2, 22 = 4, 23 = 8, 24 = 16, 25 = 32, 26 = 64, 27 = 128 corresponding to bits 0 to 7 in the standard event status enable register. When the response message is read, e.g., line 40, this register is cleared.


To be set in status byte register bit 5, an ESB bit (Event Summary Bit)
Logical OR
disabled = 0, enabled = 128(2^7)
disabled = 0, enabled = 64 (2^6)
disabled = 0, enabled = 32 (2^5)
disabled = 0, enabled = 16 (2^4)
disabled = 0, enabled = 8 (2^3)
disabled = 0, enabled = 4 (2^2)
disabled = 0, enabled = 2 (2^1)
disabled = 0, enabled = 2 (2^0)
7 6 5 4 3 2 1 0
&
Power-on
User request
Command error
Execution error
Device-dependent error
Query error
Bus control request
Operation complete
Standard event status enable register
Standard event status register

* IDN? Identification Query

(Returns the product manufacturer name, model type, serial number, and firmware level.)

■ Format

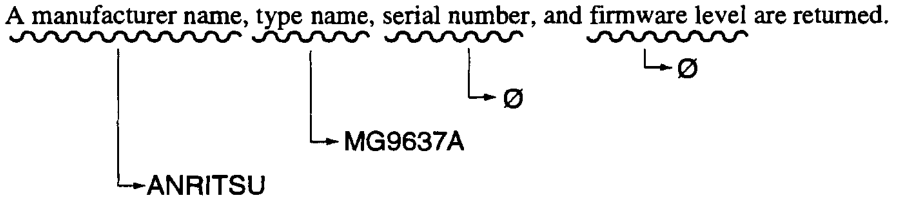
* IDN ?

■ Example

```
30  write   @ 124 : " * IDN"
40  READ @ 124 : IDEN $ !  Stores the manufacturer name, model type, serial number, and
firmware level
```

■ Description

A manufacturer name, type name, serial number, and firmware level are returned.



When the *IDN common query is sent to the device, a response message composed of these four fields is returned to the controller.

Field 1	Product manufacturer name (ANRITSU for our company)
Field 2	Model type (MG9637A or MG9638A)
Field 3	Manufacture number - serial number (0 for our company)
Field 4	Firmware version number (0 for our company)

If information need not be supplied to the serial number in field 3 and firmware version number in field 4, ASCII character 0 can be returned.

■ Response message

A response message in which four fields above are delimited by a comma (,) is returned with <ARBITRARY ASCII RESPONSE DATA>.

<field 1>, <field 2>, <field 3>, <field 4>
The total length of response message is equal to or greater than 72.

* OPC Operation Complete Command

(Sets bit 0 in the standard event status register at termination of device operation)

■ Format

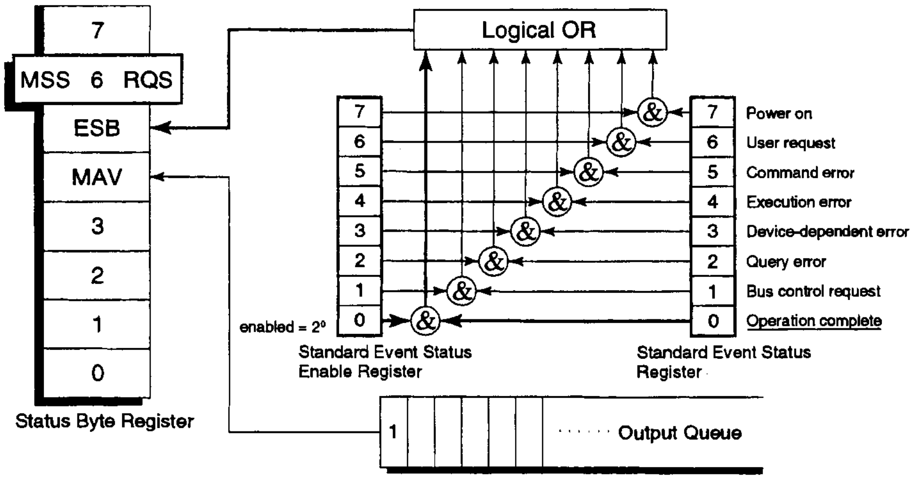
* OPC

■ Example

WRITE @ 124 : "* OPC"

■ Description

After all the selected pending device operations were completed, bit 0 in the standard event status register, that is, [operation termination bit], is set. This unit does not, however, support an overlap command, and this command has no meaning.



* OPC? Operation Complete Query

(Sets 1 into the output queue and generates an MAV summary message at termination of device operation)

■ Format

* OPC?

■ Example

WRITE @ 124 : "* OPC ?"

■ Description

After all the selected pending device operations are completed, 1 is set into the output queue and the device waits until an MAV summary message is generated.

■ Response message

Returns 1 with <NR1 NUMERIC RESPONSE DATA>.

* RST Reset Command

(Resets the device at level 3)

■ Format

```
* RST
```

■ Example

WRITE @ 124 : " * Initialize only the device at address 24.

■ Description

The *RST (Reset) command resets the device at level 3 (☞ see page 4-2). At level 3, the following items are initialized :

(1) The device functions and status are returned to a specific known state regardless of their history.
(2) A macro defined with a *DDT command is placed into the state defined by the device.
(3) The macro operation is inhibited so that no macro commands are accepted. The macro definition is returned to the state indicated by the designer.
(4) The device is placed into the operation complete command idle state (OCIS). As a result, the operation termination bit cannot be set into the standard event status register. (☞ See page 8-3.)
(5) The device is placed into the operation complete query idle state (OQIS). As a result, operation termination bit 1 cannot be set in the output queue. The MAV bit is then cleared.

The *RST command does not affect the items below.

(1) IEEE488.1 interface state
(2) Device address
(3) Output queue
(4) Service request enable register
(5) Standard event status enable register
(6) Power-on-status-clear flag setting
(7) Calibration data affecting device standard
(8) RS-232C interface conditions

* OPT? Option Identification Query

(Reports a list of mounted options.)

■ Format

*** OPT ?**

■ Example

```
30  WRITE @124" *OPT?
40  READ @124 : OPTI$        Stores the mounted option information.
```

■ Description

The *OPT? command returns the state of a mounted option with 0 or 1.

	Contents of option	State of option
OPT01	Not used	"0"
OPT02	Not used	"0"
OPT03	Not used	"0"

■ Response message

A response message in which three fields above are delimited by a comma (,) is sent with <ARBITRARY ASCII RESPONSE DATA>.
<OPT01 option state>, <OPT02 option state>, <OPT03 option state>

* SRE

Command, query

* SRE Service Request Enable Command

(Sets the bits in the service request enable register)

■ Format

*** SRE <HEADER SEPARATOR> <DECIMAL NUMERIC PROGRAM DATA>**

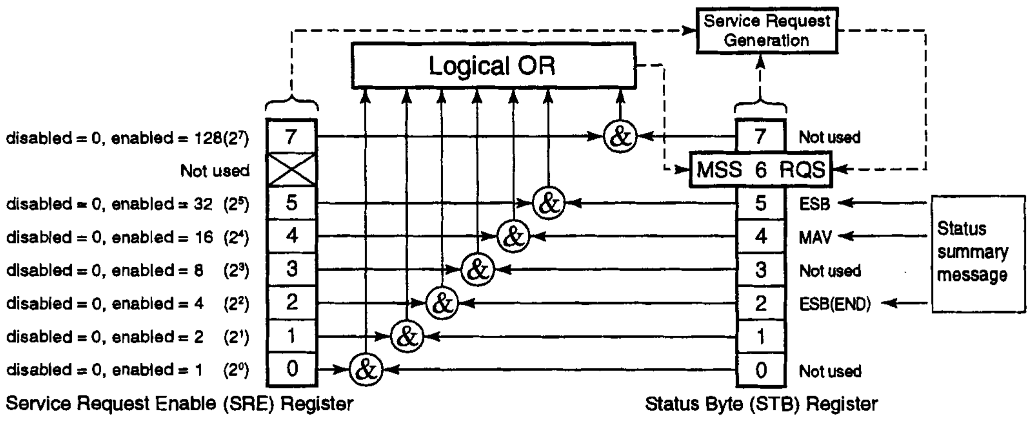
A numeric value rounded to integers 0 to 255 that is equal to <DECIMAL NUMERIC PROGRAM DATA> in this format (Must be superimposed with binary data, assuming 2 to be the base.)

■ Example

WRITE @ 124 : "* SRE 16" ! Sets bit 4 in the enable register.

■ Description

Program data is obtained from the sum of the bit digit values to be enabled of 20 = 1, 21 = 2, 22 = 4, 22 = 8, 22 = 16, 25 = 32, 26 = 64, 27 = 128 corresponding to bits 1 to 7 in the standard event status enable register. The bit digit values to be disabled are set to 0.


Service Request Generation
Logical OR
disabled = 0, enabled = 128(2^7)
Not used
disabled = 0, enabled = 32 (2^5)
disabled = 0, enabled = 16 (2^4)
disabled = 0, enabled = 8 (2^3)
disabled = 0, enabled = 4 (2^2)
disabled = 0, enabled = 2 (2^1)
disabled = 0, enabled = 1 (2^0)
7
5
4
3
2
1
0
&
Not used
MSS 6 RQS
ESB
MAV
Not used
ESB(END)
Not used
Status summary message
Service Request Enable (SRE) Register
Status Byte (STB) Register

* SRE? Service Request Enable Query

(Returns the current value of the service request enable register)

■ Format

*** SRE ?**

■ Example

If *SRE? is sent after *SRE16, 16 is returned as a response.

■ Description

Returns NR1 that is a value of the service request enable register.

■ Response message NR1

NR1 = 0 to 63 or 128 to 191 because NR1 = bit 6 (RQS bit) cannot be set.

* STB? Read Status Byte Command

(Returns the current value of the status byte including the MSS bit)

■ Format

*** STB ?**

■ Example

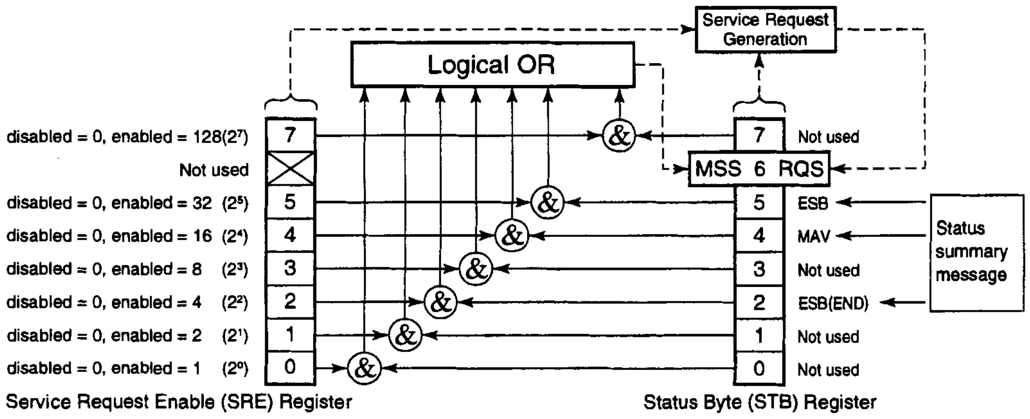
```
30  WRITE  @ 124 : "* STB?"
40  READ   @ 124 : STBV
50  PRINT  STBV
```

■ Description

Returns the sum of the status byte register value superimposed with binary data and MSS summary message value as <NR1 NUMERIC RESPONSE DATA>.

■ Response message

The response message indicates the sum of bit digit values in the status byte register that are integers 0 to 255 equal to <NR1 NUMERIC RESPONSE DATA>. Bits 0 to 5 and 7 in the status byte register are superimposed to 1, 2, 4, 8, 16, 32, and 128; the master summary status (MSS) bit to 64. MSS indicates that there is at least one reason why a service is requested. The table below gives the status byte register conditions.


Service Request Generation
Logical OR
disabled = 0, enabled = 128(2^7)
Not used
disabled = 0, enabled = 32 (2^5)
disabled = 0, enabled = 16 (2^4)
disabled = 0, enabled = 8 (2^3)
disabled = 0, enabled = 4 (2^2)
disabled = 0, enabled = 2 (2^1)
disabled = 0, enabled = 1 (2^0)
Service Request Enable (SRE) Register
7 Not used
MSS 6 RQS
5 ESB
4 MAV
3 Not used
2 ESB(END)
1 Not used
0 Not used
Status summary message
Status Byte (STB) Register

Bit	Bit superimpose value	Bit name	Status byte register conditions
7	128	——	0 = Not used
6	64	MSS	0 = Service not requested 1 = Service requested
5	32	ESB	0 = Event status not generated 1 = Event status generated
4	16	MAV	0 = There is no data in the output queue 1 = There is data in the output queue
3	8	——	0 = Not used
2	4	ESB (END)	0 = Event status not generated 1 = Event status generated
1	2	——	0 = Not used
0	1	——	0 = Not used

* TST? Self-Test Query

(Executes the internal self-test and returns error information)

■ Format

```
* TST ?
```

■ Example

```
30  WRITE   @ 124 :"* TST ?"
40  READ   @ 124 : TEST
50  PRINT   TEST
```

■ Description

The * TST? query executes the self-test in the device. The test result is written to the output queue. Data of the output queue indicates whether the test was completed without causing an error. The self-test can be executed without intervention required.

■ Response message

The response message is returned with <NR1 NUMERIC RESPONSE DATA>.
Data range = –32767 to 32767

NR1 = 0 .... Indicates that the self-test was completed without error.

NR1 = 1 .... Indicates that the self-test was not executed or an error occurred during execution of the self-test.

* WAI Wait-to-Continue Command

(Places the next command into the standby state while the device is executing a command)

■ Format

```
* WAI
```

■ Example

```
WRITE  @ 108 : "  * WAI"
```

■ Description

The *WAI common command executes an overlap command as a sequential command.

When the device can execute the next command or query sent from the controller during execution of a command or query, the first executed command or query is called an overlap command.

When the *WAI common command is executed after an overlap command, it is placed into the standby state if the device is executing another command. After the first executed command was completed, the execution of the next command is allowed. This operation is the same as a sequential command.

This unit does not however support the overlap command; therefore, this command has no meaning.

* SAV Save Command

(Saves setting conditions in the RAM in this unit)

■ Format

```
* SAV <HEADER SEPARATOR> <DECIMAL NUMERIC PROGRAM DATA>
```

In this format,
<DECIMAL NUMERIC PROGRAM DATA> = 1 to 3

■ Example

```
WRITE   @ 124 : "* SAV  1" ! Saves data in memory 1.
```

* RCL Recall Command

(Reads setting conditions from the RAM in this unit)

■ Format

* RCL <HEADER SEPARATOR> <DECIMAL NUMERIC PROGRAM DATA>

In this format,
<DECIMAL NUMERIC PROGRAM DATA> = 0 to 3

■ Example

```
WRITE  @ 124 : "* RCL  1" ! Recalls memory 1.
```

■ Description

Recalling memory 0 sets to the initialization conditions of this unit.

Section 8 Status Structure

This chapter explains the device status report defined in the IEEE488.2 standard and its data structure in addition to the synchronization method between the controller and device.

In IEEE488.2, common commands and common queries are added to supply more detailed status information as compared with IEEE488.1. For details on these commands and queries, see Chapter 7.

Section 8 Status Structure

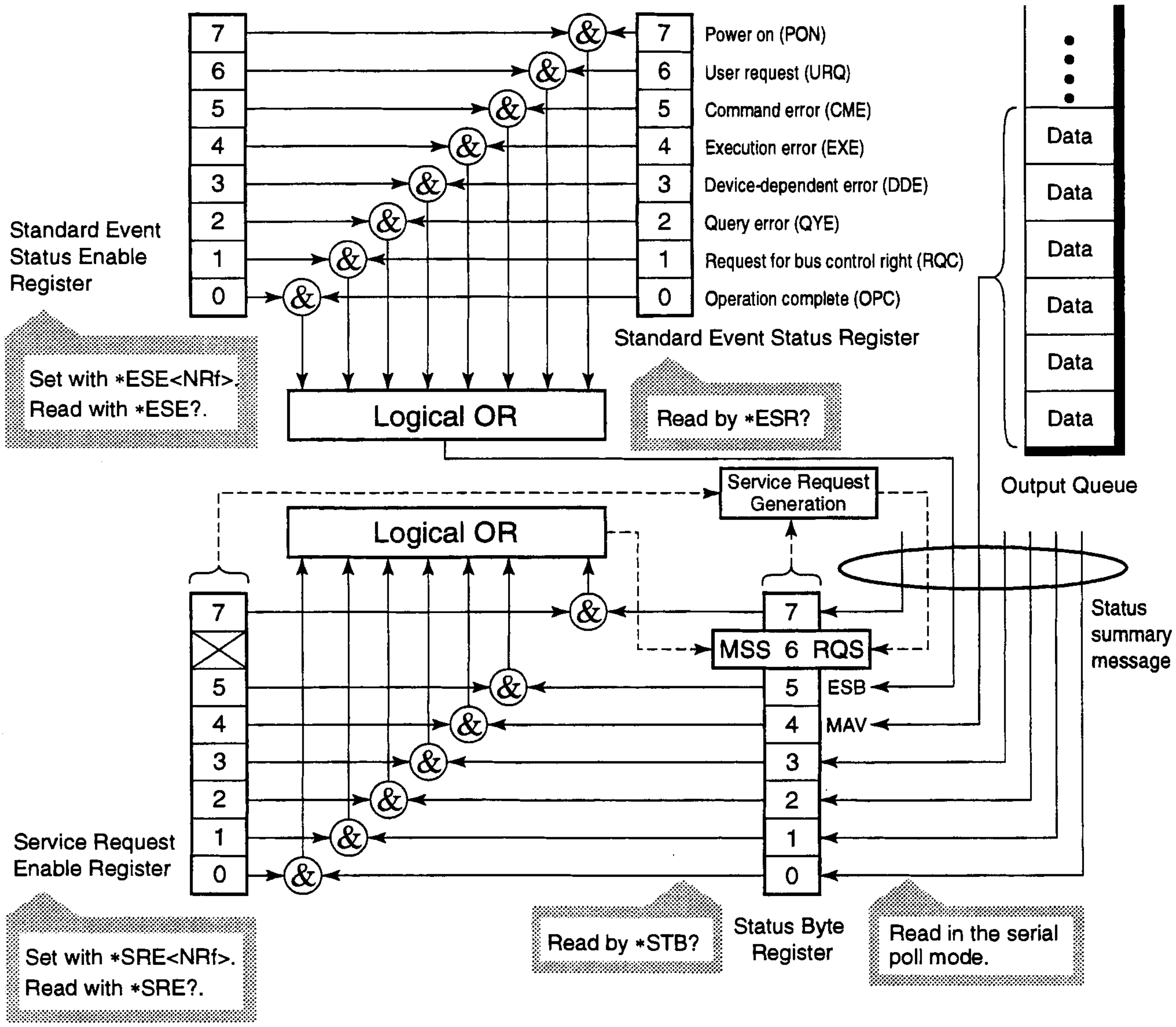
The status byte (STB) sent to the controller is based on the IEEE488.1 standard. Its configuration bit is called a status summary message that summarizes the current contents of the data saved in registers and queues.

The following sections explain a status data structure for generating the status summary message bit and status summary message bit in addition to the synchronization method between the controller and device using the status message.

This function is available to control the device from an external controller using the GPIB interface bus. It (excluding some functions) can also be used to control the device from an external controller using the RS-232C interface.

8.1 IEEE488.2 Standard Status Model

The figure below shows the standard status data structure model defined in IEEE488.2.



Power on (PON)
User request (URQ)
Command error (CME)
Execution error (EXE)
Device-dependent error (DDE)
Query error (QYE)
Request for bus control right (RQC)
Operation complete (OPC)
Standard Event Status Register
Standard Event Status Enable Register
Set with *ESE<NRf>. Read with *ESE?.
Logical OR
Read by *ESR?
Data
Output Queue
Service Request Generation
Logical OR
Status summary message
MSS 6 RQS
ESB
MAV
Service Request Enable Register
Set with *SRE<NRf>. Read with *SRE?.
Read by *STB?
Status Byte Register
Read in the serial poll mode.

Fig 8-1 Standard status model

The IEEE488.1 status byte is used in the status model. It consists of seven summary message bits supplied from the status data structure. To generate these summary message bits, the status data structure is composed of two types of models: register and queue models.

Register model	Queue model
A pair of registers for recording an event and condition the device encountered is called a register model. It consists of an event status register and event status enable register. When the AND value between these two registers is not 0, the status bit is set to 1 ; otherwise, it is set to 0. If the logical OR result is 1, the summary message bit is set to 1. When the logical OR result is 0, the summary message bit is set to 0.	A queue for recording status value or information in sequence is called a queue model. In the queue structure, the bit is set to 1 only when data exists in the queue ; otherwise, the bit is set to 0.

On the basis of the register model and queue model explained in this table, the standard model in the IEEE488.2 status data structure consists of the following two types of register models and one queue model.

(1) Standard event status register and standard event status enable register
(2) Status byte register and service request enable register
(3) Output queue

Standard event status register	Status byte register	Output queue
This register has the register model structure described above. In this register, eight types of events (1. power-on, 2. user request, 3. command error, 4. execution-time error, 5. device error, 6. query error, 7. bus control right request, 8. end of operation) are set as the standard event bits. The logical OR output bit is request-indicated in bit 5 (DI06) of the status byte register as an event status bit (EB) summary message.	This register can set the RQS bit and seven summary message bits supplied from the status data structure. It is paired with a service request enable register. When the OR value between these two registers is not 0, SRQ is set to ON. In this case, bit 6 (DI07) in the status byte register is reserved in the system to report the existence of a service request to the external controller. This SRQ structure conforms to the IEEE488.1 standard.	This register has the queue model structure described above. Its contents are summary-indicated in bit 4 (DI05) of the status byte register as a message available (MAV) summary message for reporting that data exists in the output buffer.

8.2 Status Byte (STB) Register

The STB register consists of an STB and RQS (or MSS) messages of the device. The IEEE488.1 defines how to report the STB and RQS messages, but does not cover the setting and clear protocols and STB meaning. The IEEE488.2 defines the master summary status (MSS) sent to bit 6 together with STB according to a status summary message and *STB? common query of the device.

8.2.1 ESB and MAV summary message

This section explains the ESB and MAV summary messages.

(1) ESB summary message

The event summary bit (ESB), that is a message defined in IEEE488.2, is set to bit 5 in the STB register. It indicates whether at least one or more events defined in the IEEE488.2 occurred when the service request enable register was set so that an event generation became valid after last reading or clearing the standard event status register. The ESB is set to true if at least one event registered in the standard event status register is set to true when the event generation is valid. Oppositely, the ESB is set to false if no registered event occurs even when the event generation is valid.

(2) MAV summary message

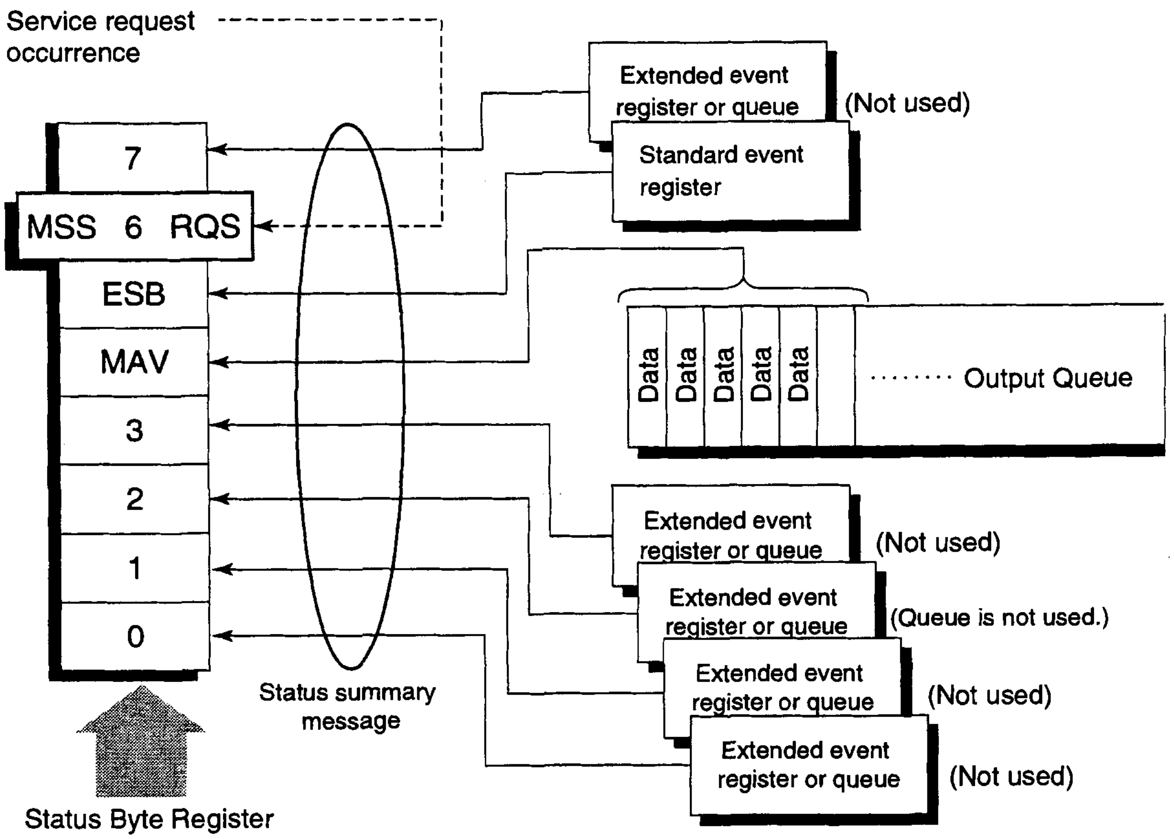
The message available (MAV) summary bit, that is a message defined in IEEE488.2, is set to bit 4 in the STB register. It indicates whether the output queue is idle. When the device is ready to accept a response message sending request from the controller, the MAV summary message bit is set to 1 (true). When the output queue is idle, it is set to 0 (false). This message is used to exchange information synchronously with the controller. For example, with this message, the controller can send a query command to a device and wait until MAV becomes true. The controller can also perform another processing while waiting for a response from the device. However, if the output queue is read without first checking MAV, all system bus operations are waited until a response is returned from the device.

8.2.2 Summary message particular to device

The IEEE488.2 does not define whether bit 7 (DIO8), bit 3 (DIO4), bit 2, bit 1, and bit 0 in the status byte register are used as summary bits in the status register or used to report that data exists in the output queue.

Each summary message particular to a device has a status data structure of a register or queue model. The status data structure is a pair of registers used to report events and status in parallel or a queue used to report status and information in sequence. The summary bit summary-indicates the current status of the status data structure. In the register model, the summary message is set to true when an event (makes one or more TRUE elements valid) exists. In the queue model, it is set to true when the queue is not idle.

This unit does not use bits 7, 3, 1, and 0, as shown below. Bit 2 is used as a summary bit for the status register. Therefore, there are two types of register models (with one type of extension) and one type of output queue (without extension).


Service request occurrence
7
MSS 6 RQS
ESB
MAV
3
2
1
0
Status Byte Register
Status summary message
Extended event register or queue
(Not used)
Standard event register
Data
Data
Data
Data
Data
Output Queue
Extended event register or queue
(Not used)
Extended event register or queue
(Queue is not used.)
Extended event register or queue
(Not used)
Extended event register or queue
(Not used)

8.2.3 Reading and clearing STB register

The contents of the STB register are read using serial polling or *STB? common query. In either method, the IEEE488.1 STB message is read; however, the value of bit 6 (position) varies depending on the selected method.
The contents of the STB register can be cleared with the *CLS command.

(1) Using serial polling for reading (Only when the GPIB interface bus is connected)

When serial polling is performed according to IEEE488.1, the device must the seven-bit status byte and RQS message bit based on IEEE488.1. In IEEE488.1, the RQS message indicates whether the device sends SRQ set to true. The value of the status byte does not vary depending on serial polling. The device must set the rsv message to false just after polling. By this operation, the RQS message is set to false when the device is polled again before a cause for requesting a new service occurs.

(2) Using *STB? common query for reading

The *STB? common query sends the contents of the STB register and one <NR1 NUMERIC RESPONSE DATA> element supplied from the master summary status (MSS) message. The response indicates the sum of the value of the STB register superimposed with binary data and MSS summary message. Bits 0 to 5 and 7 in the STB register are superimposed to 1, 2, 4, 8, 16, 32, and 128 respectively; MSS to 64. By this function, a response to the *STB? common query matches one to the serial polling except that the MSS summary message is sent to bit 6 instead of the RQS message.

(3) Definition of master summary status (MSS)

MSS indicates that there is a cause requesting at least one service in the device. The MSS message is sent to bit 6 in a response to the *STB? query returned from the device, but it is not sent as a response to the serial polling. MSS must not be assumed to be a part of the IEEE488.1 status byte. MSS is obtained by the total OR between the bits combined in the STB and SRQ enable (SRE) registers. As a result, MSS is defined as follows:

(STB Register bit 0 AND SRE Register bit 0)
OR
(STB Register bit 1 AND SRE Register bit 1)
OR
:
:
(STB Register bit 5 AND SRE Register bit 5)
OR
(STB Register bit 7 AND SRE Register bit 7)

In the MSS definition, the states of bit 6 in the STB and SRQ enable registers are ignored. To obtain the MSS value, the status byte may be processed as an 8-bit value of which bit 6 is always 0.

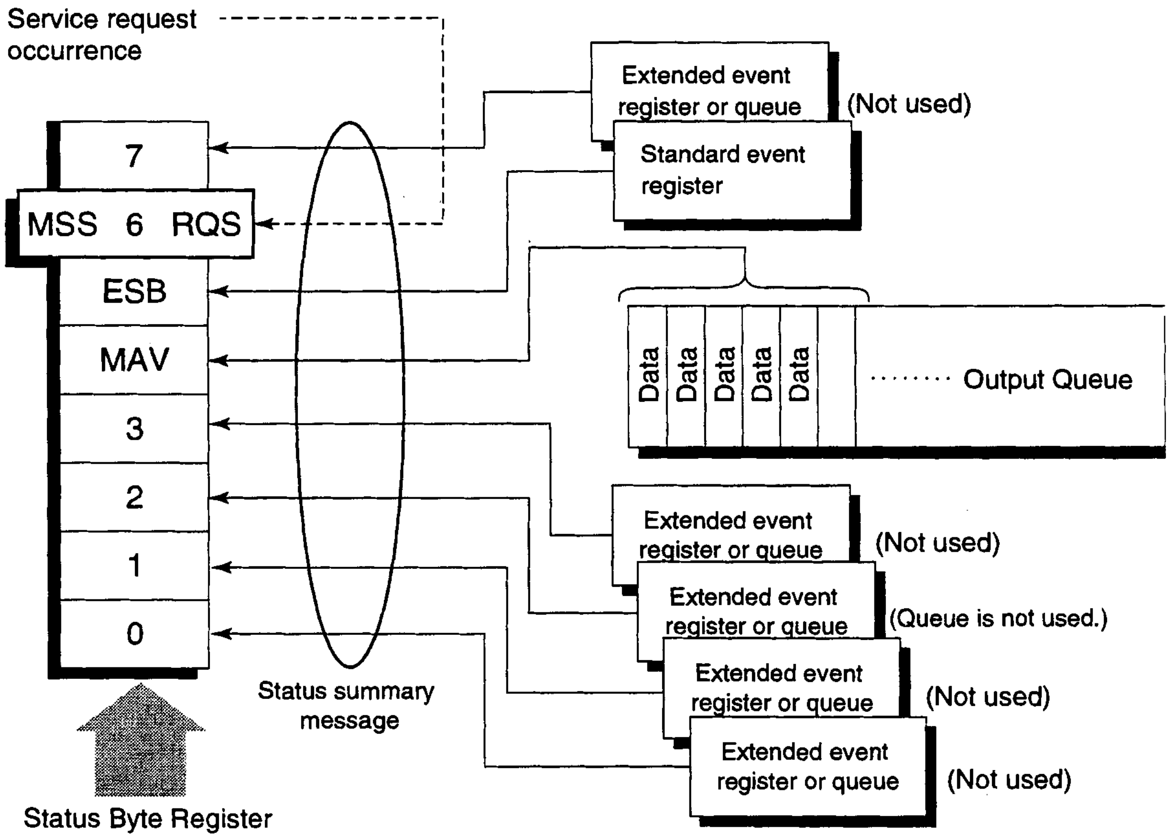
(4) Clearing STB register with *CLS common command

The *CLS common command clears all the status data structures (event registers and queues) excluding the output queue and its MAV summary message. If also clears the corresponding summary messages.

In the examples below, the output queue and MAV summary message are also cleared.

```
30  WRITE   @ ADR : "WCNT   1550NM ; POW   -10DBM"
40  WRITE   @ ADR : "*CLS ; WCNT ?"
```

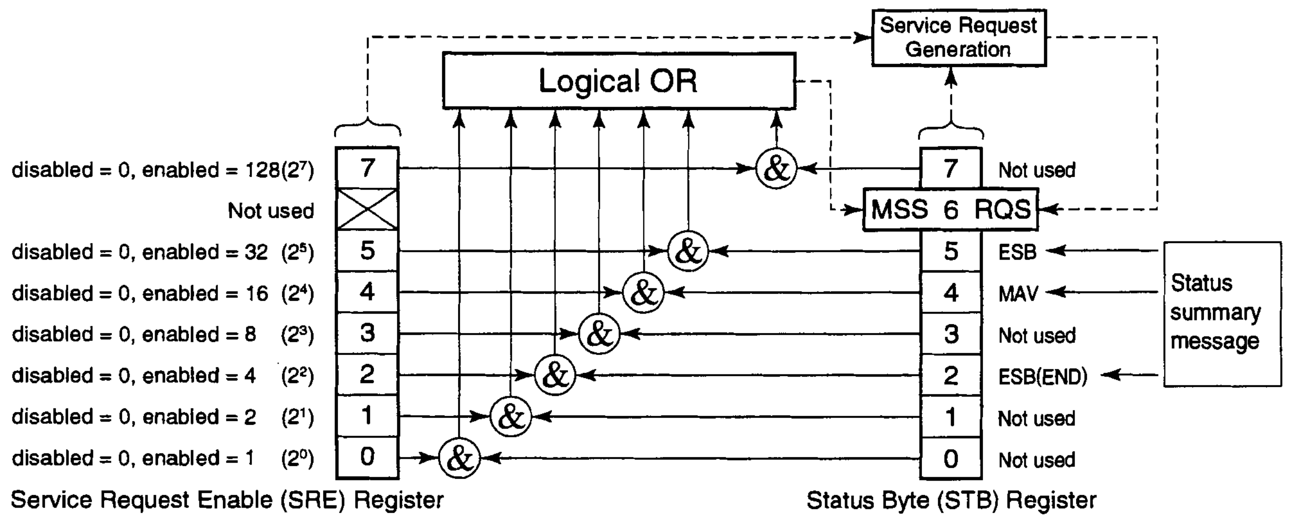
When the *CLS command is sent after <PROGRAM MESSAGE TERMINATOR> or before <Query MESSAGE UNIT> element, all the status bytes are cleared. In this method, messages not read in the output queue are also cleared. The MAV message is then set to false. The MSS message is also set to false at response to the *STB? command. The set value of each enable register is not affected by the *CLS command.


Service request occurrence
7
MSS 6 RQS
ESB
MAV
3
2
1
0
Status summary message
Status Byte Register
Extended event register or queue
(Not used)
Standard event register
Data
Data
Data
Data
Data
Output Queue
Extended event register or queue
(Not used)
Extended event register or queue
(Queue is not used.)
Extended event register or queue
(Not used)
Extended event register or queue
(Not used)

8.3 SRQ Enable

The user can select which summary message in the STB register is valid for service request depending on the SRQ enable state. The service request enable (SRE) register shown below is available to select a summary message.

Each bit in the service request enable register corresponds to one in the status byte register. When 1 is set to the bit in the status byte corresponding to a valid bit in the service request enable register, the device sets the RQS bit to 1 and outputs a service request to the controller. For example, set bit 4 in the service request enable register into the enabled state; a service request is output to the controller each time 1 is set to the MAV bit if data exists in the output queue.


Service Request Generation
Logical OR
disabled = 0, enabled = 128(2^7)
Not used
disabled = 0, enabled = 32 (2^5)
disabled = 0, enabled = 16 (2^4)
disabled = 0, enabled = 8 (2^3)
disabled = 0, enabled = 4 (2^2)
disabled = 0, enabled = 2 (2^1)
disabled = 0, enabled = 1 (2^0)
7
5
4
3
2
1
0
&
MSS 6 RQS
Not used
ESB
MAV
Not used
ESB(END)
Not used
Not used
Status summary message
Service Request Enable (SRE) Register
Status Byte (STB) Register

(1) Reading SRE register

The contents of the SRE register are read using the *SRE? common query. A response message to this query is obtained from the sum of bit digit values in the service request enable register that are integers 0 to 255 equal to <NR RESPONSE DATA>. Bits 0 to 5 and 7 in the service request enable register are superimposed to 1, 2, 4, 8, 16, 32, and 128, respectively. Bit 6 not used must be always 0.

(2) Updating SRE register

The SRE register is written using the *SRE common instruction. The *SRE common instruction is followed by <DECIMAL NUMERIC PROGRAM DATA> elements. <DECIMAL NUMERIC PROGRAM DATA> is rounded to an integer and indicated in binary notation, assuming 2 to be the base. It indicates the sum of bit digit values (wait values) in the SRE register. When this bit value is 1, the enabled state is set; otherwise, the disabled state is set. The value of bit 6 must always be ignored.

(3) Clearing SRE register

The SRE register can be cleared by executing the *SRE common command or turning the power on.

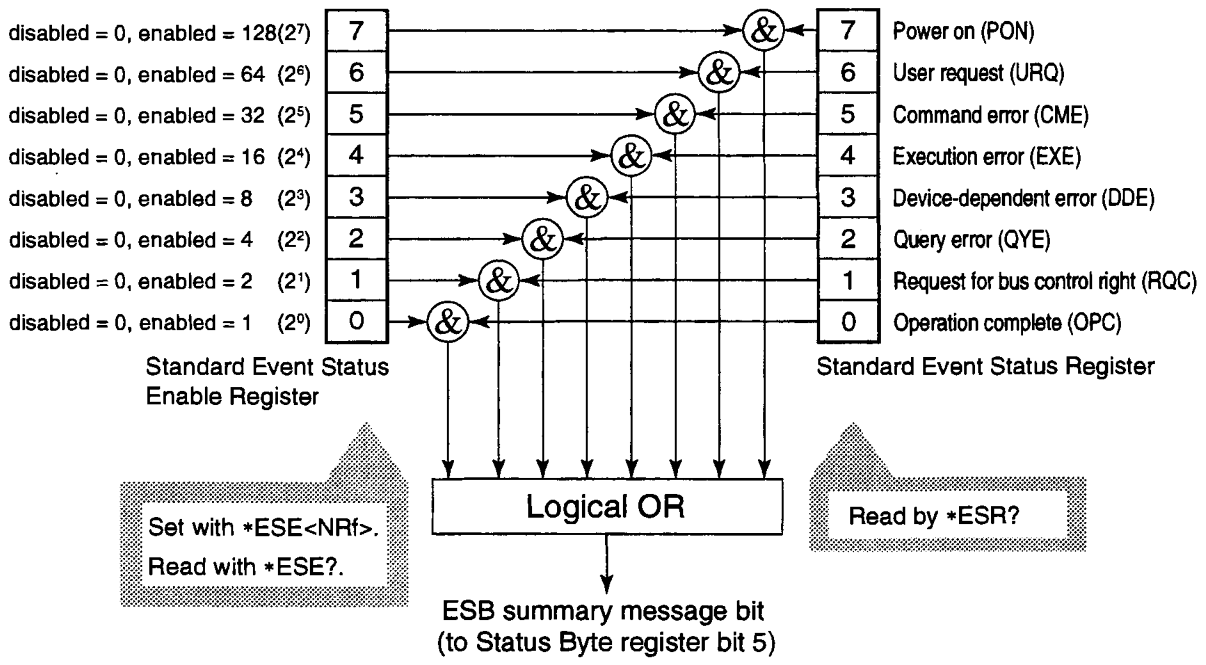
The *SRE common command can clear the SRE register by setting the value of the <DECIMAL NUMERIC PROGRAM DATA> element to 0. Clearing this register prevents an rsv local message from being sent with status information, and a service request is not reported.

When the power is turned on, the power-on status clear flag is true. Since the *PSC command is not supported in this unit, the SRE register is cleared at power-on if the clear instruction is not inhibited.

8.4 Standard Event Status Register

8.4.1 Bit definition in standard event status register

The standard event status register is necessary for all IEEE488.2 devices. The figure below shows the operation of the standard event status register model. This operation is the same as the one described above; therefore, this section explains the IEEE488.2 definition about the meaning of each bit in the standard event status register.


disabled = 0, enabled = 128(2^7)
disabled = 0, enabled = 64 (2^6)
disabled = 0, enabled = 32 (2^5)
disabled = 0, enabled = 16 (2^4)
disabled = 0, enabled = 8 (2^3)
disabled = 0, enabled = 4 (2^2)
disabled = 0, enabled = 2 (2^1)
disabled = 0, enabled = 1 (2^0)
7 6 5 4 3 2 1 0
Standard Event Status Enable Register
Power on (PON)
User request (URQ)
Command error (CME)
Execution error (EXE)
Device-dependent error (DDE)
Query error (QYE)
Request for bus control right (RQC)
Operation complete (OPC)
Standard Event Status Register
Logical OR
Set with *ESE<NRf>.
Read with *ESE?.
Read by *ESR?
ESB summary message bit
(to Status Byte register bit 5)

Bit.	Event name	Explanation
7	PON-Power on	The power state was changed from OFF to ON. Set to 1 at power-on.
6	URQ-User Request	Requests a local control (rtl). This bit is set regardless of whether the device is in the remote or local mode. Set to 1 at execution of a control command.
5	CME-Command Error	An invalid program message or misspelled command was received; or a GET command was received in a program message.
4	EXE-Execution Error	An unexecutable program message (parameter error) of which syntax was correct was received.
3	DDE-Device-dependent Error	An error occurred for a reason other than CME, EXE, or QYE. For example, this error occurs when wavelength Cal was executed with heat-up 100% or fewer or when a frequency was set in the wavelength mode.
2	QYE-Query Error	An attempt was made to read data from the output queue although no data existed in the output queue. Otherwise, data of the output queue was deleted for any cause, e.g., overflow. Always set to 0 because this bit is not used in this unit.
1	RQC-Request Control	Requests that the device itself functions as an active controller. Always set to 0 because this bit is not used in this unit.
0	OPC-Operation Complete	The device is ready to receive a new instruction after terminating the pending specified operation. Always set to 0 because this bit is not used in this unit.

8.4.2 Detailed query errors

No.	Note	Explanation
1	Incomplete program message	If the device receives MTA from the controller before a program message terminator while receiving a program message, it abandons the received incomplete program message and waits for the next one. To abandon such a program message, the device clears the input-output buffer, reports the query error to the status report unit, and sets the query error bit in bit 2 of the standard status register.
2	Response message output stop	If the device receives MLA from the controller before a response message terminator while sending a response message, it automatically stops the output of the response message and waits for the next one. To stop the output of a response message, the device clears the output buffer, reports the query error to the status report unit, and sets the query error bit in bit 2 of the standard status register.
3	Skipping a response message and sending the next program message	When the device could not output a response message because the controller sent the next program message following one including a query message, it abandons the response message and waits for the next program message. In the same way as for item 2, the device reports the query error to the status report unit.
4	Output queue overflow	During execution of a program message containing multiple query messages, too many response messages may be reported as the output queue (256 bytes) overflows. Like this, when query messages are kept reading to the output queue although it is full and the device must output a response message to each query message, the output queue enters the overflow state. If the output queue overflows, the device clears the output queue and resets the response message generation unit. The query error bit is set into bit 2 in the standard event status register of the status report unit.

8.4.3 Reading, writing, and clearing standard event status register

Reading	This register is destructively read by the *ESR? common query; in other words, this register is cleared after being read. A response message is returned with NR1 obtained by superimposing the event bit with binary data and converting the result to a decimal value.
Writing	This register cannot be written externally, excluding clearing.
Clear	This register is cleared only when: (1) the *CLS command was received; (2) the power was turned on if the power-on status clear flag was true. The device executing the power-on sequence first clears the standard event status register, and then records events (e.g., PON event bit setting) generated during this sequence. (3) events are read to the *ESR? query command.

8.4.4 Reading, writing, and clearing standard event status enable register

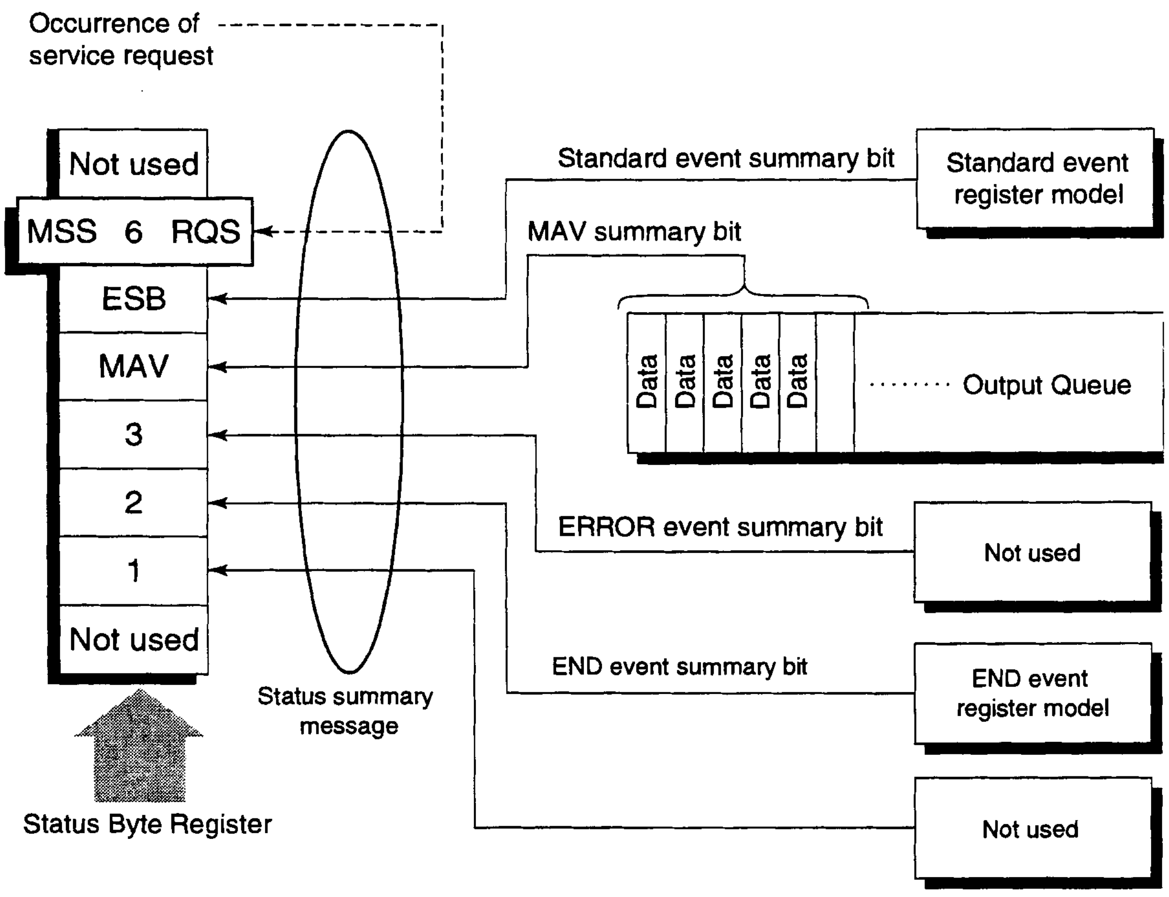
Reading	This register is non-destructively read by the *ESR? common query; in other words, this register is not cleared after being read. A response message is returned with NR1 obtained by superimposing the event bit with binary data and converting the result to a decimal value.
Writing	This register is written by the *ESS common command. Bits 0 to 8 in the register are superimposed to 1, 2, 4, 8, 16, 32, 64, and 128, respectively. Write data is sent with <decimal numeric program data> obtained by totalizing the required bit digit values of those bits.
Clear	This register is cleared only when (1) the *ESE command having data value 0 was received; (2) the power was turned on as the power-on status clear flag was true or the *PSC command was not prepared. The standard event status enable register is not affected by the items below. (1) Transition of IEEE488.1 device clear function state (2) Reception of *RST common command (3) Reception of *CLS common command

8.5 Extension Event Status Register

Each register model in the status byte register and standard event status register is necessary for the IEEE488.2 devices.

In IEEE488.2, bits 7, 3, 1, and 0 are not used, and bit 2 is allocated to the END summary bit as a status summary bit supplied from the register model.

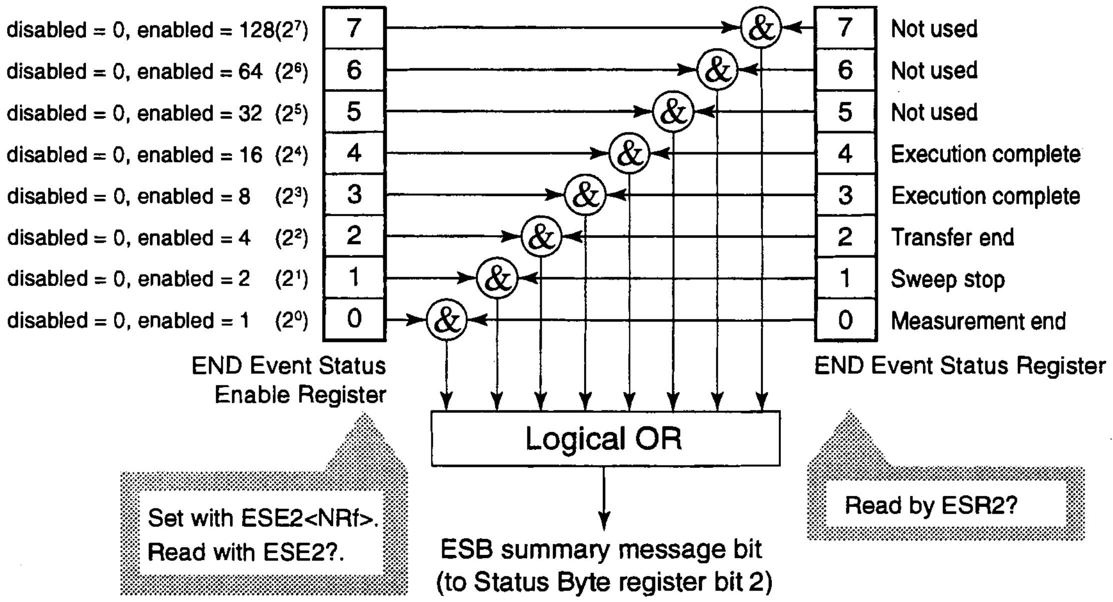
In this unit, as shown below, bits 7, 3, 1, and 0 are not used; bit 2 is allocated to an END summary bit for status summary bit supplied from an extension register model.



The following sections explain how to define, read, write, and clear the bits in the END extension event register model.

8.5.1 Bit definition in END event status register

The figure below shows the operation of the END event status register models, event bit names, and their meanings.


disabled = 0, enabled = 128(2^7)
disabled = 0, enabled = 64 (2^6)
disabled = 0, enabled = 32 (2^5)
disabled = 0, enabled = 16 (2^4)
disabled = 0, enabled = 8 (2^3)
disabled = 0, enabled = 4 (2^2)
disabled = 0, enabled = 2 (2^1)
disabled = 0, enabled = 1 (2^0)
7 6 5 4 3 2 1 0
END Event Status Enable Register
Logical OR
Set with ESE2<NRf>. Read with ESE2?.
ESB summary message bit (to Status Byte register bit 2)
7 Not used
6 Not used
5 Not used
4 Execution complete
3 Execution complete
2 Transfer end
1 Sweep stop
0 Measurement end
END Event Status Register
Read by ESR2?

Bit	Event name	Explanation
7	Not used	
6	Not used	
5	Not used	
4	End of execution	End of *RST
3	End of execution	End of wavelength cal or end of auto alignment
2	End of execution	End of level setting
1	End of execution	End of wavelength setting
0	End of sweeping	End of one sweeping

8.5.2 Reading, writing, and clearing extension event status register

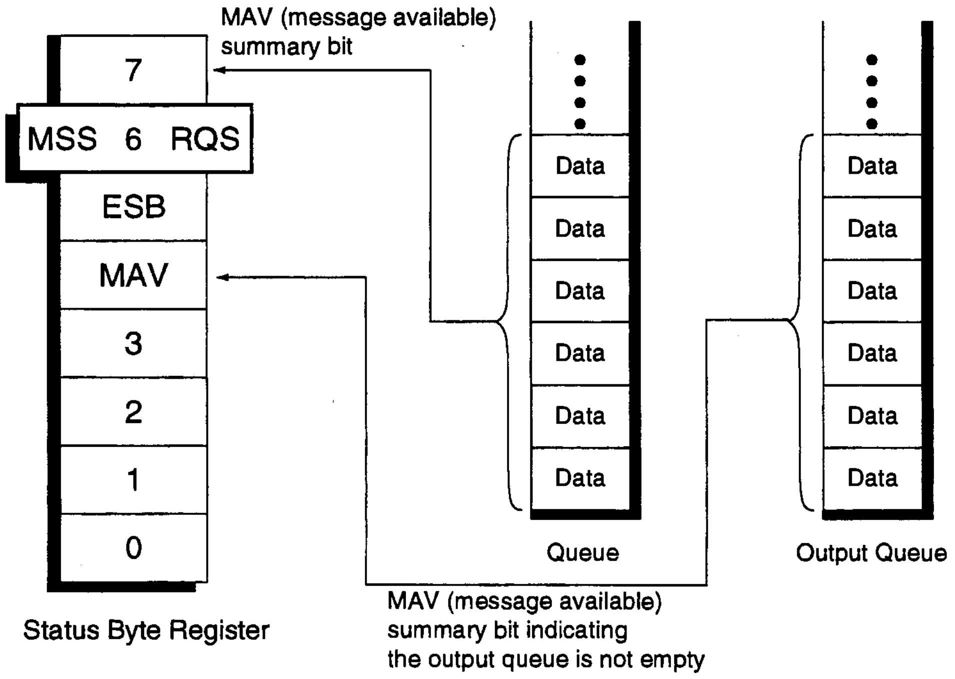
Reading	This register is destructively read by a query; in other words, this register is cleared after being read. The END event status register is read by the ESR2? query. Its value is returned with NR1 obtained by superimposing the event bit with binary data and converting the result to a decimal value.
Writing	This register cannot be written externally, excluding clearing.
Clear	This register is cleared only when: (1) the *CLS command was received; (2) the power is turned on if the power-on status clear flag is true. (3) events are read to the *ESR? query command.

8.5.3 Reading, writing, and clearing extension event status enable register

Reading	This register is non-destructively read by the *ESR? common query; in other words, this register is not cleared after being read. The END event status register is read by the ESE2? query. Its value is returned with NR1 obtained by superimposing the event bit with binary data and converting the result to a decimal value.
Writing	The END event status register is written by the *ESE2 common command. Bits 0 to 8 in the register are superimposed to 1, 2, 4, 8, 16, 32, 64, and 128, respectively. Write data is sent with <decimal numeric program data> obtained by totalizing the required bit digit values of those bits.
Clear	This register is cleared only when: (1) the *ESE2 command having data value 0 was received for the END event status register: (2) the power was turned on as the power-on status clear flag was true or the *PSC command was not prepared. The extension event status enable register is not affected by the items below. (1) Transition of IEEE488.1 device clear function state (2) Reception of *RST common command (3) Reception of *CLS common command .

8.6 Queue Model

A queue model in the status data structure is shown on the right of the figure below. The queue, that is a data structure containing information lists arranged in sequence, supplies a method of reporting the sequential status and other information. Such information in the queue is summary-indicated in the summary message. The contents of the queue are read by the hand-shake function when the device is in the talker active state (TACS).



A queue that outputs an MAV summary message to bit 4 in the status byte register is called an output queue, and it is necessary for this unit. A queue that can output an MAV summary message to one of bits 0 to 3 and 7 in the status byte register is singly called a queue, and it is optional. Bits 0 to 3 and 7 in the status byte register can be connected to a summary message sent from a register model; therefore, the number of summary message types varies depending on the device.

Bit 7 in the status byte register is provided for the summary message bit sent from a queue. However, if processing is enough only with the output queue, the queue is not used especially; bit 7 in the status byte register is not used in this unit.

The table below compares the output queue with a general queue.

Comparison of output queue with general queue

Item	Output queue	General queue
Data input-output method	FIFO system	Need not necessarily conform to the FIFO system.
Reading	Read only in the basis of the IEEE488.2 message exchange protocol. The type of the read response message unit is determined by the query.	Read by a query command particular to the device. The types of read response message units must be the same.
Writing	A program message element is not written directly. Transferred to the system interface only through the IEEE488.2 message exchange protocol.	A program message element is not written directly. Indicates the encoded device information.
Summary message	Set to true (1) when the output queue is not idle and false (0) when it is idle. The MAV summary message is used to enable a synchronous information exchange between the controller and device.	Set to true (1) when the queue is not idle and false (0) when it is idle.
Clear	Cleared when one of the following events occurred: (1) All items in the queue were read. (2) The DCL bus command was received to initialize message exchange. (3) PON was set to true by turning the power on. (4) UNTERMINATED or INTERRUPTED operation.	Cleared when one of the following events occurred: (1) All items in the queue were read. (2) The *CLS command was received. (3) Other events particular to the device

Section 9
Detailed Explanation of Device Messages

An auxiliary command is described in the detailed explanation of each device message. Each function is also operable with the header described in the auxiliary command.
In the auxiliary command, the contents enclosed with brackets [ ] and lowercase characters can be omitted.

9.1 AMEX (Amplitude Modulation External)

■ Function

Sets the external modulation.

Header	Program	Query	Response
AMEX	AMEX	none	none

■ Auxiliary command

[: SOURce] : AM : EXTernal

9.2 AMIN (Amplitude Modulation Internal)

■ Function

Sets the internal modulation and internal modulation frequency.

Header	Program	Query	Response
AMIN	AMIN p	AMIN?	n

■ • Value of p

p indicates an internal modulation frequency with a unit (Hz, kHz, MHz, GHz, or THz).
The range of p is 0.2 to 20.0 kHz.

• Value of n

n indicates an internal modulation frequency and outputs a numeric value in Hz units.
The range of n is 2.00000000E+002 to 2.00000000E+004.

■ Auxiliary command

[: SOURce] : AM : INTernal : FREQuency
[: SOURce] : AM : INTernal : FREQuency?

9.3 AMOF (Amplitude Modulation Off)

■ Function

Sets the modulation to off.

Header	Program	Query	Response
AMOF	AMOF	none	none

■ Auxiliary command

[: SOURce] : AM : OFF

9.4 AMST (Amplitude Modulation Status)

■ Function

Reads the modulation setting status.

Header	Program	Query	Response
AMST	none	AMST?	n

■ Value of n

n indicates the modulation setting status and outputs 0 to 2.

n = 0 : Modulation off
  = 1 : Modulation Int
  = 2 : Modulation Ext

■ Auxiliary command

[: SOURce] : AM : STATe
[: SOURce] : AM : STATe?

9.5 CAL (Calibration Execute)

■ Function

Executes wavelength Cal.

After the execution of wavelength Cal was completed, bit 3 (execution termination bit) in the extension event status register (ESR2) is set to 1.

Header	Program	Query	Response
CAL	CAL p	CAL?	n

■• Value of p

p =start
  =stop

• Value of n

n indicates the wavelength Cal execution status and outputs 0 to 2.

n = 0 : Normal termination
  = 1 : Executed currently
  = 2 : Abnormal termination

■ Auxiliary command

[: ADVance] : EXECute : CALibration
[: ADVance] : EXECute : CALibration?

9.6 CALF (Calibration Frequency Set)

■ Function

Sets a frequency for wavelength Cal.

Header	Program	Query	Response
CALF	CALF p	CALF?	n

■• Value of p

p indicates a frequency required to execute wavelength Cal and it is set with a unit (Hz, kHz, MHz, GHz, or THz).

The range of p is 189742.0 to 199861.6 GHz.

• Value of n

n indicates a frequency required to execute wavelength Cal and outputs a numeric value in Hz units.

The range of n is 1.89742000E+014 to 1.99861600E+014.

■ Auxiliary command

[: ADVance] : EXECute : CALibration : FREQuency
[: ADVance] : EXECute : CALibration : FREQuency?

9.7 CALW (Calibration Wavelength Set)

■ Function

Sets a wavelength required to execute wavelength Cal.

Header	Program	Query	Response
CALW	CALW p	CALW?	n

■ Value of p

p indicates a wavelength required to execute wavelength Cal and it is set with a unit (m, mm, μ m, or pm).
The range of p is 1.5 to 1.58 μ m (1500 to 1580 nm).

• Value of n

n indicates a wavelength required to execute wavelength Cal and outputs a numeric value in m units.
The range of n is 1.500000000E-006 to 1.580000000-006.

■ Auxiliary command

[: ADVance] : EXECute : CALibration : WAVElength
[: ADVance] : EXECute : CALibration : WAVElength?

9.8 COH (Coherency Control)

■ Function

Sets the coherency control state to on or off.

Header	Program	Query	Response
COH	COH s	COH?	n

■• Value of s

s = 0 or OFF : Coherency control OFF
= 1 or ON : Coherency control ON

• Value of n

n outputs the coherency control setting statue with 0 or 1.
n = 0 : Coherency control OFF
= 1 : Coherency control ON

■ Auxiliary command

[: SOURce] : COH
[: SOURce] : COH?

9.9 CONT (Sweep Continue)

■ Function

Restarts the currently stopped sweeping.

Header	Program	Query	Response
CONT	CONT	none	none

■ Auxiliary command

[: EXECute] : SWeep : CONTinue

9.10 DENA (Display Enable)

■ Function

Sets the display state to on or off.

Header	Program	Query	Response
DENA	DENA s	DENA?	n

■• Value of s

s = 0 or OFF : Display OFF
= 1 or ON : Display ON

• Value of n

n outputs the display on or off setting status with 0 or 1.

n = 0 : Display OFF
= 1 : Display ON

■ Auxiliary command

: DISPlay : ENABle
: DISPlay : ENABle?

9.11 DREV (Display Reverse)

■ Function

Sets the reverse display state to on or off.

Header	Program	Query	Response
DREV	DREV s	DREV?	n

■• Value of s

s = 0 or OFF : Reverse display OFF
=1 or ON : Reverse display ON

• Value of n

n indicates the reverse display on or off setting status with 0 or 1.

n = 0 : Reverse display OFF
= 1 : Reverse display ON

■ Auxiliary command

[: ADVance] : DISPlay : REVerse
[: ADVance] : DISPlay : REVerse?

9.12 DWEL (Dwell Time)

■ Function

Sets the laser output time (Dwell Time).

Header	Program	Query	Response
DWEL	DWEL p	DWEL?	n

■• Value of p

p indicates a Dwell Time with a unit (second).
The range of p is 0.01 to 100 s.

■• Value of n

n indicates a Dwell Time and outputs a numeric value in seconds.
The range of n is 1.00000000E-002 to 1.00000000E+002.

■ Auxiliary command

[: SOURce] : TIME : DWEL1
[: SOURce] : TIME : DWEL1?

9.13 ERR (Error)

■ Function

Reads the number of an error detected during GPIB operation.

Header	Program	Query	Response
ERR	none	ERR?	n

■ Value of n

n indicates an error code and outputs a positive integral value of four digits.
The error number is set when the ESB bit (bit 5) in the status byte register is on and the command error bit (bit 5), execution error bit (bit 4), or device dependent error bit (bit 3) in the standard event status register (ESR) is on.

■ Auxiliary command

: SYSTem : ERRor?

9.14 ESE2 (Extended Event Status Enable Register2)

■ Function

Sets and reads the enable register of extension event status register.

Header	Program	Query	Response
ESE2	ESE2 n	ESE2?	n

■ Value of n

n indicates a positive integer from 0 to 255. At n = 0, the disabled state is set.

9.15 ESR2 (Extended Event Stats Register2)

■ Function

Reads information of extension event status register 2 generated by GPIB operation.

Header	Program	Query	Response
ESR2	none	ESR2?	n

■ Value of n

n outputs an integer from 0 to 255.

9.16 FCNT (Center Frequency)

■ Function

Sets a center frequency.

This message is usable in the CE, sweep, and 1-step tuning modes.

Header	Program	Query	Response
FCNT	FCNT p	FCNT?	n

■ • Value of p

p indicates a center frequency with a unit (Hz, kHz, MHz, GHz, or THz).

The range of p is 189742.0 to 199861.6 GHz.

This range varies depending on the device.

• Value of n

n indicates a center frequency and outputs a numeric value in Hz.

The range of n is 1.89742000E+014 to 1.99861600E+014.

■ Auxiliary command

[: SOURce] : WAVElength : FREQuency

[: SOURce] : WAVElength : FREQuency?

9.17 FOFS (Frequency Offset)

■ Function

Sets an offset frequency.

The output laser frequency (wavelength) is shifted by inputting the offset frequency.

Header	Program	Query	Response
FOFS	FOFS p	FOFS?	n

■ • Value of p

p indicates an offset frequency with a unit (Hz, kHz, MHz, GHz, or THz).

The range of p is -50 to 50GHz.

• Value of n

n indicates a center frequency and outputs a numeric value in Hz.

The range of n is -500000000E+010 to 5.00000000E+010.

■ Auxiliary command

: FREQuency : OFfSet

: FREQuency : OFfSet?

9.18 FSPN (Span Frequency)

■ Function

Sets a span frequency.

Header	Program	Query	Response
FSPN	FSPN p	FSPN?	n

■• Value of p

p indicates a span frequency with a unit (Hz, kHz, MHz, GHz, or THz).
The range of p is 0.2 to 10000.0 GHz.
This range varies depending on the device.

• Value of n

n indicates a span frequency and outputs a numeric value in Hz.
The range of n is 2.00000000E+080 to 1.00000000E+013.

■ Auxiliary command

[: SOURce] : FREQuency : SPAN
[: SOURce] : FREQuency : SPAN?

9.19 FSTA (Start Frequency)

■ Function

Sets a start frequency.

Header	Program	Query	Response
FSTA	FSTA p	FSTA?	n

■• Value of p

p indicates a start frequency with a unit (Hz, kHz, MHz, GHz, or THz).
The range of p is 189742.0 to 199861.6 GHz.
This range varies depending on the device.

■• Value of n

n indicates a start frequency and outputs a numeric value in Hz.
The range of n is 1.89742000E+014 to 1.99861600E+014.

■ Auxiliary command

[: SOURce] : FREQuency : STARt
[: SOURce] : FREQuency : STARt?

9.20 FSTO (Stop Frequency)

■ Function

Sets a stop frequency.

Header	Program	Query	Response
FSTO	FSTO p	FSTO?	n

■ • Value of p

p indicates a stop frequency with a unit (Hz, kHz, MHz, GHz, or THz).
The range of p is 189742.0 to 199861.6 GHz.
This range varies depending on the device.

■ • Value of n

n indicates a stop frequency and outputs a numeric value in Hz.
The range of n is 1.89742000E+014 to 1.99861600E+014.

■ Auxiliary command

[: SOURce] : FREQuency : STOP
[: SOURce] : FREQuency : STOP?

9.21 FSTP (Step Frequency)

■ Function

Sets a sweeping step frequency.

Header	Program	Query	Response
FSTP	FSTP p	FSTP?	n

■ • Value of p

p indicates a sweeping step frequency with a unit (Hz, kHz, MHz, GHz, or THz).
The range of p is 0,1 to 1000.0 GHz.
This range varies depending on the device.

• Value of n

n indicates a span frequency and outputs a numeric value in Hz.
The range of n is 1.00000000E+008 to 1.00000000E+013.

■ Auxiliary command

[: SOURce] : FREQuency : STEP
[: SOURce] : FREQuency : STEP?

9.22 MADV (Advance Mode Set)

■ Function

Sets the advance mode.

Header	Program	Query	Response
MADV	MADV	none	none

■ Auxiliary command

[: SOURce] : MODE : ADVance

9.23 MCW (CW Mode Set)

■ Function

Sets the CW mode.

Header	Program	Query	Response
MCW	MCW	none	none

■ Auxiliary command

[: SOURce] : MODE : CW

9.24 MONE (1Step Tuning Mode Set)

■ Function

Sets the 1-step tuning mode.

Header	Program	Query	Response
MONE	MONE	none	none

■ Auxiliary command

[: ADVance] [: SOURce] : MODE : ONEStep

9.25 MOVE (Move)

■ Function

Reads the output laser wavelength setting termination state.

Header	Program	Query	Response
MOVE	none	MOVE?	n

■ Value of n

n outputs the output laser wavelength setting termination state.

n = 0	: Wavelength setting terminated
=1	: Wavelength setting currently

■ Auxiliary command

[: STATu] : CONDition : MOTor?

9.26 MST (Mode Status)

■ Function

Reads the currently specified measurement mode.

Header	Program	Query	Response
MST	none	MST?	n

■ Value of n

n outputs the currently specified measurement mode.

n = 0	: CW mode
1	: Sweep mode
2	: 1-step tuning mode
3	: Advance mode

■ Auxiliary command

[: SOURce] : MODE : STATus?

9.27 MSWP (Sweep Mode Set)

■ Function

Sets the sweep mode.

Header	Program	Query	Response
MSWP	MSWP	none	none

■ Auxiliary command

[: SOURce] : MODE : SWeep

9.28 OUTC (Optical Output Condition)

■ Function

Reads the safety function state (laser output control key, fiber connection state, and remote interlock connector) about laser output.

Header	Program	Query	Response
OUTC	none	OUTC?	b

■ Value of b

b indicates the laser output control key, fiber connection state, and remote interlock connector state. It outputs with the sum of the values obtained by superimposing bits 0 to 2 with 20 to 22.

b = b0 + b1 + b2

b0	: Laser output control key (b0 = 1 at ON, b0 = 0 at OFF)
b1	: Fiber connection state (b1 = 2 at connection, b0 = 0 at non-connection)
b2	: Remote interlock connector state (b2 = 4 at short, b2 = 0 at open)

■ Auxiliary command

[: OUTPut] : CONDition?

9.29 OUTF (Output Frequency Read)

■ Function

Reads the current output laser frequency.

Header	Program	Query	Response
OUTF	none	OUTF?	n

■ Value of n

n indicates the current output laser frequency and outputs a numeric value in Hz.
The range of n is 1.89742000E+014 to 1.99861600+014.

■ Auxiliary command

[: SOURce] : OUTput : FREQuency?

9.30 OUTP (Optical Output)

■ Function

Sets the laser output to on or off.
When the laser output is set to on with this command, if the laser output control key is off; the fiber is not connected; or the remote interlock connector is open, no laser is output.

Header	Program	Query	Response
OUTP	OUTP s	OUTP?	n

■• Value of s

s	= 0 or OFF	: Laser output OFF
	= 1 or ON	: Laser output ON

• Value of n

n outputs the laser output on or off state with 0 or 1.

n	= 0	: Laser output OFF
	= 1	: Laser output ON

■ Auxiliary command

: OUTPut [: STATe]
: OUTPut [: STATe] ?

9.31 OUTW (Output Wavelength Read)

■ Function

Reads the current output laser wavelength.

Header	Program	Query	Response
OUTW	none	OUTW?	n

■ Value of n

n indicates the current output laser wavelength and outputs a numeric value in m.
The range of n is 1.50000000E-006 to 1.58000000-006.

■ Auxiliary command

[: SOURce] : OUTput : WAVElength?

9.32 PAUS (Sweep Pause)

■ Function

Temporarily stops the wavelength (frequency) sweeping.

Header	Program	Query	Response
PAUS	PAUS	none	none

■ Auxiliary command

[: EXECute] : SWeep : PAUSe

9.33 POW (Power)

■ Function

Sets a laser output level.

Header	Program	Query	Response
POW	POW p	POW?	n

■• Value of p

p indicates a laser output level with a unit (dBm, w, mw, µ w, nw, or pw).

The range of p is –20 dBm or more and the maximum laser output level varies depending on the output wavelength and device.

• Value of n

n indicates a laser output level and outputs a numeric value in dBm for log display and in W for linear display.

The range of n is –200000000E+001 to Max. power in dBm and 1.00000000E-005 to Max. power in W.

■ Auxiliary command

[: SOURce] : POWer [: LEVel] [: IMMediate] [: AMPlitude]

[: SOURce] : POWer [: LEVel] [: IMMediate] [: AMPlitude] ?

9.34 POWM (Maximum Power)

■ Function

Sets the maximum laser output level.

Header	Program	Query	Response
POWM	POWM	POWM?	n

■ Value of n

n indicates the maximum laser output level and outputs a numeric value in dBm for log display and in W for linear display.

■ Auxiliary command

[: SOURce] : POWer : MAXimum

[: SOURce] : POWer : MAXimum?

9.35 POWU (Unit of Power)

■ Function

Sets a laser output level unit.

Header	Program	Query	Response
POWU	POWU p	POWU?	n

■ • Value of p

p indicates a unit of the set laser output level.

P = dbm : dBm
= mw : mW
= uw : μW

• Value of n

n indicates a unit of t he set laser output level and outputs a numeric value from 0 to 2.

n = 0 : dBm
= 1 : mW
= 2 : μW

■ Auxiliary command

[: SOURce] : POWer : UNIT
[: SOURce] : POWer : UNIT?

9.36 RPT (Sweep Repeat)

■ Function

Sets the wavelength (frequency) sweeping to the repeat sweeping.
After one sweeping is completed, bit 0 (execution termination bit) in the extension event status register (ESR2) is set to 1.

Header	Program	Query	Response
RPT	RPT	none	none

■ Auxiliary command

[: EXECute] : SWeep : RePeaT

9.37 SETM (Wave/Freq Set)

■ Function

Switches the wavelength and frequency modes.

Header	Program	Query	Response
SETM	SETM p	SETM?	n

■ • Value of p

p indicates the wavelength of frequency mode.

p	= wave	: Wavelength mode
	= freq	: Frequency mode

• Value of n

n indicates a wavelength of frequency mode and outputs a numeric value 0 or 1.

n	= 0	: Wavelength mode
	= 1	: Frequency mode

■ Auxiliary command

[: ADVance] : SET : MODE

[: ADVance] : SET : MODE?

9.38 SNGL (Sweep Single)

■ Function

Starts the single wavelength (frequency) sweeping.

After one sweeping is completed, bit 0 (execution termination bit) in the extension event status register (ESR2) is set to 1.

Header	Program	Query	Response
SNGL	SNGL	none	none

■ Auxiliary command

[: EXECute] : SWeep : SiNGLe

9.39 SWPT (Sweep Speed)

■ Function

Sets the sweeping speed in the 1-step tuning mode.

Header	Program	Query	Response
SWPT	SWPT p	SWPT?	n

■ Value of n

n indicates a sweeping speed and outputs a numeric value from 1 to 5.

n	= 1	: Speed 1
	= 2	: Speed 1/2
	= 3	: Speed 1/4
	= 4	: Speed 1/8
	= 5	: Speed 1/16

■ Auxiliary command

[: SOURce] : TIME : SWeeP
[: SOURce] : TIME : SWeeP?

9.40 SWST (Sweep Status)

■ Function

Reads the wavelength (frequency) sweeping state.

Header	Program	Query	Response
SWST	none	SWST?	n

■ Value of n

n indicates the wavelength (frequency) sweeping state and outputs a numeric value from 0 to 2.

n	= 0	: Stopped currently
	= 1	: Repeat sweeping currently
	= 2	: Single sweeping currently

■ Auxiliary command

[: EXECute] : SWeep : STATus?

9.41 TEMP (Heat Up)

■ Function

Reads the heat-up rate.

Header	Program	Query	Response
TEMP	none	TEMP?	n

■ Value of P

P indicates a heat-up rate with a unit (0 to 100 %).

■ Auxiliary command

[: STATus] : CONDition : TEMPerature?

9.42 WCNT (Center Wavelength)

■ Function

Sets a center wavelength.

This command is available in the CW, sweep, and 1-step tuning modes.

Header	Program	Query	Response
WCNT	WCNT p	WCNT?	n

■• Value of p

p indicates a center wavelength with a unit (m, mm, µ m, nm, or pm).

The range of p is 1.5 to 1.58 µ m (1500 to 1580 nm).

This range varies depending on the device.

• Value of n

n indicates a center wavelength and outputs a numeric value in m.

The range of n is 1.50000000E-006 to 1.58000000-006.

■ Auxiliary command

[: SOURce] : WAVElength [: CW | : FIXED]

[: SOURce] : WAVElength [: CW | : FIXED] ?

9.43 WSPN (Span Wavelength)

■ Function

Sets a start wavelength.

This command is not available in the CW mode.

Header	Program	Query	Response
WSPN	WSPN p	WSPN?	n

■• Value of p

p indicates a span wavelength with a unit (m, mm, μ m, nm, or pm).

The range of p is 2 to 80000 pm (0.002 to 80.000 nm).

This range varies depending on the device.

• Value of n

n indicates a start wavelength and outputs a numeric value in m.

The range of n is 2.00000000E-012 to 8.00000000-008.

■ Auxiliary command

[: SOURce] : WAVElength : SPAN

[: SOURce] : WAVElength : SPAN?

9.44 WSTA (Start Wavelength)

■ Function

Sets a start wavelength.

This command is not available in the CW mode.

Header	Program	Query	Response
WSTA	WSTA p	WSTA?	n

■• Value of p

p indicates a start wavelength with a unit (m, mm, μ m, nm, or pm).

The range of p is 1.5 to 1.58 μ m (1500 to 1580 μ m).

This range varies depending on the device.

• Value of n

n indicates a start wavelength and outputs a numeric value in m.

The range of n is 1.50000000E-006 to 1.58000000-006.

■ Auxiliary command

[: SOURce] : WAVElength : STARt

[: SOURce] : WAVElength : STARt?

9.45 WSTO (Stop Wavelength)

■ Function

Sets a stop wavelength. This command is not available in the CW mode.

Header	Program	Query	Response
WSTO	WSTO p	WSTO?	n

■• Value of p

p indicates a stop wavelength with a unit (m, mm, μ m, nm, or pm).
The range of p is 1.5 to 1.58 μ m (1500 to 1580 nm).
This range varies depending on the device.

• Value of n

n indicates a stop wavelength and outputs a numeric value in m.
The range of n is 1.50000000E-006 to 1.58000000-006.

■ Auxiliary command

[: SOURce] : WAVElength : STOP
[: SOURce] : WAVElength : STOP?

9.46 WSTP (Step Wavelength)

■ Function

Sets a sweeping step wavelength.
This command is not available in the CW mode.

Header	Program	Query	Response
WSTP	WSTP p	WSTP?	n

■• Value of p

p indicates a sweeping step wavelength with a unit (m, mm, μ m, nm, or pm).
The range of p is 1 to 80000 pm (0.002 to 80.000 nm).
This range varies depending on the device.

• Value of n

n indicates a start wavelength and outputs a numeric value in m.
The range of n is 1.00000000E-012 to 8.00000000-008.

■ Auxiliary command

[: SOURce] : WAVElength : STEP
[: SOURce] : WAVElength : STEP?

9.47 : SET : NOP (Not Power Off)

■ Function

Sets the laser non-cut-off function to on or off. The laser must ordinarily be cut off to change a wavelength or frequency. A function for not cutting of the laser is the laser non-cut-off function.

Header	Program	Query	Response
: SET : NOP	: SET : NOP s	: SET : NOP?	n

■• Value of s

s	= 0 or OFF	: Laser non-cut-off function OFF (Laser is turned off momentarily.)
	= 1 or ON	: Laser non-cut-off function ON (Laser is always output.)

• Value of n

n indicates whether the laser non-cut-off function is on or off with 0 or 1.

n	= 0	: Laser output OFF
	= 1	: Laser output ON

■ Auxiliary command

[: ADVance] : SET : NOPoff
[: ADVance] : SET : NOPoff?

9.48 XALN (Auto Alignment)

■ Function

Executes an auto alignment.
After the auto alignment is executed, bit 3 (execution termination bit) in the extension event status register (ESR2) is set to 1.

Header	Program	Query	Response
XALN	XALN p	XALN?	n

■• Value of p

p	= init	: Sets the positional data for alignment to the default.
	= start	: Executes the auto alignment.
	= stop	: Forcibly terminates the auto alignment.

• Value of n

n	= 0	: Normal termination
	= 1	: Executed currently
	= 2	: Abnormal termination

■ Auxiliary command

[: ADVance] : EXECute : ALIGNment
[: ADVance] : EXECute : ALIGNment?

Section 10 Program Generation Example

10.1 Notes on Program Generation

To generate a remote control program, note the following points:

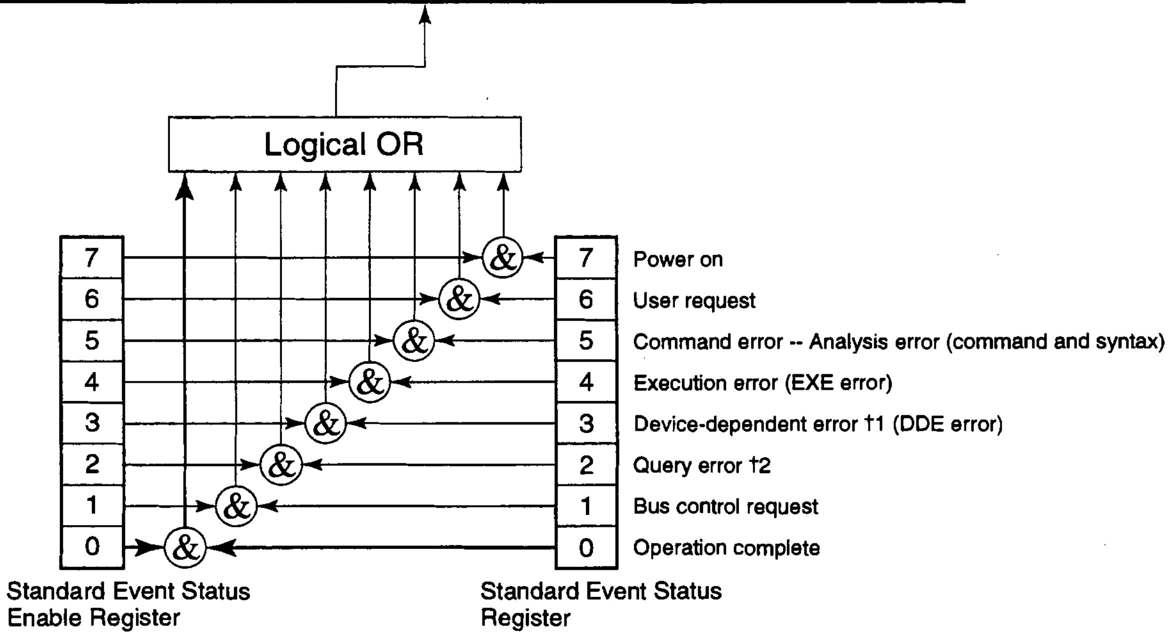
No.	Note	Explanation
1	Be sure to initialize each device.	In many cases, the state of each device is not appropriate to operate its panel and execute another program. Be sure to initialize each device before starting to use it under the predetermined conditions. Before operation, execute the following processing: (1) Initialization of interface function (IFC@) (2) Initialization of device message exchange function (DCL@) (3) Initialization of device functions (*RST) When the RS-232C interface is used, execute only item (3).
2	Set the device remote state in RWLS (remote with lockout state).	In an ordinary remote mode, if the LOCAL key is pressed erroneously, the device is placed into the local mode. In this case, if a panel key is pressed, the automatic measurement is not performed normally, and measurement data may not be reliable. Use LLO@ to place the device into the local lockout state so that the device does not return to the local mode.
3	Do not send a command related to the device other than the READ@ statement just after a query.	When a command other than the READ@ statement is sent to the controller before the query result is read, if MLA is received, the output buffer is cleared, and the response message is erased. Therefore, the READ@ statement must be written following the query.
4	Do not perform protocol exception processing.	Provide an exception processing unit in the program to process expected exceptions and prevent processing from being stopped due to an error.
5	Confirm the interface function (subset) of each device. (GPIB)	When a program is executed for a device without subset, processing does not advance. Be sure to confirm the subset for each device. Also check whether the device conforms to IEEE488.2.
6	Prevent a buffer overflow. (RS-232C)	The RS-232C interface of this unit has a 256-byte data area as an internal receiving buffer. A buffer overflow may occur depending on the processing contents. When remote-controlling the device with the RS-232C interface to prevent an overflow error, do not send a large amount of data (control commands) at one time. As a recommended method, first send a command sequence; output the *OPC? command; wait for a response; then send the next command.

Appendix A Error Messages

This appendix lists the error messages summary-indicated in bit 5 of the status byte register. Bit 5 indicates an error message reported to bits 3 to 5 in the standard event status register.

Status byte register

Bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Line	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
Summary message bit	ESB (reserved)	RQS	ESB	MAV	ESB (reserved)	ESB (END)	ESB (reserved)	ESB (reserved)


Logical OR
7
6
5
4
3
2
1
0
&
Power on
User request
Command error -- Analysis error (command and syntax)
Execution error (EXE error)
Device-dependent error †1 (DDE error)
Query error †2
Bus control request
Operation complete
Standard Event Status Enable Register
Standard Event Status Register

† 1 Device error ... A command that cannot be executed at this time

When the execution-time error (EXE error: bit 4) and device error (DDE error: bit 3) in the standard event status register are reported, this unit reports its cause to the operator with an error number and message.
In GPIB, an error number is reported by the ERR? command (see device message).

The following page shows the error numbers and error messages.

Error message list

Error No.	Error message	Status	Output conditions
Key operation errors			
1001	Span or Step Limit		Span is narrower than sweeping step.
1002	Power Limit		In the wavelength mode, a too higher output level was specified for this device.
1003	Power Limit		In the frequency mode, a too higher output level was specified for this device.
1004	Out of Limit		An invalid set value was input.
1005	Can't Find Recall Data		Recall data is not found.
Remote control errors			
2001	Invalid Command	ESE-CME	Remote command error
2002	Invalid Parameter of Command	ESE-EXE	Remote command parameter error
2003	Can't Execute Command	ESE-DDE	A request command cannot be accepted in the current mode.
2004	Can't Execute Command	ESE-DDE	The set command cannot be accepted in the current mode.
2005	Out of Cal Temperature	ESE-DDE	Since the heat-up rate does not reach 100 %, the LD module cannot be executed currently.
System errors			
4001	Out of Order (4001)		A motion control origin detection error occurred.
4002	Out of Order (4002)		A motion control operation error occurred.
4003	Out of Order (4003)		A backup checksum error occurred.
4004	Out of Order (4004)		An ND filter error occurred.

Appendix B Controller GPIB Instruction Comparison Table

Controller / Function	PACKET V	PC9801	IBM-PC	HP9000 series
Output data to device	WRITE @device-number:data	PRINT @listener-address;data	CALL IBWRT( )	OUTPUT device selector;data
Output binary data to device	BIN WRITE @device-number:data	WBYTE command;data		
Assign data input from device to variable	READ @device-number: variable	INPUT @talker-address,listener-addres-s;variable LINE INPUT @talker-address,listener-addres-s;variable	CALL IBRD( )	ENTER device selector;variab
Assign binary data input from device to variable	BIN READ @device-number:variable	RBYTE command;variable		
Initialize interface function	IFC @select-code	ISET IFC	CALL IBSIC( )	ABORT select-code
Turn on REN line	REN @select-code	ISET REN	CALL IBSRE( )	REMOTE Device selector (select-code)
Turn off REN line	LCL @LCL @select-code (sets all devices in local mode) LCL @ device-number (sets only the specified devices as listeners and issues GTL command)	IRESET REN WBYTE &H3F,listener-address, secondary-address,&H01	CALL IBSRE( ) CALL IBLOC( )	LOCAL Device selector (select-code) LOCAL Device selector (select-code + primary-address)
Output interface message and data	COMMAND @select-code :message-character-string [;data]		CALL IBCMD( ) CALL IBCMDA( ) (asynchrono-us)	SEND select-code ;message-list

Appendix B Controller GPIB Instruction Comparison Table

Function \ Controller	PACKET V	PC9801	IBM-PC	HP9000 series
Trigger the specified device	TRG @ device-number	WBYTE &H3F, listener-address, secondary-address,&08;	CALL IBTRG( )	TRIGGER Device selector
Initialize device	DCL @select-code (all devices with corresponding to specified select code) DCL @ device-number (only specified devices)	WBYTE &H3F,&H14;WBYTE &H3F, listener-address, secondary-address, H04;		CLEAR Device selector (select-code) CLEAR Device selector (select-code + primary address)
Disable switching of device from remote to local	LLO @ select-code	WBYTE &H3F,&H11;		LOCAL LOCKOUT
Transfer control right to specified device	RCT @ device-number	WBYTE talker-address,&H09	CALL IBPCT( )	PASS CONTROL
Issue service request	SRQ @ select-code	ISET SRQ	CALL IBRSV( )	REQUEST select-code
Perform serial polling	STATUS @ device-number	POLL	CALL IBRSP( )	SPOLL (Device selector) (Function)
Set terminator code	TERM IS	CMD DELIM	CALL IBEOS( ) CALL IBEOT( )	
Set limits value for timeout check		CMD TIMEOUT	CALL IBTOM( )	

Appendix C Commands Compatible with HP Wavelength Tunable Laser Source (HP8168B/C)

The following device messages of the MG9637A/MG9638A wavelength tunable laser source are the same as of the HP8168B/C wavelength tunable laser source.

● Laser output on or off

```
  : OUTPut[ : STATe]  p
                    p=OFF/ON/0/1
  : OUTPut[ : STATe] ?
                    Response    n=0/1
```

● Laser output level on or off

```
  [ : SOURce] : POWer[ : LEVel][ : IMMediate][ : AMPlitude] p
                    p=<Value+Unit>
  [ : SOURce] : POWer[ : LEVel][ : IMMediate][ : AMPlitude]?
                    Laser output level setting     n=<Value>
```

● Laser output wavelength setting

```
  [ : SOURce] : WAVElength[ : CW| : FIXED] p
                    p=<Value+Unit>
  [ : SOURce] : WAVElength[ : CW| : FIXED]?
                    Response    n=<Value>
```

● Display on or off

```
  : DISPlay : ENABle   p
                    p=OFF/ON/0/1
  : DISPlay : ENABle?
                    Response    n=0/1
```

Appendix D RS-232C Use Example

This appendix shows a remote control program sample using the RS-232C interface. This program is for PACKET V, and the RS-232C interface operates with @17.

```
100   DIM CMD$*255, RESP$*255, RES$*255                   !Declares an array.
110   SET   @17:BVAL("0000111001111101")                  !Sets the RS-232C to 9600 bps.
                                                          StopBit1, Even,  Sets character 8.
120   DO
130   LINE INPUT PROMPT "Enter a command. " :CMD$         !Command entry
140   IF POS(CMD$, "?")<>0 THEN                           A type is determined depending on whether
                                                          the command is a control or query command.
150   LET SHUBETSU=3
160   ELSE
170   LET SHUBETSU=1
180   END IF

190   LET CMD$=CHR$(2)& CHR$(LEN(CMD$))& CHR$(SHUBETSU)& CMD$ & CHR$(3)
200   LET BCC=0                                           !Generates a message format.

210   FOR I=2 TO LEN(CMD$)
220   LET BCC=EXB(BCC, ASC(CMD$(I:I)))                    !Obtains BCC. [EXB: exclusive OR]

230   NEXT I

240   LET CMD$=CMD$ & CHR$(BCC)                           !Adds BCC to the message.

250   WRITE @17:CMD$                                      !Sends the message.

260   BIN READ @17:ACK                                    !Receives ACK or NAK.

270   PRINT "ACK/NAK: "CHR$(ACK)

280   LET RES$= " "
290   LET COUNT=0
300   DO
310   BIN READ @17:A                                      !Waits for EXT.
32    LET RES$=RES$ & CHR$(A)
330   EXIT IF A=3 AND COUNT>=3
340   LET COUNT= COUNT+1
350   LOOP
360   BIN READ @17:A
```

```
370    LET RES$=RES$ & CHR$(A)                          !Reads BCC.
380    PRINT "FORMAT/RESPONCE: "& RES$
390    WRITE @17:CHR$(6)                                !Returns ACK.
400    LOOP

410    END
```

Anritsu Service and Sales offices

America

Anritsu Company (ACUS)
685-A Jarvis Drive Morgan Hill, CA
95037-2809, U.S.A.
Phone: +1-408-776-8300
Fax: +1-408-776-1738

Anritsu Company (ACUS-NJ) Dist.5
10 New Maple Avenue, P.O. Box 836
Pine Brook, NJ 07058-0836, U.S.A.
Phone: +1-973-227-8999
Fax: +1-973-575-0092

Anritsu Company (ACUS-Maryland) Dist.6
19630 Club House Road, #710
Gaithersburg, MD20879, U.S.A.
Phone: +1-301-590-0300
Fax: +1-301-216-2893

Anritsu Company (ACUS-Tx) NARO
1155 East Collins Blvd
Richardson, TX 75081, U.S.A.
Phone: +1-972-644-1777
Fax: +1-972-644-3416

Anritsu Electronics Ltd (ACCA)
5A-245 Matheson Blvd. East, Mississauga,
Ontario, L4Z 3C9, Canada
Phone: +1-905-890-7799
Fax: +1-905-890-2290

Anritsu Electronics Ltd (ACCA)
102-215 Stafford Road West Nepean,
Ontario, K2H 9C1, Canada
Phone: +1-613-828-4090
Fax: +1-613-828-5400

Anritsu Electronics Ltd (ACCA)
200-1405 Trans Canada Highway
Dorval, Quebec H9P 2V9, Canada
Phone: +1-514-421-3737
Fax: +1-514-685-9839

Anritsu Electronics Ltd (ACCA-Calgary)
Deerfoot Atrium, Suite 129, 6715-8th Street
N.E., Calgary, AB, T2E 7H7, Canada
Phone: +1-403-275-9855
Fax: +1-403-275-3609

Anritsu Eletrônica Ltd (ACBR)
Praia de Botafogo, 440-Sala 2401
CEP 22250-040, Rio de Janeiro, RJ, Brasil
Phone: +55-21-5276922
Fax: +55-21-537-1456

Anritsu Eletrônica Ltd (ACBR) Sao Paulo Branch Office
Praca Amadeu Amaral 27,
Primera Andar, Conj. 11, 12, 13, 14
Liberdade, Sao Paulo, Estado de
Sao Paulo, CEP : 01327-010, Brasil
Phone: +55-11-283-2511
Fax: +55-11-288-6940

Data Lab S.R.L
Edif. Ayfra, Pdte. Franco y Ayolas
Asuncion, Paraguay
Phone: +595-21-443-046
Fax: +595-21-441+935

Electro-Impex S.A.
P.O. Box 620-1000, San Jose, Costa Rica
Phone: +506-2-31-5701
Fax: +506-2-31-6531

Electronica 2000, S.A. DE C.V.
Blvd. Adolfo Lopez Mateos No.2016, Col.
Tlacopac, Del. Lopez Mateos, 01010,
Mexico D.F.
Phone: +525-662-8800
Fax: +525-662-5862

Electronic Engineering S.A.
Carretera de Circunvalavion, Sabanilla, Av
Novena, San Jose, Costa Rica
Phone: +506-2-25-8793
Fax: +506-2-25-1286

HDM Elquitecnica Cia. Ltda., Equitronics S.A.
Av. Republica de El Salvador, No.880,
Edificio, Almirante Colon 4to piso,
Quito, Ecuador
Phone: +593-9-704890
Fax: +593-2-48-2627

KRM Ingenieria Saciafs.
Viamonte 377,7° Piso 1053 Buenos Aires,
Argentina
Phone: +54-1-311-4165
Fax: +54-1-311-2297

SI Ltda.
E. Conchy Y Toro 65, Casilla 51888,
Santiago, Chile
Phone: +56-2-696-7534
Fax: +56-2-696-9665

Radiocom S.A.
Carrera 21 No.85-71, Conmutador
6100077, Santafe de Bogota, Columbia
Phone: +57-1-218-2054
Fax: +57-1-610-3272

Radiocomunicaciones cruz C.A.
Avda La Colina QTA. Elison, URB Colina
De Los Caobos, Caracas, Venezuela
Phone: +58-2-793-2322
Fax: +58-2-793-3429

Sakata Ingenieros S.A.
Av. Canaval Moreyra 840, Lima 27, Peru
Phone: +51-1225-7555
Fax: +51-1224-8148

Suministros Industriales S.A.
Casa 102, Calle 68 Este,
San Francisco, Balboa, Panama
Phone: +507-270-2328
Fax: +507-270-2329

Cabonorte S.A.
Colonia 1900, ESC. 603,
Montevideo, Uruguay
Phone: +598-2-430522
Fax: +598-2-418594

Europe

Anritsu Ltd (ACUK)
200 Capability Green, Luton
Bedfordshire, LU1 3LU, United Kingdom
Phone: +44-1582-433200
Fax: +44-1582-731303

Anritsu Ltd (ACUK-Manchester)
Kansas Avenue, Langworthy Park Salford,
Manchester M5 2GL, United Kingdom
Phone: +44-161-873-8041
Fax: +44-161-873-8040

Anritsu Ltd (ACUK-Bristol)
1230 Aztec West, Almondsbury
Bristol BS12 4SG, United Kingdom
Phone: +44-1454-615252
Fax: +44-1454-618017

Anritsu Ltd (ACUK-Livingston)
Unit 1, Knightsridge Industrial Estate
Turnbull Way, Knightsridge Livingston
EH54 8RB, United Kingdom
Phone: +44-1506-436111
Fax: +44-1506-436112

Anritsu GmbH (ACDE)
Grafenberger Allee 54-56
D-40237 Düsseldorf 1, Germany
Phone: +49-211-96855-0
Fax: +49-211-96855-55

Anritsu GmbH (ACDE-Sales Center South)
An der Steinernen Brücke 1 D-85757
Karlsfeld, Germany
Phone: +49-8131-3825-0
Fax: +49-8131-3825-95

July 1999

Anritsu S.A. (ACFR)

9, Avenue du Québec ZA de
Courtaboeuf 91951 Les Ulis Cedex,
France
Phone: +33-1-60-92-15-50
Fax: +33-1-64-46-10-65

Anritsu S.A. (ACFR) (Toulouse Office)

Bureau de Toulouse
Région Centre Sud Ouest
Phone: +33-5-62070484
Fax: +33-5-62070668

Anritsu S.A. (ACFR) (Toulon Office)

Bureau de Toulon
Région Centre Sude Est
Phone: +33-4-94040264
Fax: +33-4-94040265

Anritsu S.A. (ACFR) (Rennes Office)

Bureau de Rennes
Région Ouest
Phone: +33-2-99521214
Fax: +33-2-99521224

Anritsu S.p.A. (ACIT)

Via Elio Vittorini, 129
00144 Roma EUR, Italy
Phone: +39-06-509-9711
Fax: +39-06-502-2425

Anritsu S.p.A (ACIT-Milano)

C.D. Colleoni, Via Paracelso, 420041
AGRATE B.ZA(MI), Italy
Phone: +39-039-65-7021
Fax: +39-039-605-6396

Anritsu AB (ACSE)

BOTVID CENTER
145 84 STOCKHOLM, Sweden
Phone: +46-8-53470700
Fax: +46-8-53470730

Anritsu AB-Norway Branch Office (ACNO)

Leangbukta 40 1370 Asker. Norway
Phone: +47-66-901190
Fax: +47-66-901212

Anritsu AB-Finland Branch Office (ACFI)

Piispanportti 9 FIN-02240 Espoo, Finland
Phone: +358-9-435-522-0
Fax: +358-9-435-522-50

Anritsu AB-Denmark Branch Office (ACDK)

SOHOJ 11, DK-2690 KARLSLUNDE
Denmark
Phone: +45-46160330
Fax: +45-46155299

C.N. Rood B.V.

Cort van der Linddenstraat 11-13, 2288
EV Rijswijk ZH, The Netherlands
Phone: +31-70-3996360
Fax: +31-70-3905740

C.N. Rood SA/NV

Pontbeeklaan 45, 1731 Zellik, Belgium
Phone: +32-2-4668199
Fax: +32-2-4662500

ELSINCO GMBH

h.e. Strelbishte, str. Kotlenski Prohod, bl.
96/6/14, BG-1408 Sofia, Bulgaria
Phone: +359-2-58-61-31
Fax: +359-2-58-16-98

ELSINCO Praha Spol.

Novedvorska 994, CZ 142 21 Praha
4-Branik , Czecho Republic
Phone: +42-2-49-66-89
Fax: +42-2-49-54-83

ELSINCO Budapest KFT

Pannonia utca 8. IV/I.
H-1136 Budapest, Hungary
Phone: +36-1-269-18-50
Fax: +36-1-132-69-27

ELSINCO Polska Sp. Z.O.O

ul. Dziennikarska 6/1, PL 01 605
Warszawa, Poland
Phone: +48-22-39-69-79
Fax: +48-22-39-44-42

ELSINCO Bratislava Spol. s.r.o.

Kudlakova 4, SK 844 15 Bratislava,
Slovakia
Phone: +42-7-784-165
Fax: +42-7-784-454

G'Amungason Co.

Skulagata 40, 101 Reykjavik, Iceland
Phone: +354-1-677887
Fax: +354-1-625045

GMP S.A.

Av. des Baumettes 19, CH-1020 Renens
1 Lausanne, Switzerland
Phone: +41-21-6348181
Fax: +41-21-6353295

Instrutek Oeriferi A/S

Christiansholmsgade DK-8700 Horsens,
Denmark
Phone: +45-75-611100
Fax: +45-75-615658

Kostas Karayannis SA

58, Kapodistriou str., GR-142 35 Nea Ionia,
Athens, Greece
Phone: +30-1-680-0460-4
Fax: +30-1-685-3522

Omnitecnica S.A.

Estrada Alfragide 2700 Amadora, Portugal
Phone: +351-1-471-55-17
Fax: +351-1-471-36-10

Pema Ltd.

Doromiskin, Dundalk, Co. Louth, Ireland
Phone: +353-42-72899
Fax: +353-42-72376

Unitronics S.A.

Plaza Espana 18, Torre de Madrid,
Pl. 12-ofc. 9, 28019 Madrid, Spain
Phone: +34-1-5425204
Fax: +34-1-5591957

Wien CHALL GMBH

Krichbaumgasse 25, 1120 Wien, Austria
Phone: +43-1-811-55140
Fax: +43-1-811-55180

Asia, Pacific and Africa

Anritsu Private Ltd (ACSG)

6, New Industrial Rd., #06-01/02
Hoe Huat Industrial Building
Singapore 536199
Phone: 65-282-2400
Fax: 65-282-2533

Anritsu Company Ltd (ACHK)

Suite 719, 7/F., Chinachem Golden Plaza,
77 Mody Road, Tsimshatsui
East, Kowloon, Hong Kong
Phone: +852-2301-4980
Fax: +852-2301-3545

Anritsu Company Incorporated (ACTW)

6F, 96, Sec. 3, Chien Kou North Rd.
Taipei, Taiwan
Phone: +886-2-2515-6050
Fax: +886-2-2509-5519

Anritsu Corporation, Ltd (ACKR) Head Office

14F Hyunjuk Bldg. 832-41,
Yeoksam-dong, Kangnam-ku,
Seoul, Korea
Phone: +82-2-553-6603
Fax: +82-2-553-6604~5

Anritsu Proprietary Ltd (ACAU)

Unit 3, 170, Forster Rd., Mt. Waverley
Victoria 3149, Australia
Phone: +61-3-9558-8177
Fax: +61-3-9558-8255

Anritsu Proprietary Ltd (ACAU)

Suite 304/2 Rowe Street Eastwood
NSW 2122, Australia
Phone: +61-2-9874-9044
Fax: +61-2-9874-9920

Anritsu Corporation Beijing Liaison Office

Room No.1515, Beijing Fortune Bldg. 5
Dong-San-Huan-Bei-Lu Chao-Yang
District Beijing 100004, P.R. China
Phone: +86-10-6590-9230~9234
Fax: +86-10-6590-9235

Anritsu Corporation Shanghai Liaison Office

No.511 Shanghai Jing Tai Building
No.58, Mao Ming Nan Rd. Sanghai
200020 P.R. China
Phone: +86-21-6415-5137
Fax: +86-21-6472-6677

Anritsu Company Ltd Guangzhou Representative Office

Room 720, 7/F., Dongshan Plaza,
69 XianLie Road Central.
Guangzhou 510095 P.R. China
Phone: +86-20-8732-2231
Fax: +86-20-8732-2230

Anritsu Corporation New Delhi Liaison Office

Room No. 508, Prakash Deep,
7 Tolstoy Marg, New Delhi-110001, India
Phone: +91-11-331-9133
Fax: +91-11-371-3948

Associated Electric Trading Corp.

Zia Chambers, 25 McLeod Rd., Lahore,
Pakistan
Phone: +92-42-722-1716
Fax: +92-42-7221456

Chris Radiovision Ltd.

Kouloumbris Building, 23 Crete Street,
P.O. Box 1989, Nicosia, Cyprus
Phone: +357-2-466121
Fax: +357-2-365177

Electronic Equipment Marketing Co.

P.O. Box 3750, Riyadh 11481,
Saudi Arabia
Phone: +966-1-4771650
Fax: +966-1-4785140

Etecsa (Pty) Ltd.

1st Floor Montrose Place, Waterfall Park,
Bekker Rd., Midrand, South Africa
Phone: +27-11-315-1366
Fax: +27-11-315-2175

Giza Systems Engineering

2 El Mesaha Square, Dokki A.R.E.,
P.O.Box 1913, Cairo 11511, Egypt
Phone: +20-2-349-0140
Fax: +20-2-360-9932

Infotechs Ltd.

23-1, Jaya Rd., Colombo 4, Sri Lanka
Phone: +94-1-580088
Fax: +94-1-584644

Inter Muhendislik Danismanlik ve Ticaret A.S.

Farabi Sokak No: 24/14
Cankaya-06680 Ankara, Turkey
Phone: +90-312-4277792
Fax: +90-312-4277937

Jasmine Telecom Systems Co., Ltd.

333 Laksi Plaza 6th Floor, Tower 2,
Chaengwatana Rd., Donmuang, Bangkok
10210, Thailand
Phone: +66-2-576-0200
Fax: +66-2-576-0420

Meera Agencies (P) Ltd.

A-23 Hauz Khas New Delhi 110 016, India
Phone: +91-11-685-3959
Fax: +91-11-685-2275

UAE
Utmost Electronics Trading (L.L.C.)

P.O. Box 41175 Abu Dhabi, U.A.E.
Phone: +971-2-768909
Fax: +971-2-768907

Martwell Electronics Pvt., Ltd

3rd Floor, Francis House, Stanley Avenue,
P.O. Box 1737, Harare, Zimbabwe
Phone: +263-4-793578
Fax: +263-4-737956

Mandeno Electronic Equip. Co.

463 Mt. Eden Rd. Mt. Eden,
Aukland 1003, New Zealand
Phone: +64-9-630-7871
Fax: +64-9-630-1720

National Projects and Technology Co. L.L.C

P.O. Box 97, Wadi Al Kabir, Postal Code 117,
Sultanate of Oman
Phone: +968-791704
Fax: +968-791697

O'Connors Engineering & Trading (Malaysia) Bhd

3rd Floor, Wisma Siong Huat,
Lot 13, Jalan 51A/223
46100 Petaling Jaya,
Selangor Darul Ehsan, Malaysia
Phone: +60-3-757-2828
Fax: +60-3-757-7871

P.T. Subur Sakti Putera

Jalan Musi No.32, Jakarta 10150,
Indonesia
Phone: +62-21-3803644
Fax: +62-21-3845043

Qatar Communications Ltd.

P.O. Box 2481, Doha, Qatar
Phone: +974-424347
Fax: +974-324777

Trading and Agency Services

P.O. Box 1884, Doha, Qater
Phone: +974-432212
Fax: +974-422255

Rajab & Silsilah & Co.

P.O. Box 203 Jeddah 21411, Saudi Arabia
Phone: +966-2-6610006
Fax: +966-2-6610558

Salritsu International Trading Corporation

50B ODC International Plaza
Condominium, 219 Salcedo St., Legaspi
Village, Makati, Metro Manila, Philippines
Phone: +63-2-816-2646
Fax: +63-2-815-0986

Sedel

24, 26, Bd, Resistance, Casablanca,
Morocco
Phone: +212-2-302444
Fax: +212-2-449311

Superior Electronics Associated

B-98 Block H, North Nasimabad, Karachi-
33, Pakistan
Phone: +92-21-613655

Tareq Company

P.O. Box 20506 Safat, 13066 Safat, Kuwait
Phone: +965-2336-100
Fax: +965-2437-700

Tech-Cent Ltd.

Haarad St. No.7, Ramat Haahayal,
Tel-Aviv 69710, Israel
Phone: +972-3-6478563
Fax: +972-3-6478334

Test

Sehit Adem Yavuz Sokak No.6/17,
Kizilay-Ankara, Turkey
Phone: +90-41-71086
Fax: +90-41-74384

Japan

Head Office

5-10-27, Minamiazabu, Minato-ku,
Tokyo 106-8570
Phone: 03-3446-1111
Fax: 03-3442-0235

Atsugi Factory

1800, Onna, Atsugi-si, Kanagawa 243-8555
Phone: 046-223-1111
Fax: 046-225-8379

July 1999

If you have any comments on this manual, please e-mail them to Manual-support@zz.anritsu.co.jp.

Anritsu

ANRITSU CORPORATION 5-10-27, Minamiazabu, Minato-ku, Tokyo 106 Japan / Phone: 81-3-3446-1111

Document No.: [illegible]

Printed in Japan