MS9715B
WDM Tester
Remote Control
Operation Manual

First Edition

To ensure that the MS9715B WDM Tester is used safely, read the safety information in the MS9715B WDM Tester Operation Manual first. Keep this manual with the WDM Tester.

Measuring Instruments Division
Measurement Group
ANRITSU CORPORATION

Document No.: M-W1732AE-1.0

MS9715B
WDM Tester Remote Control
Operation Manual

1 July 2000 (First Edition)

About This Manual

This manual explains remote control of the MS9715B WDM Tester. You can control the MS9715B and transfer measurement results into the computer connected to the GPIB/RS-232C interface port of the MS9715B.

Table of Contents

Section 1 Introduction

This section outlines the remote control functions of the MS9715B Tester.

1.1 Overview

The MS9715B can make measurements automatically in combination with an external controller (host computer, personal computer, etc.). To connect an external controller, the MS9715B has GPIB interface bus (IEEE Standard 488.2-1987) and RS-232C interface ports.

1.2 MS9715B Remote Control Functions

The MS9715B supports the following functions:

(1) Control of almost all functions except some functions such as a POWER switch and LCAL key
(2) Read of all setting conditions
(3) Setting of the GPIB address from panel
(4) Interrupt function and serial polling (GPIB)
(5) Setting of RS-232C interface conditions from panel
(6) Selection of an interface port application from panel
(7) Configuration of an automatic measurement system by combining the MS9715B with a personal computer and another measurement instrument

1.3 Interface Port Application Selection Function

The MS9715B comes standard with a GPIB interface bus and an RS-232C interface. Application of these interface ports can be selected from the panel.

External controller connection port : Select GPIB or RS-232C.
Printer connection port : GPIB

The above two ports cannot be used at the same time.

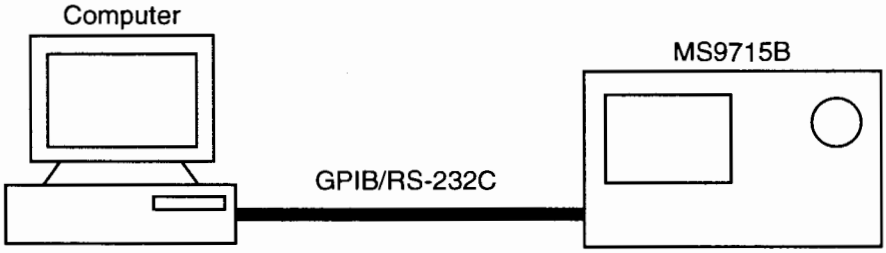
1.4 Examples of Setups Using GPIB/RS-232C

(1) Standalone type

Waveforms measured with the MS9715B are output to the printer.

(2) Control by host computer

The MS9715B is controlled by a computer automatically/remotely.



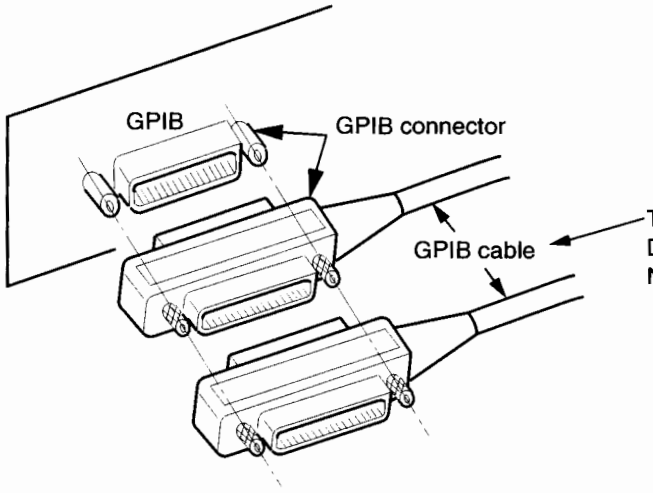
Section 2 How to Connect

This section explains how to connect GPIB and RS-232C cables between the MS9715B and external devices such as a host computer, personal computer, and printer. This section also explains how to set the interfaces of the MS9715B.

2.1 Connecting Devices Using GPIB Cables

The MS9715B has a GPIB cable connection connector on the back panel. Be sure to connect GPIB cables before turning on the power.

A maximum of 15 devices, including a controller, can be connected. Connection conditions are given shown below.



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

2.1.1 Setting interface conditions for the connection port

When controlling the MS9715B automatically/remotely from a computer, set interface conditions for the connection Press the RS-232C Prmtr function key on the "Others" card to select "GPIB" for "Interface."

=== RS232C Parameter ==========

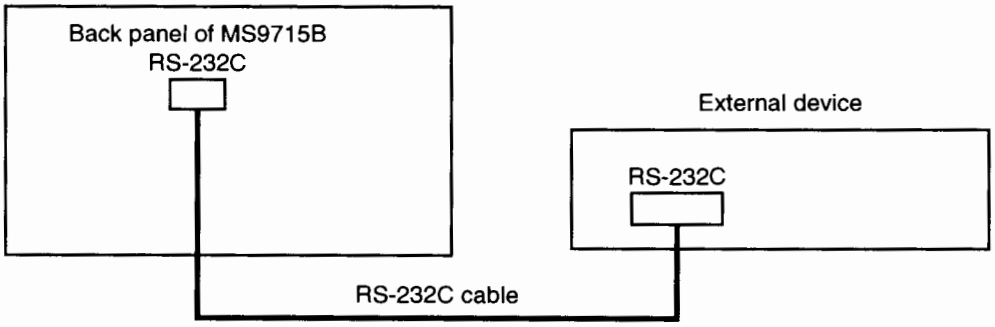
▶Interface	GPIB	RS232C			
Speed(bps)	9600	4800	2400	1200	600
Parity	None	Even	Odd		
Character Length	7Bit	8Bit			
Stop Bit	1Bit	2Bit			

2.1.2 Confirming and Setting the Address

Be sure to set the MS9715B's GPIB address after turning on the power. The factory-set address "08" is battery-backed up. If you use this address, the address need not be set again. If you want to change the address, place the MS9715B in the local mode, press the GPIB Address function key on the "Others" card, then enter a new address with keyboard keys or an encoder. Immediately after the power is turned on, the devices on the GPIB automatically enters the local mode.

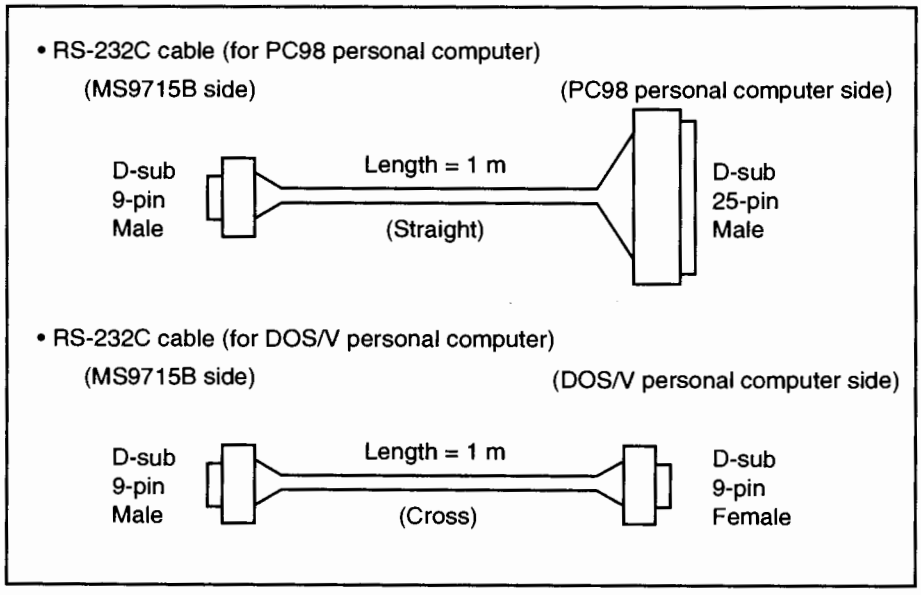
2.2 Connecting a Device Using an RS-232C Cable

Connect the RS-232C connector (D-sub, 9-pin, female) and the RS-232C connector with an RS-232C cable.



Note:

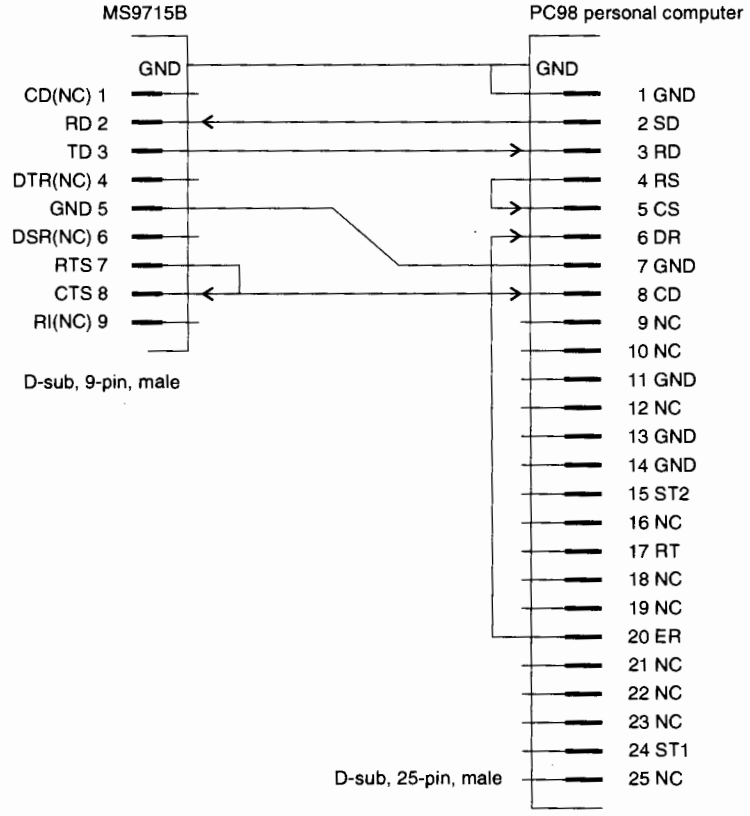
RS-232C connectors are available in 9-pin and 25-pin types. Before purchasing an RS-232C cable, check the number of pins of the RS-232C connector on the external device. The following two types of RS-232C cables are available as application parts for this analyzer.



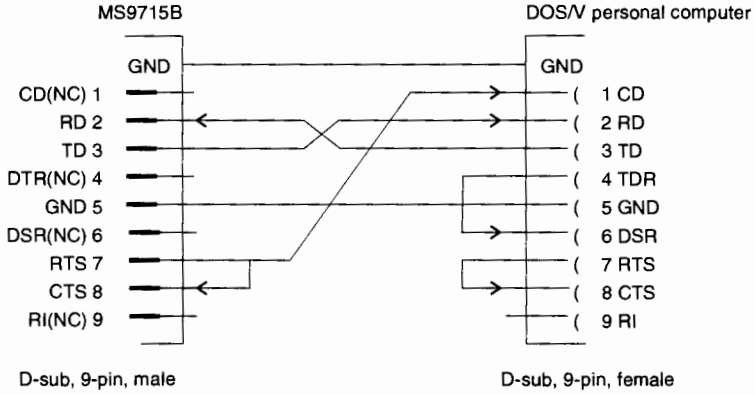
2.2.1 RS-232C interface signal connection diagrams

The following diagrams show connections of RS-232C interface signals between the MS9715B and two types of personal computers.

- Connection with PC98 personal computer



- Connection with DOS/V personal computer



2.2.2 Setting interface conditions for the connection port

When controlling the MS9715B automatically/remotely from a computer, set interface conditions for the connection port.
Press the RS-232C Prmtr function key on the "Others" card and select "RS232C" for "Interface."

2.2.3 Setting RS-232C interface conditions

Set interface conditions for the RS-232C port of this analyzer so that they match the interface conditions of the connected external device.

Pressing the RS-232C Prmntr function key on the "Others" card will bring up the following screen:

```
=== RS232C Parameter ==========================================
▶Interface ..... GPIB   RS232C
 Speed(bps) ..... 9600   4800   2400   1200   600
 Parity ..... None   Even   Odd
 Character Length ..... 7Bit   8Bit
 Stop Bit ..... 1Bit   2Bit
```

Using ↑ and ↓ function keys, move the cursor to the item you want to change.

Item	Meaning of setting	
Speed	Select a communication speed among 600, 1200, 2400, 4800, and 9600 bps.	
Parity	Select a parity bit type.	
	None	No parity bit is added.
	Even	An even parity bit is added.
	Odd	An odd parity bit is added.
Stop Bit	Select a stop bit type.	
	1	1 stop bit is added.
	2	2 stop bits is added.
Character Length	Select a character length.	
	7	7 bits
	8	8 bits

Section 3 Standards

This section explains the MS9715B's GPIB standard, RS-232C standard, and device message list.

3.1 GPIB Standard

The standard for the GPIB of the MS9715B is summarized below.

Item	Standard value and description
Function	Conforms to IEEE 488.2. This analyzer can be controlled from an external controller. This analyzer can control a printer.
Interface functions (Note)	SH1: All of source handshake functions are supported. Data send timing is controlled. AH1: All of acceptor handshake functions are supported. Data receive timing is controlled. T6: Basic talker functions are supported. A serial port function is supported. A talk-only function is not supported. The function of releasing the talker with MLA is supported. L4: Basic listener functions are supported. A listen-only function is not supported. The function of releasing the listener by MTA is supported. SR1: All of service request and status byte functions are supported. RL1: All of remote/local functions are supported. A local lockout function is supported. PP0: A parallel poll function is not supported. DC1: All of device clear functions are supported. DT0: A disk trigger function is not supported. C0: A controller function is not supported. A controller function is performed during external plot output.

Note:

For details on the interface function subset, refer to Section 1, "Basic Knowledge of GPIB," in the separate GPIB Basic Guide (available optionally).

3.2 RS-232C Standard

The standard for the RS-232C of the MS9715B is summarized below.

Item	Standard value
Function	Control from external controller
Communication method	Asynchronous (start-stop), half-duplex
Communication control method	No flow control
Baud rate	600, 1200, 2400, 2800, 9600 bps
Number of data bits	7 bits, 8 bits
Parity	Odd parity (ODD), even parity (EVEN), non-parity (NON)
Number of start bits	1 bit
Number of stop bits	1 bit, 2 bits
Connector	D-sub, 9-pin, female

3.3 Device Message List

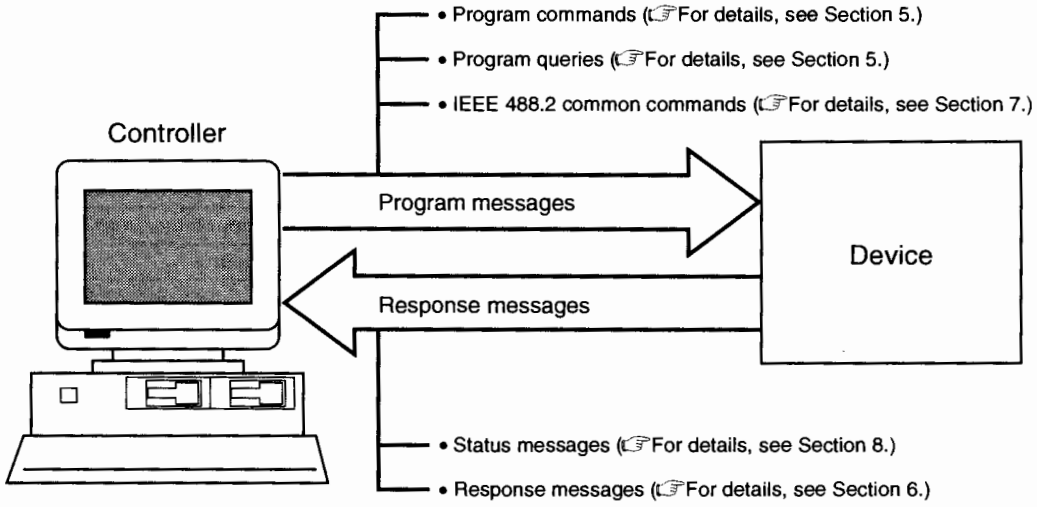
Device messages are data messages which are transferred between a controller and devices. They are classified into program messages and response messages.

Program messages are ASCII messages transferred from a controller to devices. Program messages are further classified into program commands and program queries. These two types of commands are explained on the following pages.

Program commands include device-dependent commands which are exclusively used for controlling the MS9715B and IEEE 488.2 common commands. IEEE 488.2 common commands are program commands which are commonly applicable to other IEEE 488.2-ready measuring instruments (including the MS9715B) on the GPIB interface bus.

Program queries are commands used to get response messages from devices. Program queries must be transferred from a controller to a device in advance so that the controller can receive response messages from the device later.

Response messages are ASCII data messages which are transferred from a device to a controller. Among response messages, status messages and response messages corresponding to program queries are listed on the following pages.



In program and response messages, numeric data may end with a suffix (unit).

The above messages are transferred through the device input/output buffer. The output buffer is also called an output queue. A brief description of the output buffer is given below.

Input buffer	Output queue
A FIFO (first in first out) type memory area that stores DABs (program and query messages) temporarily before analysis of syntax and execution. The input buffer size of the MS9715B is 256 bytes.	An FIFO-type queue memory area. All DABs (response messages) output from a device to a controller are stored in this memory until they have been read by the controller. The output queue size of the MS9715B is 256 bytes.

3.3.1 IEEE 488.2 common commands and the commands supported by the MS9715B

The table below lists 39 common commands specified by IEEE 488.2. Among these commands, the commands supported by the MS9715B are marked with ◎.

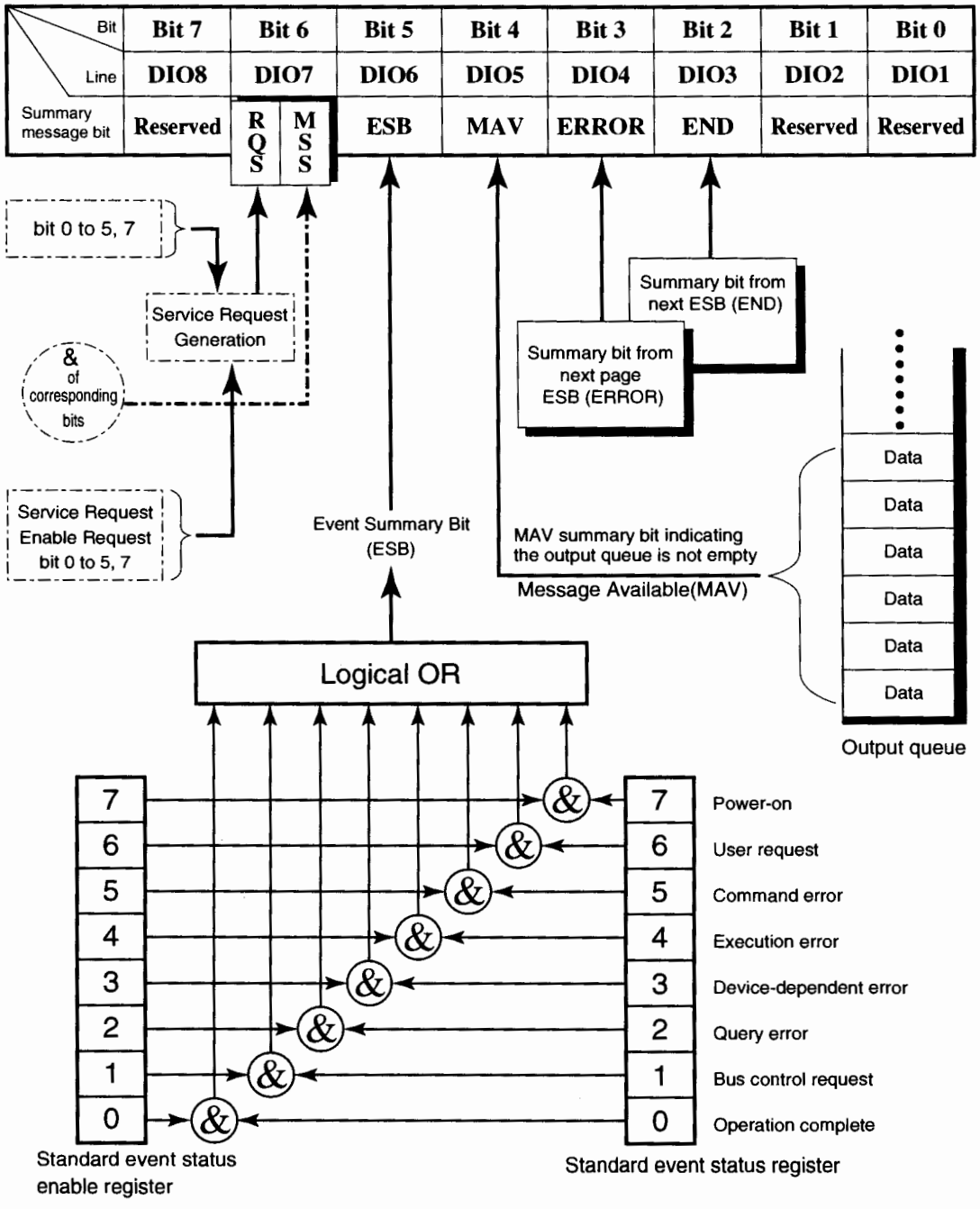
Mnemonic	Full spelled command name	Specification by IEEE 488.2	Support by MS9715B
* ADD	Accept Address Command	Optional	
* CAL	Calibration Query	Optional	
* CLS	Clear Status Command	Required	◎
* DDT	Define Device Trigger Command	Optional	
* DDT?	Define Device Trigger Query	Optional	
* DLF	Disable Listener 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	◎

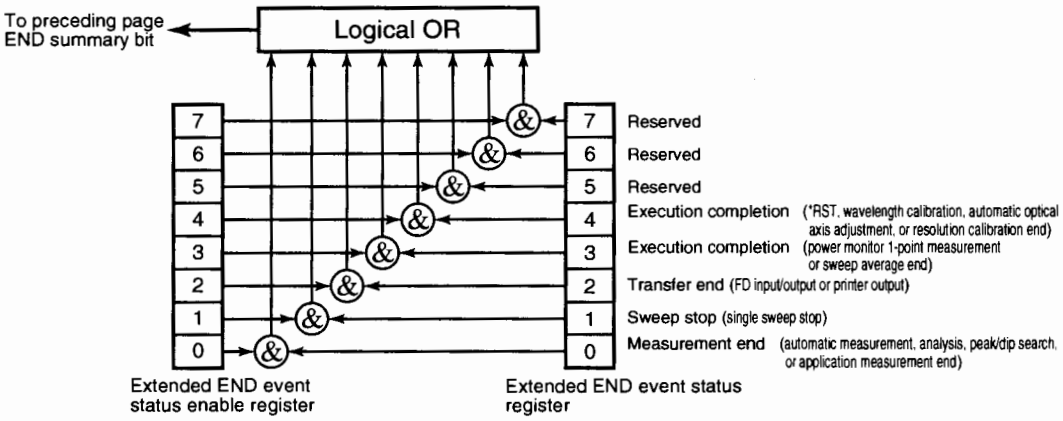
☞ IEEE 488.2 commands begin with *. For more details, see Section 7.

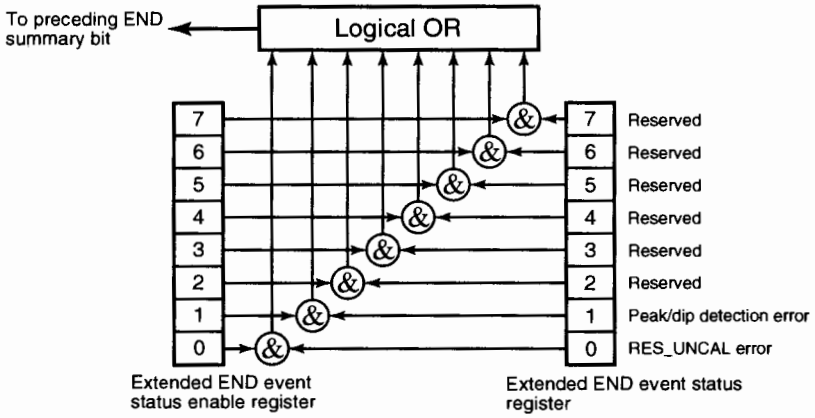
3.3.2 Status Messages

Shown below is the structure of the service request summary message set in the status byte register of the MS9715B.

Summary Bit Configuration of Status Byte Register




To preceding page
END summary bit
Logical OR
7
6
5
4
3
2
1
0
Reserved
Reserved
Reserved
Execution completion (*RST, wavelength calibration, automatic optical axis adjustment, or resolution calibration end)
Execution completion (power monitor 1-point measurement or sweep average end)
Transfer end (FD input/output or printer output)
Sweep stop (single sweep stop)
Measurement end (automatic measurement, analysis, peak/dip search, or application measurement end)
Extended END event status enable register
Extended END event status register


To preceding END summary bit
Logical OR
7
6
5
4
3
2
1
0
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved
Peak/dip detection error
RES_UNCAL error
Extended END event status enable register
Extended END event status register

3.3.3 MS9715B device message list

A list of MS9715B-dependent program commands, program queries, and response messages is shown on the following pages.

MS9715B Device Message List (1/5)

Item		Device message			Remarks
		Command	Data request	Response	
Sweep	Single	SSI			
	Repeat	SRT			
	Stop	SST			
Zoom Marker	Zoom Marker	ZMKW λ c, λ s	ZMKW?	λ c, λ s λ c=xxxx. xxx λ s=xxxx. xxx	λ c, λ s : Unit (nm) λ c : Zoom center λ s : Zoom span
	Zoom In/Out	ZMKW Zoom, s s=IN =OUT			
	Marker Erase	ZMKW ERS			
Measurement Status			MODE?	n n=0 : No measurement of spectrum n=1 : Single sweep of spectrum n=2 : Repeat sweep of spectrum n=3 : Under continuous testing	
Spectrum	Peak No /Wl/Lvl /SNR/L/R		APR? WDM, PKC APR? WDM, SNR, x	WDM, PKC, d WDM, PKC, λ, L, S, rl λ=xxxx. xxx L=xxxx. xx S=xxx. xx rl=1 : Left 2 : Right	d : Peak count value (0.1.2•••) x : Peak No. λ : Peak No. x Wavelength (nm) L : Peak No. x Level (dBm) S : Peak No. x SNR (dB) rl : PeakNo. x Left or Right Peak Count=case 0 λ=-1 L=-999.99 S=-999.99 rl=0
	Gain Val		APR? WDM, SNR, GAT	WDM, SNR, GAT a, mx, mi a=xxx. xx mx=xxx. xx mi=xxx. xx	a=mx-mi (dB) mx : Maximum value (dB) mi : Minimum value (dB) Peak Count=case 0 a=999.99 mx=999.99 mi=999.99

MS9715B Device Message List (2/5)

Item		Device message			Remarks
		Command	Data request	Response	
Spectrum	Slope On/Off	SLP s s=ON, OFF	SLP?	s s=ON, OFF	
	Slope Value		SLPD?	n n=xx, xxx	n : Unit (dB/nm)
	Total Power On/Off	TPW s s=ON, OFF	TPW?	s s=ON, OFF	
	Total Power Value		TPWD?	n n=xx, xxx	n : Unit (dBm)
	Marking	PMK s, x s=ON, OFF x=Perk No.(0,1,2•••) 0 : All peak No.	PMK? x x=Perk No. (1,2,3•••)	s s=ON, OFF	Same as L-Team Test Marking
	Res	RES n n is the value indicated in the right column.	RES?	n n=0.05, 0.07, 0.1, 0.2, 0.5, 1	n: resolution (nm)
	VBW	VBW s s is the value indicated in the right column.	VBW?	s s=1 MHz, 100 kHz, 10 kHz, 1 kHz, 100 Hz, 10 Hz	s: VBW value In Hz unit when the unit is omitted.
	Trace Marker	TMK λ λ=xxxx.xxxx	TMK?	λ, l λ=xxxx.xxxx l=xx.xx (dBm) =Shipment 4 digits	λ: wavelength (nm) l: level (dBm)
Save/Recall	Format	FMT			
	File Delete	DEL n			n : File name
	File Option	FOPT a, b, c a=NOTE =BMP =TXT =BMP&TXT b=NUMBER =NAME c=1.44M =1.22M	FOPT?	a, b, c a=NOTE =BMP =TXT =BMP&TXT b=NUMBER =NAME c=1.44M =1.22M	a : Option file specification b : File specification method c : FFD mode c : Omissible
	Save	SAV n n : File name			n : File name
	Recall	RCL n n : File name			n : File name

MS9715B Device Message List (3/5)

Item		Device message			Remarks
		Command	Data request	Response	
Memory Data	Data	d + terminator	DMA?	LOG scale ± xxx. xx	LOG : Unit (dBm)
		d + Separator	DQA?		
		Binary	DBA?	LOG : 2 bytes/1 data	LOG : × 0.01 (dBm)
	Data Condition		DCA?	λ 1. λ 2. n λ 1=xxx. xx λ 2=xxx. xx n=501 to 2001	
Cal	WI Cal	WCAL n n=0 : W-CAL INITIAL =2 : W-CAL2 =3 : Forcible termination	WCAL?	m m=0 : Calibration complete =1 : During calibration =2 : Insufficient optical level =3 : Other faults	
	Auto Align	ALIN n n=0 : ALIGN INITIAL =1 : ALIGN =2 : Forcible termination	ALIN?	m m=0 : Calibration complete =1 : During calibration =2 : Insufficient optical level =3 : Other faults	
Others	Date	DATE yy. mm. dd	DATE?	yy. mm. dd	yy : 0 to 99 mm : 01 to 12 dd : 01 to 31 hh : 00 to 23 mi : 00 to 59
	Time	TIME hh. mi	TIME?	hh. mi	
	Time&Date On/Off	TDSP s s=ON. OFF	TDSP?	s s=ON. OFF	
	Buzzer	BUZ s s=ON. OFF	BUZ?	s s=ON. OFF	

MS9715B Device Message List (4/5)

Item		Device message			Remarks
		Command	Data request	Response	
L-T Test		s=ON. OFF	LTS?	s s=ON. OFF	ON : Under continuous testing OFF : Spectrum screen
	Sampling Time	SPT n n=1 to 99. OFF	SPT?	n n=1 to 99. OFF	
	Marking	PMK s. x s=ON. OFF x=Perk No. (0.1.2•••) 0 : All Peak No.	PMK? x x=Perk No. (1.2.3•••)	s s=ON. OFF	Same as spectrum marking
	Test Start	TSA n n : File name			
	Test Stop	TSO			
			LTSS?	m m=0 : Normal termination =1 : During operation =2 : Abnormao termination	Normal termination : • Save of 1000 data • FD Full • Stop key is pressed During operation : Continuous testing inprogress Abnormao termination : • FD Error • Cant Find Peak • Optifal Block Error
	Save Data number		LTSD? DTA	DTA. n n=0 to 1000	
	Wl. Lvl. SNR Measurement Data		LTSD? PKD. x x=Perk No. (1.2.3•••)	PKD. a λ. mx λ. mi λ. aL. mxL. miL. aS. mxS. miS a λ=xxxx. xxx mx λ=xxxx. xxx mi λ=xxxx. xxx aL=xxxx. xx mxL=xxxx. xx miL=xxxx. xx aS=xxx. xx mxS=xxx. xx miS=xxx. xx	Peak No. data saved on FD. a λ : Wavelength average value (nm) mx λ : Wavelength maximum value (nm) mi λ : Wavelength minimum value (nm) aL : Level average value (dBm) mxL : Level maximum value (dBm) miL : Level minimum value (dBm) aS : SNR average value (dB) mxS : SNR maximum value (dB) miS : SNR minimum value (dB)

MS9715B Device Message List (5/5)

Item		Device message			Remarks
		Command	Data request	Response	
L-T Test	Wl, Lvl, SNR Measurement start time data		LTSD? SPKD, x x=Perk No. (1,2,3•••)	SPKD, λ, L, S λ=xxxx. xxx L=xxxx. xx S=xxx. xx	Start time peak No. data saved on FD λ : Wavelength value (nm) L : Level value (dBm) S : SNR value (dB)
	Total Power		LTSD? TPWD	TPWD, n, mx, mi n=xxxx. xx mx=xxxx. xx mi=xxxx. xx	Total power saved on FD n : Average value (dBm) mx : Maximum value (dBm) mi : Minimum value (dBm)
	Gain Val		LTSD? GAT	GAT, n, mx, mi n=xxx. xx mx=xxx. xx mi=xxx. xx	Gain tilt saved on FD n : Average value (dB) mx : Maximum value (dB) mi : Minimum value (dB)
	Slope		LTSD? SLP	SLP, n, mx, mi n=xxx. xx mx=xxx. xx mi=xxx. xx	Sope saved on FD n : Average value (dB/nm) mx : Maximum value (dB/nm) mi : Minimum value (dB/nm)
	Start/Stop Time		LTSD? TIM	TIM, h, mi, sh, smi	Start time and stop time saved on FD h, sh : 00 to 23 mi, smi : 00 to 59
	Specing	SPG S S =FRQ =WAVE FRQ=Frequency display WAVE=Wavelength display	SPG?	S =FRQ =WAVE	
L-T Test Recall	Recall	LRCL? n, m n : File name m : Data No. (1 to 1000)			

Section 4 Initial Setting

The GPIB interface system is initialized at three levels. At level 1, "bus initialization" is performed to place the system bus in the idle state. At level 2, "message exchange initialization" is performed to enable devices to receive program messages. At level 3, "device initialization" is performed to initialize device-dependent functions.
At these three initialization levels, preparations are made for starting devices.

E 488.1 defined the following two levels of GPIB system initialization:

- Initialization of bus: Interface functions of all devices connected to the bus are initialized by an IFC message from the controller.

- Initialization of devices: All devices on the GPIB are initialized with a GPIB bus command "DCL", or only the specified devices are initialized to their specified states with a GPIB bus command "SDC."

IEEE 488.2 defines three levels. At level 1, "bus initialization" is performed. This is the highest level. "Device initialization" is divided into "message exchange initialization" (level 2) and "device initialization" (level 3). IEEE 488.2 also defines the device power-on status.

The following table provides a summary of the above explanation:

Level	Initialization type	Outline	Combination and priority of levels
1	Bus initialization	Interface functions of all devices connected to the bus are initialized by an IFC message from a controller.	This level may be combined with other levels. However, initialization at level 1 must be performed before initialization at other levels.
2	Message exchange initialization	Message exchange is initialized and the function of reporting completion of operation to the controller is disabled for all devices on the GPIB (with GPIB bus command "DCL") or only for the specified devices (with a GPIB bus command "SDC").	This level may be combined with other levels. However, initialization at level 2 must be performed before initialization at level 3.
3	Device initialization	Only the specified devices on the GPIB are initialized to their known states with an *RST command irrespective of the past use state.	This level may be combined with other levels. However, initialization at level 3 must be performed after initialization at level 1 and 3.

When controlled from a controller via the RS-232C interface port, the MS9715B can use the "device initialization" function (level 3). However, it cannot use "bus initialization" (level 1) and "message exchange initialization" (level 2) functions. When controlled from a controller via a GPIB interface bus, the MS9715B can use all the above initialization functions (levels 1-3).

Let's take a look at the commands for performing initialization at levels 1-3 and the items to be initialized as well as the known states set at power-on.

4.1 Initialization of Bus by IFC Statement

■ Format

IFC Δ@select-code

■ Application example

IFC @ 1

■ Explanation

This function can be used when the MS9715B is controlled from a controller via a GPIB interface bus.

On the GPIB corresponding to the specified select code, the IFC line is activated for about 100 μs (electrically set at the low level). When IFC@ is executed, interface functions of all devices connected to the GPIB bus line corresponding to the specified select code are initialized. Only the system controller can send this command.

"Initialization of interface functions" refers to the processing in which controller-set device interface functions (talker, listener, etc.) are reset to their initial states. Functions marked with ○ in the following table are initialized. The function marked with Δ is initialized partially.

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	○

If the IFC statement is true (the IFC line is set at the low level through execution of the IFC@ statement), initialization is not performed at levels 2 and 3. That is, device operating states are not affected.

Let's take a look at some device states set by the IFC statement.

[1] Talker/listener: ..... All talkers and listeners are set in the idle state (TIDS, LIDS) within 100 µs.

[2] Controller: ..... If the controller is not active (SACS: System control ACtive State), it enters the idle state "CIDS" (Controller IDle State) within 100 µs.

[3] Return of control right: If the system controller (the first device on the GPIB which is used as a controller) has granted the control right to another device when IFC@ is executed, the control right is returned to the system controller. Generally, pressing the [RESET] key on the system controller allows an IFC message to be output from the system controller.

[4] Devices issuing service request:

The state in which an SRQ message is issued by a device (the SRQ line is set at the low level by the device) is not canceled, but the state in which all devices on the system bus are placed in the serial poll mode by the controller is canceled.

[5] Devices in remote state:

For the devices currently in the remote state, the remote state is not canceled by the IFC message.

4.2 Initialization of Message Exchange by DCL and SDC Bus Commands

■ Format

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

■ Application example

DCL @1	Initializes message exchange for all devices on the bus. (Issue of DCL)
DCL @103	initializes message exchange only for the device at address 3. (Issue of SDC)

■ Explanation

This function can be used when the MS9715B is controlled by a controller via the GPIB interface bus.
This statement initializes message exchange for all device on the GPIB corresponding to the specified select code or only for the specified devices.

The purpose of message exchange is to allow the controller to send new commands when the controller cannot control message-exchange-related parts inside the devices due to execution of programs although it is not necessary to change the panel settings.

■ When only a select code is specified

Message exchange is initialized for all the devices on the GPIB corresponding to the specified select code. DCL@ issues a DCL (Device Clear) bus command to the GPIB.

■ When an address is also specified

Message exchange is initialized only for the specified device. Listeners on the GPIB corresponding to the specified select code are canceled, only the specified device is set as a listener, and an SDC (Selected Device Clear) bus command is issued.

■ Items subject to initialization of message exchange

[1] Input buffer and output queue: ... Cleared.

[2] Syntax analysis, execution control, and response generation parts:
Reset

[3] Device commands including *RST:
All commands interfering with execution of these commands are cleared.

[4] Paired parameter/program message:
All commands and queries whose execution has been suspended due to paired parameters are discarded.

[5] *OPC command processing: ..... The specified device is set in the OCIS (Operating Complete Command Idle State). The operation complete bit cannot be set in the standard event status register. (☞P7-7)

[6] *OPC? query processing: ..... The specified device is set in the OCIS (Operating Complete Command Idle State). The operation complete bit cannot be set in the output queue. The MAV bit is cleared. (☞P7-7)

[7] Automatic system configuration: . *ADD and *DFL common commands are invalidated. (The MS9715B does not support these commands.)

[8] Device function: ..... All parts related to message exchange are set in the idle state. The device waits for a message from the controller.

The following operations are prohibited:
[1] Changing the current device settings and stored data
[2] Interrupting front panel I/O
[3] Changing status bits other than the MAV bit when clearing the output queue
[4] Affecting or interrupting the device operation currently being performed

■ Orders of issue of GPIB bus commands using DCL@ statements

Orders of issue of GPIB bus commands using DCL@ statements are summarized below.

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 Initialization of Devices by *RST Command

■ Format

*RST

■ Application example

WRITE @103:"*RST" Only the device at address 3 is initialized at level 3.

■ Explanation

The *RST (Reset) command, an IEEE 488.2 common command, is used to reset a specified device at level 3.

Generally, devices are set in various states using device-dependent commands (device messages). Among these commands, the *RST command is used to reproduce a known state of a device. Completion of device operation is invalidated like level 2.

■ Specification of device number in WRITE @ statement

The device at the specified address is initialized at level 3.

■ Items subject to device initialization

[1] Device-dependent functions and states:
The specified device is set in a known state irrespective of its history. (☞See the lists on the following pages.)

[2] *OPC command processing: The specified device is set in the OCIS (Operation Complete Command Idle State). The operation complete bit cannot be set in the standard event status register. (☞P7-7)

[3] *OPC? query processing: ..... The specified device is set in the OCIS (Operating Complete Command Idle State). The operation complete bit cannot be set in the output queue. The MAV bit is cleared. (☞P7-7)

[4] Macro command: ..... Macro operation is disabled, setting the state in which macro commands cannot be accepted. The designer can show macro definitions.

Notes:

*RST command does not affect the following items:

[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 device standard

[8] RS-232C interface condition

Table 4-1 lists MS9715B-dependent initial settings.
The "Set condition" column lists device's initial states set by the *RST command.
In the "Battery backup" column, items battery-backed-up after power-off are marked with ○.

Table 4-1 MS9715B-dependent initial settings

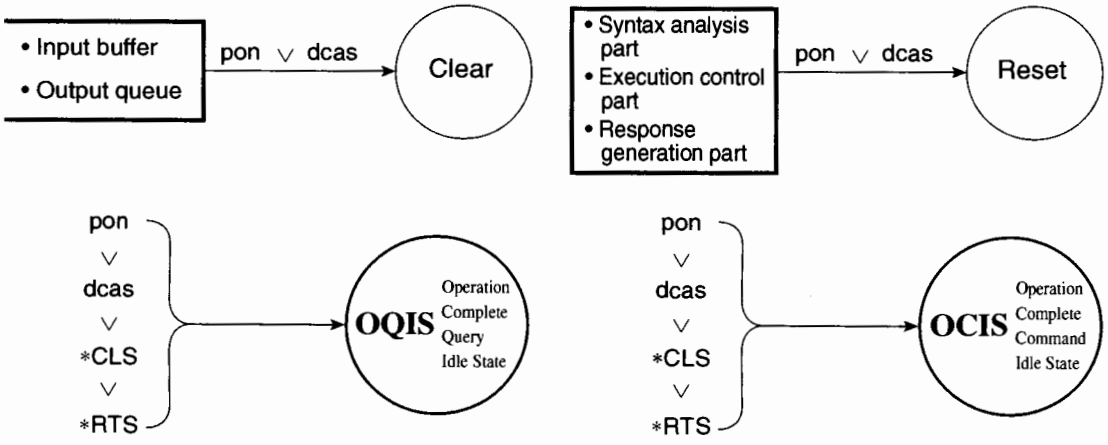
Item group	Item	Set condition	Battery backup
Wavelength	Center	1550 nm	○
	Span	200 nm	○
	Start	1450 nm	○
	Stop	1650 nm	○
Level Scale	Scale	Log	–
	Log/div	10 dB/div	–
	Reference Level	0 dBm	–
Res/VBW	Res	0.1 nm	○
	VBW	1 kHz	○
Save/Recall	File Option	File Option: None	○
		File ID: Number	○
		FDD Mode:1.44 M	○
Title			○
Others	Printer Prmtr	Device Address: 17	○
Status Register	Service request enable register	0 (All inhibited)	
	Standard event status enable register	0 (All inhibited)	
	Extended event status enable register	0 (All inhibited)	

4.4 Device States at Power-on

When the power is turned on:

[1] The MS9715B is restored to the last power-off state.
[2] The input buffer and output queue are cleared.
[3] Syntax analysis, execution control, and response parts are reset.
[4] The device is set in the OCIS (Operation Complete Command Idle State).
[5] The device is set in the OQIS (Operation Complete Query Command Idle State).
[6] The MS9715B does not support a *PSC command. So the standard event status register and standard event status enable register are cleared. Events are recorded after being cleared.

States [2] to [5] are set except when the power is turned on. The state diagram is shown below.



■ Items not changes at power-on

[1] Address

[2] Associated calibration data

[3] Data and states that change with the responses to the following common query commands

*IDN?	(☞P7-6)
*OPT?	(☞P7-9)
*PSC?	(Not supported by the MS9715B)
*PUD?	(Not supported by the MS9715B)
*RDT?	(Not supported by the MS9715B)

■ Items related to power-on status clear (PSC) flag

When the PSC flag is false, the service request enable register (☞P8-9), standard event status enable register (☞P8-11), and parallel poll enable register are not affected.

When the PSC flag is true or the *PSC command has not been executed, the above registers are not cleared.

(☞The PSC command is not supported by the MS9715B)

■ Items that change at power-on

[1] Current device function test

[2] Status information

[3] *SAV/*RCL register (Not supported by the MS9715B)

[4] Macro definition made with a *DDT command (Not supported by the MS9715B)

[5] Macro definition made with a *DMC command (Not supported by the MS9715B)

[6] Macro definition made with an *EMC command (Not supported by the MS9715B)

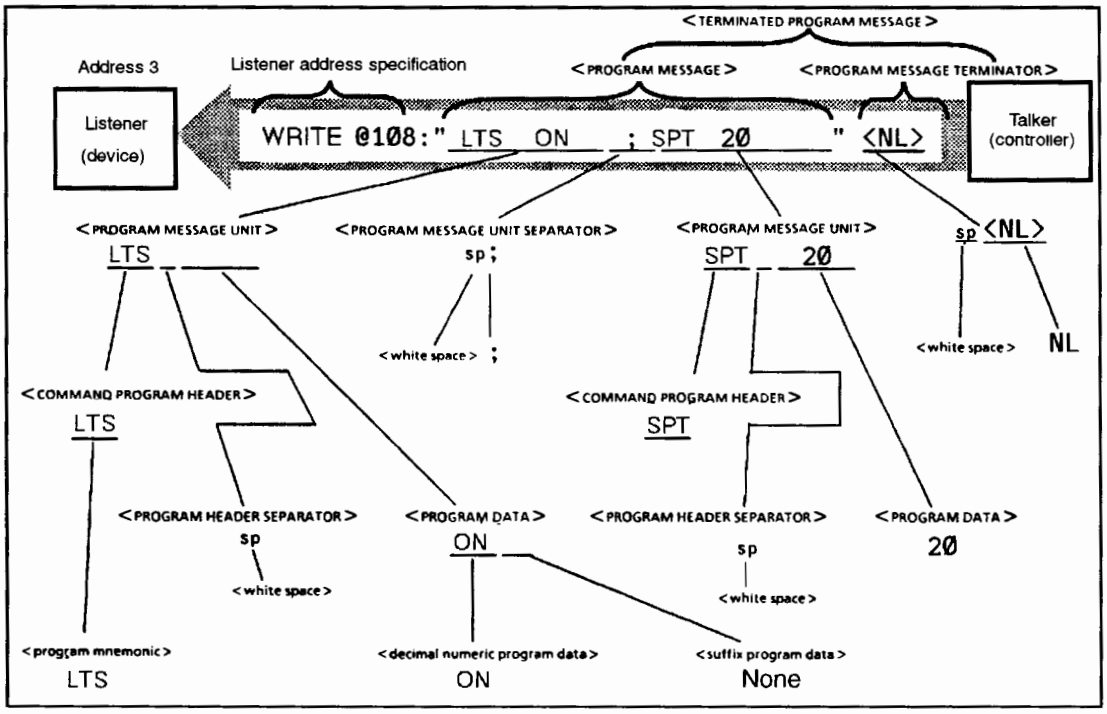
[7] Address received with a *PCB command (Not supported by the MS9715B)

Section 5 Listener Input Formats

Device messages transferred between the controller and devices are classified into program messages and response messages. This section explains the formats of the program messages received by listeners.

A program message is a sequence of program message units. Each unit is a program command or query.

The following figure shows that a program message made by connecting two program message units LTS ON and SPT 20 using a program message unit separator is sent from the controller to a device to select the L-T Test Screen and set the sampling time to 20 minutes.



A program message is a sequence of functional elements, the minimum units that can represent functions. In the above figure, functional elements are indicated by capital characters with them enclosed in < >. Functional elements are further classified into coding elements which are indicated by lowercase characters with them enclosed in < >.

The chart indicating the route of selection of functional elements is called a functional syntactical chart. The chart indicating the route of selection of coding elements is called a coding syntactical chart. On the following pages, program message formats are explained using these functional and coding syntactical charts.

Coding elements indicate coding of the actual bus which is required to send functional element data byte to a device. Upon receipt of a functional element data byte, the listener checks whether individual elements follow the coding syntax rules. If they do not follow the rules, the listener causes a command error without regarding the elements as functional elements.

5.1 Summary of Listener Input Program Message Syntactical Notation

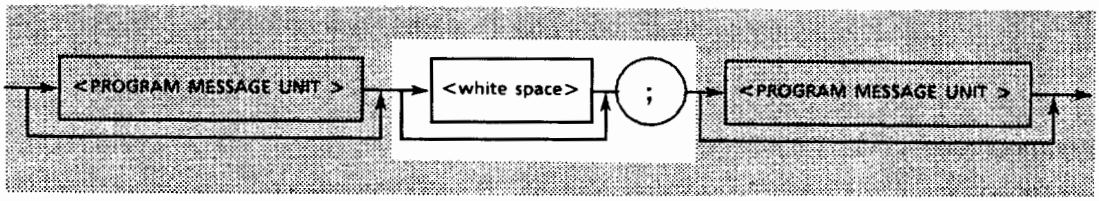
This section gives a general description of program messages functional units (☞P5-8) and program data formats (☞P5-19). (Compound commands and common commands are excluded.)

5.1.1 Separator, terminator, and space before header

(1) PROGRAM MESSAGE UNIT SEPARATOR

Link two or more program message units using zero or more spaces and a semicolon.

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



<Example 2> One space + Semicolon

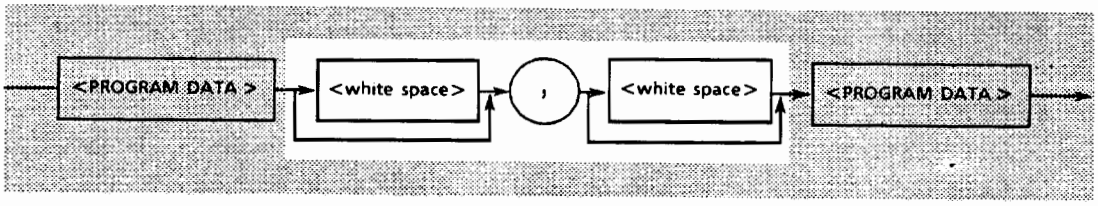
LTS ΔONΔ;SPTΔ20

Select the L-T Test screen and set the sampling time to 20 minutes.

(2) PROGRAM DATA SEPARATOR

When there are two or more pieces of program data, separate two contiguous pieces of program data using zero or more spaces, a command, and zero or more spaces.

<Example 1> General format for separating two pieces of program data



<Example 2> Comma only

TIMEΔ10, 15

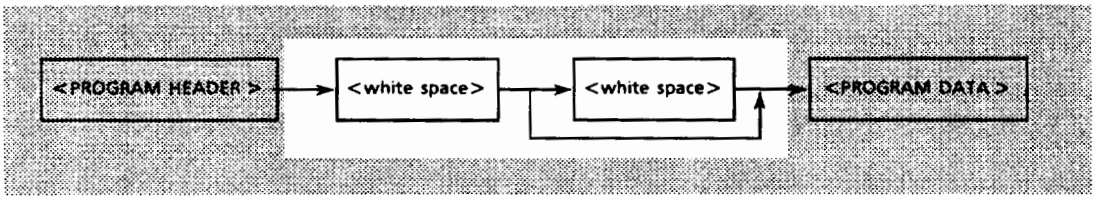
<Example 3> Comma + One space

TIME Δ10, Δ15 Set the times to 10:15.

(3) PROGRAM HEADER SEPARATOR

Separate a program header and program data using one space and zero or more spaces.

<Example 1> General format of simple command program header



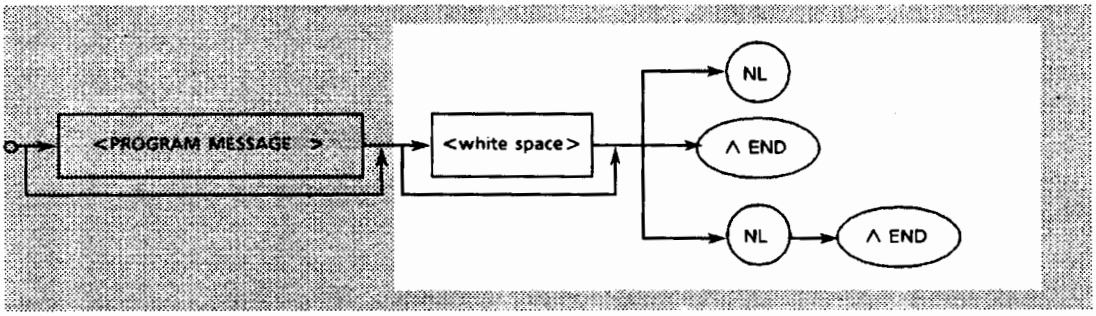
<Example 2> One space

SPT △20

(4) PROGRAM MESSAGE TERMINATOR

Add zero or more spaces and one of NL, EOI and a combination of NL and EOI at the end of a program message.

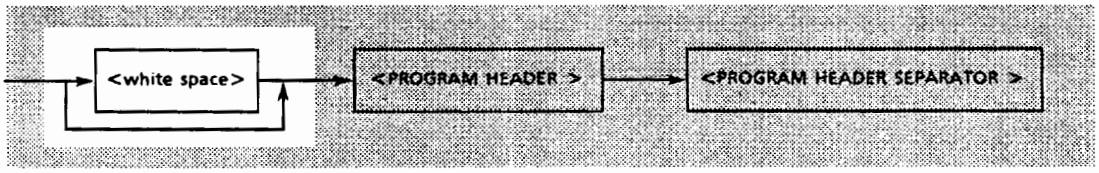
<General format>



(5) Space before header

Zero or more spaces can precede a program header.

<General format>



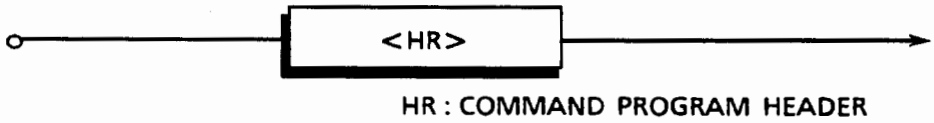
<Example> One space before second program header RLV

LTS ΔON ;△ SPT Δ 20

Select the L-T Test screen and set the sampling time to 20 minutes.

5.1.2 General format of program command message

(1) Message without data specification



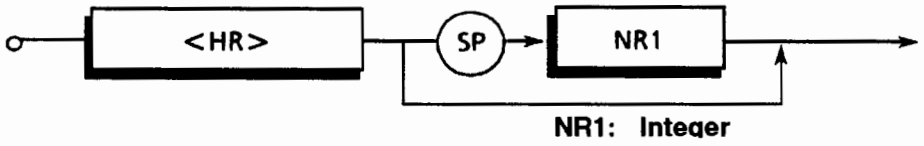
HR : COMMAND PROGRAM HEADER

<Examples>

SSI: Single sweep start

SPT: Start the repeat sweep.

(2) Message with integer data

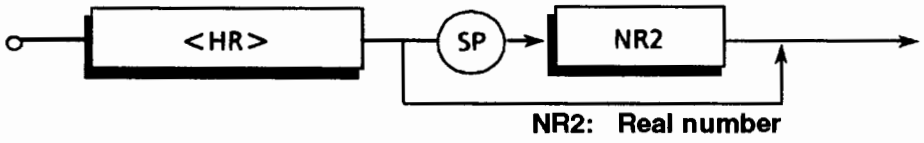


NR1: Integer

<Example>

SPT Δ20 Set the sampling time to 20 minutes.

(3) Message with real number

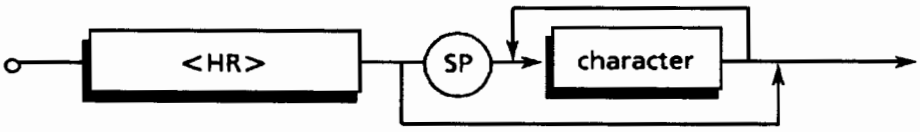


NR2: Real number

<Example>

ZMKWΔ1550.2, 10.5 Set the zoom center wavelength and zoom span width to 1550.2 nm and 10.5 nm, respectively.

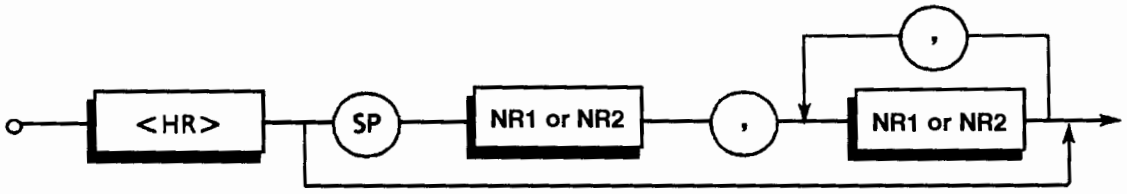
(4) Message with fixed or arbitrary character string data (data length ≤12 characters)



<Example>

SLPΔON Display in slope display.

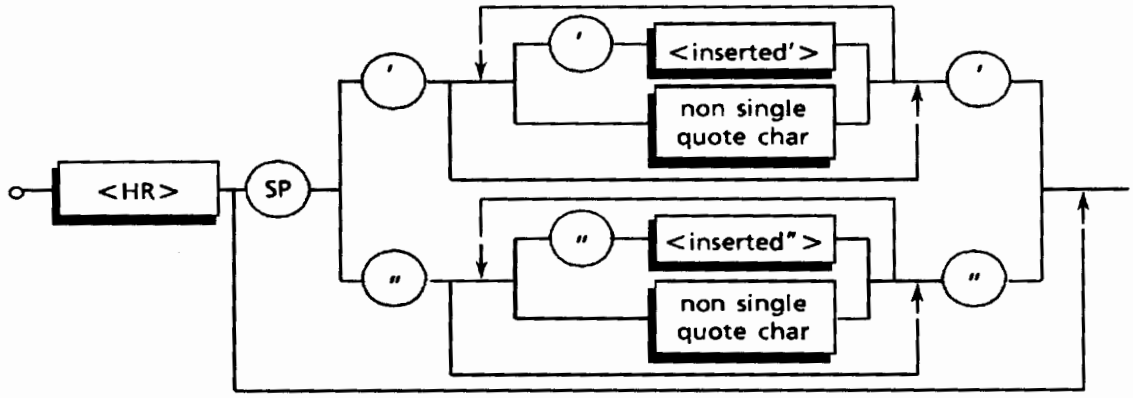
(5) Message with multiple pieces of program data (first: NR1)



<Example>

DATE△96, 10 Set the date to Oct. 10, 1996.

(6) Character-only message that can use all seven ASCII bits

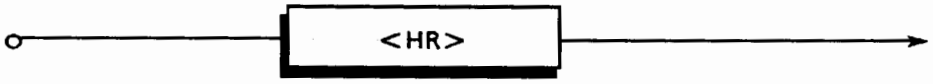


<inserted'>: A single ASCII code representing a value 27
non-single quote char: A single ASCII code representing a value other than 27
<inserted">: A single ASCII code representing a value 22
non-single quote char: A single ASCII code representing a value other than 22

5.1.3 General format of query message

Add ? at the end of a query program header.

(1) Message without query data specification



<Example>

ZMKW? Zoom marker value

(2) Message with query data specification

<Example>

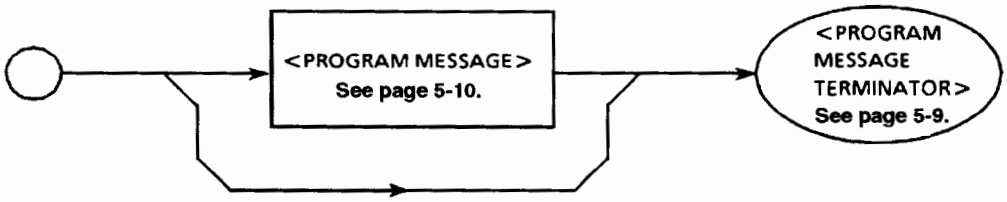
PMK?Δ1 Inquire marking ON/OFF of peak No.1.

5.2 Program Message Functional Elements

A device accepts a program message by detecting the terminator added at the end of the program message. Functional elements of the program message is described below.

5.2.1 <TERMINATED PROGRAM MESSAGE>

<TERMINATED PROGRAM MESSAGE> is defined as follows:

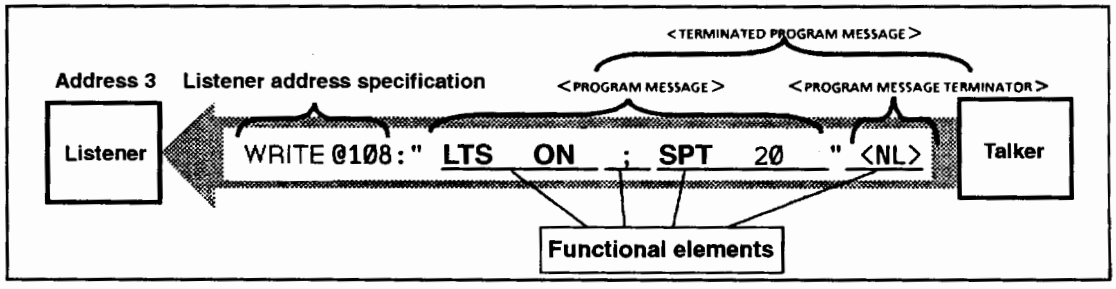


<TERMINATED PROGRAM MESSAGE> is a data message having all the necessary functional elements to be sent from a controller to a device.

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

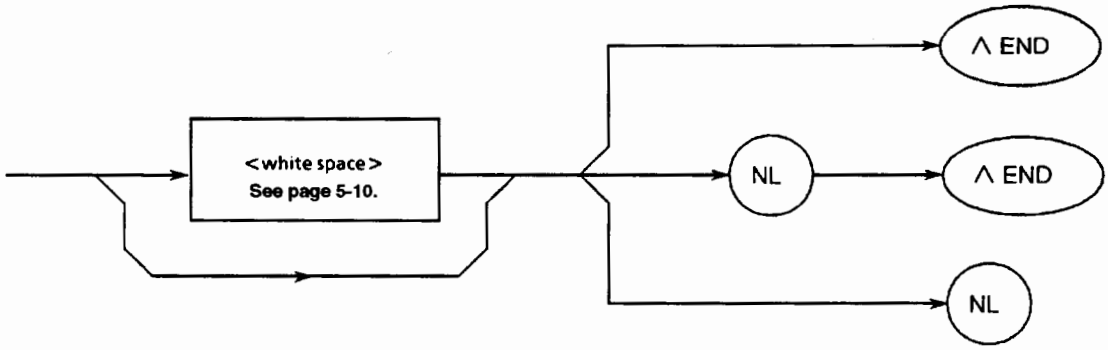
<Example>

<TERMINATED PROGRAM MESSAGE> for sending two pieces of commands with a WRITE statement



5.2.2 <PROGRAM MESSAGE TERMINATOR>

<PROGRAM MESSAGE TERMINATOR> is defined as follows:

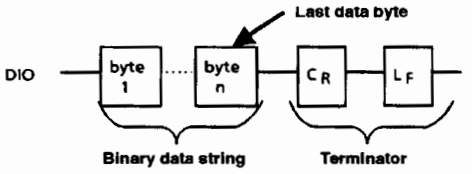


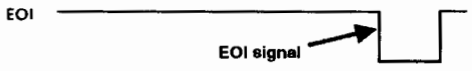
<PROGRAM MESSAGE TERMINATOR> terminates a sequence of one or more fixed-length <PROGRAM MESSAGE UNIT> elements.

NL: Defined as a single ASCII code byte 0A (decimal 10). That is, it is an ASCII control character LF (Line Feed) that moves the printing position down one line. As printing starts at a new line, it is also called NL (New Line). When sending <PROGRAM MESSAGE> with a WRITE@ statement, the WRITE@ statement automatically issues CR/LF. So the CR/LF codes need not be written in the program. To generate only the LF code, the following statement must be executed at the beginning of the program:

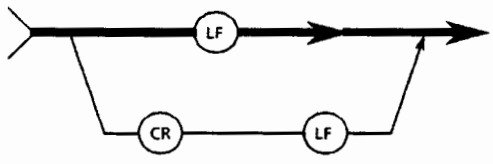
```
TERM IS CHR$ (10)
```

END: Sets the EOI line, one of GPIB control buses, at the LOW level (TRUE), generating an EOI signal.





An EOI ON/OFF statement can be used to control the EOI line. EIO OFF is the default (the EOI line is not controlled). If the EOI ON statement is executed in advance, an EOI signal is issued along with the terminator LF when the last byte of the WRITE@ statement is issued. It is also possible to terminate <PROGRAM MESSAGE> using only an END signal without generating an LF code.

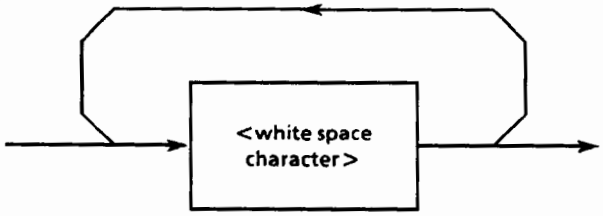


Note:

The CR code is used to return the printing position to the first character position on the same line; however, most listeners ignore it. Some products available on the market uses CR-LF code, so most controllers are so designed that CR and LF codes are issued in succession.

5.2.3 <white space>

<white space> is defined as follows:

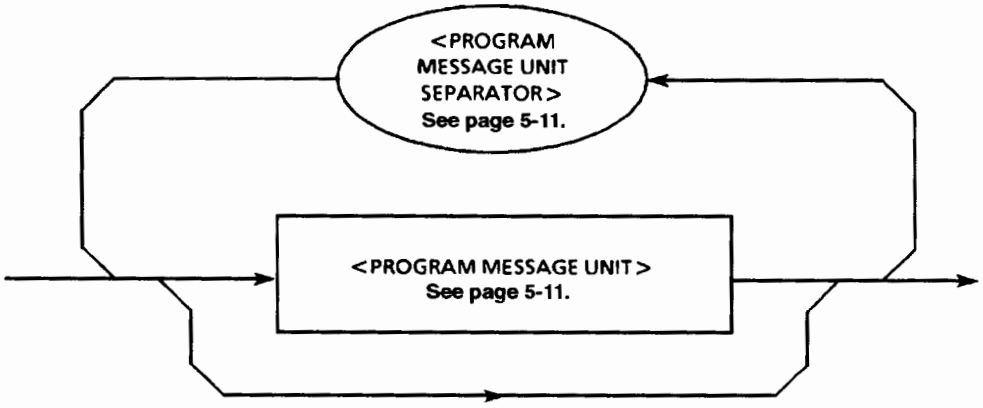


<white space character> is one of ASCII code bytes 00 to 09 and 0B to 20 (decimal values 0 to 9 and 11 to 32).

This range includes ASCII control codes and space signals (except NL). The device does not regard these codes as ASCII control codes, but it regards them as spaces or skips them.

5.2.4 <PROGRAM MESSAGE>

<PROGRAM MESSAGE> is defined as follows:



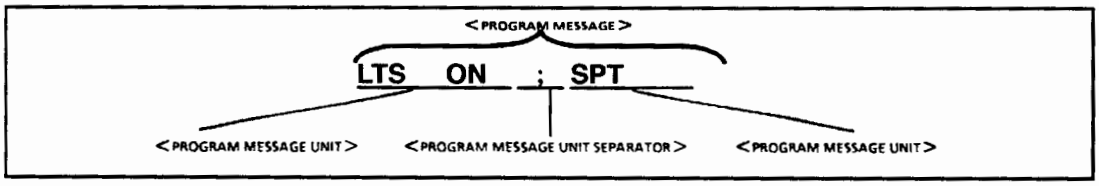
<PROGRAM MESSAGE> is zero, a <PROGRAM MESSAGE UNIT> element, or a sequence of <PROGRAM MESSAGE UNIT> elements. A <PROGRAM MESSAGE UNIT> element is a programming command or data which is sent from a controller to a device.

A <PROGRAM MESSAGE UNIT SEPARATOR> element is used to separate two or more <PROGRAM MESSAGE UNIT> elements.

<Example 1> Program message for setting to the L-T Test screen

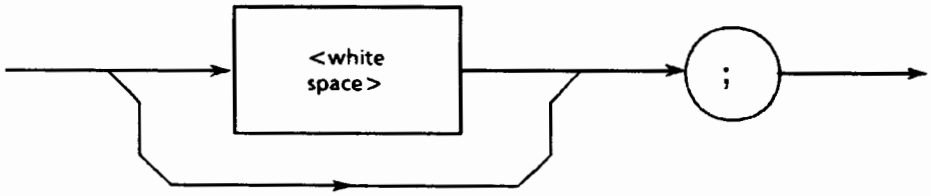
LTS ON

<Example 2> Program message for setting the sampling time to 20 minutes

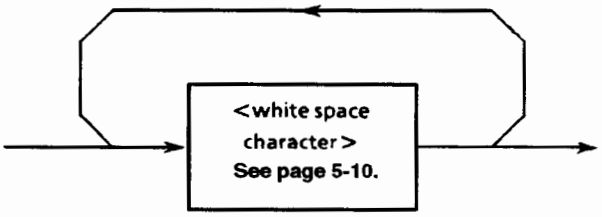


5.2.5 <PROGRAM MESSAGE UNIT SEPARATOR>

<PROGRAM MESSAGE UNIT SEPARATOR> is defined as follows:



<white space> is defined as follows:

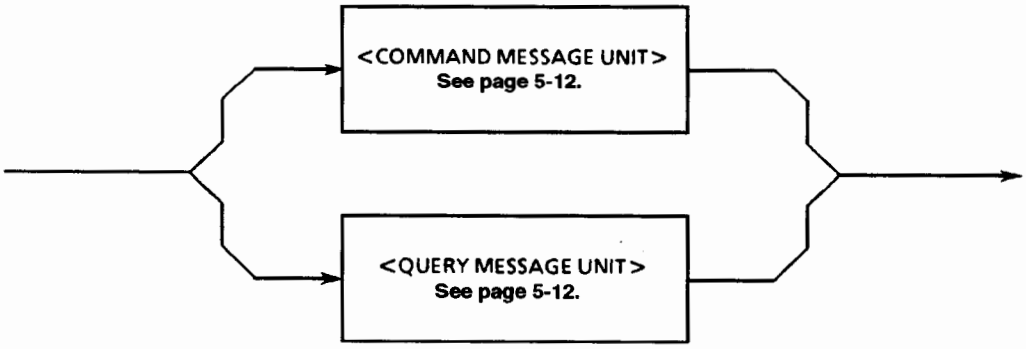


<PROGRAM MESSAGE UNIT SEPARATOR> divides a sequence of <PROGRAM MESSAGE UNIT> elements within the <PROGRAM MESSAGE> range.

A device interprets a semicolon (;) as the separator between <PROGRAM MESSAGE UNIT> elements. Accordingly, <white space character> before and after the semicolon are ignored. It should be noted that <white space character> improves program readability. <white space> following a semicolon is also used as a <white space> for the next program header. (☞See <Example 2> on the preceding page or page 5-13.)

5.2.6 <PROGRAM MESSAGE UNIT>

<PROGRAM MESSAGE UNIT> is defined as follows:



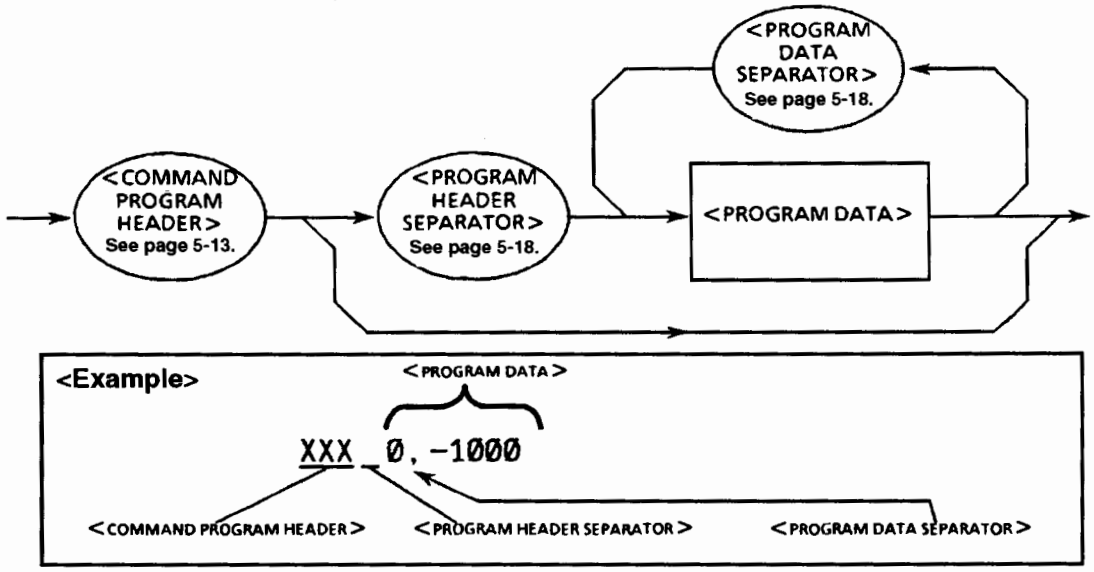
<PROGRAM MESSAGE UNIT> is a single command message received by a device.

It consists of <COMMAND MESSAGE UNIT> or <QUERY MESSAGE UNIT>, a single query message.

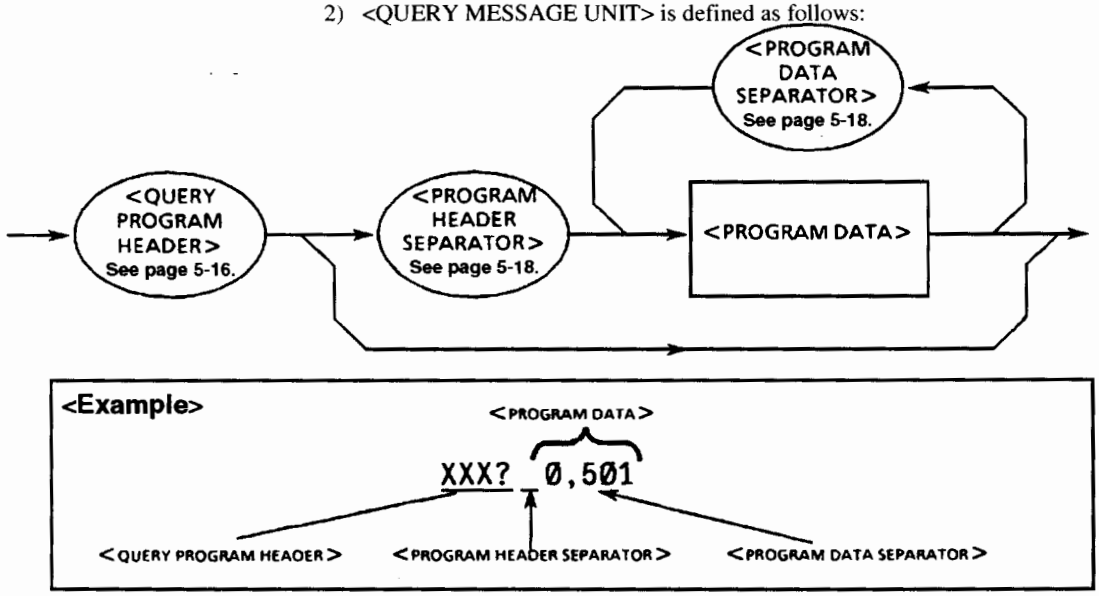
For details on <COMMAND MESSAGE UNIT> and <QUERY MESSAGE UNIT>, see the next page.

5.2.7 <COMMAND MESSAGE UNIT>/<QUERY MESSAGE UNIT>

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



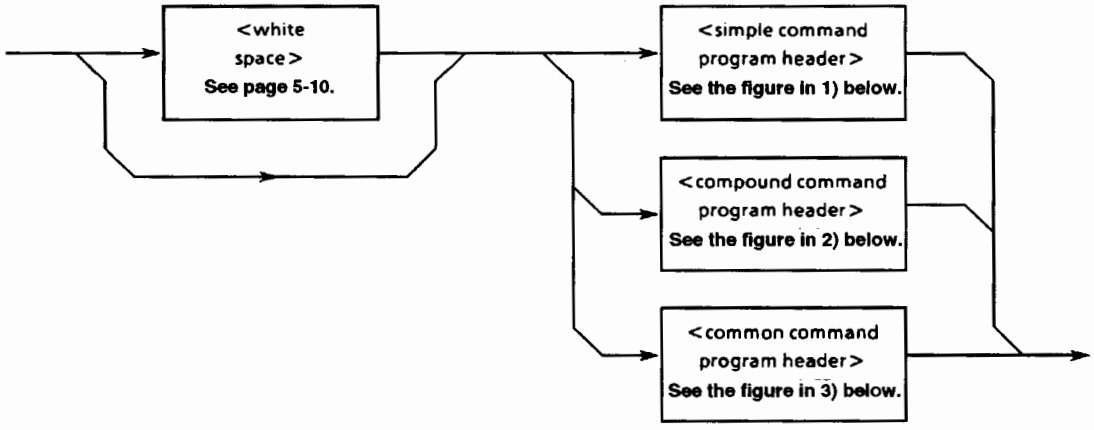
2) <QUERY MESSAGE UNIT> is defined as follows:



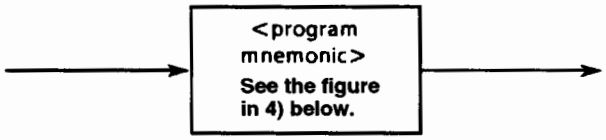
When a program header <COMMAND MESSAGE UNIT> or <QUERY MESSAGE UNIT> is followed by program data, a space is inserted between them. A program header indicates the application, function, and operation of the program. If a program header is not followed by program data, the program header solely indicates the application, function, and operation to be performed in the device. Among program headers, <COMMAND PROGRAM HEADER> is a control command issued from a controller to a device and <QUERY PROGRAM HEADER> is a query command that is issued from a controller to a device in advance so that the controller can receive responses from the device. These headers always end with a query indicator "?".

5.2.8 <COMMAND PROGRAM HEADER>

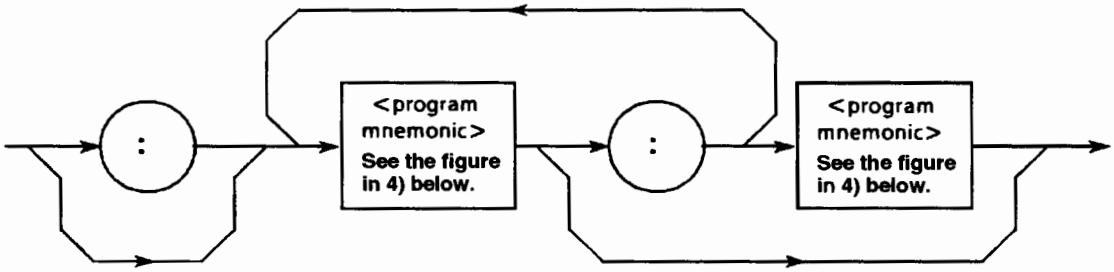
<COMMAND PROGRAM HEADER> is defined below.
Each header can be followed by <white space>.



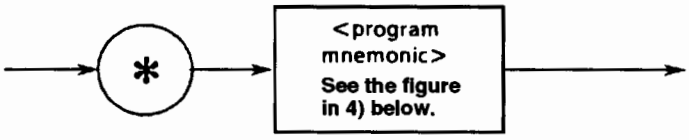
1) <simple command program header> is defined as follows:



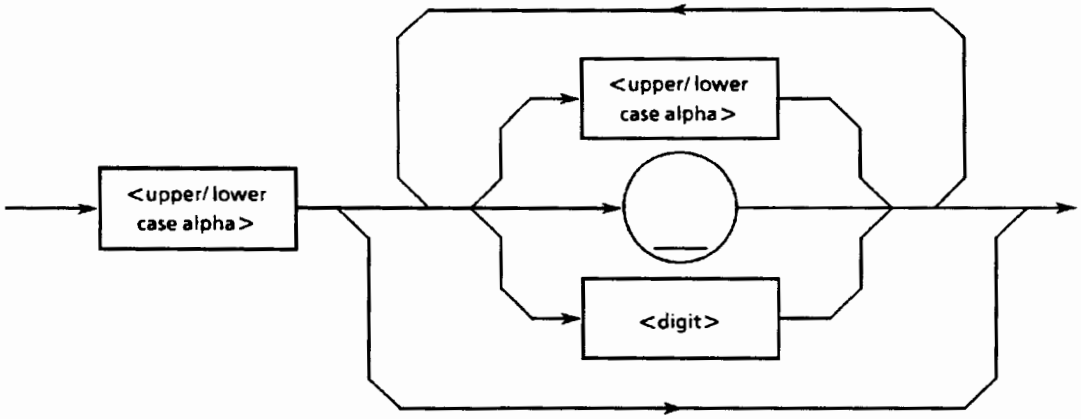
2) <compound command program header> is defined as follows:



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



4) <program mnemonic> is defined as follows:



■ <COMMAND PROGRAM HEADER>

This element indicates the application, function, and operation of the program data to be executed by the device. When it is not followed by program data, the header solely indicates the application, function, and operation to be performed in the device.

The meanings of an application, function, or operation is represented by <program mnemonic> which is widely called a mnemonic. Mnemonics and the command program headers defined in 1) to 3) above are explained below.

■ <program mnemonic>

A mnemonic begins with an uppercase or lowercase character, which is followed by an arbitrary combination of characters such as uppercase characters (A to Z) or lowercase characters (a to z), underline (_), and numeric characters (0-9). A mnemonic can contain a maximum of 12 characters; however, most mnemonics contain 3-4 characters. (No space is inserted between characters.)

- <upper/lower case alpha> — One of ASCII code bytes 41 to 5A and 61 to 7A (decimal values 65 to 90 and 97 to 122 = uppercase characters A to Z and lowercase characters a to z). The device can accept a header irrespective of whether it is represented by uppercase or lowercase characters.

- <digit> — One of ASCII code bytes 30-39 (decimal values 48 to 57 = characters 0-9).

- (_) — An ASCII code byte, i.e., ASCII code byte 5F (decimal value 95 = underline).

■ <simple command program header>

The above rules for <program mnemonic> applies. For example, the MS9715B uses "SSI" as a mnemonic indicating "sweep." It is also used as a "simple command program header" which means execution of sweep without program data. "ZMKW" is a mnemonic which means a zoom wavelength; however, it can be used as a "simple command program header" to set a zoom wavelength only when it is provided with the program data indicating a zoom waveform.

■ <compound command program header>

<compound command program header> is a command program header that executes a compound function. <program mnemonic> is always preceded by a colon (:) to separate it from <compound command program header>. When only one <compound command program header> is used, the succeeding colon (:) may be omitted.

The MS9715B does not support this compound command program header. However, it is explained here taking into account future extension.

● Function

On a complex device, a device command set is organized logically by providing a compound function instead of limiting the number of unique headers. A hierarchical command structure can be handled effectively.

● <Example 1>

To allow the MS9715B to use all device commands of another model (e.g., model MSXXXX), the compound program header would be
:MSXXXX

● <Example 2>

To allow the MS9715B to use a WXYZ device command of another model (e.g., model MSXXXX), the compound program header would be
MSXXXX:WXYZ or :MSXXXX:WXYZ

● <Example 3>

The name of a white buck rabbit living in a FOREST is WHITE.
The name of a white doe rabbit living in a GROVE is WHITE, too. If only WHITE is used as a command, we cannot distinguish between the above rabbits.
FOREST:WHITE or :FOREST:WHITE ··· White buck rabbit
GROVE:WHITE or :GROVE:WHITE ··· White doe rabbit

■ <common command program header>

An asterisk (*) is always added before <program mnemonic> of <common command program header>. "Common" means that this command is a program command which commonly used for other IEEE 488.2-ready measuring instruments connected to the bus.

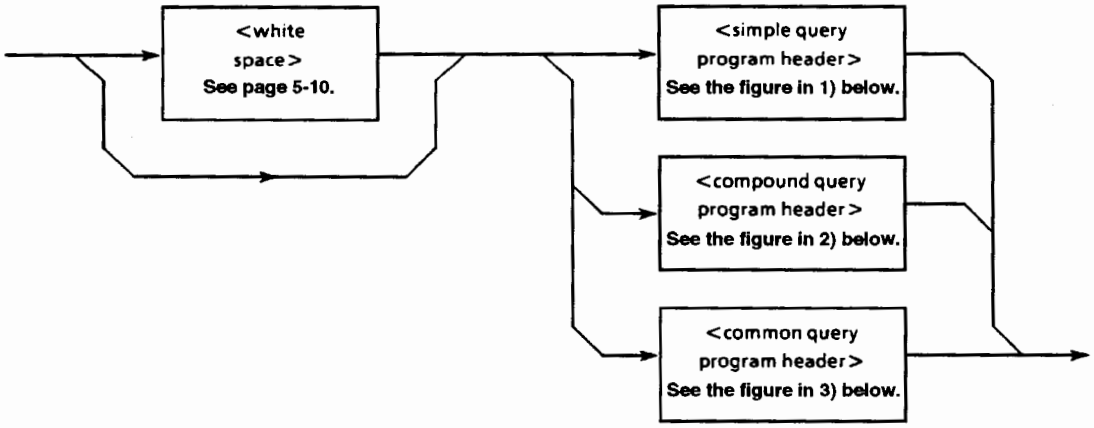
● <Example>

To idle completion of operation of the device at address 8, which is connected to the GPIB interface corresponding to select code 1, and restore devices to their initial states, the following common command is used:

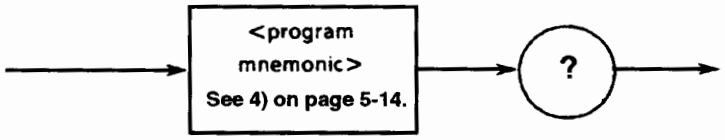
WRITE @108:"*RST" ..... The character string enclosed with quotation marks (" ") is an IEEE 488.2 common command *RST for executing the above processing.

5.2.9 <QUERY PROGRAM HEADER>

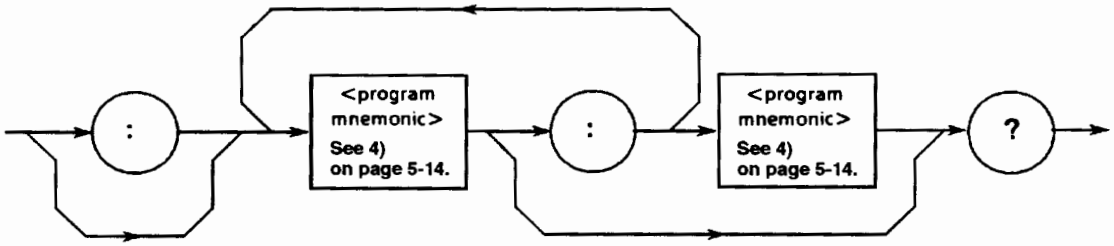
<QUERY PROGRAM HEADER> is defined as follows:
<white space> may be written before each header.



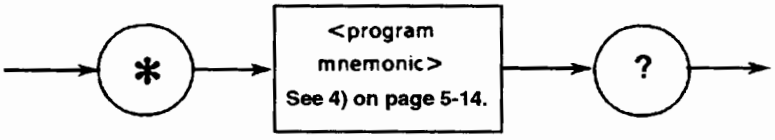
1) <simple query program header> is defined as follows:



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



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



■ <QUERY PROGRAM HEADER>

<QUERY PROGRAM HEADER> is a query command which is sent from a controller to a device in advance so that the controller can receive response messages from the device. This header always ends with a query indicator "?". It is explained below using examples of programs.

☞The format of <QUERY PROGRAM HEADER> is the same as that of <COMMAND PROGRAM HEADER> with the exception that a query indicator "?" is added at the end. See page 5-13.

● <Example 1> Setting and reading sampling time

```
10 WRITE @108:"SPT 20"
20 WRITE @108:"SPT?"!..... Query message SPT?
30 READ @108:A
40 PRINT A;"min"
```

Line 10: A command header SPT. for setting sampling time and a program message consisting of program data 1. 20 nm is set for the device.

Line 20: A program message that requires the device to send the set 20 nm to the controller. A query header "SPT?" is used.

Line 30: The listener device MS9715B that received the query header "SPT?" from the controller becomes a talker. The device is a controller that has become a listener, and it sends a response message 20 in response to SPT?. The listener reads the response message into the numeric variable A.

Line 40: The wavelength "20 nm" is displayed on the CRT. However, if HEAD ON is specified with a HEAD command, "SPT 20" is sent.

● <Example 2> Reading measurement data on 501 measuring points from memory A and printing the measurement data

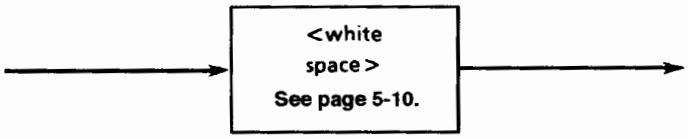
```
100 WRITE @108:"DMA?"
110 FOR K=0 TO 500
120 READ @108:DT(K)
130 PRINT DT(K);"dBm"
140 NEXT
150 END
```

Line 100: A query message "DMA?" is sent to the listener to store 501 pieces of data, starting at address 0.

Line 120: Line 100 causes the device to reply, response messages at points 0-500 are sent to the controller, and they are read into a numeric array variable DT (K).

5.2.10 <PROGRAM HEADER SEPARATOR>

<PROGRAM HEADER SEPARATOR> is defined as follows:



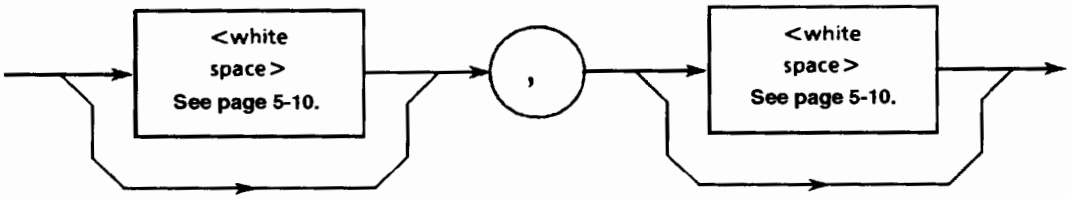
<PROGRAM HEADER SEPARATOR> is used as the separator between <COMMAND PROGRAM HEADER> (or <QUERY PROGRAM HEADER>) and <PROGRAM DATA>.

When there are two or more <white space character> elements between the program header and the program data, the first <white space character> is interpreted as a separator and the remaining <white space character> is ignored. It should be noted that <white space character> improves program readability.

That is, at least one header separator must exist between the header and the data. It indicates both the end of the program header and the beginning of the program data.

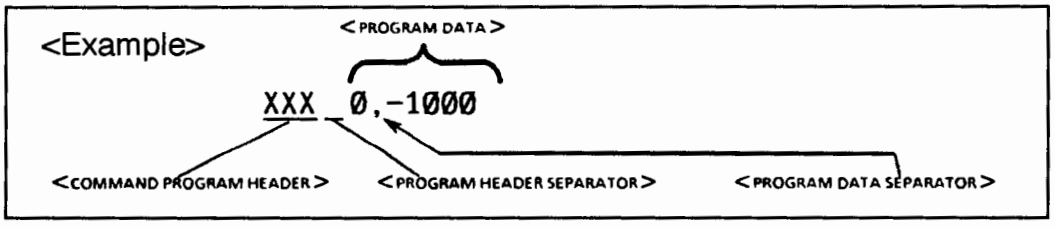
5.2.11 <PROGRAM DATA SEPARATOR>

<PROGRAM DATA SEPARATOR> is defined as follows:



When <COMMAND PROGRAM HEADER> or <QUERY PROGRAM HEADER> has many parameters, <PROGRAM DATA SEPARATOR> is used to separate them.

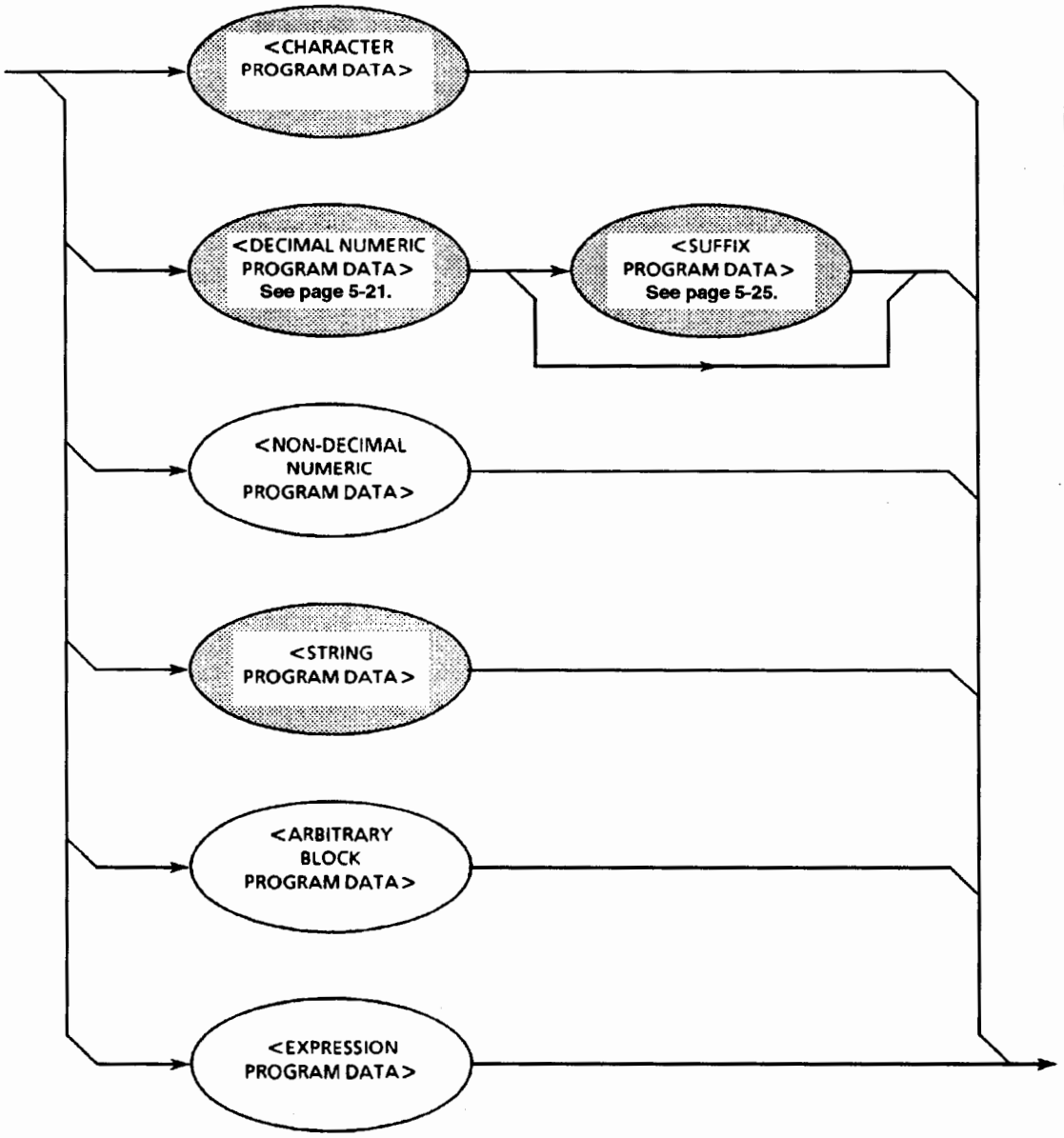
When this data separator is used, a comma is mandatory but <white space character> is omissible. The <white space character> before a comma and the <white space character> after a comma are ignored. It should be noted that <white space character> improves program readability.



5.3 Program Data Format

This section explains the format of the <PROGRAM DATA> shown in the functional syntactical charts (→ page 5-12), which is one of terminated program message formats.

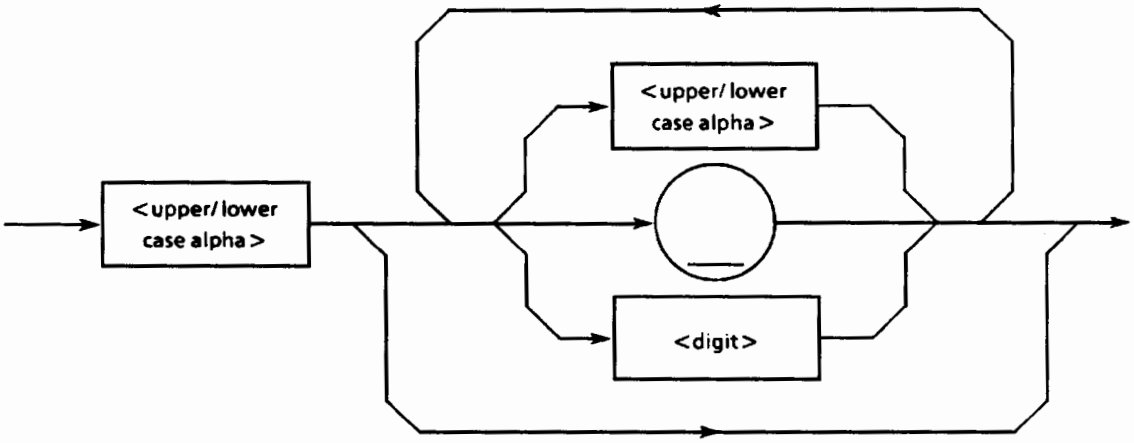
The functional element <PROGRAM DATA> is used to transfer various types of parameters related to the program header. Program data types are shown below. The MS9715B accepts the program data shown in the hollow squares surrounded by a shade. For the program data not supported by the MS9715B, read this section just for reference.



5.3.1 <CHARACTER PROGRAM DATA>

The functional element <CHARACTER PROGRAM DATA> is used to perform remote control by transferring short alphabetic or alphanumeric data. It is defined as follows:

Details on character data are the same as those on program mnemonics. So far, we discussed control data focusing on numeric data. However, program data can also be used to perform control. A coding syntactical chart is as follows:



Data always begins with an uppercase or lowercase character, which is followed by an arbitrary combination of characters such as uppercase characters (A to Z) or lowercase characters (a to z), underline (_), and numeric characters (0-9). Since combinations of alphanumeric characters are used as mnemonic-like symbols, the maximum data length is 12 characters.

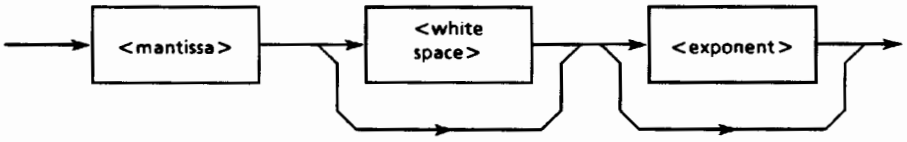
- <upper/lower case alpha> — One of ASCII code bytes 41 to 5A and 61 to 7A (decimal values 65 to 90 and 97 to 122 = uppercase characters A to Z and lowercase characters a to z). The device can accept a header irrespective of whether it is represented by uppercase or lowercase characters.
- <digit> — One of ASCII code bytes 30-39 (decimal values 48 to 57 = characters 0-9).
- (_) — A single ASCII code byte, i.e., ASCII code byte 5F (decimal value 95 = underline).

Therefore, <CHARACTER PROGRAM DATA> is program data used to transfer relatively short mnemonic-type alphanumeric codes.

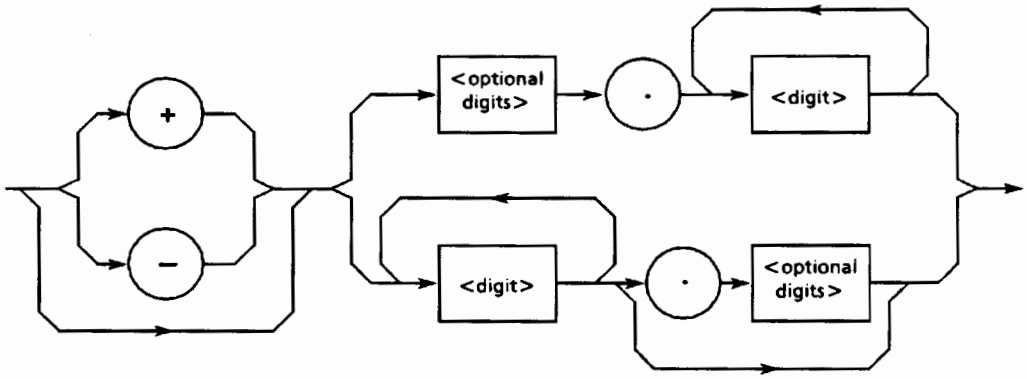
5.3.2 <DECIMAL NUMERIC PROGRAM DATA>

<DECIMAL NUMERIC PROGRAM DATA> is program data used to transfer numeric constants represented in decimal notation. There are three types of decimal numeric representation: integer, fixed- point, and floating-point.

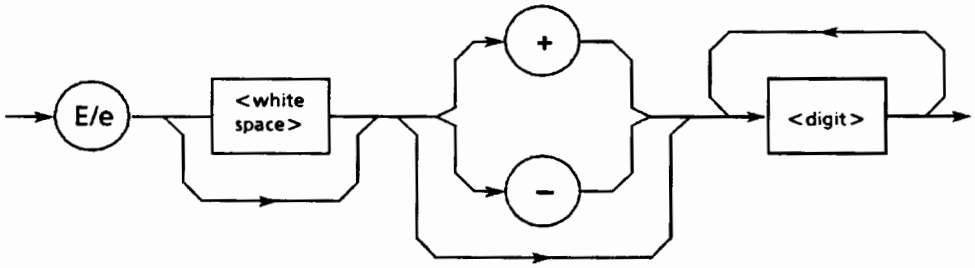
These three types of numerics represent decimal numeric program data, which can contain spaces, flexibly (NRF: flexible numeric representation), so they are defined as follows:



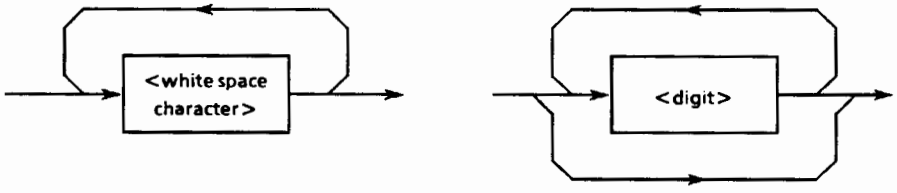
<mantissa> is defined as follows:



<exponent> is defined as follows:



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



For <white space>, see page 5-10. For <digit>, see page 5-20.

Let's take a look at coding syntactical charts of decimal numeric program data with respect to integer, fixed-point, and floating- point notations respectively.

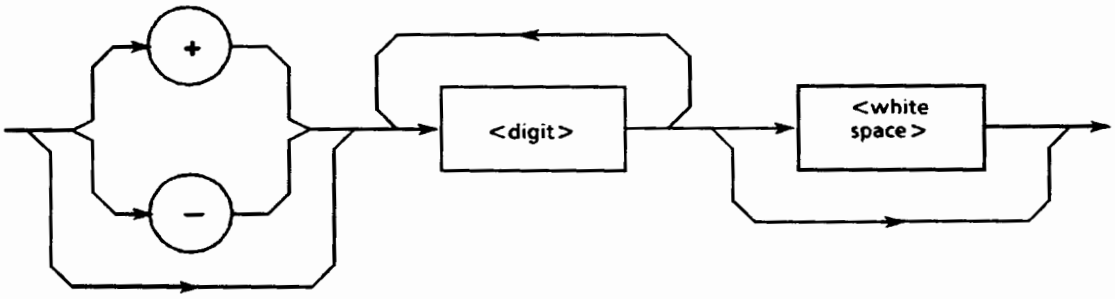
Note that the following processing is performed during transfer of any type of numeric representation:

● Rounding of numeric element: When a device receives a <DECIMAL NUMERIC PROGRAM DATA> element having too many digits to handle, it ignores the sign of the element value and rounds it off.

● Data outside the range: If the <DECIMAL NUMERIC PROGRAM DATA> element value is outside the range permitted in relation to the program header, an execution error is reported.

(1) Integer NR1 transfer

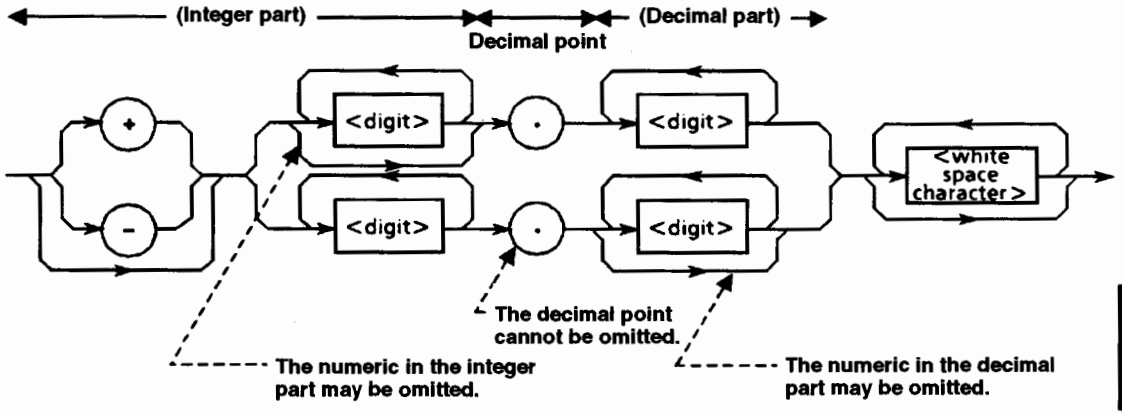
A decimal value not including a decimal point and exponent, i.e., an integer (NR1) in a real number, is transferred.



- 0 (s) may be added at the beginning. → 005, + 000045
- A space (+ or –) must not be inserted between a sign and a numeric. → +5, Δ5 (wrong)
- Spaces may be added after a numeric. → +5ΔΔΔ
- The + sign may be omitted. → +5, 5
- Commas must not be used to indicate decimal places. → 1,234,567 (wrong)

(2) Fixed-point NR2 transfer

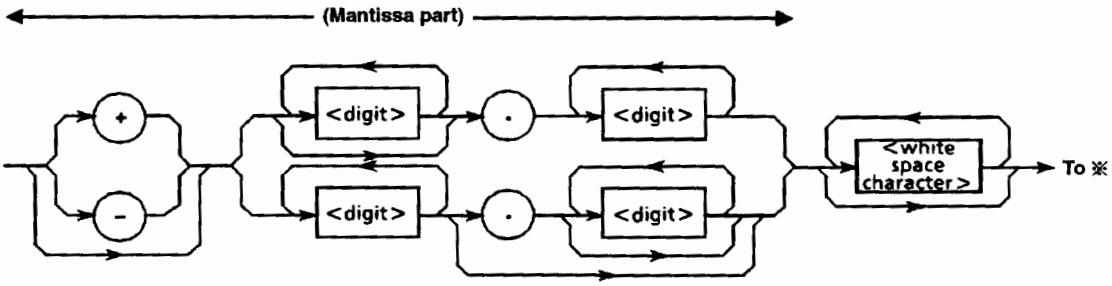
A decimal number having digits below the decimal point, i.e., an integer and a real number (NR2) except an exponent, is transferred.
The syntactical chart shows an integer part and a decimal point (and a decimal part).

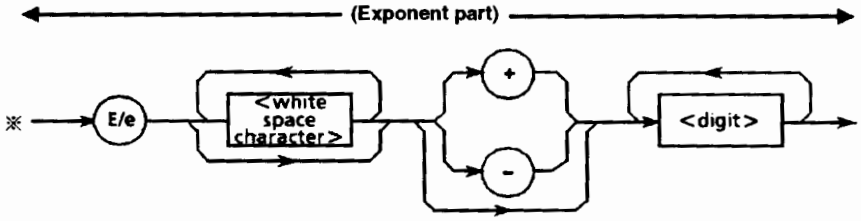


- An integer representation is applied to the integer part.
- A space must not be inserted between a numeric and a decimal point. → +753Δ.123 (wrong)
- Spaces may be added after the numeric in the decimal part. → +753.123 ΔΔΔΔ
- The decimal point need not follow a numeric. → .05
- A sign may be written before a decimal point. → +.05, −.05
- A numeric may end with a decimal point. → 12.

(3) Floating-point NR3 transfer

A decimal numeric having an exponent, i.e., a real number (NR3) represented in floating-point notation, is transferred. The syntactical chart consists of a mantissa part and an exponent part. The exponent part is represented in integer and floating-point notation to indicate precision of the numeric. The exponent part begins with E. On the right of E is a number to the power of 10.

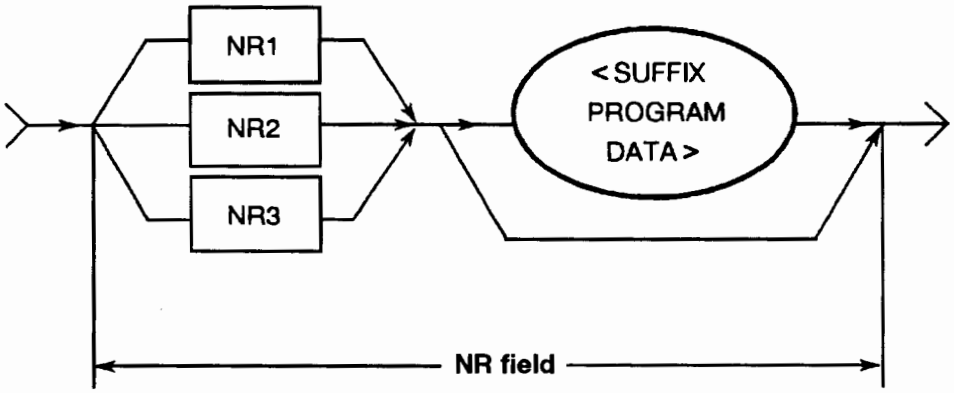




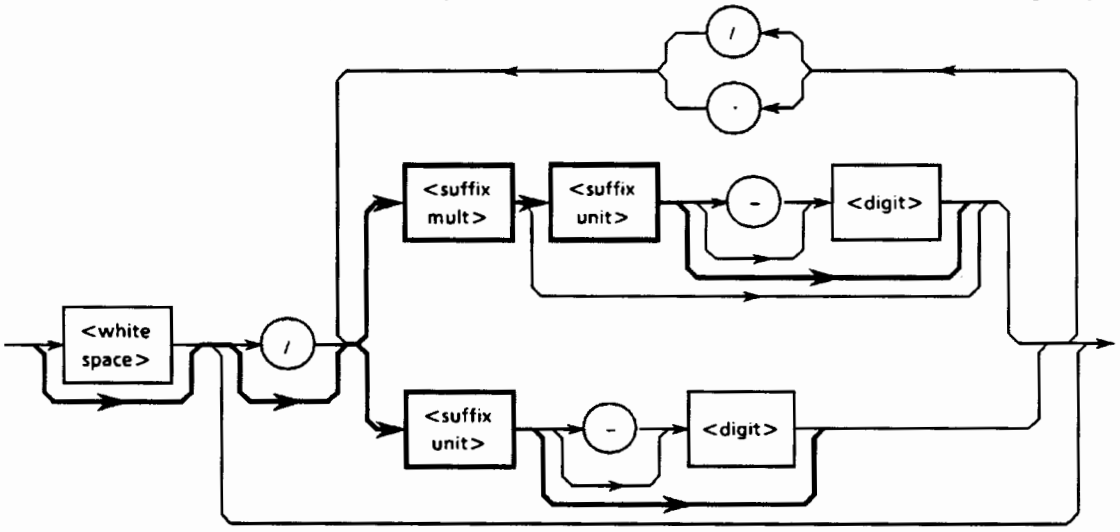
- E indicates power of 10. It indicates the beginning of the exponent part.
- E may be either an uppercase or lowercase character. → 1.234 E+12, 1.234 +12
- A space may be written before or after E/e. → 1.234ΔEΔ+12
- If the sign is +, it may be omitted in mantissa and exponent parts. → +1.234 E+4, 1.234E4
- The numeric in the exponent part cannot be omitted. → −1E2, −E2(×), −.E2(×)

5.3.3 <SUFFIX PROGRAM DATA>

<SUFFIX PROGRAM DATA> follows <DECIMAL NUMERIC PROGRAM DATA> (integer NR1, fixed-point NR2, or floating-point NR3). The NR1, NR2, and NR3 may be followed by a suffix.



A suffix is added at the end of decimal numeric program data only when the data requires a unit of measure. It is a combination of a suffix unit and a suffix multiplier. The syntactical chart is shown below. Bold-line routes are used frequently.



- A suffix multiplier is represented by an uppercase or lowercase character. For example, 1E3 Hz is represented by 1 kHz assuming 1E3 = k.
- A suffix unit is represented by an uppercase or lowercase character.
- Placing E at the beginning of <SUFFIX PROGRAM DATA> is prohibited because it may be confused with the E used for floating-point decimal numerics.

Suffix multipliers and units are listed in the table below.

(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:

According to convention, Hz to the sixth power of 10 is MHz (mega-hertz) and OHM to the six power of 10 is MOHM (megaohm). These are not listed in the above table, but they are listed in Table 5-2, "Suffix units."

(2) Relative units (dB)

- Decibel relative to 1 μV ..... DBUV
- Decibel relative to 1 μW ..... DBUW
- Decibel relative to 1 mW ..... DBMW

(3) Suffix units

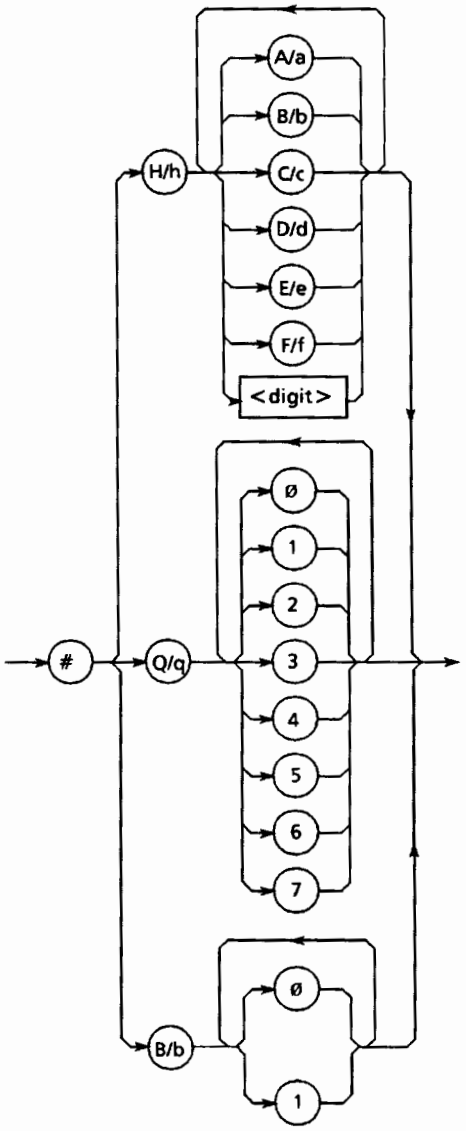
Table 5-2 Suffix units

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 transfer decimal, octal, and binary numeric data as non-decimal numeric values. Non-decimal data always begins with #. It is defined as shown in the coding syntactical chart below.

When an unspecified character string is sent, a command error occurs.



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

```
#Habc1230          (11,256,099D)
#hAbC123
#H2DC3             (11,715D)
#h2dc3
#H8301             (33,537D)
#h8301
```

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

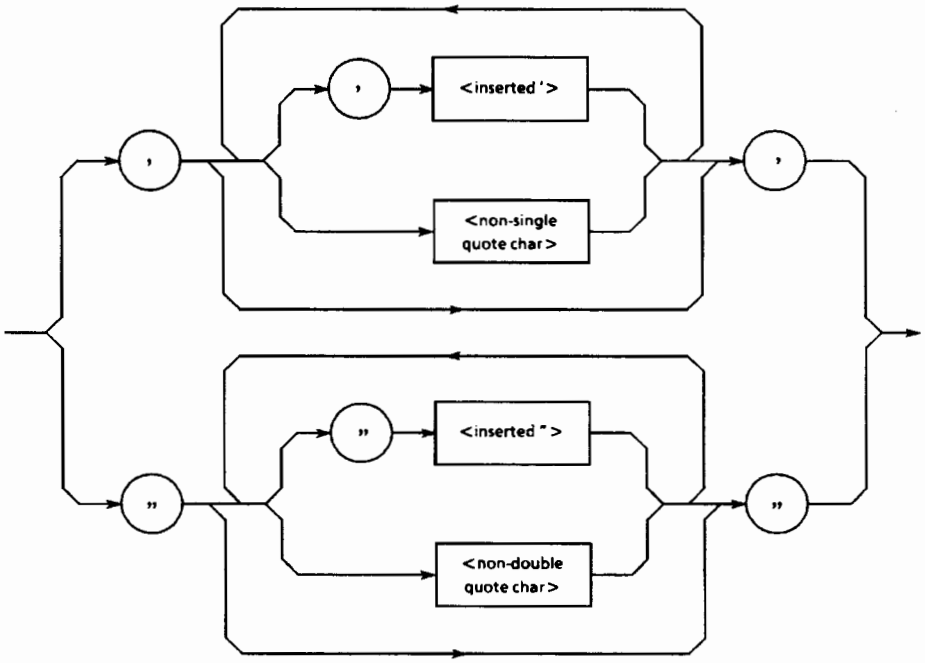
```
#Q37               (31D)
#q37
#Q26703            (11,715D)
#q26703
```

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

```
#B10101011110000010010011(11,256,099D)
#b00101110111000011        (11,715D)
```

5.3.5 <STRING PROGRAM DATA>

<STRING PROGRAM DATA> is program data consisting of only character strings. All ASCII 7-bit codes can be used. When a character string includes single or double quotation marks, two identical quotation marks must be written in succession per quotation mark.



- A character string must be enclosed with single or double quotation marks irrespective of whether the character string contains any quotation mark. For example,

  It's a nice day. → "It's a nice day."
  → 'It' 's a nice day.'

- When a character string is enclosed with single quotation marks, each single quotation mark contained in the character string must be doubled. Other characters, including double quotation marks, must be written as they are. For example,

  "I shouted, 'Shame'." → ' "I shouted, ' 'Shame' '." '

- When a character string is enclosed with double quotation marks, these double quotation marks must be doubled. Other characters, including single quotation marks, must be written as they are. For example,

  "I shouted, 'Shame'." → " " "I shouted, 'Shame'." " "

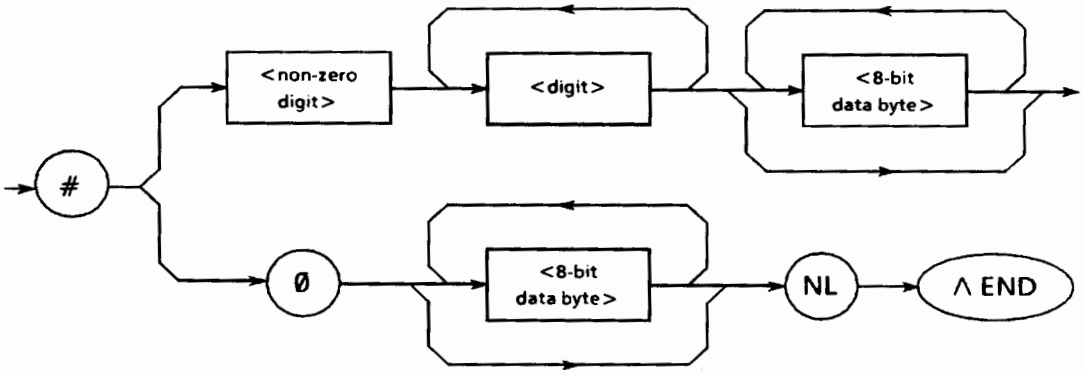
- <inserted '> is an single ASCII code set in ASCII code byte 27 (decimal 39 = symbol '). <inserted "> is a single ASCII code set in ASCII code byte 22 (decimal 34 = symbol "). <non-single quote char> and <non-double quote char> are single ASCII codes other than single and double quotation marks.

5.3.6 <ARBITRARY BLOCK PROGRAM DATA>

<ARBITRARY BLOCK PROGRAM DATA> is non-decimal program data starting with #. Binary data is transferred directly in 1-byte (8-bit) blocks. Differences from the non-decimal numeric program data (<NON-DECIMAL NUMERIC PROGRAM DATA>) mentioned on page 5-28 are as follows:

- Data is not limited to numeric data, but character string data and numeric data can be handled.
- The number of data bytes to be transferred can be written between # and the first data.

The non-decimal data is program data that can specify the data bytes to be transferred.



- <digit>
  One of ASCII code bytes 30-39 (decimal values 48-57 = characters 0-9).

- <non-zero digit>
  One of ASCII code bytes 31-39 (decimal values 49-57 = characters 1-9).

- <8-bit data byte>
  An 8-bit byte within the range from 00 to FF (decimal values 0-255).

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

The upper-right route in the above syntactical chart is applied.
Specify the number of <8-bit data byte> bytes to be transferred at the <digit> position, i.e., just before writing data. Write the number of digits of the specified number of bytes between # and <non-zero digit>. For example, to send four data bytes (DABs), write <ARBITRARY BLOCK PROGRAM DATA> as follows:

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

```
          ↓
# 14 <DAB> <DAB> <DAB> <DAB>
          ↑
```

The number of digits of the value 4 at the <digit> position is 4. So specify 1 at the <non-zero digit> position.

To send four bytes, specify 4 at the <digit> position. Leading 0s may be specified.

↓

3004 <DAB> <DAB> <DAB> <DAB>

↑

The number of digits of the value 4 at the <digit> position is 3. Specify 3 at the <non-zero digit> position.

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

The lower-right route in the syntactical chart on page 5-31 is applied. Write #0 before the first data and write NL^END after the last data, causing exitless termination.

0 <DAB> <DAB> <DAB> <DAB> <DAB> NL ^ END

If the following statements are specified for NL and ^END at the beginning of the program, then an EOI signal (END signal) is issued along with the terminator LF when the last byte has been transferred. (☞See page 5-9.)

- For NL, TERM IS CHR$ (10)
- For END, EOI ON

(3) Handling integer-precision binary data

Integer-precision binary data is used as <ARBITRARY BLOCK>-type transfer data, whether it is program data or response data, and has the specifications summarized below. Negative values are processed as two's complements.

Number of transfer bytes	1, 2, 4, or 8 bytes
Byte transfer order	Bytes are transferred sequentially, starting at the most significant byte.
Signed binary code	● LSD ... Right-justify ● MSB .. Sign bit ● When the data length is shorter than the field length, pad the remaining field with MSBs.
Unsigned binary code	● LSD ... Right-justify ● MSB .. Not a sign bit ● Pad unused high-order bits with 0s.

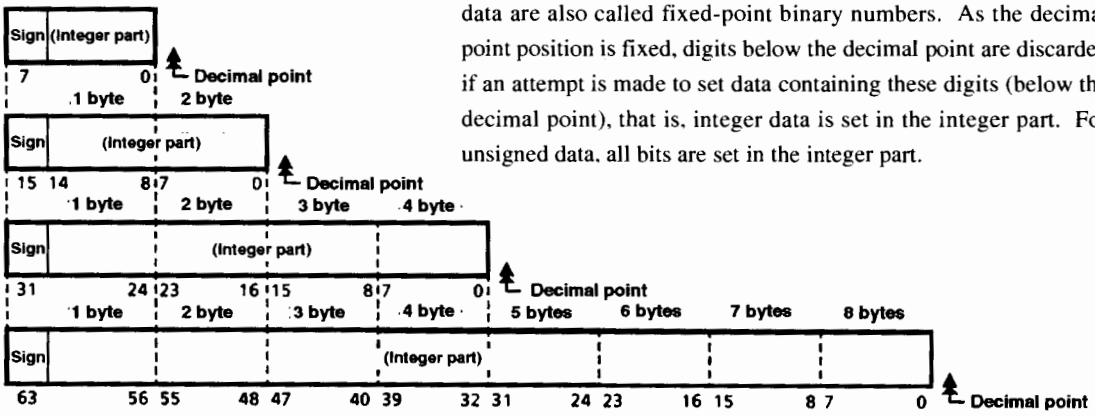
Ranges of signed and unsigned 1-byte (8-bit) and 2-byte (16-bit) integer data are shown below.

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

Internal representations of signed 1-, 2-, 3-, 4-, and 8-byte integer data are shown below. When the sign bit is 0, it indicates positive data. When a sign bit is 1, it indicates negative data.

The decimal point position is fixed at the right of the LSB bit, these data are also called fixed-point binary numbers. As the decimal point position is fixed, digits below the decimal point are discarded if an attempt is made to set data containing these digits (below the decimal point), that is, integer data is set in the integer part. For unsigned data, all bits are set in the integer part.



(4) Floating-point binary data

Floating-point binary data, whether it is program data or response data, is used as <ARBITRARY BLOCK>-type transfer data. Our products do not support floating-point binary data; however, general specifications are explained below.

Floating-point binary data must consists of the following three fields:

[1] Sign field (sign bit)
[2] Exponent field (exponent bit)
[3] Mantissa field (mantissa bit)

Numeric data having a decimal point is handled here. It has two types of precision: single precision and double precision. Field structures and transfer orders are shown below. Meanings of symbols are as follows:

S: Sign bit
EM: Most significant exponent bit
EL: Least significant exponent bit
FM: Most significant mantissa bit
FL: Least significant mantissa bit

Single precision — 4 bytes

Transfer byte	DIO line							
	8	7	6	5	4	3	2	1
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

Transfer byte	DIO line							
	8	7	6	5	4	3	2	1
1st byte	S	EM	E	E	E	E	E	E
2nd byte	E	E	E	EL	FM	F	F	F
3rd to 7th 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 sends the expression for obtaining a scalar, vector, matrix, or string value to a device, allowing the device to calculate a value in place of the controller. Its coding syntactical chart is as follows:

- <expression>:
  A sequence of ASCII characters represented by ASCII code bytes 20-7E (decimal values = 32-126), excluding the following six characters in [ ]:
  [" # ' ( ) ;]
  That is, a double quotation mark, number code (sharp), single quotation mark, left parenthesis, right parenthesis, and semicolon are excluded.

If a+b+c is written as <expression>, then the above syntactical chart will be expressed as

(a+b+c)

To transfer this to a device, program data discussed on pages 4-16 to 4-35 can be used with the exception of the <INDEFINITE LENGTH ARBITRARY BLOCK PROGRAM DATA>. Upon receipt of (<expression>), the device obtains the solution to this expression.

Note:

The MS9715B does not support the <expression> function. If calculation of an expression is required, the solution to the expression must be obtained by the controller and the resultant numeric data must be transferred to the device as program data.

Section 6 Talker Output Format

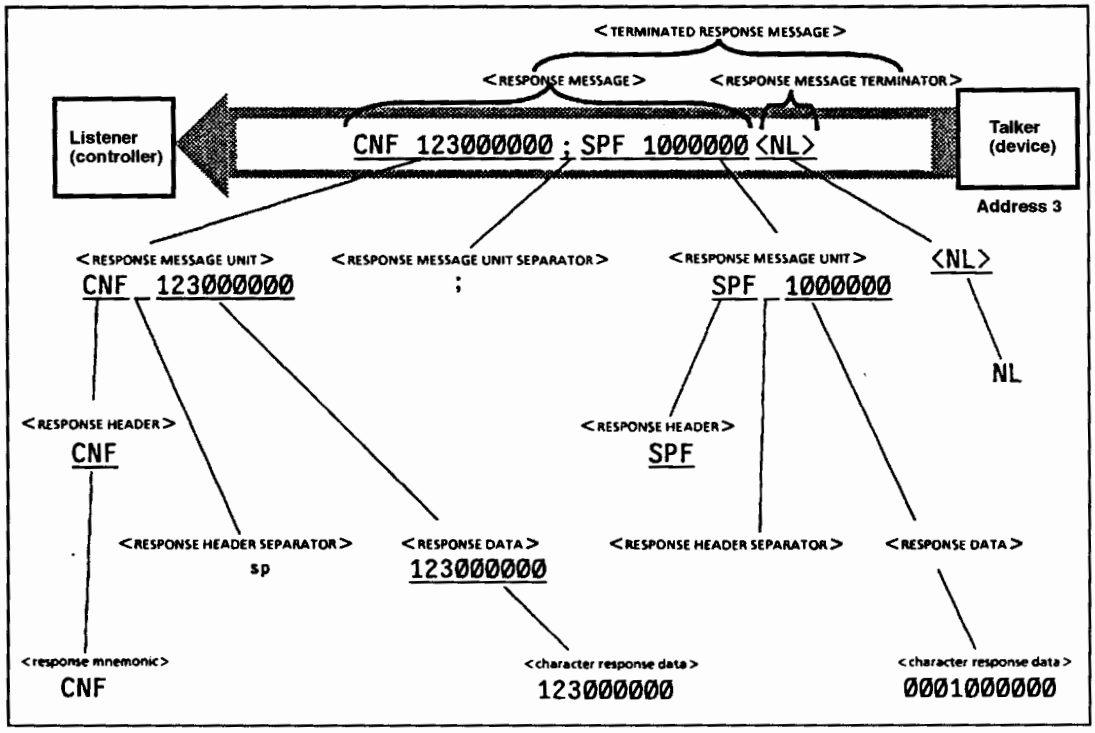
Device messages transferred between the controller and devices are classified into program messages and response messages. This section explains the formats of the program messages sent from a talker to a listener.

Note:

In this section, CNF? and SPF? are used to explain talker output formats. The MS9715B does not support these commands.

Typical response messages are: measurement result, setting, and status information. Response messages are classified into those with header and those without header.

The following figure shows that messages, ASCII character strings with header, are sent from a device to a controller in response to a center frequency message unit CNF? and a span frequency response message unit SPF?.



Only the operation-related parts is programmed as follows:

```
100  WRITE   @103 :          "CNF?; SPF?"! Center and span frequency query message
110  READ    @103 : A$! ←    When a terminator NL is detected, a response message "CNF
                             123000000; SPF 1000000" is read into A$.
```

A response message is a sequence of functional elements, the minimum units that can represent functions, as is the case with the program message. In the above figure, functional elements are indicated by uppercase characters with them enclosed in < >. Functional elements are further classified into coding elements which are indicated by lowercase characters with them enclosed in < >.

Let's take a look at talker output formats focusing on the differences from listener input formats.

6.1 Differences in Syntax between Listener Input Formats and Talker Output formats

Significant differences in syntax between the listener and the talker are as follows:

- Listener format:
  Program can be written flexibly so that devices can accept program messages from the controller. If a program message involves some description errors, it can execute its function normally. For example, you can joint as many <white space> elements as you want to make an easy-to-read program.

- Talker format:
  Messages are output following strictly defined syntactical rules to allow the controller to accept the response messages from the device. Therefore, the syntax of response messages permits only one notation for a function.

The table below summarizes the differences in output format between the listener and the talker. In this table, "0/1 or more spaces" means <white space>.

Item	Listener input program message syntax	Talker output response message syntax
Characteristic	(Flexible)	(Strict)
Alphabetic characters	No difference between uppercase	Uppercase characters only and lowercase characters
Character before and after NR3 exponent part E	0 or more spaces + E/e + 0 or more spaces	Uppercase character E only
+ sign of NR3 exponent part	Omissible	Required
<white space>	Two or more white spaces can be written before/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
Space before header	0 or more spaces + Header	Header only
Header separator	Header + 1 or more spaces	Header + One $20*
Data separator	0 or more spaces + Comma + 0 or more spaces	Comma only
Terminator	0 or more spaces + One of NL, EOI, and NL+EIO	NL+EOI

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

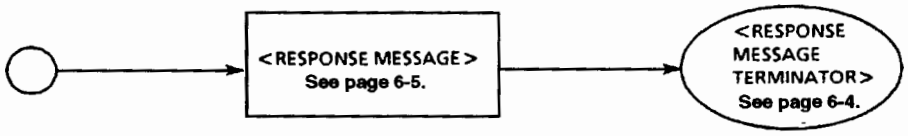
6.2 Response Message Functional Elements

Response messages output from a talker are terminated with an NL^END signal, allowing the controller to accept them. Functional elements of these response messages are explained here.

Rules for syntactical chart notation are the same as those for program messages, so see section 5. Functional and coding elements which are the same as those of program messages are not explained in this section, so see section 5 for them.

6.2.1 <TERMINATED RESPONSE MESSAGE>

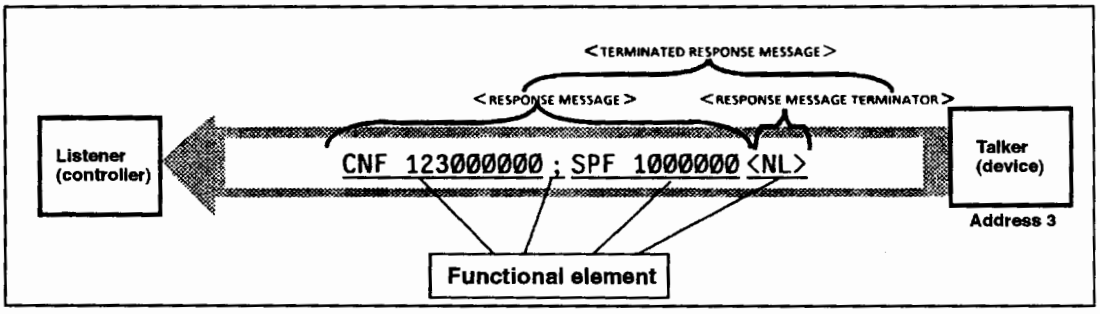
<TERMINATED RESPONSE MESSAGE> is defined as follows:



<TERMINATED RESPONSE MESSAGE> is a data message having all the necessary functional elements to be sent from a talker to a device.

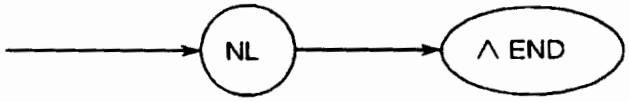
To complete transfer of <RESPONSE MESSAGE>, <RESPONSE MESSAGE TERMINATOR> is added at the end of <RESPONSE MESSAGE>.

<Example> <TERMINATED RESPONSE MESSAGE> in which two message units are connected



6.2.2 <RESPONSE MESSAGE TERMINATOR>

<RESPONSE MESSAGE TERMINATOR> is defined as follows:



<RESPONSE MESSAGE TERMINATOR> is placed after the last <RESPONSE MESSAGE UNIT> to terminate the sequence of one or more fixed-length <RESPONSE MESSAGE UNIT> elements.

If the following statements are specified for NL and ^END at the beginning of the program, then an EOI signal (END signal) is issued along with the terminator LF when the last data byte has been transferred. (☞See page 5-9.)

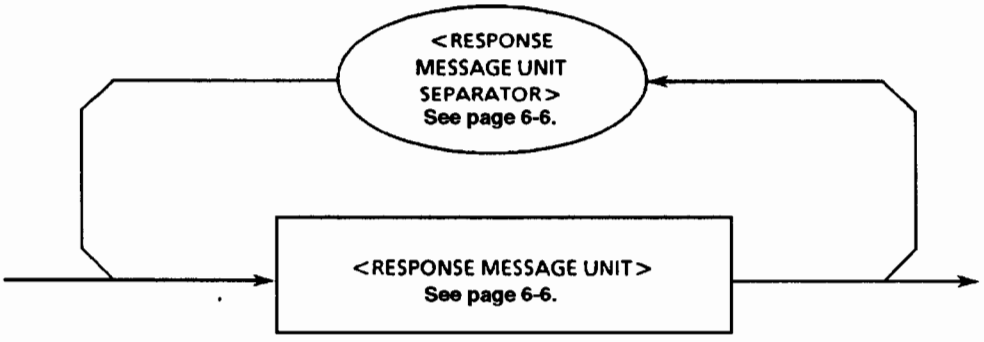
- For NL, TERM IS CHR$ (10)
- For END, EOI ON

<Example> Reading the currently set center frequency

```
10 LET ADR=101
20 TERM IS CHR$ (10)! ..... Specify LF (New Line) as a terminator code.
30 EOI ON! ..... Output a EOI signal for making the EOI line true when the last data byte has been transferred.
40 WRITE @ADR : "CNT?"! ..... Center wavelength read query
50 READ @ADR : A$! ..... Terminate response data read with an EOI signal.
60 PRINT A$
70 END
```

6.2.3 <RESPONSE MESSAGE>

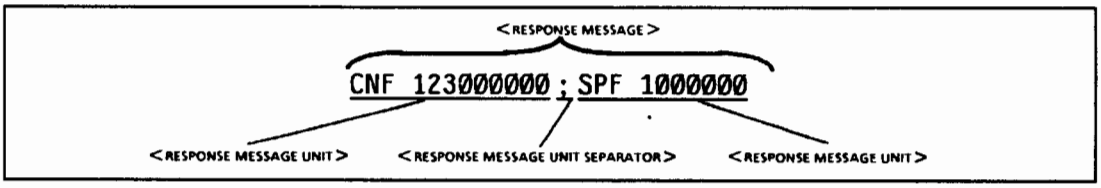
<RESPONSE MESSAGE> is defined as follows:



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

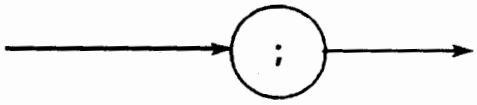
The <RESPONSE MESSAGE UNIT> element is a single message sent from a device to a controller. A <RESPONSE MESSAGE UNIT SEPARATOR> is used as a separator for separating multiple <RESPONSE MESSAGE UNIT> elements.

<Example> Adding CNF to the center frequency, adding SPF to the response data, and transferring them using a I-character fixed format



6.2.4 <RESPONSE MESSAGE UNIT SEPARATOR>

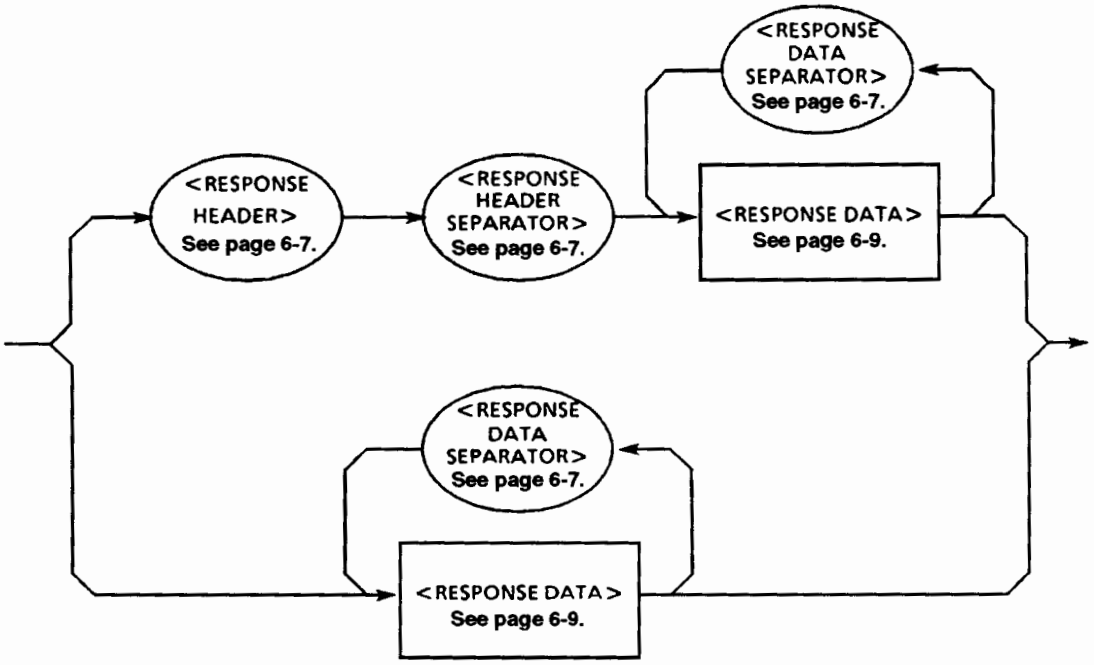
<RESPONSE MESSAGE UNIT SEPARATOR> is defined as follows:



<RESPONSE MESSAGE UNIT SEPARATOR> is used to separate <RESPONSE MESSAGE UNIT> elements with a <UNIT SEPARATOR> (semicolon (;)) when outputting a sequence of multiple <RESPONSE MESSAGE UNIT> elements as one <RESPONSE MESSAGE>.

6.2.5 <RESPONSE MESSAGE UNIT>

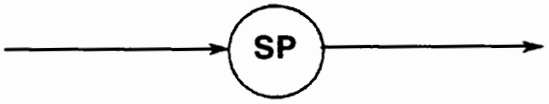
<RESPONSE MESSAGE UNIT> is defined as follows:



One is a response message unit with header, which returns the result of processing the program-message-set information accurately. The other is a response message unit without header, which returns only the measurement result.

6.2.6 <RESPONSE HEADER SEPARATOR>

<RESPONSE HEADER SEPARATOR> is defined as follows:



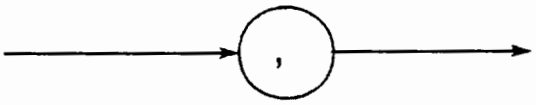
<RESPONSE HEADER SEPARATOR> is a space written after <RESPONSE HEADER> to be separated from <RESPONSE DATA>.

The space SP corresponds to ASCII code byte 20 (decimal 32).

In a response message with header, a space must always exist between the header and the data as a response header separator. It indicates the end of the header and the beginning of response data at the same time.

6.2.7 <RESPONSE DATA SEPARATOR>

<RESPONSE DATA SEPARATOR> is defined as follows:



When multiple <RESPONSE DATA> elements are to be output, <RESPONSE DATA SEPARATOR> must be placed between them.

6.2.8 <RESPONSE HEADER>

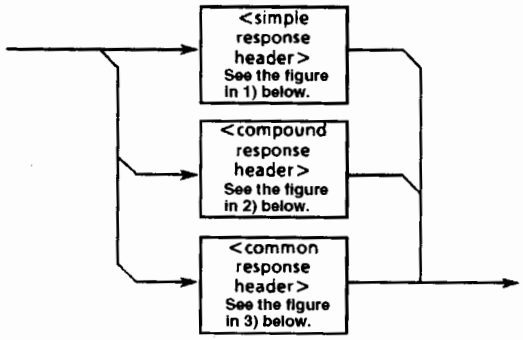
The format of <RESPONSE HEADER> is the same as that of <COMMAND PROGRAM HEADER> stated on pages 5-9 and 5-13 with the exception of the following three points:

[1] Characters that can be used in <response mnemonic> are specified. For alphanumeric characters, only uppercase characters must be used. Other points are the same as those of <program mnemonic>.
[2] A space cannot be written before a response header while it can be written before a program header.
[3] Only one space can be written before a response header while two or more spaces can be written before a program header.

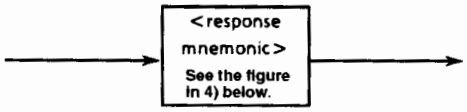
On the next page, the response header is explained up to <response mnemonic>.

(☞It should be noted that only uppercase characters must be used in <response mnemonic>. Other points are the same as those of <program mnemonic> discussed on page 5-14.)

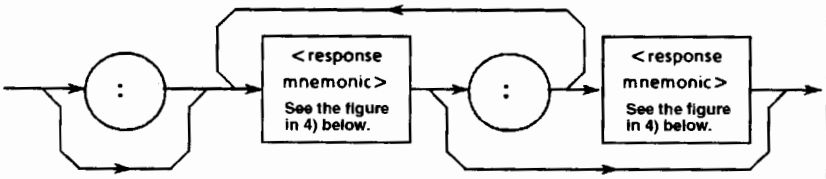
Item	Function
RESPONSE HEADER	**A header indicates a function of response data.** **It explains the function with a 12-character-long character string or a <response mnemonic> element that consists of uppercase characters, numeric characters, and/or underline.**



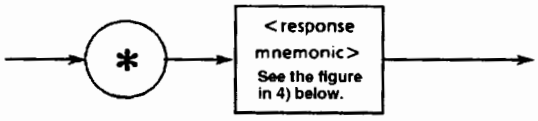
1) <simple response header> is defined as follows:



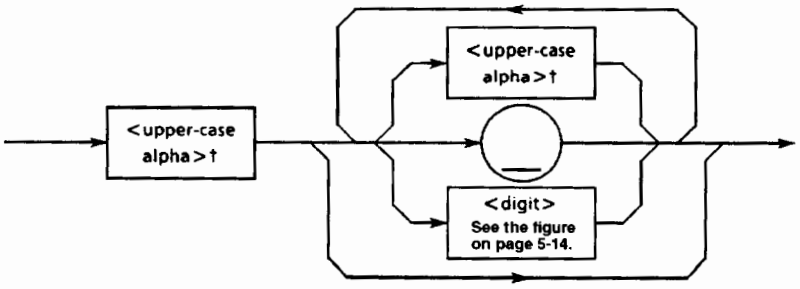
2) <compound response header> is defined as follows:



3) <common response header> is defined as follows:



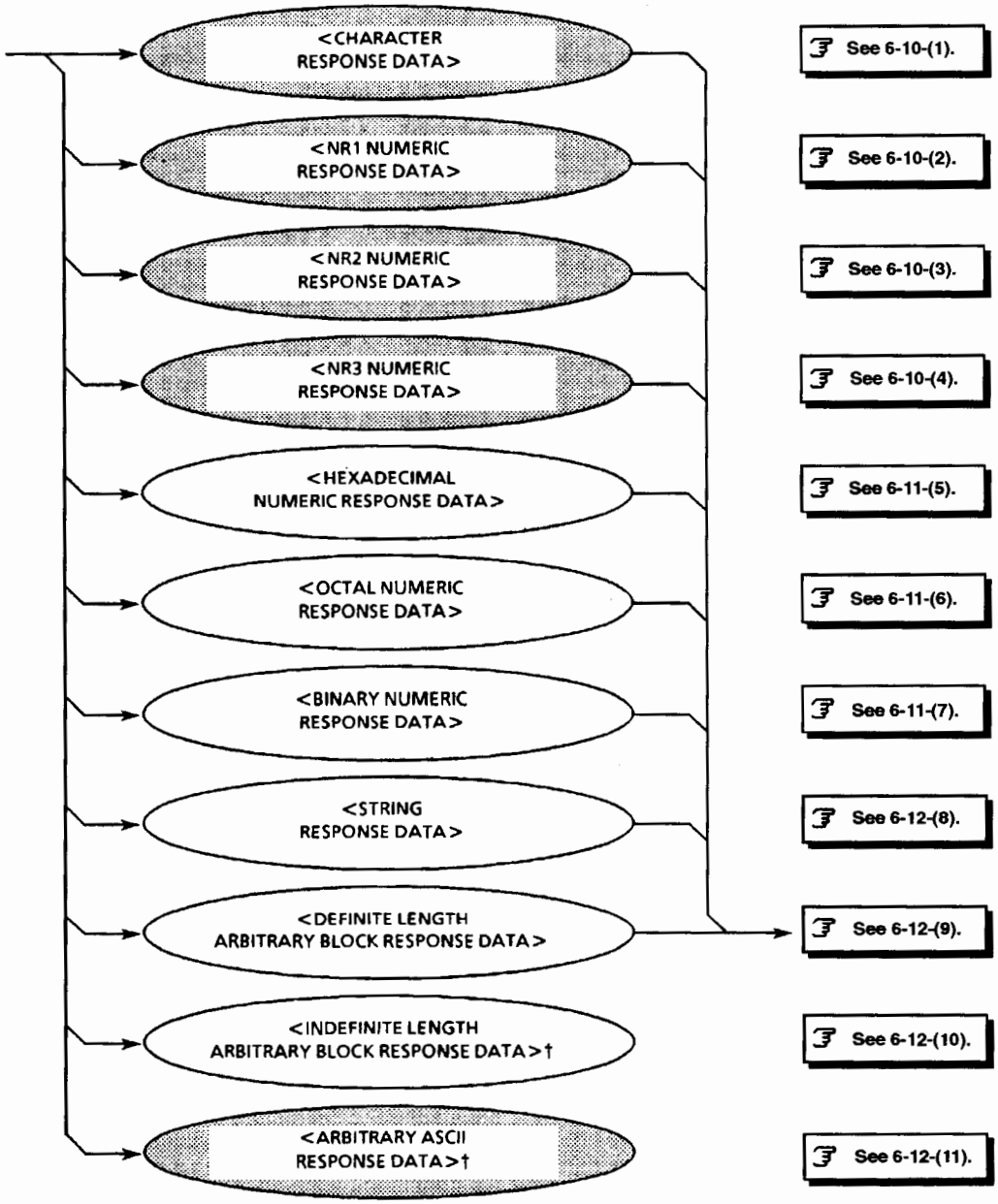
4) <response mnemonic> is defined as follows:



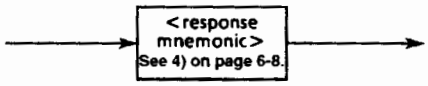
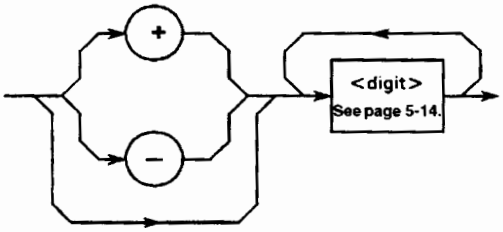
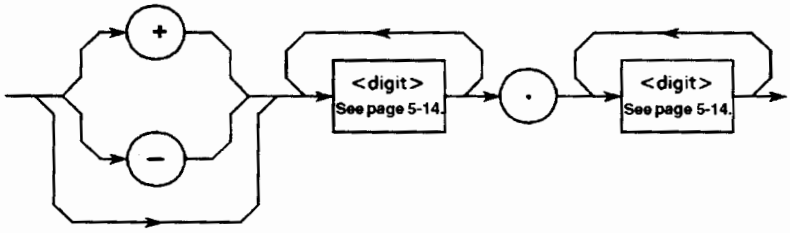
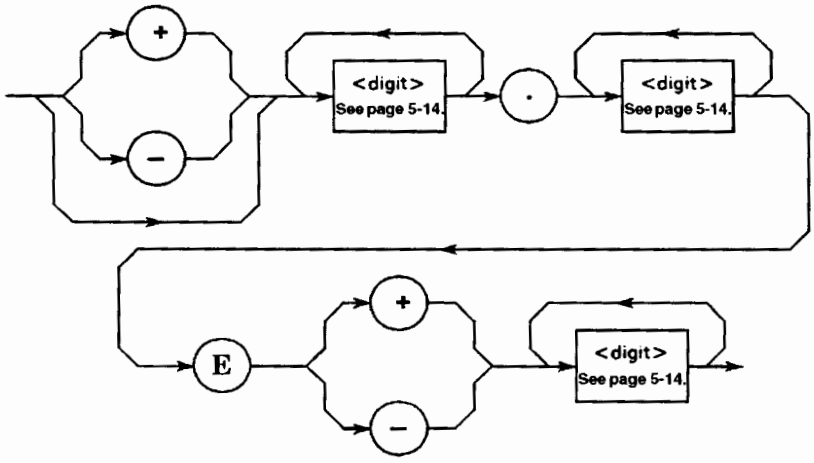
† <upper-case alpha>: ASCII code bytes 41 to 5A (decimal values 65-90 = uppercase characters A to Z)

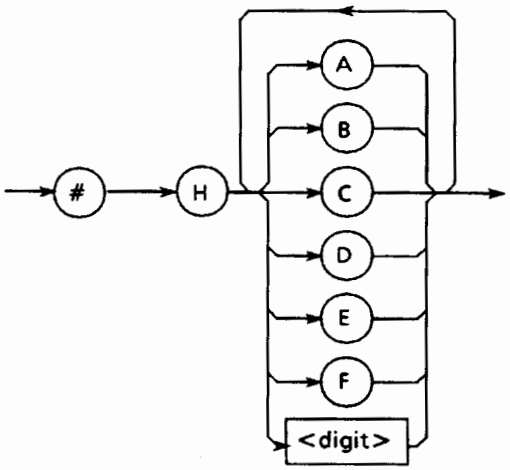
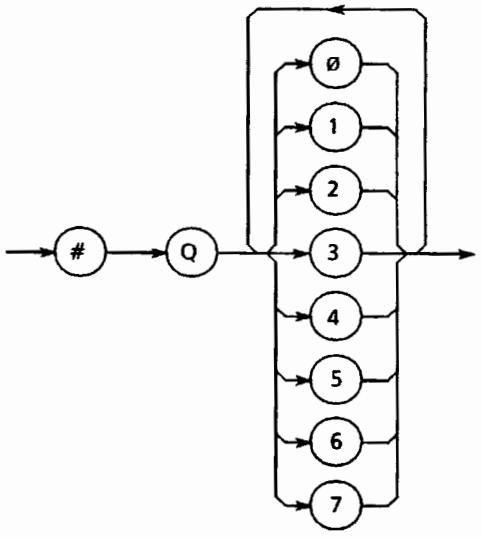
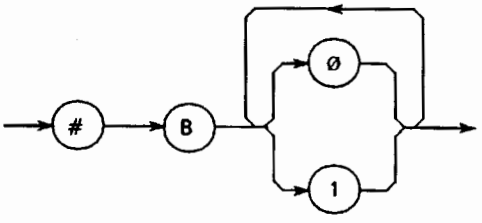
6.2.9 <RESPONSE DATA>

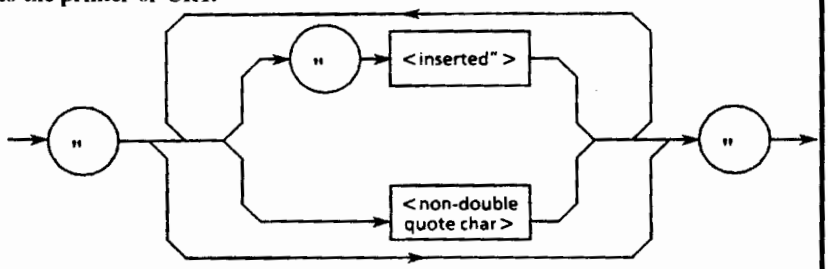
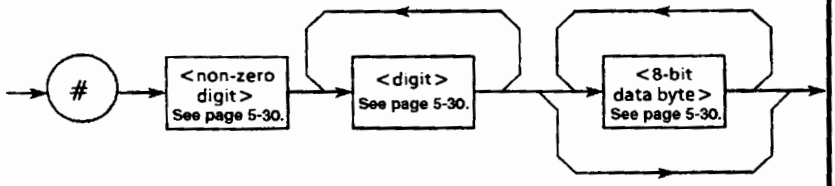
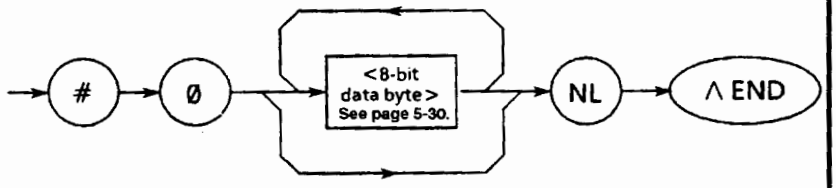
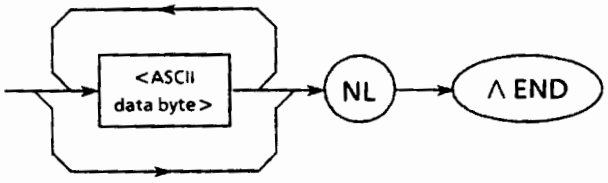
There are 11 types of <RESPONSE DATA> elements. Among them, the MS9715B transfers the response data shown in the hollow squares surrounded by a shade. The response data to be returned depends on the query message.



† <INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> and <ARBITRARY ASCII RESPONSE DATA> is terminated with NL^END after the last byte has been transferred.

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. 
(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. 

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""." **(9) DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA** **<Example>** Transferring 11256099D in a 4-byte blocks ↓ #1400ABC123 **(10)INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA** **<Example>** Indefinite-length -250, -50, 120, ... are transferred. ↓ #0FF06FFCE0078 **(11)ARBITRARY ASCII RESPONSE DATA** **<Example 1>** <ASCII Byte><ASCII Byte>NL^END **<Example 2>** NL^END	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.  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 5-31.)  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.  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 section explains common commands and common query commands specified by IEEE 488.2. These common commands are not bus commands which are used as interface messages. Like device messages, the common commands are data messages used when the bus data mode (or the ATN line) is false. They can be applied to all measuring instruments, including those of other companies, that comply with IEEE 488.2. IEEE 488.2 common commands always begin with *.

7.1 Classification of MS9715B-Supported Common Commands byGroup Function

The table below shows classification of MS9715B-supported IEEE 488.2 common commands by group function. Common commands to be supported are explained in an alphabetical order on the following pages.

7.2 Classification of Supported Commands and References

MS9715B-supported commands discussed previously are classified by function group as shown below. Details on these commands are given in alphabetical order on the next and subsequent pages.

Group	Function by group	Mnemonic
System data	Information about device connected to the system (e.g., manufacturer name, type name, and serial number) is returned.	*IDN? *OPT?
Internal operation	Control inside the device: 1) Resetting of device at level 3; 2) Self-test and error detection inside the device	*RST *TST?
Synchronization	A device is synchronized with the controller by: 1) Service request wait 2) Device output queue wait 3) Forced sequential execution	*OPC *OPC? *WAI
Status and event	A status byte consists of a status summary message. Summary bits of the status summary message are set by a standard event register, output queue, and extended event register (or an extended queue). Three commands and four queries are provided to set, clear, validate, and invalidate the data in these registers and queues and to know the register settings using queries.	*CLS *ESE *ESE? *ESR? *SRE *SRE? *STB?

*CLS Clear Status Command

(Clears status byte registers)

■ Format

*CLS

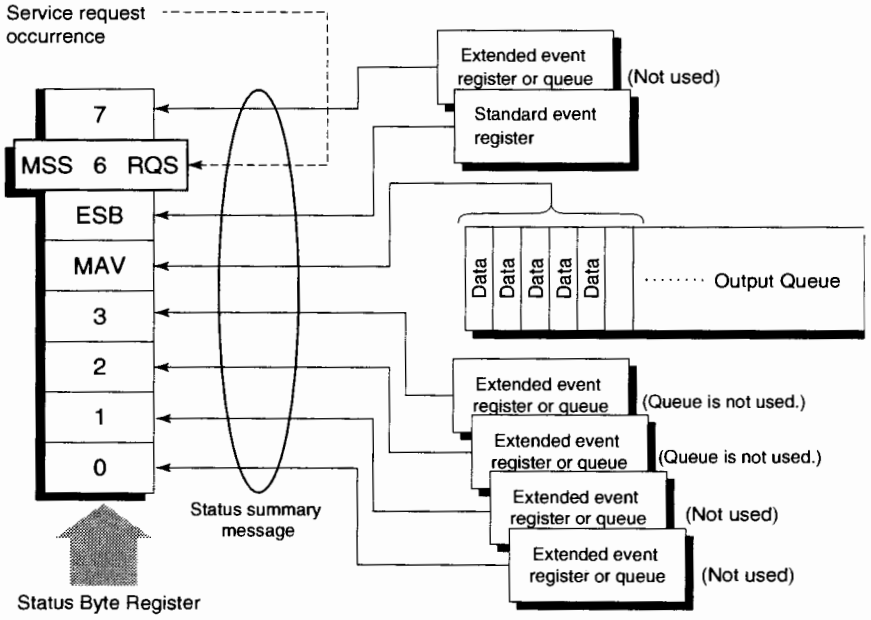
■ Application example

WRITE @108 : "*CLS"

■ Explanation

The *CLS command clears all status structures (i.e., event registers and queues) except an output queue and its MAV summary messages, thus clearing the corresponding summary messages.

Issuing a *CLS command after <PROGRAM MESSAGE TERMINATOR> or before <QUERY MESSAGE UNIT> will clear all status bytes. With this method, all unread messages in the output queue will also be cleared. Values set in enable registers are not changed by the *CLS command.



*ESE Standard Event Status Enable Command

(Sets or clears the standard event status enable register)

■ Format

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

<DECIMAL NUMERIC PROGRAM DATA>: A value rounded to an integer, 0-255 (base is 2 and binary weights are assigned).

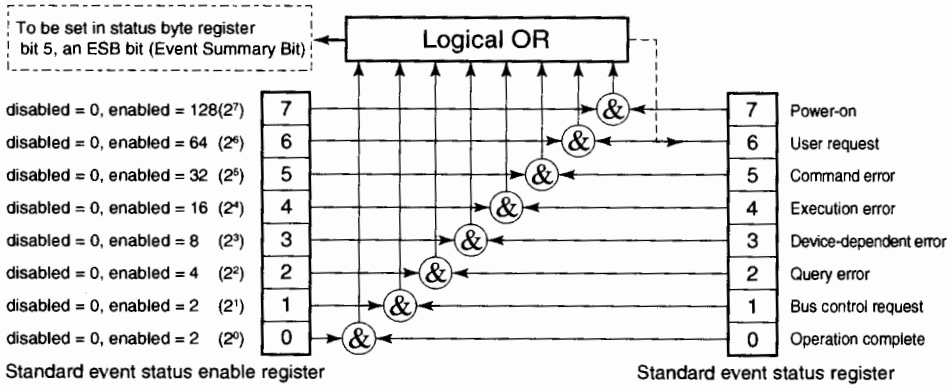
■ Application example

WRITE @108 : "*ESE 20"! Sets enable register bits 2 and 4.

■ Explanation

The total of values ($2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, and/or $2^7 = 128$) corresponding to the standard event status enable register bits 1, 2, 3, 4, 5, 6, and/or 7 that are to be enabled becomes program data.

The value of the bit to be disabled is 0.



*ESE? Standard Event Status Enable Query

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

■ Format

*ESE?

■ Application example

Issuing *EST? after executing *ESE 20 will return 20.

■ Explanation

The value (NR1) of the standard event status enable register is returned.

■ Response message

NR1=0-255

*ESR? Standard Event Status Register Query

(Returns the current value of the standard event status register)

■ Format

*ESR?

■ Application example

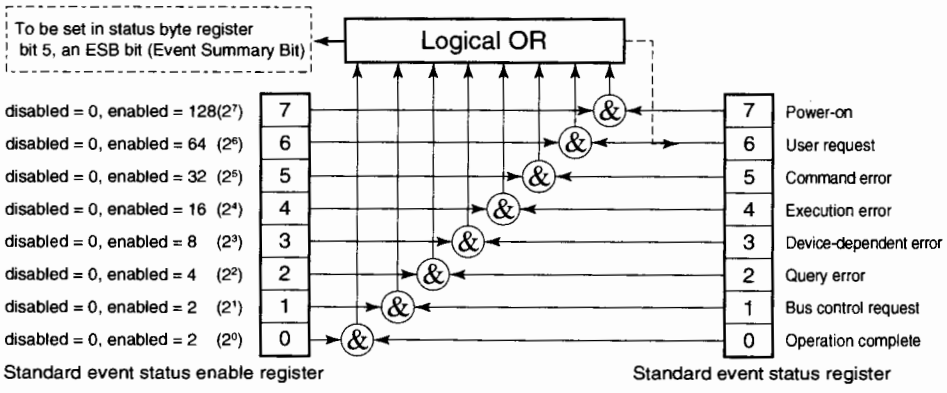
```
30  WRITE  @108 : "*ESR?"
40  READ   @108 : STEVET!      A command error occurs if the variable value is 32.
50  PRINT  STEVET
```

■ Response message NR1

NR1=0-255

■ Explanation

The current value NR1 of the standard event status register is returned. The total of values ($2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, and/or $2^7 = 128$) corresponding to the standard event status enable register bits 1, 2, 3, 4, 5, 6, and/or 7 that are enabled becomes NR1. When the response has been read (e.g., line 40), this register is cleared.



*IDN? Identification Query

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

■ Format

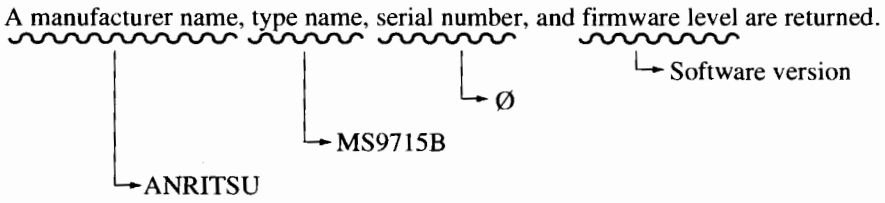
*IDN?

■ Application example

```
30  WRITE   @108 : "*IDN?"
40  READ    @108 : IDEN$ !
```

Stores the manufacturer name, type name, serial number, and irmware level.

■ Explanation



When the manufacturer of the product whose type name, serial number, and software/hardware version number are Anritsu, 0, and 1 respectively, sending a common query *IDN? to a device will return a response message consisting of the above four fields.

Field 1: Product manufacturer (e.g., ANRITSU)
Field 2: Type name
Field 3: Serial number (e.g., 0)
Field 4: Firmware version No. (control software version and optical software version)

If you don't want to return a serial number and firmware version in fields 3 and 4, you can return ASCII character "0."

■ Response message

A response message which consists of the above four fields separated by commas is sent as <ARBITRARY ASCII RESPONSE DATA>.

Overall length of the response message comprising fields 1-4 ≤ 72 characters

*OPC Operation Complete Command

(Sets bit 0 of the standard event status register when device operations have been completed)

■ Format

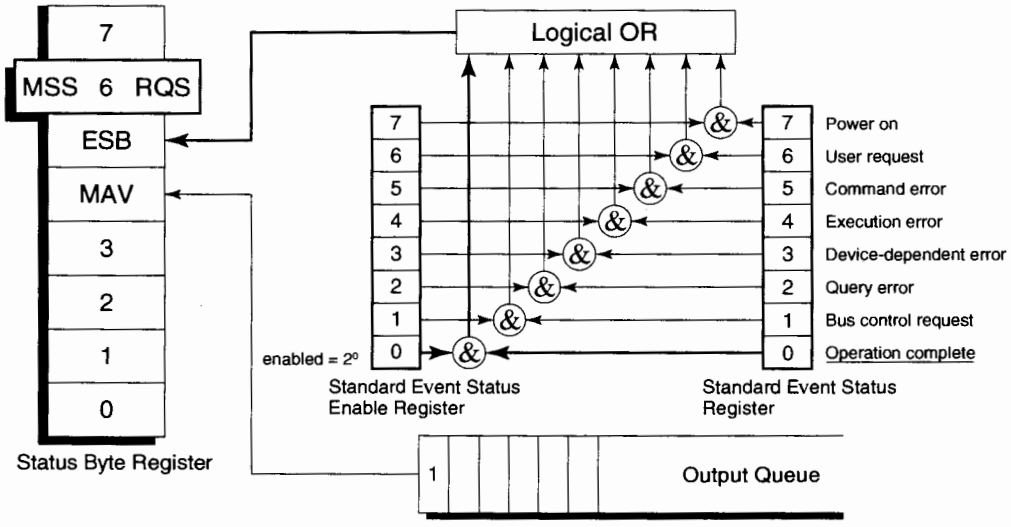
*OPC

■ Application example

WRITE @108 : "*OPC"

■ Explanation

When all the pending device operations have been completed, standard event status register bit 0 (i.e., operation complete bit) is set. However, since the MS9715B does not have an overlap command, the *OPC command counts for nothing.



*OPC? Operation Complete Query

(When device operations have been completed, sets 1 in the output queue to generate an MAV summary message)

■ Format

*OPC?

■ Application example

WRITE @108 : "*OPC?"

■ Explanation

When all the pending device operations have been completed, 1 is set in the output queue, waiting for an MAV summary message to occur.

■ Response message

1 is returned as <NR1 NUMERIC RESPONSE DATA>.

*RST Reset Command

(Rests a device at level 3)

■ Format

*RST

■ Application example

WRITE @108 : "*RST" Initializes only the device at address 3.

■ Explanation

The *RST (Reset) command resets a device at level 3 (☞See page 4-8).
At level 3, the following items are initialized:

[1] Device-dependent functions and states are restored to known states irrespective of the device history.

[2] The *DDT-command-defined macro is restored to the device-defined state.

[3] A mode in which macro operation is disabled and macros are not accepted, is set. Macro definitions are restored to the designer-specified states.

[4] The specified device is set in the OCIS (Operation Complete Command Idle State). The operation complete bit cannot be set in the standard event status register. (☞P8-3)

[5] The specified device is set in the OQIS (Operating Complete Query Idle State). The operation complete bit cannot be set in the output queue. The MAV bit is cleared.

The *RST command does not affect the following:

[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 device standard

[8] RS-232C interface condition

*SRE Service Request Enable Command

(Sets a service request enable register bit)

■ Format

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

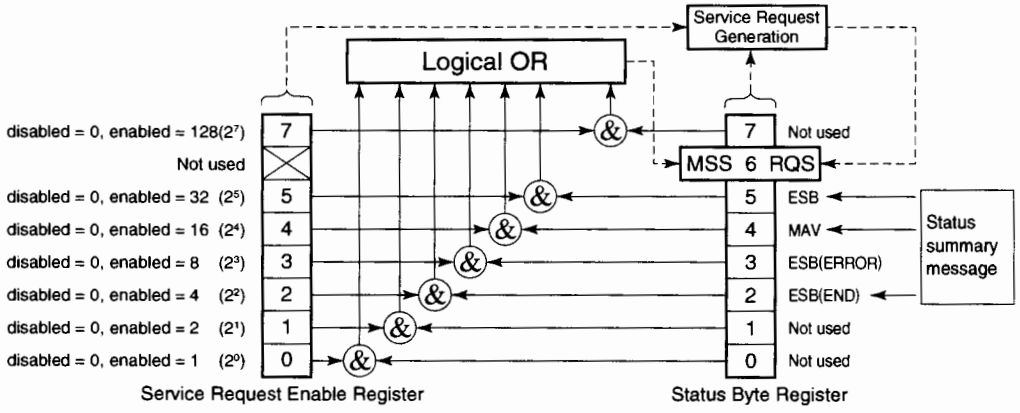
<DECIMAL NUMERIC PROGRAM DATA>: A value rounded to an integer, 0-255 (base is 2 and binary weights are assigned).

■ Application example

WRITE @108 : "*SRE 16"! Sets enable register bit 4.

■ Explanation

The total of values (2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^5 = 32, and/or 2^7 = 128) corresponding to the service request enable register bits 1, 2, 3, 4, 5, 6, and/or 7 that are to be enabled becomes NR1. The value of the bit to be disabled is 0.



*SRE? Service Request Enable Query

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

■ Format

*SRE?

■ Application example

Issuing an *SRE? command after executing an *SRE16 returns 16.

■ Explanation

The value NR1 of the service request enable register is returned.

■ Response message NR1

Since NR1 = bit 6 (RQS bit) cannot be set, NR1 = 0-63 or 128-191.

*STB? Read Status Byte Query

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

■ Format

*STB?

■ Application example

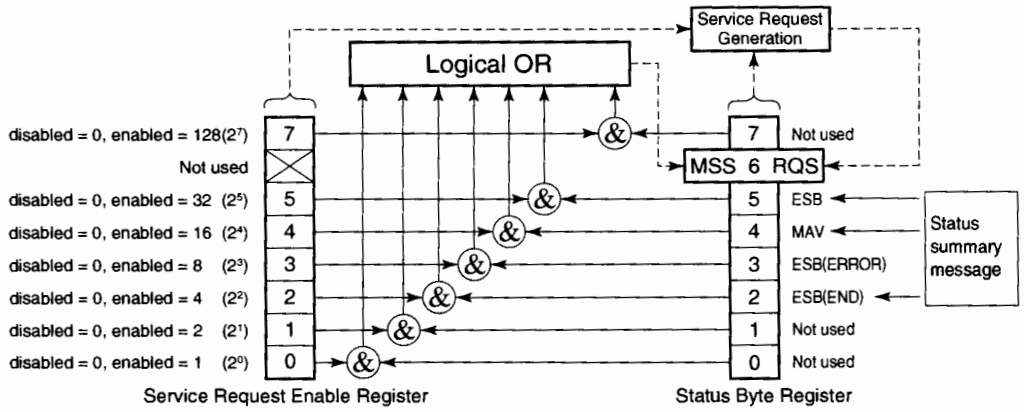
```
30  WRITE  @108 : "*STB?"
40  READ   @108 : STBV
50  PRINT  STBV
```

■ Explanation

The *STB? command returns the total of the status register value assigned binary weights and MSS summary message value as <NR1 NUMERIC RESPONSE DATA>.

■ Response message

A response message (<NR1 NUMERIC RESPONSE DATA>) is an integer ranging from 0 to 255. It is the total of status byte register bit values. Status byte register bits 0-5 and 7 is assigned weights 1, 2, 4, 8, 16, 32, and 128 respectively, and the MSS (Master Summary Status) bit is assigned weight 64. The MSS indicates that there is at least one reason for requesting a service. MS9715B's status byte register conditions are summarized in the table below.



Bit	Bit weights	Bit name	Status byte register conditions	
7	128	——	0 = Not used	
6	64	MSS	0 = No service is not requested.	1 = A service is requested.
5	32	ESB	0 = No event service has occurred.	1 = An event status has occurred.
4	16	MAV	0 = Data does not exist in the output queue.	1 = Data exists in the output queue.
3	8	ESB(ERROR)	0 = No event status has occurred.	1 = An event status has occurred.
2	4	EBS(END)	0 = No event status has occurred.	1 = An event status has occurred.
1	2	——	0 = Not used	
0	1	——	0 = Not used	

*TST? Self - Test Query

(Conducts an internal self-test and indicates whether any error has occurred)

■ Format

*TST?

■ Application example

```
30  WRITE  @108 : "*TST?"
40  READ   @108 : TEST
50  PRINT  TEST
```

■ Explanation

The *TST? command conducts a self-test inside the device. The test result is set in the output queue. The data in the output queue indicates that the test has been completed without causing any error. The self-test does not require operator intervention.

The MS9715B conducts a self-test on the optical unit.

■ Response message

A response message is sent as <NR1 NUMBER RESPONSE DATA>.
Data range = -32767 to 32767

NR1 = -: The test has been completed without causing any error.

NR1 = 1: The test has not been conducted or any error occurred during the test.

*WAI Wait - to - Continue Command

(Causes the next command to wait until the current command has been executed by the device)

■ Format

*WAI

■ Application example

WRITE @108 : "*WAI"

■ Explanation

The *WAI command executes overlap commands as sequential commands.

If the device can start executing the next command while processing a command or query from the controller, the command or query is called an overlap command.

If a *WAI command is executed after an overlap command, the next command must wait for the *WAIT common command to end. This also applies to sequential commands.

However, since the MS9715B does not support overlap commands. The *WAI command counts for nothing.

Section 8 Status Structure

This section explains the device status data specified by IEEE 488.2, the status data structure, and the technique of synchronization between a device and a controller.

IEEE 488.2 additionally provides common commands and common queries to get more detailed information compared with IEEE 488.1. For details on these commands and queries, see Section 7.

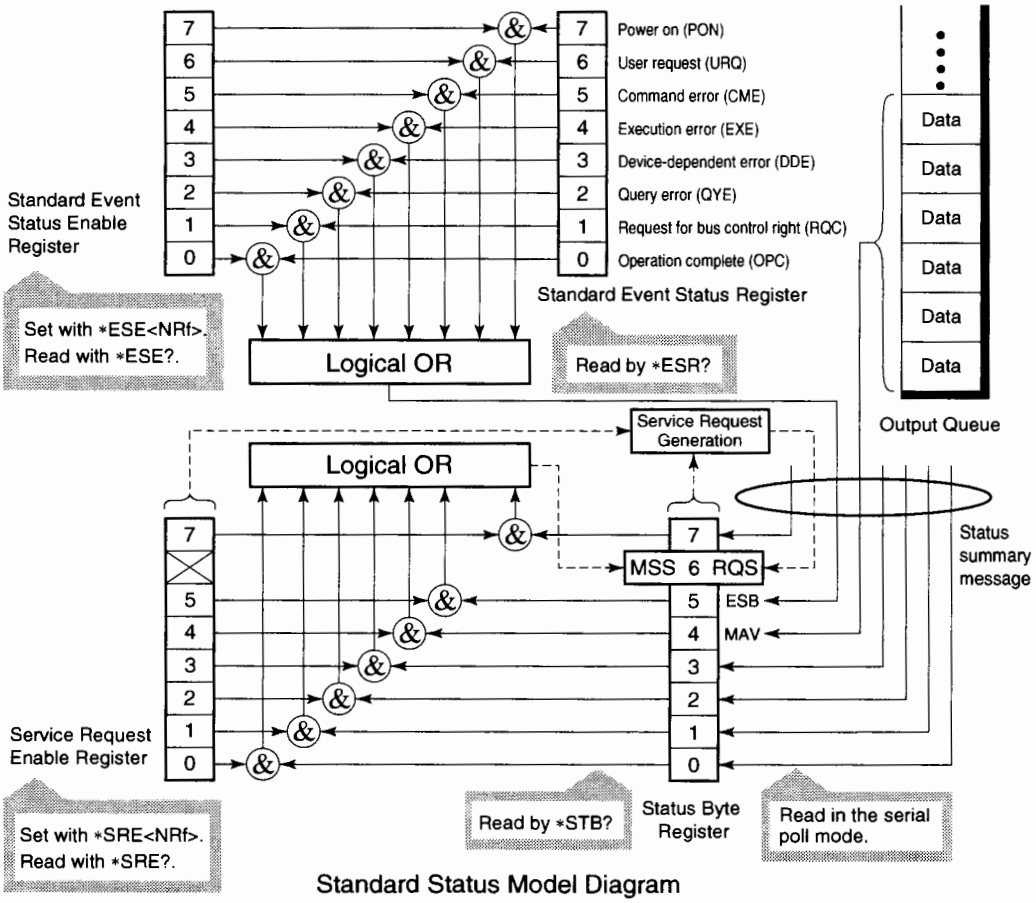
The status byte (STB) sent to the controller is specified by IEEE 488.1. The bits of the status byte represent a status summary message, providing a summary of the current contents of the data stored in a register or queue.

The following sections explain the status summary message bits, the status data structure for generating these status summary message bits, and the technique of synchronizing a device with the controller using the status messages.

These functions are used to control devices from an external controller via the GPIB interface. These functions, except a few, can also be used to control devices from an external controller via the RS-232C interface.

8.1 IEEE 488.2 Standard Status Model

Shown below is the standard model of the status data structure specified by IEEE 488.2



Standard Status Model Diagram

Fig. 8-1 Standard status model

The status model uses an IEEE 488.1 status byte. This status byte consists of seven summary message bits provided by the status data structure. To generate these summary message bits, the status data structure is comprised of two models: a register model and a queue model.

Register model	Queue model
A pair of registers used to record an event that a device has encountered and a condition. It consists of an event status register and an event status enable register. When the results of ANDing the values of bits of these registers is not 0, the corresponding status register bits are set to 1s. In other cases, the corresponding status register bits are set to 0s. If the result of ORing the values of status register bits is 1, the summary message bit is set to 1. If the result of ORing these bits is 0, the summary message bit is set to 0.	A data structure in which status values or information are removed in the same order they were entered. Only when the queue structure contains data, the corresponding bit is set to 1. If it is empty, the corresponding bit is set to 0.

Based on the concept of the above register model and queue model, the IEEE 488.2 standard status model is constructed from two types of register models and a 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 mentioned above. It has eight bits corresponding to eight standard events encountered by the device: 1) power on, 2) user request, 3) command error, 4) execution error, 5) device dependent error, 6) query error, 7) bus control request, 8) operation complete. The result of logical OR is output to the status byte register bit 5 (DIO 4) as an event status bit (ESB) summary message.	The status byte register consists of an RQS bit and seven summary message bits for setting status summary messages from the status data structure. It is used in combination with a service request enable register. When the result of ORing the values of these two registers is 0, the SRQ is set ON. In this case, the status byte register bit "DIO 7" is reserved by the system as an RSQ bit, so this bit indicates to an external controller that a service request exists. The function of the SRQ conforms to IEEE 488.1.	This queue has the queue model structure mentioned above. Its contents are summarized and transferred to the status byte register bit 4 (DIO 5) as a message available (MAV) summary message.

8.2 Status Byte (STB) Register

The STB register consists of device STB and RQS (or MSS) messages. IEEE 488.1 defines the method of reporting STB and RQS messages, but it does not define the setting and clearing protocols and STB meaning. IEEE 488.2 defines device status summary messages and the master summary status (MSS) transferred to bit 6 along with an STB in response to the *STB? common query.

8.2.1 ESB and MAV summary message

Let's take a look at an ESB summary message and an MAV summary message.

(1) ESB summary message

The ESB (event summary bit) summary message is defined by IEEE 488.2. It appears in STB register bit 5. This bit indicates whether one or more IEEE 488.2 defined events have occurred, with the service request enable register set to allow events to occur, after the standard event status register was read or cleared last. The ESB summary message bit becomes true when at least one event registered in the standard event status register becomes true with event occurrence enabled. Conversely, the ESB summary bit becomes false when none of the registered events has occurred even if event occurrence is enabled.

(2) MAV summary message

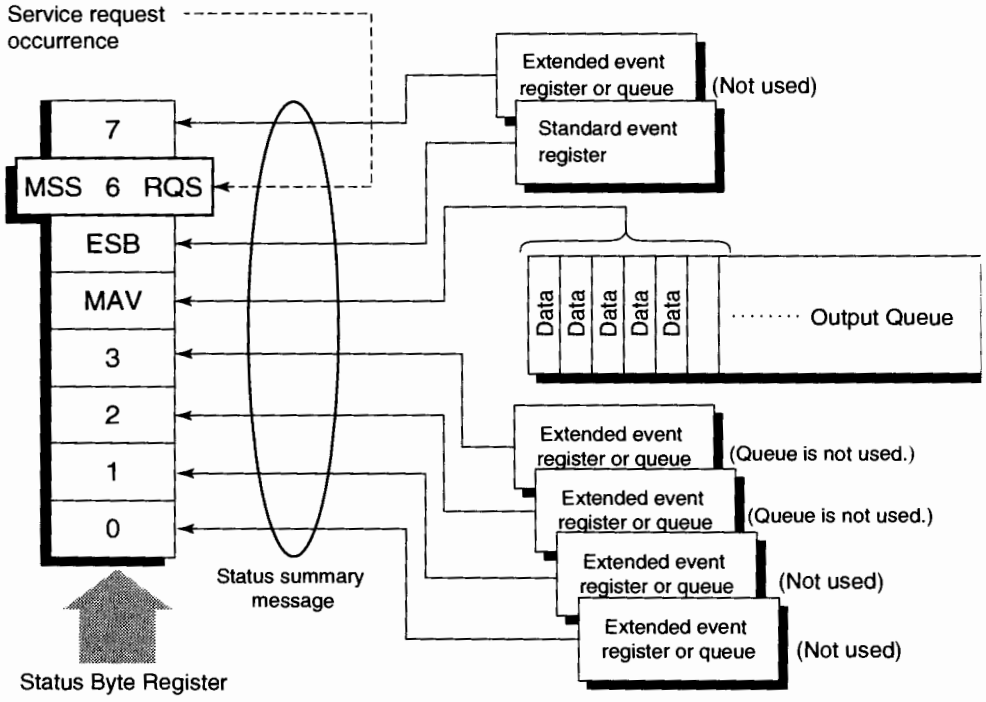
The MAV (message available) summary message is defined by IEEE 488.2. It appears in STB register bit 4. This bit indicates whether the output queue is empty. When a device is ready for accepting response messages from the controller, the MAV summary message bit becomes 1 (true). When the output queue is empty, this bit becomes 0 (false). This message is used to synchronize information exchange with the controller. For example, the controller can send a query message to the device and wait for the MAV to become true. The controller can perform another processing while waiting for a response from the device. If the controller has started reading the output queue without checking the MAV, all system bus operations are suspended until a response is received from the device.

8.2.2 Device dependent summary messages

IEEE 488.2 does not define whether status register bit 7 (DIO 8) and bits 3 (DIO 4) to 0 (DIO 1) are used as status register summary bits or the bits indicating existence of data in the queue. Accordingly, these bits can be used as device dependent summary message bits.

Device dependent summary messages have a register model or queue model status data structure. This status register is a pair of registers used to report events and states in parallel or a queue used to report states and information sequentially. The summary bit provides a summary of the current status of the corresponding status data structure. For the register model, the summary message bit becomes true when one or more events have become true with occurrence of events enabled. For the queue model, the summary message bit becomes true when the queue is not empty.

The MS9715B does not use bits 7, 1, and 0 and uses bits 2-3 as status register summary bits. So the register model has four types of status data structures (two extended status data structures), and the queue model has an output queue (no extended status data structure).



8.2.3 Reading and Clearing the STB register

STB register contents can be read using serial polling or an *STB? common inquiry. IEEE 488.1 defined STB messages can be read by either method, but the value transferred to bit 6 (position) varies depending on the method.

STB register contents can be cleared using a *CLS command.

(1) Reading the STB register using serial polling (only when a GPIB interface bus is used)

When IEEE 488.1 defined serial polling is carried out, the device must return a 7-bit status byte and IEEE 488.1 defined RQS message bit. According to IEEE 488.1, the RQS message indicates whether the device has issued SRQs in the true state. The status byte value is not affected by serial polling. Immediately after being polled, the device must set the rsv message in the false state. If the device is polled again before a cause of issuing a new service request occurs, the RQS message has already been set in the false state.

(2) Reading the STB register using an *STB? common query

The *STB? common query causes the device to output STB register contents and one <NR1 NUMERIC RESPONSE DATA> from the MSS (master summary status) summary message. The response is the total of the status register value assigned binary weights and MSS summary message value. STB register bits 0 to 5 and 7 are assigned weighs 1, 2, 4, 8, 16, 32, and 128 respectively, and the MSS is assigned weights 64. The response to the *STB? is the same as that to serial polling with the exception that an MSS summary message appears in bit 6 instead of an RQS message.

(3) Definition of MSS (Master Summary Status)

The MSS indicates that the device has at least one cause of issuing a service request. In the device's response to the *STB? query, the MSS message appears in bit 6. However, it does not appear in the response to serial polling. It must not be regarded as part of the IEEE 488.1 defined status byte. The MSS is the result of ORing the values of STB register and SRQ enable (SRE) register bits totally. Specifically, the MSS is defined as follows:

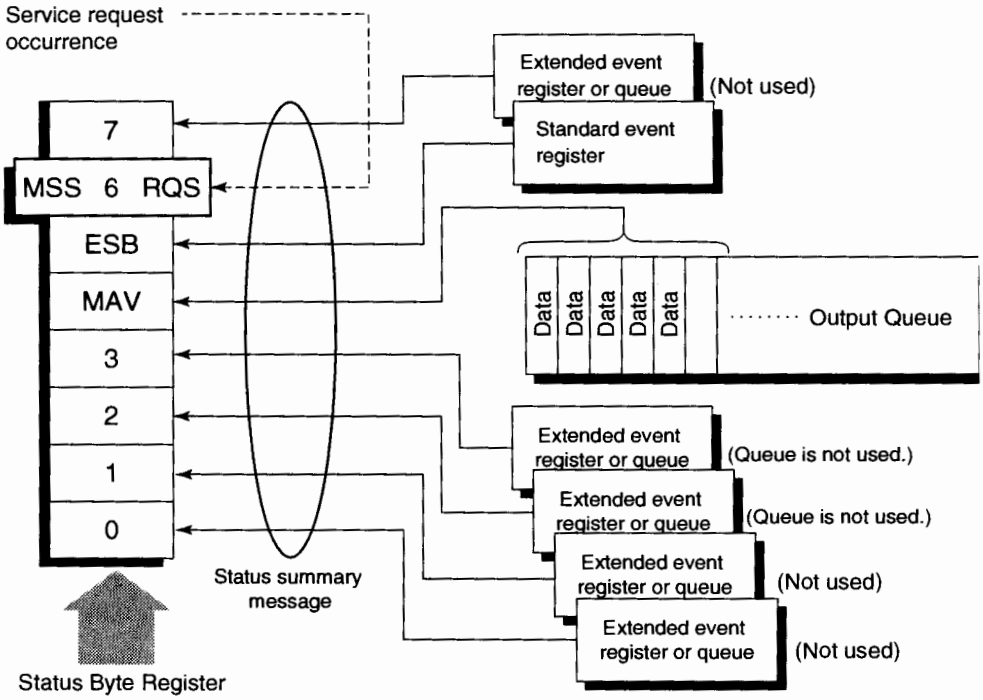
(STB Register bit 0 AND SRE Register bit 0)
OR
(STB Register bit 1 AND SRE Register bit 1)
OR
:
:
(STB Register bit 5 AND SRE Register bit 5)
OR
(STB Register bit 7 AND SRE Register bit 7)

In the definition of the MSS, the values of bits 6 of the STB register and SQR enable register are ignored. Accordingly, when calculating the MSS value, the status byte may be handled assuming that it is represented by 8 bits and bit 6 is always 0.

(4) Clearing the STB register using a *CLS common command

The *CLS common command clears all status structures, except the output queue and MAV summary message (i.e., event registers and queues), and the corresponding summary messages.

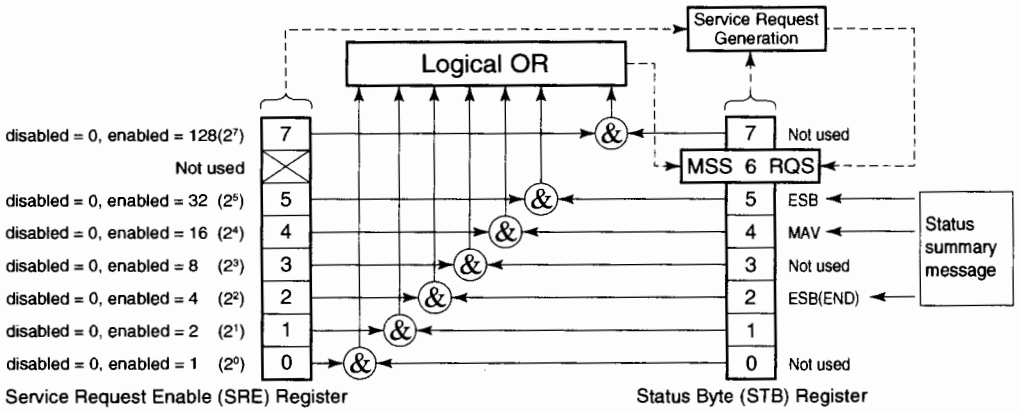
Issuing a *CLS command after the <PROGRAM MESSAGE TERMINATOR> element or before the <Query MESSAGE UNIT> element clears all status bytes. With this method, all unread messages in the output queue are cleared and the MAV message becomes false. When replying to the *STB?, the MSS message becomes false, too. Values of enable registers are not affected by *CLS.



8.3 Enabling the SRQ

Enabling the SRQ allows a summary message in the STB register to be selected in response to a service request. The service request enable (SRE) register shown below can be used to select a summary message.

Bits of the service request enable register correspond to the bits of the status byte register. When 1 is set in a status byte bit corresponding to a significant bit of the service request enable register, the devices sets the RQS bit to 1 and issues a service request to the controller. For example, when bit 4 of the service request enable register is set (enabled) in advance, a service request can be issued to the controller each time the MAV bit is set to 1 (if the output queue has data).



(1) Reading the SRE register

SRE register contents can be read using an *SRE? common inquiry. The response message to this query is <NR1 NUMERIC RESPONSE DATA>, an integer ranging from 0 to 255. It is a total of values of the service request enable register. Service request enable register bits 0 to 5 and 7 are assigned weights 1, 2, 4, 8, 16, 32, and 128, respectively. Unused bit 6 must always be 0.

(2) Updating the SRE register

The SRE register is written using an *SRE common command. The *SRE common instruction is followed by a <DECIMAL NUMERIC PROGRAM DATA> element. <DECIMAL NUMERIC PROGRAM DATA> is rounded to an integer. It is represented in binary notation using a base 2, indicating the total of values of SRE register bits (weight value). When the value of this bit is 1, it indicates the enabled state. When the value of this bit is 0, it indicates the disabled state. The value of bit 6 must always be ignored.

(3) Clearing the SRE register

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

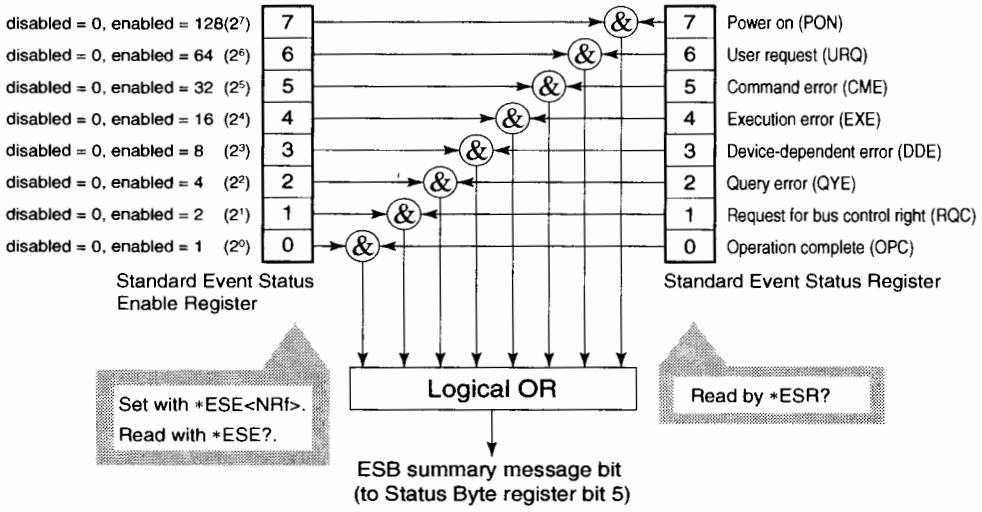
When an *SRE common command is used, the SRE register can be cleared by bringing the <DECIMAL NUMERIC PROGRAM DATA> element value to 0. Clearing the SRE register disables the status information to generate an rsv local message, suppressing issue of a service request.

When the power is turned on, the SRE register is cleared if the power-on status clear flag is true and the *PSC command for disabling clearing of this register is not supported.

8.4 Standard Event Status Registers

8.4.1 Definition of standard event status register bits

Any device conforming to IEEE 488.2 must have the standard event register. Operation of the standard event register model is shown below. As it has already been explained, here we will explain the meaning of standard event status register bits given in IEEE 488.2.



Bit	Event name	Description
7	Power-on (PON)	The power has been turned on.
6	User request (URQ)	Local control is requested. This bit is set irrespective of the remote/local state of the device. Since this bit is not supported by MS9715B, it is always 0.
5	Command error (CME)	A program message including a syntax error or a misspelled command has been received or a GET command has been received in a program message.
4	Execution error (EXE)	A program message which is syntactically OK but cannot be executed has been received.
3	Device-dependent error (DDE)	An error other than CME, EXE, and QYE has occurred.
2	Query error (QYE)	An attempt was made to read data from the output queue while it has no data, or the data in the output queue has been lost due to overflow, etc.
1	Request control (RQC)	The device is required to be an active controller. Since this bit is not used by MS9715B, it is always 0.
0	Operation complete (OPC)	The device has completed the specified pending operation and ready for receiving a new instruction. This bit responds only to the *OPC command and sets the operation complete bit.

8.4.2 Details on query errors

No.	Item	Description
1	Incomplete program message	When a device receives an MTA from the controller before receiving a program message terminator, it discards the incomplete message which has been received so far and waits for the next program message. To discard the incomplete program message, the device clears the input/output buffer, reports a query error to the status report part, and sets the standard status register bit 2 (query error bit).
2	Interruption of response message output	When a device receives an MLA from the controller before completing output of a response message terminator, it automatically interrupts output of the response message and waits for a next program message. To interrupt output of the response message, the device clears the input/output buffer, reports a query error to the status report part, and sets the standard status register bit 2 (query error bit).
3	When the next program message is sent without reading a response message	When the device cannot output a response message because the controller has output a program message (including a query message) and the next program message in succession, the device discards the response message and waits for the next program message. A query error is reported to the status report part like item 2.
4	Output queue overflow	When a program message containing many query messages is executed one after another, too many response messages to be stored in the output queue (256 bytes) may be generated. If more query messages are input and the response messages to them must be output, the output queue overflows. When this happens, the device clears the output queue and resets the response message generation part. The device also sets the standard event status register bit 2 (query error bit) in the status report part.

8.4.3 Reading, writing, and clearing the standard event status register

Read	This register is read destructively in response to the *ESR? common command. That is, this register is cleared after being read. The event bit assigned binary weights and converted to a decimal value <NR1> is the response message.
Write	This register cannot be written externally; however, it can be cleared.
Clearing	This register is cleared in the following cases: [1] A *CLS command is received. [2] The power is turned on if the power-on status clear flag is true. The device executing a power-on sequence first clears the standard event status register, then records the events that have occurred in this sequence (e.g., PON event bit setting). [3] An event is read in response to an *ESR? query command.

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

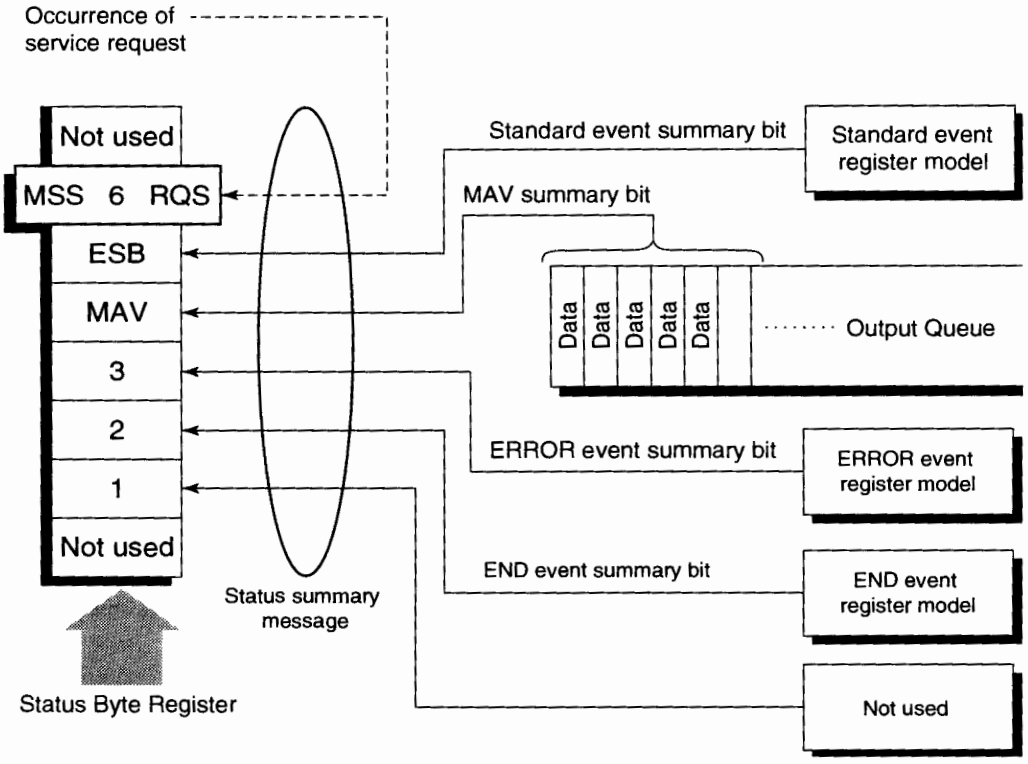
Read	This register is read non-destructively in response to the *ESR? common command. That is, this register is not cleared after being read. The response message is assigned binary weights, converted from a binary value to a decimal value <NR1>, and returned.
Write	This register is written using an *ESS common command. Register bits 0 to 8 are assigned weights 1, 2, 4, 8, 16, 32, 64, and 128 respectively, so a total of values of the desired write data bits is sent as <DECIMAL NUMERIC PROGRAM DATA>.
Clearing	This register is cleared in the following cases: [1] An *ESE command whose data value is 0 is received. [2] The power is turned on with the power-on status clear flag in the true state or the power is turned on when a *PSC command is not supported. The standard event status register is not affected by the following: [1] Change in the state of the IEEE 488.1-defined device clear function [2] Reception of an *RST common command [3] Reception of a *CLS common command

8.5 Extended Event Status Registers

Devices conforming to IEEE 488.2 require register models for status byte and standard event status registers including an enable register.

IEEE 488.2 assigns status byte register bit 7 (DIO 8) and bits 3 (DIO 4) to 0 (DIO 1) to the status summary bits transferred from an extended register model and extended queue model.

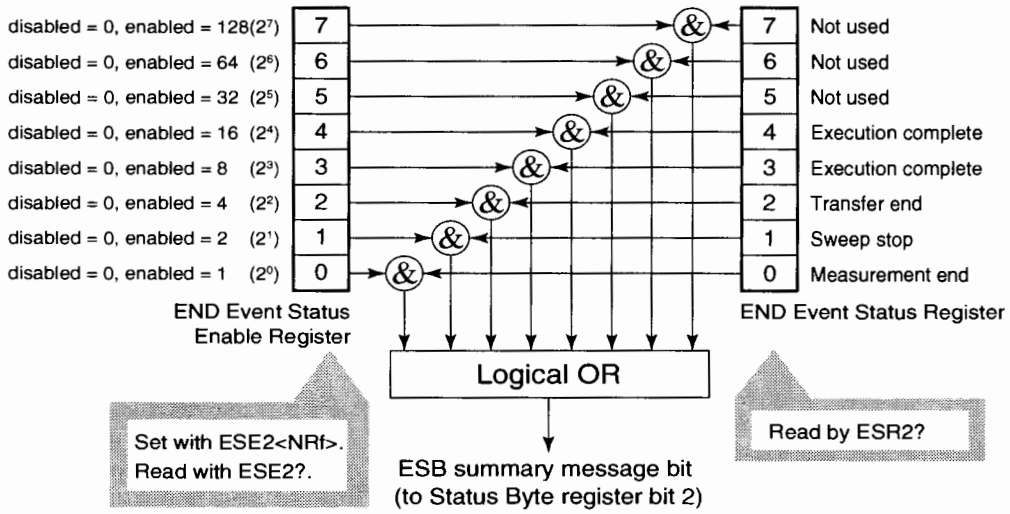
As shown below, the MS9715B does not use bits 7, 1, and 0. It assigns bits 3 and 2 to END and ERROR summary bits for status summary bits transferred from the extended register model.



Let's take a look at definition, read, write, and clearing of END and ERROR extended event register model bits.

8.5.1 Definition of END event status register bits

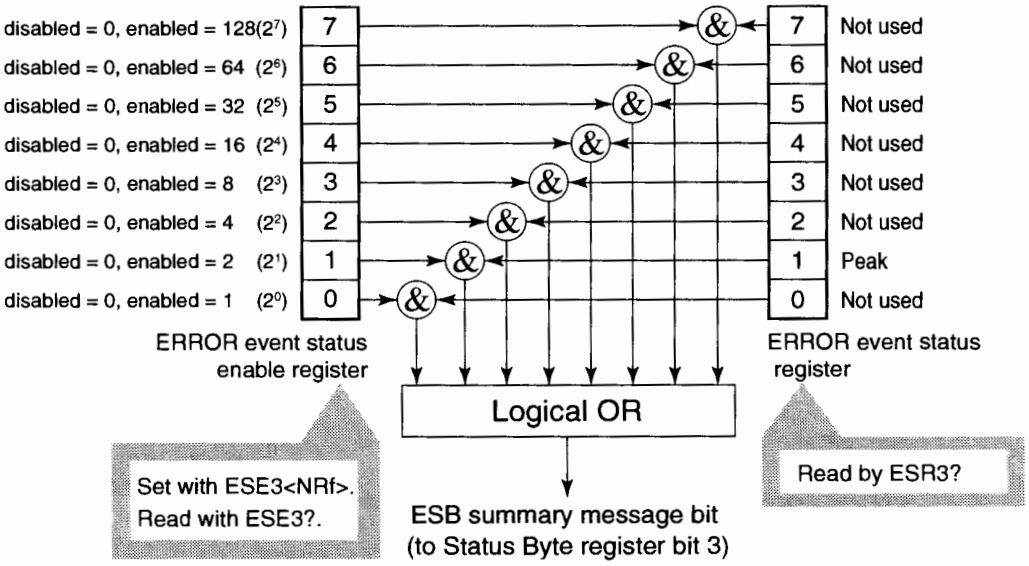
This section explains END event status register model operation and names and meanings of events.



Bit	Event name	Description
7	Not used	
6	Not used	
5	Not used	
4	Execution	Completion of *RST, wavelength calibration, automatic axis alignment complete
3	Not used	
2	Transfer end	Completion of transfer to FD or printer output
1	Sweep stop	Single sweep stop
0	Measurement end	Completion of analysis

8.5.2 Definition of ERROR event status register bits

This section explains ERROR event status register model operation and names and meanings of event bits.



Bit	Event name	Description
7	Not used	
6	Not used	
5	Not used	
4	Not used	
3	Not used	
2	Not used	
1	Peak Error	Occurrence of peak detection error
0	Not used	

8.5.3 Reading, writing, and clearing the extended event status register

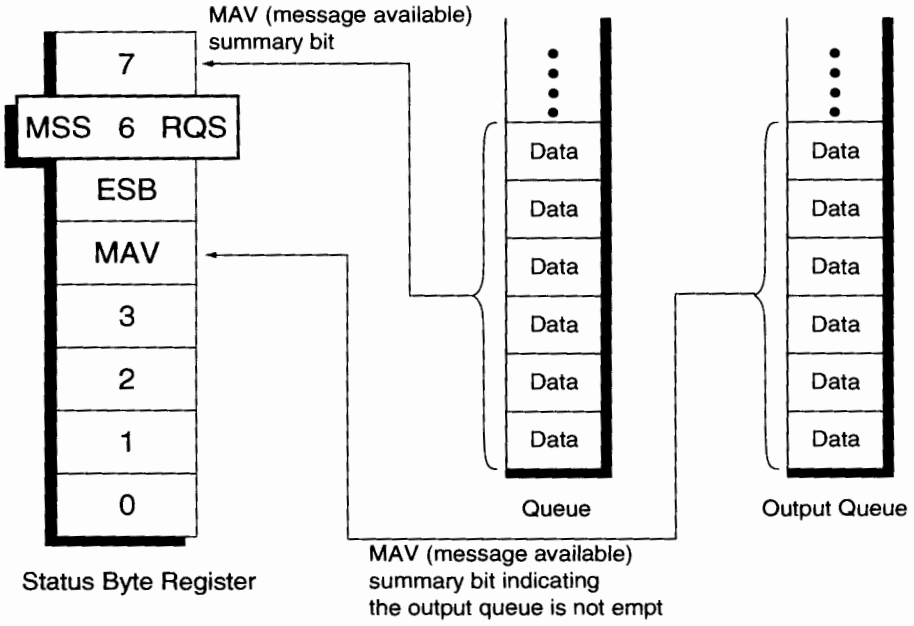
Read	This register is read destructively in response to a query. That is, this register is cleared after being read. END and ERROR event registers are read with ESR2? and ESR3? queries respectively. The event bit assigned binary weights and converted to a decimal value <NR1>.
Write	This register cannot be written externally; however, it can be cleared.
Clearing	This register is cleared in the following cases: [1] A *CLS command is received. [2] The power is turned on if the power-on status clear flag is true. [3] An event is read in response to a query.

8.5.4 Reading, writing, and clearing the extended event status enable register

Read	This register is read non-destructively in response to a query. That is, this register is not cleared after being read. END and ERROR event registers are read with ESR2? and ESR3? queries respectively. The value is assigned binary weights, converted from a binary value to a decimal value <NR1>, and returned.
Write	END and ERROR registers are written with an ESE2, ESE3, and program commands. Register bits 0-8 are assigned weights 1, 2, 4, 8, 16, 32, 64, and 128 respectively, so a total of values of the desired write data bits is sent as <DECIMAL NUMERIC PROGRAM DATA>.
Clearing	This register is cleared in the following cases: [1] For END and ERROR event status registers, an ESE2, ESE3, or program command whose data value is 0 is received. [2] The power is turned on with the power-on status clear flag in the true state or the power is turned on when a *PSC command is not supported. The extended event status register is not affected by the following: [1] Change in the state of the *IEEE 488.1-defined device clear function [2] Reception of an *RST common command [3] Reception of a *CLS common command

8.6 Queue Model

The right-hand illustration shown below is a queue model having a status data structure. A queue is a data structure in which data is arranged sequentially, providing information such as sequential status. A summary message indicates that such information exists in the queue. Queue contents are read by an handshake when the device is in the talker active state (TACS).



The queue that outputs an MAV summary bit to status byte register bit 4 is called an "output queue." This queue is mandatory. The queue that can output an MAC summary message to one of status byte register bits 0 to 3 and 7 is simply called a "queue." It is optional. A summary message from the register model can also be output to status byte register bits 0 to 3 and 7, so the summary message type depends on the device type.

We use status byte register bit 7 for the summary message bit transferred from the "queue." However, we do not use this bit if only the "output queue" suffices and therefore the "queue" need not be used.

The table on the next page provides a comparison of the "output queue" to general queues.

Table 8-1 Comparison of Output Queue to General Queues

Item	Output queue	Queue
Data input/output type	FIFO type	Not necessary be FIFO type
Read	Response message units are read using only an IEEE 488.2 message exchange protocol. The type of these response message units depends on the query type.	Response message units are read with device-dependent query commands. These response message units must be of the same type.
Write	Program message elements are not written directly. This queue communicates with the system interface using only an IEEE 488.2 message exchange protocol.	Program message elements are not written directly. Coded device information is indicated.
Summary message	When the output queue is not empty, the summary message bit becomes true (1). When it is empty, the summary message bit becomes false (0). The MAV summary message is used to synchronize information exchange between a device and the controller.	When the queue is not empty, the summary message bit becomes true (1). When it is empty, the summary message bit becomes false (0).
Clearing	This queue is cleared in the following cases: [1] All items in the queue are read. [2] A DCL bus command is received for message exchange. [3] The PON bit becomes true at power-on. [4] Operation is unterminated or interrupted.	This queue is cleared in the following cases: [1] All items in the queue are read. [2] A *CLS command is received. [3] Other device-dependent means

Section 9 Details on Device Messages

9.1 SSI [Single Sweep]

■ Function

Sets single sweep start.

Upon sweep end, bit 1 of the expansion event status register (ESR2)(sweep end bit) is set to "1".

Header	Program	Inquiry	Response
SSI	SSI	None	None

9.2 SRT [Repeat Sweep]

■ Function

Sets repeat sweep start.

Header	Program	Inquiry	Response
SRT	SRT	None	None

9.3 SST [Sweep Stop]

■ Function

Sets sweep stop.

Header	Program	Inquiry	Response
SST	SST	None	None

9.4 ZMK [Zoom Marker]

9.4.1 ZMKW [Zoom Marker Wavelength]

■ Function

It determines the center of a zoom maker and sets the span width.

Header	Program	Inquiry	Response
ZMKW	ZMKW, λc, λs	ZMKW? WL	λc, λs

■ • λc value

λc is the center wavelength of the zoom marker.

The unit is fixed to nm, and numeric data including up to three digits after the decimal point is input.

Data range: start wavelength ≤ λc ≤ stop wavelength.

• λs value

λs is the span width of the zoom marker.

The unit is fixed to nm, and numeric data including up to three digits after the decimal point is input.

Set the zoom marker range so as not to exceed the range from start wavelength to stop wavelength.

■ Initial setting value

λc = center wavelength

λs = 2 × span width /10

9.4.2 ZMKW ZOOM [Zoom Maker→Zoom In/Out]

■ Function

It executes zoom in/zoom out.

Header	Program	Inquiry	Response
ZMKW	ZMKW Zoom, s	ZMKW? Zoom	Zoom, s

■ s value

s = IN:Displays zoom of zoom marker range.

= OUT:It returns state to the Full span.

■ Initial setting value

s = OUT

9.4.3 ZMKW ERS [Maker Erase]

■ Function

Deletes marker.

Header	Program	Inquiry	Response
ZMKW	ZMKW ERS	None	None

9.5 MODE [Measure Mode]

■ Function

Outputs measuring mode with numeric values.

Header	Program	Inquiry	Response
MODE	None	MODE?	n

■ n value

n is output as a value between 0 and 3, and has the following meaning.

0 : When spectrum is not measured

1 : Spectrum is being measured (single sweep)

2 : Spectrum is being measured (repeat sweep)

3 : Under continuous testing

9.6 APR [Application Result]

■ Function

It reads the results of the analysis after spectrum measurement.

Header	Program	Inquiry	Response
APR	None	APR? WDM, PKC	WDM, PKC, d
			d=Peak No. (1,2,3・・・) Type 1
		APR? WDM, SNR, x	WDM, SNR
		x=1 to 50	l, L, S, rl
			l=xxxx.x
			L=xxxx.x
			S=xxx.xx
			rl=1: left
			=2: right
			Type 2
		APR? WDM, SNR,	WDM, SNR, GAT
		GAT	a, mx, mi
			a=xxx.xx
			mx=xxx.xx
			mi=xxx.xx Type3

■ Response data

Numeric values are tail-zero suppressed.

Type 1

It outputs a peak count value from the measurement results.

• d value

d is the peak cont value.

Type 2

From the measurement results, it determines the specified peak number and outputs a level and SNR.

• x value

x is the peak No.

• λ value

l is the wavelength of peak No. x.

The unit is fixed to nm, and numeric values including two digits after the decimal point are output.

If analysis cannot be performed, -1 is output.

• L value

L is the level of peak No. x.

The unit is fixed to dBm and numeric values including two digits after the decimal point are output.

• S value

S is the SNR of peak No. x.

The unit is fixed to dB, and numeric values including two digits after the decimal point are output.

If analysis cannot be performed, 999.99 is output.

• rl value

rl is Left or Right of peak No. x.

Type 3

From the measurement results, it outputs a gain tilt and a maximum and minimum peak levels.

• a value (gain tilt)

a is the difference between the maximum value and the minimum value.

The unit is fixed to dB, and numeric values including two digits after the decimal point are output.

If analysis cannot be performed, 999.99 is output.

• mx value (maximum peak levels)

mx is the maximum value.

The unit is fixed to dB, and numeric values including two digits after the decimal point are output.

If analysis cannot be performed, 999.99 is output.

• mi value (minimum peak levels)

mi is the minimum value.

The unit is fixed to dB, and numeric values including two digits after the decimal point are output.

If analysis cannot be performed, 999.99 is output.

9.7 SLP [Slope]

■ Function

Links the measurement waveform peaks with a line (line approximation), and performs display and non-display of the line display.

Header	Program	Inquiry	Response
SLP	SLP s	SLP?	s

■ s value

s = ON : Display

= OFF : Non-display

9.8 SLPD [Slope Data]

■ Function

It reads a ramp value as the slope value by connecting each peak point of a WDM measured wave pattern with line (linear approximation).

Header	Program	Inquiry	Response
SLPD	None	SLPD?	n n=xx.xxx

■ n value

n is a slope value of line.

The unit is fixed to dB/nm, and numeric values including three digits after the decimal point are output.

9.9 TPW [Total Power]

■ Function

Performs display and non-display of total power (integral power) from WDM measurement in wavelength range of 1520 to 1580 nm.

Header	Program	Inquiry	Response
TPW	TPW s	TPW?	s

■ s value

s= ON : Display

= OFF : Non-display

9.10 TPWD [Total Power Data]

■ Function

Calculates total power (integral power) from WDM measurement in wavelength range of 1520 to 1580 nm, and reads the value.

Header	Program	Inquiry	Response
TPWD	None	TPWD?	n n=xx.xxx

■ n value

n is total power value.

The unit is fixed to dBm, and numeric values including three digits after the decimal point are output.

9.11 PMK [Peak Marking]

■ Function

Displays wavelength mark to specific peak wavelength for WDM measurement waveform.

Header	Program	Inquiry	Response
PMK	PMK s, x	PMK? x	s

Same as L-Term Test marking

■ • s value

s= ON : Displays wavelength mark

= OFF : Non-displays wavelength mark

• x value

x is Peak No. (0,1,2,3•••)

but in case of inquiry (1,2,3•••)

0 : All Peak No.

9.12 RES [Resolution]

■ Function

Sets the measurement resolution.

Header	Program	Inquiry	Response
RES	RES n	RES?	n

■ n value

Unit is fixed to nm. Input any of the following resolutions.

1.0, 0.5, 0.2, 0.1, 0.07, 0.05

■ Initial setting value

n is backup value.

■ Default value

n = 0.1 (nm)

9.13 VBW [Video Band Width]

■ Function

Sets the Video Band Width (VBW).

Header	Program	Inquiry	Response
VBW	VBW s	VBW?	s

■ s value

s is VBW. Input any of the following integers with unit.

Or, input the corresponding integer with Hz unit.

1 MHz, 100 kHz, 10 kHz, 1 kHz, 100 Hz, 10 Hz

■ Initial setting value

s is backup value.

■ Default value

s = 1 kHz

9.14 TMK [Trace Marker]

■ Function

Sets the trace marker with a wavelength value.

Header	Program	Inquiry	Response
TMK	TMK λ	TMK?	λ , l

■ λ value

λ is wavelength value.

Unit is fixed to nm. Input a value with 4 digits under decimal point.

Data range is from start wavelength ≤ λ ≤ stop wavelength.

l value

l is the level value of the trace marker with the level unit on the set scale.

If the level cannot be calculated on the linear scale, "-1" is output.

9.15 FMT [FD Format]

■ Function

Performs FD formatting.

Set bit 2 (transmission end bit) of expansion event status register (ESR2) to "1" upon completion of file formatting.

Header	Program	Inquiry	Response
FMT	FMT	None	None

9.16 DEL [FD File Delete]

■ Function

Deletes the specified file from FD.

Set bit 2 (transfer end bit) of the expansion event status register (ESR2) to "1" upon completion of file delete.

Header	Program	Inquiry	Response
DEL	DEL n	None	None

■ n value

n is file name.

9.17 FOPT [FD File Option]

■ Function

Sets FD file options.

Header	Program	Inquiry	Response
FOPT	FOPT a, b, c	None	None

■ • a value

a specifies an Add&Save file

a = NONE: None (no add file)

= BMP: *bimp file output

= TXT: *txt (txt (text) file output

= BMP&TXT: bimp and text file output

• b value

b specifies the file ID.

b = NUMBER: File No. input

= NAME: File name input

• c value

c sets the FDD mode. It can be omitted.

c = 1.44 M: PC compatible machine

= 1.2 M: PC98 (NEC/EPSON)

Note:

C changes take effect by turning the power OFF and then ON again.

9.18 SAV [FD File Save]

■ Function

Saves measurement data to a specified FD file.

Set bit 2 (transmission end bit) of the expansion event status register (ESR2) to "1" upon completion of file saving.

Header	Program	Inquiry	Response
SAV	SAV n	None	None

■ n value

Input a file name of 8 characters or less in a format recognizable by DOS, as shown below.

xxxxxxxx. dat

Extension .dat can be omitted.

9.19 RCL [FD File Recall]

■ Function

Recalls specified FD file.

Set bit 2 (transfer end bit) of the expansion event status register (ESR2) to "1" upon completion of file recall.

Header	Program	Inquiry	Response
RCL	RCL n	None	None

■ n value

Input a file name of 8 characters or less in a format recognizable by DOS, as shown below.

xxxxxxxx. dat

Extension .dat can be omitted.

9.20 DMA [Memory Data A]

■ Function

Outputs memory A measurement data in ASCII format for the number of measurement points.

The number of memory A data is the number of measurement points used during measurement.

Header	Program	Inquiry	Response
DMA	None	DMA?	d+terminator (Number of measurement points)

■ d value

d indicates the measurement data; The data format differs depending on the scale.
No header is used for response data, and x is zero suppressed.
○ LOG scale...d = ±xxx.xx : Unit dBm
(–120 dBm ≤ d ≤ +30 dBm)

9.21 DQA [Memory Data A]

■ Function

Outputs memory A measurement data in ASCII format for the number of measurement points.
The number of memory A data is the number of measurement points used during measurement.

Header	Program	Inquiry	Response
DQA	None	DQA?	d+separator (Number of measurement points)

■ d value

d indicates the measurement data; The data format differs depending on the scale.
No header is used for response data, and x is zero suppressed.

○ LOG scale d = ±xxx.xx : Unit dBm
(–120 dBm ≤ d ≤ +30 dBm)

9.22 DBA [Memory Data A]

■ Function

Outputs memory A measurement data in binary format for the number of measurement points.
The number of memory A data is the number of measurement points used during measurement.

Header	Program	Inquiry	Response
DBA	None	DBA?	d (number of measurement points, binary data)

■ d value

d indicates the measurement data. The data format differs depending on the scale.
See Appendix B Binary Data Transfer Format.

9.23 DCA [Data Condition Memory A]

■ Function

Reads data measurement conditions for memory A measurement data.

Header	Program	Inquiry	Response
DCA	None	DCA?	λ1, λ2, n

■ • λ1 value

λ1 is the start wavelength, its unit is fixed to nm, and it is a numeric value with up to two digits after the decimal point.

• λ2 value

λ2 is the stop wavelength, its unit is fixed to nm, and it is a numeric value with up to 2 digits after the decimal point.

• n value

n is a measurement point within the range of 501 to 2001.

9.24 WCAL [Wavelength Calibration]

■ Function

Executes wavelength calibration using a reference light source and creates wavelength calibration data. Set bit 4 (execution end bit) of the expansion event status register (ESR2) to "1" upon completion of wavelength calibration.

Header	Program	Inquiry	Response
WCAL	WCAL n	WCAL?	m

■ • n value

n = 0 : Wavelength calibration data is used as the initial setting value.
= 2 : Wavelength calibration is performed using reference light source, and wavelength calibration data is created.
= 3 : Forcible termination of wavelength calibration

• m value

m = 0 : Normal termination of wavelength calibration
= 1 : Wavelength calibration in progress
= 2 : Interruption of wavelength calibration due to insufficient light level
= 3 : Interruption of wavelength calibration due to other abnormality

9.25 ALIN [Auto Alignment]

■ Function

Performs auto-alignment while measured light is incoming and creates alignment position data. Set bit 4 (execution end bit) of the expansion event status register (ESR2) to "1" upon completion of auto-alignment execution.

Header	Program	Inquiry	Response
ALIN	ALIN n	ALIN?	m

■ • n value

n = 0 : Alignment position data is used as the initial setting value.
= 1 : Executes auto-alignment and creates alignment position data.
= 2 : Forcible termination of auto-alignment.

• m value

m = 0 : Normal termination of wavelength calibration
= 1 : Wavelength calibration in progress
= 2 : Interruption of wavelength calibration due to insufficient light level
= 3 : Interruption of wavelength calibration due to other abnormality.

9.26 DATE [Date Set]

■ Function

Sets year, month, day.

Header	Program	Inquiry	Response
DATE	DATE yy, mm, dd	DATE?	yy, mm, dd

■ • yy:

Input 2-digit numeric value for the year according to Western calendar (00 to 99)

• mm:

Input 2-digit numeric value for the month (01 to 12).

• dd:

Input 2-digit numeric value for the day (01 to 31).

9.27 TIME [Time Set]

■ Function

Sets hours and minutes.

Header	Program	Inquiry	Response
TIME	TIME hh,mi	TIME?	hh, mi

■ • hh:

Input a 2-digit numeric value (00 to 23) for the hour.

• mi:

Input a 2 digit numeric value (00 to 59) for the minutes.

9.28 TDSP [Time & Data Display On/Off]

■ Function

Sets timer display ON/OFF.

Header	Program	Inquiry	Response
TDSP	TDSP s	TDSP?	s

■ s value

s indicates the ON/OFF status of timer display.

s = ON :Timer display ON

= OFF :Timer display OFF

9.29 BUZ [Buzzer On/Off]

■ Function

Sets buzzer ON/OFF.

Header	Program	Inquiry	Response
BUZ	BUZ s	BUZ?	s

■ s value

s = ON :Buzzer ON

= OFF :Buzzer OFF

■ Initial setting value

s is backup value.

■ Default value

s = ON

9.30 LTS [Long Term Test]

9.30.1 LTS [Long Term Test]

■ Function

Selects either spectrum measurement or continuous testing.

Header	Program	Inquiry	Response
LTS	LTS s	LTS?	s

■ s value

s = ON : Continuous testing screen with ON

= OFF : Spectrum measurement screen with OFF

9.30.2 SPT [Sampling Time]

■ Function

Sets sampling time.

Header	Program	Inquiry	Response
SPT	SPT n n=1 to 99, OFF	SPT?	n n=1 to 99, OFF

■ n value

n is sampling time.

The unit is minutes.

The range is $1 \leq x \leq 99$, OFF.

9.30.3 PMK [Peak Marking]

■ Function

Emphasizes display of specified peak No.

Header	Program	Inquiry	Response
PMK	PMK s, x	PMK? x	s

Same as spectrum marking.

■ • s value

s = ON : emphasized

= OFF : non-emphasized

• x value

x is Peak No. (0,1,2,3•••)

but in case of inquiry (1,2,3•••)

0 : All Peak No.

9.30.4 TSA [Test Start]

■ Function

Starts continuous testing.

Header	Program	Inquiry	Response
TSA	TSA n	None	None

■ n value

n is file name.

9.30.5 TSO [Test Stop]

■ Function

Stops continuous testing.

Header	Program	Inquiry	Response
TSO	TSO	None	None

9.30.6 LTSS [Long Term Test State]

■ Function

Displays continuous testing status.

Header	Program	Inquiry	Response
LTSS	None	LTSS?	m

■ m value

m =0 : Normal termination
=1 : During operation
=2 : Abnormal termination

■ • Normal termination

- Save of 1000 data
- Floppy is full
- Stop key is pressed

• During operation

- Continuous testing in progress

• Abnormal termination

- FD Error
- Can't Find Peak
- Optical block error

9.30.7 SPG [Spacing]

■ Function

Sets whether to display the frequency difference with the neighboring peak No.

Header	Program	Inquiry	Response
SPG	SPG s	SPG?	s

■ s value

s = FRQ : Frequency display
= WAVE : Wavelength display

9.30.8 LRCL [Long Term Test Recall]

■ Function

It the Long Term Test, it recalls data saved in an FD.
Set bit 2 (transfer end bit) of the expansion event status register (ESR2) to "1" upon completion of file recall.

Header	Program	Inquiry	Response
LRCL	LRCL n, m	None	None

■ • n value

Input a file name of 8 characters or less in a format recognizable by DOS, as shown below.
xxxxxxxx. lst
Extension .lst can be omitted.

• m value

m is the data No. (1 to 1000)

9.30.9 LTSD DTA [Long Term Test Data]

■ Function

It reads the number of Long Team Test data saved in an FD.

Header	Program	Inquiry	Response
LTSD DTA	None	LTSD? DTA	n

■ n value

n is the data No.
Data range : 0 to 1000.

9.30.10 LTSD PKD [Long Term Test Peak Data]

■ Function

In performing the Long Term Test, it reads peak data with specified number out of the most current measured data.

Header	Program	Inquiry	Response
LTSD PKD	None	LTSD? PKD x x=Peak No. (1,2,3・・・)	PKD, aλ, mxλ, miλ, aL, mxL, miL, aS, mxS, miS aλ =xxxx. xxx mxλ=xxxx. xxx miλ =xxxx. xxx aL =xxxx. xx mxL=xxxx. xx miL=xxxx. xx aS =xxx. xx mxS=xxx. xx miS =xxx. xx

■ x value

Peak No. data saved on FD immediately before.

■ Response data

- aλ value

  aλ is the wavelength average value.

  The unit is fixed to nm, and numeric data including up to three digits after the decimal point is output.

- mxλ value

  mxλ is the wavelength maximum value.

  The unit is fixed to nm, and numeric data including up to three digits after the decimal point is output.

- miλ value

  miλ is the wavelength mimimum value.

  The unit is fixed to nm, and numeric data including up to three digits after the decimal point is output.

- aL value

  aL is the level average value.

  The unit is fixed to dBm, and numeric data including up to two digits after the decimal point is output.

- mxL value

  mxL is the level maximum value.

  The unit is fixed to dBm, and numeric data including up to two digits after the decimal point is output.

• miL value

miL is the level mimimum value.

The unit is fixed to dBm, and numeric data including up to two digits after the decimal point is output.

• aS value

aS is the SNR averaga value.

The unit is fixed to dB, and numeric data including up to two digits after the decimal point is output.

• mxS value

mxS is the SNR maximum value.

The unit is fixed to dB, and numeric data including up to two digits after the decimal point is output.

• miS value

miS is the SNR mimimum value.

The unit is fixed to dB, and numeric data including up to two digits after the decimal point is output.

9.30.11 LTSD SPKD [Long Term Test Start Peak Data]

■ Function

It reads the Peak No. data that was saved in an FD at time of starting the Long Term Test.

Header	Program	Inquiry	Response
LTSD SPKD	None	LTSD? SPKD, x x=Peak No. (1,2,3・・・)	SPKD, λ, L, S λ = xxxx. xxx L = xxxx. xx S = xxx. xx

■ x value

Peak No.data saved on FD.

■ Response data

• λ value

λ is wavelength value of start-time peak No.

The unit is fixed to nm, and numeric data including up to three digits after the decimal point is output.

• L value

L is level value of start-time peak No.

The unit is fixed to dBm, and numeric data including up to two digits after the decimal point is output.

• S value

S is SNR value of start-time peak No.

The unit is fixed to dB, and numeric data including up to two digits after the decimal point is output.

9.30.12 LTSD TPWD [Long Term Test Total Power Data]

■ Function

It reads the most current total power data in performing the Long Term Test.

Header	Program	Inquiry	Response
LTSD TPWD	None	LTSD? TPWD	TPWD, n, mx, mi n = xxxx. xx mx= xxxx. xx mi = xxx. xx

■ Response data

- n value

  n is average value of the total power .

  The unit is fixed to dBm, and numeric data including up to two digits after the decimal point is output.

- mx value

  mx is maximam value of the total power.

  The unit is fixed to dBm, and numeric data including up to two digits after the decimal point is output.

- mi value

  mi is minimam value of the total power.

  The unit is fixed to dBm, and numeric data including up to two digits after the decimal point is output.

9.30.13 LTSD GAT [Long Term Test Gain Tilt]

■ Function

It reads the most current gain tilt data in performing the Long Term Test.

Header	Program	Inquiry	Response
LTSD GAT	None	LTSD? GAT	GAT, n, m, mi n = xxx. xx mx= xxx. xx mi = xxx. xx

■ Response data

- n value

  n is average value of the Gain tilt.

  The unit is fixed to nm, and numeric data including up to two digits after the decimal point is output.

• mx value

mx is maximam value of the Gain tilt.

The unit is fixed to nm, and numeric data including up to two digits after the decimal point is output.

• mi value

mi is mimimam value of the Gain tilt.

The unit is fixed to nm, and numeric data including up to two digits after the decimal point is output.

9.30.14 LTSD SLP [Long Term Test Slope]

■ Function

It reads the most current slope data in performing the Long Term Test.

Header	Program	Inquiry	Response
LTSD SLP	None	LTSD? SLP	LTSD SLP, n, mx, mi n = xxx. xx mx= xxx. xx mi = xxx. xx

■ Response data

• n value

n is average value of the Slope.

The unit is fixed to dB/nm, and numeric data including up to two digits after the decimal point is output.

• mx value

mx is maximamu value of the Slope.

The unit is fixed to dB/nm, and numeric data including up to two digits after the decimal point is output.

• mi value

mi is mimimamu value of the Slope.

The unit is fixed to dB/nm, and numeric data including up to two digits after the decimal point is output.

9.30.15 LTSD TIM [Long Term Test Time]

■ Function

It reads the most current Start and Stop times in performing the Long Term Test.

Header	Program	Inquiry	Response
LTSD TIM	None	LTSD? TIM	TIM h, mi, sh, smi

■ Response data

- h/sh values : The unit is hours.
  The range is 0 to 23.
- mi/smi values : The unit is minutes.
  The range is 0 to 59.

Section 10 Program Examples

10.1 Precautions on Creating a Program

Precautions on creating a remote control program are as follows:

No.	Precaution	Description
1	Be sure to initialize devices.	Devices may be in various states after they have been operated by their own operator panels and other programs. In many cases, these states may not be proper at the start of use. Therefore, these devices must be initialized so that they can be used under certain conditions. 1) Initialization of interface function (IFC@) 2) Initialization of device message exchange function (DCL@) 3) Initialization of device-dependent functions (*RST) When the RS-232C interface is used, only 3) is required.
2	Set devices in the RWLS (Remote With Lockout State).	If devices are set in a simple remote state, they will enter the local state when the LOCAL key is pressed. If a panel key is pressed with the device in the local state, the device cannot carry out automatic measurement properly and therefore the measurement data becomes inaccurate. Execute an LLO@ statement to lock out devices to prevent devices from returning to the local state.
3	Immediately after sending a query, do not send any device-related command other than READ@	If a command other than a READ@ statement is sent to the controller before reading the query result, the output message is cleared at reception of MLA and therefore the response message disappears. Be sure to write a READ@ statement directly after the query.
4	Avoid exception handling in the protocol.	Expected exceptions must be handled in the exception handling section in the program so that execution does not stop due to errors.
5	Check interface functions (subset) of individual devices (GPIB).	If a created program is executed for a device that does not have a subset, processing will not proceed. Be sure to check subsets of devices. Also check the devices conform to IEEE 488.2.
6	Prevent buffer overflow (RS-232C).	The RS-232C interface of the MS9715B has a 256-byte data area as an internal receive buffer. However, overflow may occur depending on the processing type. To prevent errors from occurring due to overflow, do not send a large volume of data (control commands) at a time when performing remote control using the RS-232C interface. After sending a sequence of commands, you can send an *OPC? command, wait for a response to be received, then send the next command for synchronization.

Section 11 LabVIEW Measuring Instrument

This section explains the measuring instrument drivers (MX971502G/S) used to control the MS9715B remotely under LabVIEW.

LabVIEW measuring instrument drivers are modules in which command send and receive functions are incorporated, allowing measuring instruments to be controlled under the U.S. National Instruments Graphic Programming System "LabVIEW." Using these drivers, you can control the MS9715B remotely without remembering control commands.
To use this measuring instrument, a controller in which National Instruments LabVIEW software (Windows version) is installed is required.

The measuring instrument drivers have been created using LabVIEW Ver. 4.0.1/J (Windows version).
For how to use LabVIEW, see the LabVIEW User's Guide.

LabVIEW is a trademark of U.S. National Instruments Corporation.
Windows is a trademark of U.S. Microsoft Corporation.

About LabVIEW

LabVIEW is a graphical program language suitable for controlling measuring instruments and saving and analyzing data.

LabVIEW allows you to create a program as if you drew a circuit diagram, so you can easily get used to use it compared with text-based program languages. The execution speed is almost the same as the C language.

LabVIEW supports various libraries related to measuring instrument control and data saving, analysis, and display. Using LabVIEW and measuring instrument drivers, you can create a graphical user interface (GUI) program with ease.

11.1 Installation

The two floppy disks (MX971502G/S) that come standard with the MS9715B store the following files:

MS9715BG. LLB (GPIB driver)
MS9715BS. LLB (RS-232C driver)

Use the library for a desired driver by copying it to an appropriate directory.

11.2 Measuring Instrument Drivers

The measuring instrument driver file name is assigned as follows:

MS9715□ (card name);(function key name).vi

or

MS9715□ (name corresponding to panel key or function).vi

(For he GPIB driver, □ is left blank. For the RS-232C driver, □ is entered with S.)
Icons resemble the keys on the main unit.

You can select drivers according to the main unit key operation. In most case, you can select the drivers to be used by imagining the main unit key operations.

11.3 Description of Measuring Instrument Driver Functions

This section explains functions and input/output parameters of measuring instrument drivers.

A measuring instrument driver VI receives data and setting values through the terminals on the left of the icon, performs the specified processing according to the input parameters, and outputs the processing results through the terminals on the right side of the icon.

11.3.1 Common parameters

This section explains most of the input/output parameters used with measuring instrument drivers.

instr handle in

instr handle out

"instr handle" is generated by "initialize.vi." It becomes the index for referencing information such as an RS-232C port number.

Arrange drivers in the order of execution and connect "instr handle out" terminals and "instr handle in" terminals, one after another, with wires.

error in

error out

The error information before execution is input to the "error in" terminal. When information indicating an error has occurred is input to the "error in" terminal, the VI performs nor processing and outputs the "error in" value through the "error out" terminal. When information indicating that no error has occurred is input to the "error in" terminal, VI performs the specified processing and outputs the post-processing error state through the "error out" terminal. Thus, errors can be checked. Connecting "error in" and "error out" terminals of VIs, one after another, with wires and using an MS9715 Error Message VI at the end of the diagram allow an error location, error code, and error message to be displayed.

status:	True if an error has occurred.
code:	Error code
source:	Error location

timeout (ms)

If processing is not completed within the timeout time, it is terminated and an error message is output. For processing requiring long time, set a sufficiently long time.

11.3.2 Description of functions

For input parameter setting ranges, output data formats, and so forth, see the help window.
( ) following an input parameter indicates an initial value (default).

MS9715 Close

This driver terminates communication with the device.
Execute it at the end of the program.

MS9715 Error Message

MS9715 Error Message Japanese

Executing this VI after executing a measuring instrument driver will display an error location, error code, and error message if any error has occurred.
If the error is an MS9715B-specific error, the value "MS9715B error code + 5000" is output.
For details on MS9715 error codes, see Appendix A.
("101 Can't Find Peak" and "102 Can't Find Dip" are not handled as errors.)
"MS9715 Error Message Japanese" displays error messages in Japanese.

Input parameter:

type of dialog (OK msg:1):Select an error message display dialog type.

Output parameters:

status:	Indicates whether an error has occurred.
code:	Outputs an error code.
error message:	Outputs an error message.

MS9715 Initialize

This driver initializes the device and generates "instr handle."
It sets the response data header to OFF.
To use a measuring instrument, this VI must be executed first.
Terminate the VI after completion of initialization.
RS-232C interface conditions are as follows:
Parity = Even; Stop Bit = 1; Character length = Fixed at 8 bits
Set MS9715B interface conditions to the above conditions. (For the setting method, see 2.2.3, "Setting RS-232C interface conditions.")

Input parameters:

<GPIB>	
GPIB address (8):	Input a GPIB address.
ID query (Yes:T):	Request the ID of the measuring instrument to check the device against it.
reset (No:F):	Reset the measuring instrument.
<RS-232C>	
Port No. (COM1:0):	Input an RS-232C port number.
Speed (bps) (9600:4):	Set a transmission speed (600/1200/2400/4800/9600 bps).
ID query (Yes:T):	Request the ID of the measuring instrument to check the device against it.
reset (No:F):	Reset the measuring instrument.

MS9715 Reset

This driver resets the measuring instrument.
After completion of resetting, the VI is terminated.
Input parameter:

timeout (ms) (600000) :	Set a reset timeout time.

MS9715 Read Memory Data

This driver outputs the measurement data (Data/Suffix) and data measurement conditions (Condition Data) from the memory.

Input parameter:

Memory A/B (Memory A:F): Select a memory from which data is to be read.

Output parameters:

Condition Data [cluster]The following cluster elements are output:

1. Start Wavelength (nm):	Start wavelength
2. Stop Wavelength (nm):	Stop wavelength
3. Sampling Points:	Number of sampling points
Data (dBm):	Measurement data is output. The number of arrays is equal to the number of sampling points.

MS9715 Sweep Start

This driver starts single/repeat sweep.

In the single sweep mode, the VI is terminated after completion of sweep.

Input parameters:

Single/Repeat (Single:F):	Select single or repeat sweep.
timeout (ms) (600000):	Set a single sweep timeout time.

MS9715 Sweep Stop

This driver stops sweep.

MS9715 Cal ;Auto Alignment

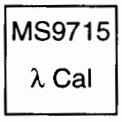
This driver aligns the optical axis automatically.
After completion of alignment, the VI is terminated.

Input parameters:

Mode Select (Initial:0): "Initial" sets the alignment position data to the de fault value.
Execute starts calibration.

timeout (ms) (600000): Set an auto alignment timeout time.

MS9715 Cal ;Wl Calibration



This driver creates wavelength calibration data by performing calibration using reference light source.
After completion of calibration, the VI is terminated.

Input parameters:

Mode Select (Initial:0): "use int.Light" calibrates the wavelength using the internal reference light source, creating wavelength calibration data.
"Initial" sets the wavelength calibration data to the default value.

timeout (ms)(600000): Set a wavelength calibration timeout time.

MS9715 Save/Recall;FD

This driver saves data, recalls data, deletes data, or formats the FD.
When saving, recalling, deletion, or formatting is complete, the VI is terminated.

Input parameters:

Mode (Save:0): Select data saving, data recalling, data deletion, or FD for matting.

Data Addition (no change): Determine whether bit-map-format data and text-format data are to be saved in addition to the basic-format data.
When no value is input, the previous setting is used.

File Name (""): Input a file name.

timeout (ms) (600000): Set a timeout time.

MS9715 Zoom Marker ; Set/Erase

Sets or deletes zoom marker.
If there is no input to Center or Span, the zoom marker is deleted.

Input parameters:

Center (nm)(Erase):	Sets zoom center value.
Span (nm)(Erase):	Sets span value.

MS9715 Zoom Maker ; Zoom Out/In

Sets Zoom In/Zoom Out display for zoom marker.

Input parameter:

Zoom Out/In (Out : F): Sets Zoom In/Zoom Out.

MS9715 Res/BW/Avg ; Res/BW/SmplPt

Sets the measurement resolution and the reception light bandwidth. Without input, the setting value will not be changed.

Input parameters:

Resolution (nm) (no change):	Sets the measurement resolution.
Video Band Width (no change):	Sets the reception light bandwidth.

Output parameters:

Sampling Points:	Outputs the number of measurement points in the span.
Resolution (nm):	Outputs the measurement resolution.
Video Band Width (Hz):	Outputs the reception light bandwidth.

MS9715 Marker ; Trace Marker MS9715 TMkr

Sets the Trace Marker and outputs the Marker values (wavelength, level/unit).

Input parameter:

Wavelength (nm) (no change): Sets the wavelength of the Trace Marker. Without input, the setting conditions will not be changed. (Mandatory when the Trace Marker is not is not displayed.)

Output parameters:

Wavelength (nm):	Outputs the wavelength of the Trace Marker.
Level Suffix:	Outputs the level and unit of the Trace Marker.

MS9715 Spectrum ; Peak MS9715 Spectrum Peak

Outputs number of peaks detected from measurement data.
Also outputs the wavelength, level, SNR, DIP direction (long wavelength or short wavelength side) when a specific peak No. is input.

Input parameter:

Peak No. (1 to 32):	Sets peak No. for data to be output.

Output parameters:

Peak Count:	Outputs the number of detected peaks.
Wavelength (nm):	Outputs the wavelength of the input peak No.
Level (dBm):	Outputs the level value of the input peak No.
SNR (dB):	Outputs the SNR value of the input peak No.
Left/Right 1:Left 2:Right	Left/RightOutputs whether the DIP level when cal culating the SNR value for an input peak No. is on the long wavelength or short wavelength side.

MS9715 Spectrum ; Gain Val

Outputs gain val value, slope value, and total power value from measurement data.

Output parameters:

Gain Val (dB):	Outputs the gain val value, slope value, and total power value from the measurement data.
Max Lvl (dBm):	Outputs the maximum level from the peak level of the measurement data.
Min Lvl (dBm):	Outputs the minimum level from the peak level of the measurement data.

MS9715 Spectrum ; Slo/T-pow

Outputs gain val value, slope value, and total power value from measurement data.

Input parameters:

Slop Off/On (Off: F):	Sets whether or not to calculate the slope value.
Totals Power Off/On (Off: F):	Sets whether or not to perform total power measure ment and calculation.

Output parameters:

Slop (dB/nm):	Outputs slope value from measurement data when slope value On is input.
Total Power (dBm):	Measures total power when total power On is input, calculates total power value based on measurement data and outputs this value.

MS9715 Marking

Displays marker to peak No. specified in case of spectrum display. In case of continuous testing table data, emphasizes display of specified peak No.

Input parameters:

Marking No. (0 to 32):	Sets peak No. to be marked.
Off/On (Off: F):	Sets On/Off of peak No. to be marked.

MS9715 L-T Test ; Display MS9715 L-T Test Display

Selects whether to perform spectrum measurement or continuous testing.

Input parameters:

L-Term Test

Off/On (Off: F): Performs continuous testing at On. Performs spec trum measurement at Off.

MS9715 L-T Test ; Start/Stop MS9715 L-T Test Start/Stop

Start/stops continuous testing.
Also sets sampling time.

Input parameters:

Sampling Time (1 to 99): Set input value from 1 to 99 minutes.

Mode (Start:0): Start/stop continuous testing.

File Name (" "): Input file name.

MS9715 L-T Test ; WDM Data MS9715 L-T Test WDM Data

Outputs number of detected peaks from measurement data of continuous testing. Also outputs wavelength (average, maximum, and minimum during measurement inside sampling time), level (average, maximum, and minimum during measurement inside sampling time), SNR (average, maximum, and minimum during measurement inside sampling time) upon input of specific peak No. (latest measurement data)

Input parameter:

Peak No. (1 to 32): Sets peak No. of data to be output.

Output parameters:

Wavelength (nm): Outputs the wavelength values (average, maximum, and minimum) of the wavelength value of the input peak No.

Level (dBm): Outputs the level values (average, maximum, and minimum) of the input peak No.

SNR (dB): Outputs SNR value of input peak No. (average, maximum, and minimum)

MS9715 L-T Test ; Start Data

Outputs the continuous testing start time wavelength, level, and SNR upon input of a specific peak No.

Input parameter:

Peak No. (1 to 32):	Sets peak No. of data to be output.

Output parameters:

Wavelength (nm):	Outputs the wavelength at the start of continuous testing for the input peak No.
Level (dBm):	Outputs the level value at the start of continuous testing for the input peak No.
SNR (dB):	Outputs the SNR value at the start of continuous testing for the input peak No.

MS9715 L-T Test ; T-Pow

Outputs the latest total power data in continuous testing (latest data saved on FD).

Output parameters:

Avg T-power (dBm):	Outputs total power average during measurement within sampling time.
Max T-power (dBm):	Outputs total power maximum during measurement within sampling time.
Min T-power (dBm):	Outputs total power minimum during measurement within sampling time.

MS9715 L-T Test ; Gain Val

Outputs the latest gain val data during continuous testing (latest data saved on FD).

Output parameters:

Avg Gain Val (dB):	Outputs gain val average during measurement within sampling time.
Max Gain Val (dB):	Outputs gain val maximum during measurement within sampling time.
Min Gain Val (dB):	Outputs gain val minimum during measurement within sampling time.

MS9715 L-T Test ; Slop

Outputs latest slope data during continuous testing (latest data saved on FD).

Output parameters:

Avg Slop (dB): Outputs slope average during measurement within sam pling time.

Max Slop (dB): Outputs slope maximum during measurement within sam pling time.

Min Time (dB): Outputs slope maximum during measurement within sam pling time.

MS9715 L-T Test ; Time

Outputs previous measurement start time and stop time during continuous testing (measurement start time and stop time for latest data saved on FD).

Output parameters:

Start Time: Outputs measurement start time.

Stop Time: Outputs measurement stop time.

MS9715 L-T Test; Recall

Outputs data saved during continuous measurement to screen.

Input parameters:

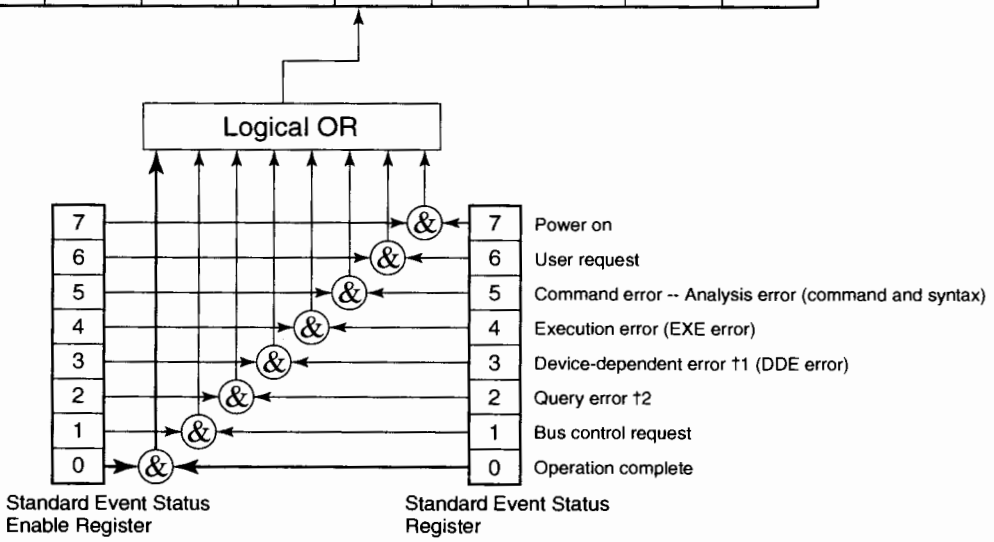
File Name (" "): Sets the file name of the file to be accessed.

Data No. (1 to 1000): Sets the No. of the data to be accessed.

Appendix A Error Messages

This appendix lists error messages summarized by bits 5 and 3 of the status byte register. Bit 5 indicates error messages reported by bits 2 to 5 of the standard event status register. Bit 3 indicates RES-Uncal and Peak/Dip errors.

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 (ERROR)	ESB (END)	ESB (reserved)	ESB (reserved)



† 1 Device-dependent error : Condition error (setting condition error, etc.)
: Hardware error (CAL error)

† 2 Query error : The host has not read a received query message.

When an execution error (EXE error: bit 4) or device-specific error (DDE error: bit 3) of the standard event status register is set, the MS9715B reports its error number and message to the operator as the information about the error cause. GPIB error numbers can be displayed using a ERR? command (see the explanation of device messages).

The lists on the following pages summarizes error numbers and messages.

1. System Errors (000 to 099)

No.	Error message	Status	Output condition
000	No error		
001	Optical Error (RAM)		RAM error
002	Optical Error (Slit-1)		Slit 1 error
003	Optical Error (Slit-2)		Slit 2 error
004	Optical Error (WI Align)		Wavelength alignment error
005	Optical Error (Opt Att)		Optical ATT error
006	Reserved		
007	Optical Error (Light Source)		Light source error
008	Optical Error (Grating)		Grating error
009	Optical Error (Offset)		Offset error
010	Optical Error (Over Power)		Excessive light input error

2. Measurement Errors (100 to 199)

No.	Error message	Status	Output condition
101	Can't Find Peak	ESE-DDE	No peak was found.
102 to 109	Reserved		
110	WI Cal Error (Optical Level)	ESE-DDE	Insufficient optical level during wavelength calibration
111	WI Cal Error	ESE-DDE	Wavelength calibration error
112	Align Error (Optical Level)	ESE-DDE	Insufficient optical level during optical axis automatic alignment
113	Align Error	ESE-DDE	Optical axis automatic alignment error

3. Key Operation Errors (200 to 299)

No.	Error message	Status	Output condition
200	Reserved		
201	Input Value Error	ESE-EXE	The input value exceeds the range.
202 to 209	Reserved		
210	Valid Only In Spectrum Mode	ESE-DDE	Valid only during the spectrum measurement.
211 to 214	Reserved		
215	Invalid In Others Input	ESE-DDE	Invalid during Others input.
216 to 269	Reserved		
270	Invalid In L-Term Test	ESE-DDE	Invalid during testing
271	Invalid Cal	ESE-DDE	Invalid during calibration
272	Zoom Marker Not Display	ESE-DDE	Invalid when zoom marker displayed

4. Device Errors (300 to 499)

No.	Error message	Status	Output condition
	–Errors related to FD–		
300	FD Does Not Exist	ESE-DDE	An FD has not been set.
301	FD Format Error	ESE-DDE	The FD format is invalid.
302	Can't Find File	ESE-DDE	The specified file does not exist on the FD.
303	Undefined		
304	FD Write Protected	ESE-DDE	The FD is write-protected
305	File Incomplete	ESE-DDE	Files on the FD are incomplete.
	–Errors related to printer–		
	–Errors related to GPIB/RS-232C		
400	Reserved		
401	Command Error	ESE-CME	An undefined header has been received.
402	Command Error	ESE-CME	The integer part of the numeric data is invalid.
403	Command Error	ESE-CME	The real part of the numeric data is invalid, or invalid real-type data has been input.
404	Command Error	ESE-CME	The integer part of the numeric data is invalid, or invalid exponent-type data has been input.
405	Command Error	ESE-CME	The suffix part (unit) is invalid.
406	Command Error	ESE-CME	The number of arguments does not follow command syntax.
407	Command Error	ESE-CME	A *PCB command was received but there is no controller function for this command.

Appendix B Binary Data Transfer Formats

This appendix explains formats of the binary data transferred in response to query commands DBA? and DBB?.

- Log scale

Data structure	16 bits/data
Transfer order	High-order byte → Low-order byte
Numeric representation	Signed 16-bit value (0.01 dBm = 1) Measured value = Input value × 0.01 dBm
Unit	dBm

Example: When 2-byte input values are 233 and 162
Hexadecimal notation : E9A2
Decimal notation : –5726
Measured value : –5726 × 0.01 dBm = –57.26 dBm

Appendix C Comparison Table of GPIB Commands of Controller

Function \ Controller	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-address;variable LINE INPUT @talker-address,listener-address;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( ) (asynchronous)	SEND select-code ;message-list

Controller / Function	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 D Example of Program Used on PC9801

This appendix gives an example of a program that reads measurement data using a PC-9801 personal computer. The following program lines correspond to those of the PACKET V sample program on page 10-2.

```
10  '  ******  GPIB  initialize  ******
20     ISET       IFC
30     ISET       REN
40     CMD        DELIM = 2
50     CMD        TIMEOUT = 5
60  '
70     DIM        D $(500)
80  '
90  '  ******  DATA  READ  TEST  ******
100    PRINT  @8 ; DAM?"
110    FOR  1 = 0  TO  500
120    INPUT  @8 ; D $ (1)
130    PRINT  I , D $(1)
140    NEXT  I
150 '
160    END
```

Lines 20 to 50 : GPIB initialization
Line 70 : Array declaration
Lines 100 to 140 : Equivalent to PACKET V sample program

Notes:

- When controlling the MS9715B from the PC-9801 via the GPIB, be sure to initialize the GPIB at the beginning of the program.
- DBA?, DBB?, DQA?, and DQB? are intended for personal computers that can input/output arrays. If your personal computer does not have an array input/output function or it does not use an equivalent program, use DMA? and DMB? commands.

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., Equltronics 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

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, Auklend 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

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