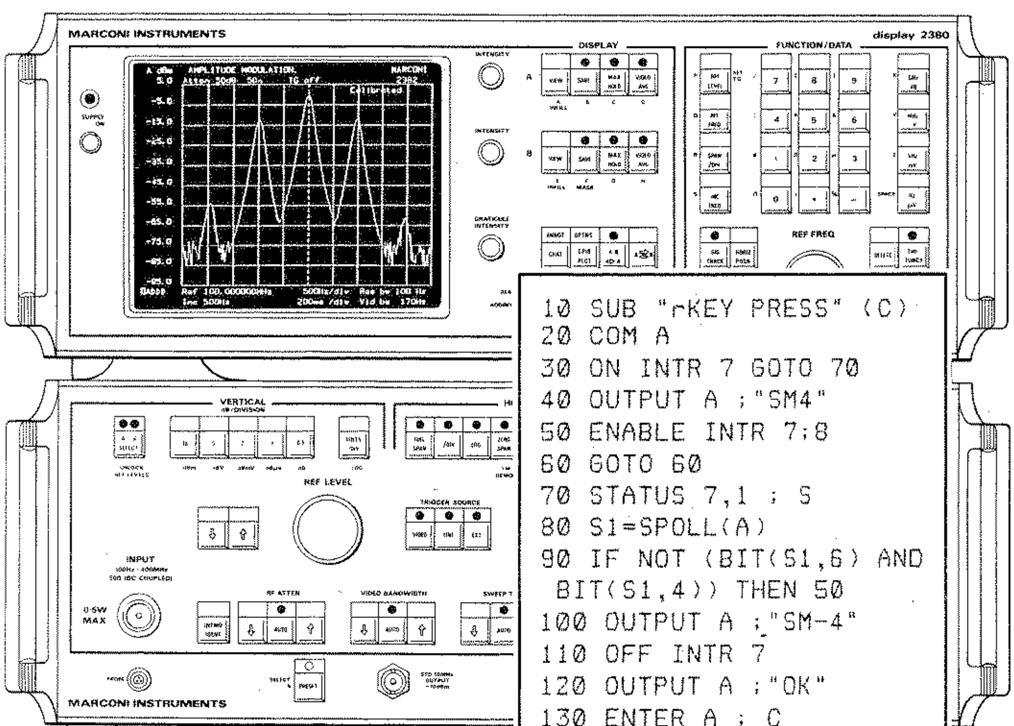
Marconi
Instruments

C 3451 C 3462

2382/2380

100Hz - 400MHz SPECTRUM ANALYZER AND DISPLAY




```
10 SUB "rKEY PRESS" (C)
20 COM A
30 ON INTR 7 GOTO 70
40 OUTPUT A ;"SM4"
50 ENABLE INTR 7;8
60 GOTO 60
70 STATUS 7,1 ; S
80 S1=SPOLL(A)
90 IF NOT (BIT(S1,6) AND
 BIT(S1,4)) THEN 50
100 OUTPUT A ;"SM-4"
110 OFF INTR 7
120 OUTPUT A ;"OK"
130 ENTER A ; C
140 SUBEND
```

GPIB Operating Manual

H 52382-900A
Vol. 1A
2nd Edition

100 Hz — 400 Hz
SPECTRUM ANALYZER and DISPLAY
2382/2380

GPIB Operation

Printed in the UK

Part No. 46881-583Y
Print code: A–11/86

CONTENTS

Note:

Each page bears the date of its original issue or, if it has been amended, the date and status number of the amendment. Any changes subsequent to the latest amendment status are included on Manual Change sheets coded C1, C2 etc. at the front of the manual.

Chapter 1

INTRODUCTION

FEATURES

The 2382 100 Hz to 400 MHz Spectrum Analyzer is equipped for remote operation. All functions (apart from power ON, A and B intensity controls, graticule intensity control and those related to the second marker) on both units are GPIB controllable via the 24-way socket on the rear panel. This socket allows the instrument to be coupled to the GPIB Controller. Any Controller which interfaces with the IEEE-488 bus is suitable and Controllers with IEC 625-1 interfaces may be used with a Marconi Instruments adapter block (part number 46883-408K).

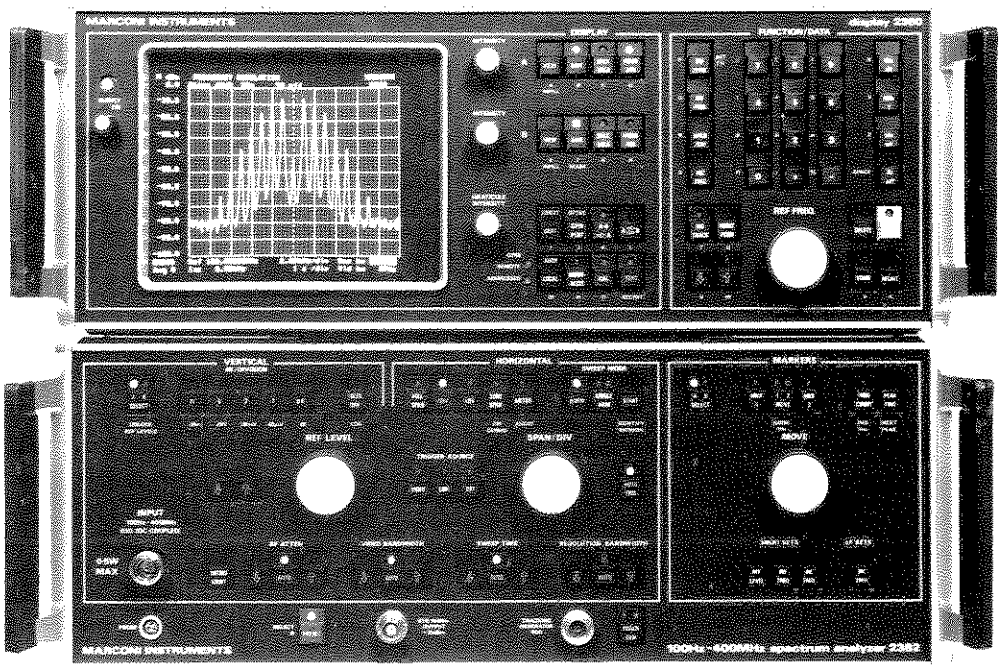


Fig. 1. The 2382 Spectrum Analyzer with 2380 Display

The Spectrum Analyzer responds to a remote command from the Controller in a similar way as to a local front panel command. General programming concepts and procedures are set out in IEEE Standard 488-1978 and IEC Publication 625-1 and an introduction to the instrument bus, principles of operation, commands, data transfer etc. is given in 'The GPIB Manual' H 54811-010P (an optional accessory). Programming commands are summarized in Table 2–2 and on the pull-out card at the bottom of the instrument and more detailed information is provided in the 'Command functions' section of this manual.

The 2382/2380 has talker, listener and talk only capabilities. One address is used for talk and listen and this is initially set by means of the address switch on the rear panel beside the GPIB connector. The address is set as the sum of the numbers switched e.g. for address 12, the 8 and 4 switches are pressed to the left and all the others to the right. Addresses 0 to 30 are available; 31 puts the analyzer in talk only. From power up the instrument's internal address register is updated continuously to the setting of this address switch. In LOCAL mode The address appears in the CRT display if the 2ND FUNCT key and then the LOCAL key is pressed. The status of the instrument's software can be read over the GPIB and also displayed on the CRT by pressing the 2ND FUNCT key and then the RECALL key on the front panel.

Capability codes to identify the interface functions applicable to the 2382/2380 are marked on the rear panel, just above the GPIB address switch. They are also listed below.

Indicator lights on the front panel show when the instrument is under remote operation (REMOTE indicator light ON) and when it is being addressed by the Controller (ADDRESSED indicator light ON). Under remote operation all front panel controls except LOCAL are disabled to guard against manual intervention. Pressing the LOCAL key restores manual control under these circumstances, but if the controller sends the local lock-out command the LOCAL key is disabled and to restore local operation a further command, GTL – go to local, is necessary.

AVAILABLE GPIB FUNCTIONS

The fundamental communication capabilities provided by the GPIB are as follows:-

Source handshake(SH1)	– complete capability
Acceptor handshake (AH1)	– complete capability
Talker (T5)	– basic talker, serial poll, talk only, unaddress if MLA
Listener (L4)	– basic listener, unaddress if MTA
Service request (SR1)	– complete capability
Remote/local (RL1)	– complete capability, when in remote or local lock out, selected groups of keys can be enabled and disabled using commands sent over the bus.
Device clear (DC1)	– complete capability, the instrument adopts its preset state.
Device trigger (DT1)	– complete capability
Parallel poll (PP0)	– no capability
Controller (C0)	– no capability
Open collector drivers (E1)	– as opposed to tri-state drivers

The essential purposes of the above GPIB interface functions are described below.

SH1: Source handshake. The source handshake sequences the transmission of each data byte over the bus data lines. The sequence is initiated when the instrument becomes active as a talker and the purpose of the function is to synchronize the rate at which bytes become available to the rate at which accepting devices on the bus can receive the data.

AH1: Acceptor handshake. The acceptor handshake sequences the reading of the data byte from the bus data lines.

T6: Talker. The talker function provides the 2382/2380 with the ability to send device dependent messages over the bus to other devices. The ability of the instrument to talk exists only when it has been addressed as a talker.

L4: Listener. The listener function provides the 2382/2380 with the ability to receive device dependent messages over the bus. The capability exists only when the instrument is addressed to listen via the bus by the Controller.

SR1: Service request. The service request function gives the instrument the capability to inform the Controller when it requires attention.

RL1: Remote/local. The remote/local function allows the 2382/2380 to be controlled either by the local front panel keys or by device dependent messages over the bus. When in remote or local lock-out, selected groups of buttons can be enabled and disabled using commands sent over the bus.

DC1: Device clear. The DC1 function provides this instrument with the capability to be cleared (initialized) either individually (SDC) or as part of a group of devices (DCL). On receipt of 'selected device clear' (SDC) or 'device clear' (DCL) the 2382/2380 performs a software reset, which returns it to its preset state.

DT1: Device trigger. The device trigger function provides the instrument with the ability to have its basic operation started either individually or as part of a group of devices.

The GPIB electrical interface within the 2382/2380 comprises open collector bus drivers (E1) rather than tristate devices.

Chapter 2

WRITING A PROGRAM

The analyzer is operated remotely by a command or series of commands from a controller over the bus. These commands must conform to a specified message format.

SYNTAX

Before studying message format it is useful to consider the various and symbols which may be used.

Notation

‹ ›	An item enclosed in angular brackets is considered to be an element of the language being defined.
:=	This symbol means 'is defined as'.
[ ]	Items enclosed in square brackets are optional.
|	A list of items separated by vertical lines indicates a choice of one and only one of the items.
. . . or ( )..	Indicate a repetition of the previous item(s).

Definitions

‹listen›:=	Analyzer addressed to listen.
‹talk›:=	Analyzer addressed to talk.
‹C›:=	General ASCII character. Specific characters are printed literally.
‹nB›:=	8-bit binary byte; n shows the value of the byte in decimal (0 to 255). It is omitted if the value of the byte is not defined.
‹CS›:=	Modulo 256 check byte.
‹eoi›:=	EOI line true.
‹space›:=	ASCII space character (ASCII 32_{10}).
‹header›:=	Two character function code i.e. ‹letter›‹letter›.
‹number›:=	A number either implicit, explicit or floating point (IEEE Std. 728 NR1,NR2,NR3).
‹separator›:=	; (ASCII 59_{10}) or , (ASCII 44_{10}).
‹delimiter›:=	Carriage return ‹cr› (ASCII 13_{10}), line feed ‹lf› (ASCII 10_{10}) or ‹eoi›.

‹string›:= Sequence of characters within double quotes (ASCII 34_{10}).
‹"›‹C›[‹C›] . . . ‹"›
A literal double quotes character within a string is represented by two consecutive double quotes
‹"›‹C›‹"›‹"C"›‹"›‹C›‹"›

‹units code›:=

Power	DB			
voltage	VL	MV	UV	NV
frequency	GZ	MZ	KZ	HZ
time	SC	MS	US	
resistance	MO	KO	OH	

‹parameters›:= Function parameter values.

‹response›:= Form of the replay sent to the Controller.

MESSAGE FORMAT

The analyzer is first addressed to listen and then addressed to talk if a response is required. The form of a command entry is therefore:

‹listen›(‹header›‹parameters›[‹separator›]) . . . ‹delimiter›[‹talk›‹response›]

e.g. Using the HP85 Controller the message format to set the Spectrum Analyzer to 10 MHz Span and -10 dBm Reference Level is:

OUTPUT 701; "SP10MZ;RL-10DB"

The header consists of two letters and serves to identify the command. For ease of programming, the codes have been chosen to be similar to the labels associated with front panel keys and controls e.g. VS for vertical scale. In general the first letter shows the area with which the command is concerned (refer to Table 2–1).

The syntax of the ‹parameters› field varies for each command (refer to Table 2–2). Where a ‹number› is included this may be expressed in either integer, real or exponent notation. There is a limit of 80 digits before or after the decimal point and the exponent must not exceed the range +19 to −19.

Where a single line contains a number of commands, to improve clarity, a separator (comma or semi-colon) may be used to separate one command from the next. Alternatively commands may be separated by spaces as these are ignored by the instrument.

Each line must be terminated by a delimiter (carriage return ‹cr›, line feed ‹lf› or EOI (End Or Identify) set true). EOI is a line in the GPIB bus, it can be set true by inputing an appropriate command from the Controller. The Controller manual should be consulted for this information.

Table 2–2 lists all the recognized commands and where additional information has been provided a paragraph reference is given. As an aid to programming, command mnemonics are shown associated with their respective analyzer keys in Fig 2–3 at the end of this chapter.

TABLE 2–1 COMMAND AREA IDENTIFICATION

First header letter	Area
A	A trace
B	B trace
C	Calibration, colour pallette
D	Display (general)
F	Frequency
G	Graticule
H	Horizontal axis
I	Identify
L	Learn
M	Markers
N	Normalize
O	Output
P	Stores
R	RF attenuation, reference level
S	Sweep, span
T	Tracking generator, trigger
U	Unlock
V	Vertical axis, video bandwidth
W	Resolution bandwidth

COMMAND EXECUTION

The Controller manual should be consulted for commands to set the Spectrum Analyzer into a required state e.g.

Controller to talk, Analyzer to listen
Analyzer to talk, Controller to listen
Analyzer local lock-out (LLO)
Analyzer 'go to local' (GTL)
Controller to talk to other instruments, Analyzer to stand by. etc.

Providing the REN (remote enable) bus signal is present, addressing the Analyzer as a listener causes transition to the REMOTE state. The REMOTE and ADDRESSED lights on the analyzer front panel go ON and the instrument is ready to accept commands from the Controller. The REN bus signal is a function of the Controller.

If REN is not present when the Analyzer is addressed commands from the Controller are discarded.

Except for block data, commands within a line are stored until a delimiter is received providing the input store is not full. A line must not exceed 256 characters or bytes. On receipt of the delimiter the commands are scanned and executed in turn. Execution stops when all complete lines in the input store have been dealt with. If more than 256 characters or bytes of data are sent without a delimiter then the content of the store is discarded and an error reported. Thereafter the Analyzer continues to receive data as normal. Block data is processed as it is received and the length restriction of 256 characters does not apply.

When the data store is full the rate at which data is allowed to enter it from the bus is retarded to match the rate at which the Analyzer processes it.

If an error is found when the commands are being scanned, an error message is generated.

When the Spectrum Analyzer is in the REMOTE state it performs a sweep abort if certain operational parameters are altered e.g. reference frequency, RF attenuation etc. Sweeping remains aborted until a Take sweeps (TS) or a Take sweeps with initial first sweep (TI) command is sent by the Controller.

Commands that generate a response cause the response to be written into an output store. This is cleared before the response is written into it and it is not output until command execution stops. This guarantees that data received by the controller corresponds to that last requested. Two output stores are provided for use under certain circumstances. If the response is not available an empty string is returned, ‹cr› ‹lf + eoi› only, and an error is generated.

Programming examples to illustrate how the 2382/2380 Spectrum Analyzer can be operated remotely via the GPIB are given in Chap. 3.

TABLE 2–2

<table>
<tr><th>Header</th><th>Syntax</th><th>Description and parameter limitations</th></tr>
<tr><td>A</td><td></td><td></td></tr>
<tr><td>AA</td><td>‹listen› AA ‹number›</td><td>A video average. Factor = 0 to 7.</td></tr>
<tr><td>OPAA</td><td>‹listen›OPAA‹talk›AA‹number›</td><td>Output A video average factor.</td></tr>
<tr><td>AB</td><td>‹listen›AB0|1</td><td>A minus B to A. (A-B→A) Off or on.</td></tr>
<tr><td>OPAB</td><td>‹listen›OPAB‹talk›AB0|1</td><td>Output state of A-B→A.</td></tr>
<tr><td>AH</td><td>‹listen›AH0|1</td><td>A max hold. Off or on.</td></tr>
<tr><td>OPAH</td><td>‹listen›OPAH‹talk›AH0|1</td><td>Output state of A max hold.</td></tr>
<tr><td>AI</td><td>‹listen›AI0|1</td><td>A infill. Off or on.</td></tr>
<tr><td>OPAI</td><td>‹listen›OPAI‹talk›AI0|1</td><td>Output state of A infill.</td></tr>
<tr><td>AN</td><td>‹listen›AN0|1</td><td>Annotation. Off or on.</td></tr>
<tr><td>OPAN</td><td>‹listen›OPAN‹talk›AN0|1</td><td>Output state of annotation.</td></tr>
<tr><td>AS</td><td>‹listen›AS0|1</td><td>A save. Off or on.</td></tr>
<tr><td>OPAS</td><td>‹listen›OPAS‹talk›AS0|1</td><td>Output state of A save.</td></tr>
<tr><td>AU</td><td>‹listen›AU0|1</td><td>Audio. Off or on.</td></tr>
<tr><td>OPAU</td><td>‹listen›OPAU‹talk›AU0|1</td><td>Output state of audio.</td></tr>
<tr><td>AV</td><td>‹listen›AV0|1</td><td>A view. Off or on.</td></tr>
<tr><td>OPAV</td><td>‹listen›OPAV‹talk›AV0|1</td><td>Output state of A view.</td></tr>
<tr><td>B</td><td></td><td></td></tr>
<tr><td>BA</td><td>‹listen›BA‹number›</td><td>B video average. Factor = 0 to 7.</td></tr>
<tr><td>OPBA</td><td>‹listen›OPBA‹talk›BA‹number›</td><td>Output B video average factor.</td></tr>
<tr><td>BE</td><td>‹listen›BE‹number›</td><td>Beeper. Number = 0 to 8.
0 Off
1 High momentary beep
2 Low momentary beep
3 High continuous tone
4 Low continuous tone
5 High intermittent tone
6 Low intermittent tone
7 Warble
8 Not used</td></tr>
<tr><td>BH</td><td>‹listen›BH0|1</td><td>B max hold. Off or on.</td></tr>
<tr><td>OPBH</td><td>‹listen›OPBH‹talk›BH0|1</td><td>Output state of B max hold.</td></tr>
<tr><td>BI</td><td>‹listen›BI0|1</td><td>B infill. Off or On.</td></tr>
<tr><td>OPBI</td><td>‹listen›OPBI‹talk›BI0|1</td><td>Output state of B infill.</td></tr>
<tr><td>BO</td><td>‹listen›BO‹number›[(,|;‹number›) . . .]</td><td>Beeper options. Number = 1 to 3.
1 On error
2 On end of sweep
3 On overload
Positive number enables option, negative number disables option.</td></tr>
<tr><td>OPBO</td><td>‹listen›OPBO‹talk›BO‹number›(,‹number›). . .</td><td>Output beeper options.</td></tr>
<tr><td>BS</td><td>‹listen›BS0|1</td><td>B save. Off or on.</td></tr>
<tr><td>OPBS</td><td>‹listen›OPBS‹talk›BS0|1</td><td>Output state of B save.</td></tr>
<tr><td>BV</td><td>‹listen›BV0|1</td><td>B view. Off or on.</td></tr>
<tr><td>OPBV</td><td>‹listen›OPBV‹talk›BV0|</td><td>Output state of B view.</td></tr>
</table>

<table>
<tr><th>Header</th><th>Syntax</th><th>Description and parameter limitations</th></tr>
<tr><td>C</td><td></td><td></td></tr>
<tr><td>CA</td><td>‹listen›CA</td><td>Perform self calibration routine SRQ(5) upon completion. Ref. OI command.</td></tr>
<tr><td>CL</td><td>‹listen›CL‹number›</td><td>Colour palette. Number = 0 or 1.
0 Normal
1 Strong.</td></tr>
<tr><td>OPCL</td><td>‹listen›OPCL‹talk›CL‹number›</td><td>Output state of colour palette.</td></tr>
<tr><td>CR</td><td>‹listen›CR‹number›,|;‹number›</td><td>Enter frequency response array point. 1st number 0 to 899, 2nd number − 32768 to 32767.</td></tr>
<tr><td>OPCR</td><td>‹listen›OPCR‹number›‹talk›CR‹space›
‹number›‹cr›‹lf›‹eoi›</td><td>Output frequency response array point. Number = 0 to 899. Ref. Chap. 4.</td></tr>
<tr><td>D</td><td></td><td></td></tr>
<tr><td>DC</td><td>‹listen›DC[‹number›(,|;‹number›) . . .]</td><td>Display clear. Up to 4 numbers which are start line, start column, stop line, stop column respectively. Region between start line/column and stop line/column is cleared. (filled with nulls).
Lines = 1 to 24, columns = 1 to 65.
Default values are 1,1,24,65.</td></tr>
<tr><td>DS</td><td>‹listen›DS0|1</td><td>Display select. Main or alternative.†</td></tr>
<tr><td>OPDS</td><td>‹listen›OPDS‹talk›DS0|1</td><td>Output state of display select.</td></tr>
<tr><td>DT</td><td>‹listen›DT‹number›,|;‹number›#I‹B›. . .
‹eoi›</td><td>Display text. Numbers give starting line & column respectively. Line = 1 to 24, column = 1 to 65.†</td></tr>
<tr><td>OPDT</td><td>‹listen›OPDT[‹number›(,|;‹number›) . . .]
‹talk› DT#I‹B› . . . ‹eoi›</td><td>Read display text. Up to four numbers which are start li ne, start column, stop line, stop column respectively. Region between start line/column and stop line/column is output.</td></tr>
<tr><td>DX</td><td>‹listen›DX</td><td>AB display exchange (A↔B)</td></tr>
<tr><td>F</td><td></td><td></td></tr>
<tr><td>FA</td><td>‹listen›FA0|1</td><td>Auto incremental frequency. Off or on. (Auto setting gives 10th of span).</td></tr>
<tr><td>OPFA</td><td>‹listen›OPFA‹talk›FA0|1</td><td>Output state of auto incremental frequency</td></tr>
<tr><td>FD</td><td>‹listen›FD</td><td>Decrement reference frequency by value of increment frequency. (↓).</td></tr>
<tr><td>FI</td><td>‹listen›FI‹number›[HZ|KZ|MZ|GZ]</td><td>Set increment frequency. 0 − 400 MHz.</td></tr>
<tr><td>OPFI</td><td>‹listen›OPFI‹talk›FI‹number›HZ</td><td>Output increment frequency.</td></tr>
<tr><td>FR</td><td>‹listen›FR‹number›[HZ|KZ|MZ|GZ]</td><td>Set reference frequency 0 − 400 MHz.</td></tr>
<tr><td>OPFR</td><td>‹listen›OPFR‹talk›FR‹number›HZ</td><td>Output reference frequency.</td></tr>
<tr><td>FT</td><td>‹listen›FT0|1</td><td>Signal track . Off or on. SRQ(1) − error if signal is lost.†</td></tr>
<tr><td>OPFT</td><td>‹listen›OPFT‹talk›FT0|1</td><td>Output state of signal track.</td></tr>
<tr><td>FU</td><td>‹listen›FU</td><td>Increment reference frequency by value of increment frequency (↑).</td></tr>
</table>

†For detailed information on this command see 'Command functions'

<table>
<tr><th>Header</th><th>Syntax</th><th>Description and parameter limitations</th></tr>
<tr><td>G</td><td></td><td></td></tr>
<tr><td>GM</td><td>‹listen›GM0|1</td><td>Minor gratiule lines. Off or on.</td></tr>
<tr><td>OPGM</td><td>‹listen›OPGM‹talk›GM0|1</td><td>Output state of minor graticule lines.</td></tr>
<tr><td>GR</td><td>‹listen›GR0|1</td><td>Graticule. Off or on.</td></tr>
<tr><td>OPGR</td><td>‹listen›OPGR‹talk›GR0|1</td><td>Output state of graticule.</td></tr>
<tr><td>H</td><td></td><td></td></tr>
<tr><td>HL</td><td>‹listen›HL‹number›HZ|KZ|MZ|GZ|,|;
number›[HZ|KZ|MZ|GZ]</td><td>Logarithmic sweep. Frequencies are start frequency and stop frequency respectively 100 Hz – 1 GHz. Decade values only.</td></tr>
<tr><td>OPHL</td><td>‹listen›OPHL‹talk›HL‹number›HZ,
‹number›HZ</td><td>Output logarithmic sweep start and stop frequencies.</td></tr>
<tr><td>HP</td><td>‹listen›HP‹number›</td><td>Horizontal position. Number = 0 to 2.
0 Left
1 Centre
2 Right</td></tr>
<tr><td>OPHP</td><td>‹listen›OPHP‹talk›HP‹number›</td><td>Output horizontal position.</td></tr>
<tr><td>HR</td><td>‹listen›HR‹number›</td><td>Horizontal mode. Number = 0 to 5.
0 Full span
1 Per division
2 Log span
3 Zero span
4 Meter
5 FM demod</td></tr>
<tr><td>OPHR</td><td>‹listen›OPHR‹talk›HR‹number›</td><td>Output horizontal mode.</td></tr>
<tr><td>I</td><td></td><td></td></tr>
<tr><td>ID</td><td>‹listen›ID‹number›‹talk›ID‹C› . . .</td><td>Output identification. Number = 1 to 8.†
1 Display type number
2 Display software version number
3 Analyzer type number
4 Analyzer software version number
5 GPIB address
6 Options fitted.
7 Display store version number
8 Analyzer store version number</td></tr>
<tr><td>IP</td><td>‹listen›IP0|1</td><td>Intermodulation products identification (Intmd Ident). Off or on.</td></tr>
<tr><td>OPIP</td><td>‹listen›OPIP‹talk›IP0|1</td><td>Output state of Intmd Ident.</td></tr>
<tr><td>IR</td><td>‹listen›IR0|1</td><td>Identify refresh. Off or on.</td></tr>
<tr><td>OPIR</td><td>‹listen›OPIR‹talk›IR0|1</td><td>Output state of identify refresh.</td></tr>
<tr><td>K</td><td></td><td></td></tr>
<tr><td>KL</td><td>‹listen›KL[(‹number›DB|KV|VL|MV|UV|
NV|,|;‹number›HZ|KZ|MZ|GZ|,|;) . . .]</td><td>Enter lower mask limits. Omitting parameters clears previous entries.†</td></tr>
<tr><td>OPKL</td><td>‹listen›OPKL‹talk›KL‹number›‹cr›‹lf›
[(‹number›DB|VL,‹number›HZ‹cr›‹lf›) . . .]
‹eoi›</td><td>Output lower mask limits. First number is the number of lower mask limits.</td></tr>
</table>

†For detailed information on this command see 'Command functions'

Header	Syntax	Description and parameter limitations
K		
KM	‹listen›KM0|1	State of mask. Off or on.
OPKM	‹listen›OPKM‹talk›KM0|1	Output state of mask off or on.
KU	‹listen›KU[(‹number›DB|KV|VL|MV|UV|NV|,|;‹number›HZ|KZ|MZ|GZ|,|;) . . .]	Enter upper mask limits. Omitting parameters clears previous entries.†
OPKU	‹listen›OPKU‹talk›KU‹number›‹cr›‹lf› [(‹number›DB|VL,‹number›HZ‹cr›‹lf›) . . .] ‹eoi›	Output upper mask limits. First number is the number of upper mask limits.
L		
LD	‹listen›LD#J‹B› . . . ‹CS›‹eoi›	Learn display annotation and graticule, 197 byte data block (plus ‹CS›).†
OPLD	‹listen›OPLD‹talk›LD#J‹B› . . . ‹CS›‹eoi›	Output display annotation and graticule, 197 byte data block (plus ‹CS›).
LI	‹listen›LI#J‹B› . . . ‹CS›‹eoi›	Learn instrument settings, 237 byte data block (plus ‹CS›).†
OPLI	‹listen›OPLI‹talk›LI#J‹B› . . . ‹CS›‹eoi›	Output instrument settings, 237 byte data block (plus ‹CS›).
LT	‹listen›LT#B› . . . ‹eoi›	Learn trace, 1004 byte data block.
OPLT	‹listen›OPLT‹talk›LT#I‹B› . . . ‹eoi›	Output trace , 1004 byte data block.
M		
MC	‹listen›MC‹number›.	Frequency count 0 Off 1 On 2 1 Hz resolution
OPMC	‹listen›OPMC‹talk›MC‹number›	Output state of frequency count.
MF	‹listen›MF‹number›[HZ|KZ|MZ|GZ]	Position marker by frequency. Illegal on METER mode.
MI	‹listen›MI	Set increment frequency to marker 1 frequency.
MK	‹listen›MK0|1	Marker 1. Off or on.†
OPMK	‹listen›OPMK‹talk›MK0|1	Output state of marker 1.
ML	‹listen›ML	Set reference level to marker 1 level.
MP	‹listen›MP‹number›	Position marker 1 by ordinate. Number = 1 to 500 (or -2500 to 2500 on METER mode).
OPMP	‹listen›OPMP‹talk›MP‹number›	Output marker 1 ordinate.
MR	‹listen›MR	Set reference frequency to marker 1 frequency.
MS	‹listen›MSA|B	Marker AB select.
OPMS	‹listen›OPMS‹talk›MSA|B	Output state of marker AB select.
N		
NZ	‹listen›NZ0|1	Normalize. Off or on and take normalizing sweep. SRQ (3) at end of sweep.†
OPNZ	‹listen›OPNZ‹talk›NZ0|1	Output state of sweep, un-normalized or normalized.

†For detailed information on this command see 'Command functions'

<table>
<tr><th>Header</th><th>Syntax</th><th>Description and parameter limitations</th></tr>
<tr><td>O</td><td></td><td></td></tr>
<tr><td>OA</td><td>‹listen›OA‹talk›OA‹number›DBM|DBV|
DBMV|DBUV|DB|VL</td><td>Output marker 1 amplitude.</td></tr>
<tr><td>OB</td><td>‹listen›OB‹number›</td><td>Select output buffer. Number = 0 or 1.†
0 Main
1 Alternative.</td></tr>
<tr><td>OPOB</td><td>‹listen›OPOB‹talk›OB‹number›</td><td>Output state of output buffer.</td></tr>
<tr><td>OC</td><td>‹listen›OC‹talk›OC‹space›‹B› . . . ‹cr›‹lf›‹eoi›</td><td>Output calibration state (60 bytes).</td></tr>
<tr><td>OE</td><td>‹listen›OE‹talk›OE‹number›,‹CC›</td><td>Output error information, including error number and corresponding header.†</td></tr>
<tr><td>OF</td><td>‹listen›OF‹talk›OF‹number›HZ</td><td>Output marker 1 frequency.</td></tr>
<tr><td>OI</td><td>‹listen›OI‹talk›OI‹number›</td><td>Output instrument status word.†</td></tr>
<tr><td>OK</td><td>‹listen›OK‹talk›OK‹number›</td><td>Output number of last key pressed.†</td></tr>
<tr><td>OM</td><td>‹listen›OM‹talk›OM‹number›DBM|DBV|
DBMV|DBUV|DB|VL,‹number›HZ</td><td>Output marker 1 amplitude and marker 1 frequency.</td></tr>
<tr><td>OS</td><td>‹listen›OS‹number›‹talk›OS‹C› . . .</td><td>Output non-volatile store status.
Number = 1 to 9. 45 characters are sent.†</td></tr>
<tr><td>OT</td><td>‹listen›OT‹talk›(‹number›‹cr›‹lf›) . . . ‹eoi›</td><td>Output selected trace in ASCII.
502 numbers sent.†</td></tr>
<tr><td>P</td><td></td><td></td></tr>
<tr><td>PC</td><td>‹listen›PC‹number›</td><td>Clear non-volatile store. Store numbers= 1 to 9.</td></tr>
<tr><td>PE</td><td>‹listen›PE‹number›</td><td>Enable (unprotect) store. Store numbers= 1 to 9.</td></tr>
<tr><td>PF</td><td>‹listen›PF‹number›</td><td>Peak find. Peak number = 1 to 9.</td></tr>
<tr><td>PL</td><td>‹listen›PL‹number›[(,|;‹number›) . . .]‹talk›
‹C› . . .‹cr›‹lf›‹eoi›</td><td>Output command string for GPIB plotter.
Numbers identify the items to be included:-
(See Programming Example 21)
1 A display (2500)
2 B display (2500)
3 Major graticule (750)
4 Minor graticule (2500)
5 A display annotation (1000)
6 B display annotation (1000)
7 General annotation and markers (800)
Bracketed numbers give a high estimate of the typical number of characters sent.</td></tr>
<tr><td>PM</td><td>‹listen›PM‹number›(,1;‹number›) . . .</td><td>Store (push) limit mask into non-volatile store. Store numbers = 1 to 9.</td></tr>
<tr><td>PN</td><td>‹listen›PN‹number›(,1;‹number›) . . .</td><td>Set pen numbers. Five pen numbers are sent, allocating a pen number in the range 1 to 9 to the following items in the order they appear:–
A trace and annotation
B trace and annotation
Major graticule
Minor graticule
General annotation</td></tr>
</table>

†For detailed information on this command see 'Command functions'

<table>
<tr><th>Header</th><th>Syntax</th><th>Description and parameter limitations</th></tr>
<tr><td>OPPN</td><td>‹listen›OPPN‹talk›PN‹number›
(,‹number›) . . .</td><td>Output pen assignments for GPIB plot.
Five numbers are sent.</td></tr>
<tr><td>PO</td><td>‹listen›PO‹number›</td><td>Recall (pop) settings from non-volatile store. Store numbers = 1 to 9.</td></tr>
<tr><td>PP</td><td>‹listen›PP‹number›</td><td>Protect store. Store numbers = 1 to 9.</td></tr>
<tr><td>PR</td><td>‹listen›PR</td><td>Preset selected unit.</td></tr>
<tr><td>PS</td><td>‹listen›PS[‹number›]</td><td>Set pen speed for GPIB plot. 0 to 999.99 cm/s or default speed if number omitted.</td></tr>
<tr><td>OPPS</td><td>‹listen›OPPS‹talk›PS[‹number›]</td><td>Output pen speed set for GPIB plot.
Number is omitted if speed is unspecified.</td></tr>
<tr><td>PU</td><td>‹listen›PU‹number›</td><td>Store (push) settings into non-volatile store. Store numbers = 1 to 9.</td></tr>
<tr><td>R</td><td></td><td></td></tr>
<tr><td>RA</td><td>‹listen›RA0|1</td><td>Set auto RF attenuation. Off or on.</td></tr>
<tr><td>OPRA</td><td>‹listen›OPRA‹talk›RA0|1</td><td>Output state of auto RF attenuation.
Off or on.</td></tr>
<tr><td>RF</td><td>‹listen›RF‹number›[DB]</td><td>Set RF attenuation. 0 to 60 dB in 10 dB steps.</td></tr>
<tr><td>OPRF</td><td>‹listen›OPRF‹talk›RF‹number›DB</td><td>Output RF attenuation.</td></tr>
<tr><td>RL</td><td>‹listen›RL‹number›[DB]</td><td>Set reference level for selected display.</td></tr>
<tr><td>OPRL</td><td>‹listen›OPRL‹talk›RL‹number›DB</td><td>Output reference level for selected display.</td></tr>
<tr><td>S</td><td></td><td></td></tr>
<tr><td>SA</td><td>‹listen›SA0|1</td><td>Set auto sweep time. Off or on.</td></tr>
<tr><td>OPSA</td><td>‹listen›OPSA‹talk›SA0|1</td><td>Output state of auto sweep time. Off or on.</td></tr>
<tr><td>SD</td><td>‹listen›SD0|1</td><td>Set service request display. Off or on.</td></tr>
<tr><td>OPSD</td><td>‹listen›OPSD‹talk›SD0|1</td><td>Output state of service request display.
Off or on.</td></tr>
<tr><td>SE</td><td>‹listen›SE0|1</td><td>Set secret. Off or on.</td></tr>
<tr><td>OPSE</td><td>‹listen›OPSE‹talk›SE0|1</td><td>Output state of secret. Off or on.</td></tr>
<tr><td>SG</td><td>‹listen›SG‹number›[(,|;‹number›) . . .]</td><td>Switch groups enable and disable. Groups are numbered 1 to 17 is used to denote all groups. Positive numbers enable, negative numbers disable.†</td></tr>
<tr><td>OPSG</td><td>‹listen›OPSG‹talk›SG‹number›
(,‹number›) . . .</td><td>Output state of switch group enable.
16 numbers are sent.</td></tr>
<tr><td>SL</td><td>‹listen›SLA|B</td><td>Set AB select.</td></tr>
<tr><td>OPSL</td><td>‹listen›OPSL‹talk›SLA|B</td><td>Output state of AB select.</td></tr>
<tr><td>SM</td><td>‹listen›SM‹number›[(,|;‹number›) . . .]</td><td>Set service requests enable/disable.
Positive numbers enable negative numbers disable.†
1 error
2 instrument status word change
3 sweep completed
4 key pressed
5 end of calibration
7 data ready</td></tr>
</table>

†For detailed information on this command see 'Command functions'

Header	Syntax	Description and parameter limitations
OPSM	‹listen›OPSM‹talk›SM‹number›(, ‹number›) . . .	Output state of service request enable. 5 numbers are sent.
SP	‹listen›SP‹number›[HZ|KZ|MZ|GZ]	Set span 100 Hz to 200 MHz in 1,2,5 sequence and 400 MHz. (Enter total span not span/div.).
OPSP	‹listen›OPSP‹talk›SP‹number›HZ	Output span.
SR	‹listen›SR0|1	Switch remote use. Off or on.†
OPSR	‹listen›OPSR‹talk›SR0|1	Output state of switch remote use. or on.
ST	‹listen›ST‹number›[US|MS|SC]	Set sweep time. 100 ms to 200 s in 1,2,5 sequence. 50 μs to 200 s on ZERO SPAN. (Enter total sweep time not sweep time/div.).
OPST	‹listen›OPST‹talk›ST‹number›SC	Output sweep time.
SU	‹listen›SU‹number›	Select lower unit. Unit number = 0 for single lower unit.
OPSU	‹listen›OPSU‹talk›SU‹number›	Output selected lower unit number.
SW	‹listen›SW‹number›	Set sweep mode. Number = 0 or 1. 0 Single arm 1 Normal.
OPSW	‹listen›OPSW‹talk›SW0|1	Output sweep mode.
T		
TA	‹listen›TA0|1	Set auto trigger. Off or on.
OPTA	‹listen›OPTA‹talk›TA0|1	Output state of auto trigger. Off or on.
TE	‹listen›TE‹number›	Set trigger edge. Number = 0 or 1. 0 Positive 1 Negative
OPTE	‹listen›OPTE‹talk›TE‹number›	Output state of trigger edge.
TG	‹listen›TG0|1	Set tracking generator. Off or on.
OPTG	‹listen›OPTG‹talk›TG0|1	Output state of tracking generator. Off or on.
TI	‹listen›TI[‹number›]	Take sweeps with initial first sweep. Default to one sweep. SRQ (3) at end of sweep.†
TL	‹listen›TL‹number›[DB]	Set tracking generator level (dBm). Range −9.7 dBm to −20.3 dBm (0·1 dB steps).
OPTL	‹listen›OPTL‹talk›TL‹number›DB	Output level of tracking generator.
TR	‹listen›TR‹number›	Set trigger source. Number = 0 to 3. 0 None 1 Video 2 Line 3 Ext
OPTR	‹listen›OPTR‹talk›TR‹number›	Output trigger source.
TS	‹listen›TS[‹number›]	Take sweeps. Default to one sweep SRQ (3) at end of sweep.†

†For detailed information on this command see 'Command functions'

<table>
<tr><th>Header</th><th>Syntax</th><th>Description and parameter limitations</th></tr>
<tr><td>TX</td><td>‹listen›TX‹string›</td><td>Enter text to selected display. Up to 38 characters.</td></tr>
<tr><td>OPTX</td><td>‹listen›OPTX‹talk›TX‹string›</td><td>Output text of selected display.</td></tr>
<tr><td>U</td><td></td><td></td></tr>
<tr><td>UL</td><td>‹listen›UL0|1</td><td>Unlock reference levels. Off or on.</td></tr>
<tr><td>OPUL</td><td>‹listen›OPUL‹talk›UL0|1</td><td>Output state of unlock reference levels. Off or on.</td></tr>
<tr><td>V</td><td></td><td></td></tr>
<tr><td>VA</td><td>‹listen›VA0|1</td><td>Auto video bandwidth. Off or on.</td></tr>
<tr><td>OPVA</td><td>‹listen›OPVA‹talk›VA0|1</td><td>Output state of auto video bandwidth. Off or on.</td></tr>
<tr><td>VB</td><td>‹listen›VB‹number›[HZ|KZ|MZ|GZ]</td><td>Set video bandwidth 1 Hz,2 Hz,5 Hz,11 Hz, 22 Hz,43 Hz, 87 Hz, 170 Hz, 350 Hz, 700 Hz, 1·4 kHz, 2·8 kHz, 5·4 kHz, 11 kHz, 25 kHz, 50 kHz</td></tr>
<tr><td>OPVB</td><td>‹listen›OPVB‹talk›VB‹number›HZ</td><td>Output video bandwidth.</td></tr>
<tr><td>VG</td><td>‹listen›VG‹number›[NV|UV|MV|VL]</td><td>Set log volts reference level of selected display. Decade value 100 nV to 10 V.</td></tr>
<tr><td>OPVG</td><td>‹listen›OPVG‹talk›VG‹number›VL</td><td>Output log volts reference level of selected display.</td></tr>
<tr><td>VS</td><td>‹listen›VS‹number›[DB]</td><td>Set dB/div for vertical scale of selected display.
Number = 0·5 , 1,2,5 or 10 dB/div.</td></tr>
<tr><td>OPVS</td><td>‹listen›OPVS‹talk›VS‹number›DB</td><td>Output value of vertical dB/div for selected display.</td></tr>
<tr><td>VU</td><td>‹listen›VU‹number›</td><td>Set vertical scale units for selected display. Number = 0 to 6.
0 dBm
1 dBV
2 dBmV
3 dBμV
4 dB
5 log volts
6 linear volts</td></tr>
<tr><td>OPVU</td><td>‹listen›OPVU‹talk›VU‹number›</td><td>Output vertical scale units for selected display.</td></tr>
<tr><td>VV</td><td>‹listen›VV‹number›[NV|UV|MV|VL]</td><td>Set top of scale linear volts for selected display. (Enter total volts range not volts/div.).
1 μV to 5 V in 1,2,5 sequence.</td></tr>
<tr><td>OPVV</td><td>‹listen›OPVV‹talk›VV‹number›VL</td><td>Output top of scale linear volts for selected display.</td></tr>
<tr><td>W</td><td></td><td></td></tr>
<tr><td>WA</td><td>‹listen›WA0|1</td><td>Set auto filter bandwidth. Off or on.</td></tr>
<tr><td>OPWA</td><td>‹listen›OPWA‹talk›WA0|1</td><td>Output state of auto filter bandwidth. Off or on.</td></tr>
</table>

<table>
<tr><th>Header</th><th>Syntax</th><th>Description and parameter limitations</th></tr>
<tr><td>WF</td><td>‹listen›WF‹number›[HZ|KZ|MZ|GZ]</td><td>Set filter bandwidth. 3 Hz to 1 MHz in 1,3,10 sequence.</td></tr>
<tr><td>OPWF</td><td>‹listen›OPWF‹talk›WF‹number›Hz</td><td>Output filter bandwidth.</td></tr>
<tr><td>WN</td><td>‹listen›WN0|1</td><td>Set noise in 1 Hz bandwidth. Off or on.</td></tr>
<tr><td>OPWN</td><td>‹listen›OPWN‹talk›WN0|1</td><td>Output state of noise in 1 Hz bandwidth. Off or on.</td></tr>
</table>

DS Display select

The displays referred to are the character displays. Each character display is composed of 24 lines of characters with 65 characters per line. The Main Display overlays the A trace, B trace and graticule and is used by the analyzer to display alphanumeric information, e.g. horizontal and vertical scaling. You may send text to and read text off the Main Display. However, such text may be overwritten by the analyzer.

The Alternative Display is provided solely for the GPIB operator to display text. When the Alternative Display is selected the A trace, B trace and graticule displays are inhibited. Text sent to the Alternative Display will not be interfered with by the analyzer in any way.

Both displays exist independently and simultaneously and the operator may switch from one to the other at any time using the DS command.

The GPIB commands for manipulating screen text (DC DT OPDT) always apply to the currently selected trace. If the analyzer changes from remote to local operation while the Alternative Display is selected then the Main Display is automatically selected.

DT Display text

The display text command enables you to display alpha-numeric text and symbols on the currently selected display. The alphanumeric displays on the Analyzer are usually thought of as being composed of 24 lines of characters with 65 characters per line, making a total of 1560 characters per screen. However, when considering displaying text, the screen should be considered as a linear array of 1560 cells. Each cell may contain a byte value (a number in the range 0 to 255), the assignment of numbers to characters is detailed below. Codes 32 to 93 and 97 to 122 follow the normal ASCII assignment. The remaining codes have been redefined to satisfy the individual requirements of the analyzer display.

Code	Character
0	Null. Null produces a blank on the alphanumeric display but allows the trace and graticule displays to appear as normal, i.e. no 'hole' is cut in these displays.
1	Transparent character. This is a special character for the GPIB user. Displaying a transparent character at a position on the screen results in the character currently at that position remaining unchanged. Its purpose is for producing screens of text which may be overlayed on existing text. Where a transparent character is present the old text will 'show through'.
2 – 13	Graticule characters. These characters are used internally for producing the graticule and serve no useful purpose for the GPIB user.
14	Down arrow character.
15	Block character.

Code	Character
16 – 26	Graticule characters. These characters are used internally for producing the graticule and serve no useful purpose for the GPIB user.
27	Corner block character.
28	Horizontal line character.
29	Vertical line character.
30	Exponential 'n' character as used in 2^n.
31	Graticule character. These characters are used internally for producing the graticule and serve no useful purpose for the GPIB user.
32 – 93	ASCII equivalent. Note that printing characters cuts a 'hole' in the graticule and trace displays. A space is considered to be a printing character and as such will cut a 'hole'. The null character however will not cut a 'hole'.
94	Up arrow character.
95	Omega character.
96	Micro character.
97 – 122	ASCII equivalent.
123	Outline character.
124	Delta character.
125	'A' arrow character (›).
126	'B' arrow character (‹).
127	Infill character.
128 – 255	Field attribute codes.

Field attribute codes appear on the display in the same way as the null character. They affect the attributes with which any following printing characters will be displayed. Field attribute codes are bit encoded as follows.

Bit. No.	7	6	5	4	3	2	1	0
Assignment	Always one	Always zero	Always zero	Reverse video	Colour bit 1	Colour bit 0	Blink	Always zero

Colour bits 0 and 1 only have meaning when a colour monitor display is being used. They are further encoded as follows:

Colour bit 1	Colour bit 0	Displayed colour
0	0	Green
0	1	Red
1	0	Blue
1	1	White

Two charts, Figs. 2–1 and 2–2, show the screen arrangment of the 2380 Main Display. One shows the arrangement for linear sweeps and the other the arrangement for log. sweeps. The crossed boxes indicate the location of attribute characters, although fields may contain additional attribute characters. The central field marked 'MESSAGES, PROMPTS AND MENUS' may be used by the GPIB operator for displaying text. This will not be over-written by the analyzer providing that the function activated is not one for which the analyzer uses this field. A printing character placed within the two columns defined by dotted lines to the left and right of this field will cut the left and right bounds of the graticule. Non-printing characters, attribute characters for instance, can be placed in these columns leaving the vertical graticule lines intact.

The display text commanduses a block data transfer format to allow sections of the array to be filled quickly. A complete screen of 1560 characters takes about 0.7 s to transfer if speed is limited only by the Analyzer and not the GPIB Controller but the data block can be as short as one character. Characters received in excess of 1560 are ignored. The starting point of the required array is specified in terms of the line and column position of the first character in the block to allow its positioning to be related to the actual form of the display.

ID IDENTIFY

The ID command is used to identify features of the Spectrum Analyzer. These are listed below and are numbered. This number is included in the ID command to output the particular feature required.

Number	Feature	Example reply from Analyzer
1	Part number of the Display unit	ID 52380-900E
2	Software version of the Display unit	ID 001A
3	Part number of the Analyzer unit	ID 52382-900A
4	Software version of the Analyzer unit	ID 001A
5	GPIB address	ID 28
6	Options fitted	ID 240
7	Display store version number	ID 0001
8	Analyzer store version number	ID 0001

The reply to a request to identify 'Options fitted' is a binary encoded number, with a zero indicating that an option is fitted. Bits have the following assignments:

Bit. No.	7	6	5	4	3	2	1	0
Assignment	Not used	Not used	Not used	Not used	Pen plot fitted	Colour palette fitted	Beeper fitted	Options board fitted

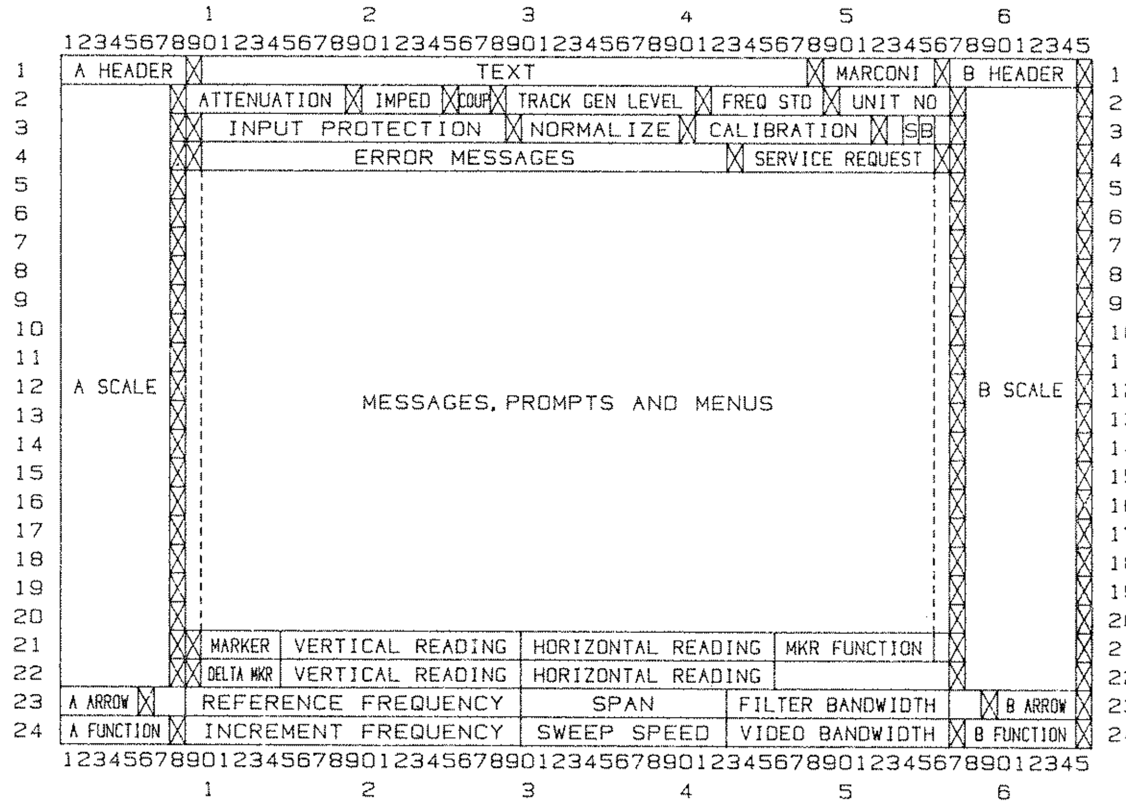


Fig. 2–1. 2380 screen arrangement – linear sweeps

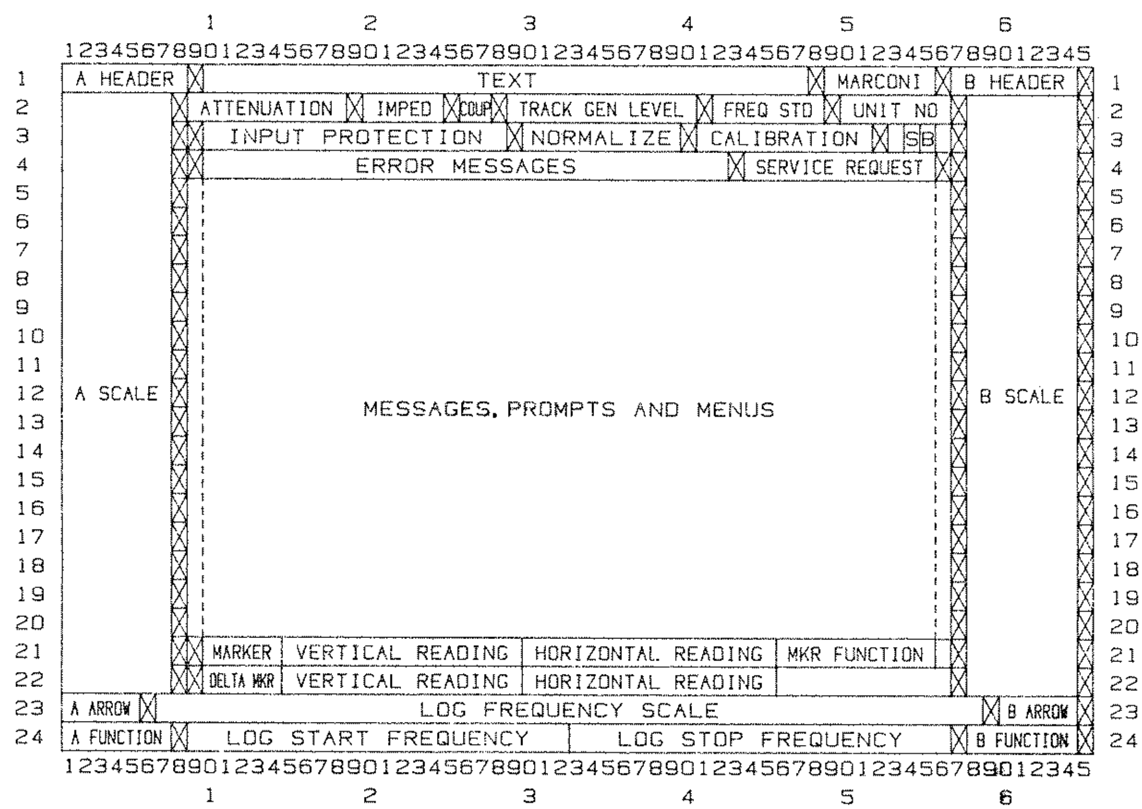


Fig. 2–2. 2380 screen arrangement – log. sweeps

KM, KU, KL Mask

The state of the mask mode – off or on – may be established using the KM 0/1 command. Mask limits may not be entered unless MASK is ON.

The KU and KL commands are used to enter up to eight amplitude, frequency coordinate pairs for the maximum and minimum boundaries of the mask respectively. The vertical units of a mask are established as the vertical units of the trace when the first mask point is entered and remain until all mask points are cleared.

A set of maximum and minimum mask points is cleared by sending the commands KU or KL respectively, omitting the coordinate pairs.

MK Markers

Only marker 1 can be controlled remotely. It is recommended that the marker is switched on prior to use (command K1) because the reaction of the instrument to a command directed at the marker while it is off varies from one command to another. The instrument may reject the command as illegal, accept and execute the command leaving the marker off or accept and turn the marker on automatically prior to executing the command.

All the commands used to control the marker have headers beginning with the letter M except for three dedicated commands used to read the marker value OA, OF and OM. OA is used to read the marker amplitude, OF is used to read the marker frequency while OM allows the amplitude and frequency to be read together. OM sends the amplitude first followed by the frequency – the same order as they are displayed on the screen. Note that the amplitude is returned in dBm if the instrument is scaled in log volts vertically.

Selecting METER mode changes the commands MF and MP used for positioning the markers. MF becomes illegal, and MP takes an argument in the range −2500 to 2500. On METER mode the 2382 sets and counts the currently tuned frequency more accurately and this is reflected in the increased resolution of the marker position.

NZ Normalize

The test fixture must be connected either automatically or via manual intervention using the text display (DT) command to display this instruction on the Analyzer screen. Pressing a units key on the Analyzer is used to signal 'fixture connected' refer to 'Command functions – Switch remote use' (SR). Pressing the units key sets bit 4 of the status byte and initiates a service request. Command NZ1 can now be sent.

Issuing the normalize ON command (NZ1) is similar to sending a 'Take Sweeps' (TS) command, except that the sweep data is loaded into the error store. NZ0 cancels the normalize function.

OB Select output buffer

The instrument buffers data that has been requested over the GPIB before it sends it out. Only the response to the last request for data is retained however and if a further request is received before the response to a previous request has been output then the new response overwrites the old one. This is usually desirable since it guarantees that the data received back always corresponds to the last request, but there is a situation where this behaviour is undesirable.

If the GPIB Controller has requested data from the Analyzer and the Analyzer then issues a service request, the GPIB Controller may service the Analyzer before reading back the data. If in the course of servicing the Analyzer some new data is buffered, then the original data will be lost.

To avoid this situation the Analyzer is provided with two output buffers. They are termed 'main' and 'alternative'. Data existing in one buffer is not lost by data being written into the other. Usually the main buffer is used, but the GPIB operator can instruct the Analyzer to switch between main and alternative buffers using the Output Buffer (OB) command. The GPIB operator can now avoid the occurrence of the situation outlined above by switching to the alternative buffer when reading data within an interrupt routine. Service requests are not issued for data in the alternative buffer.

There are some instances where data output does not go through the output buffer. These exceptions are when reading :

- the instrument settings store
- the display annotation stores
- the trace stores
- GPIB plot
- limit mask points
- text off the screen

A request for these items is nullified by a request for further data of any kind, irrespective of which output buffer is in use.

OE

The OE command enables the GPIB operator to read back information about the last operator error that occurred at the analyzer. The information comprises the error number and the header of the last GPIB command to be processed before the error occurred. The latter information may be helpful in pin-pointing the cause of the error within a GPIB program.

When an error occurs the above information is stored and the input to the store is then blocked so that the information is not overwritten if a further error occurs. In this way, where there is a string of errors, the information relating to the first error in the string is returned. The input to the store may be re-enabled by:

1) Reading the store by means of the OE command;
2) Unmasking service request on error by means of the SM command or
3) Executing a device clear.

When the input to the store is enabled the data is reurned to the 'no error' condition, which is, error number = 0, error header = ‹NUL›‹NUL› (‹NUL› is the ASCII character with code 0). This provides a means of checking for the occurrence of an error within a specific segment of a program. A 'flush read', that is a read of the store in order to clear it, is done at the start of the particular segment of program. The error store is examined on completion of the segment of program and if the store is not in the 'no error' state, then the error occurred within the program segment.

An interpretation of the error numbers it is possible to obtain with the OE command is given in Table 2–3.

TABLE 2–3 ERROR NUMBER INTERPRETATION

Error number	Description
4	Invalid operation
5	Options not installed
7	Invalid STORE operation
8	Illegal DATA entry
9	DATA entry out of range
11	Invalid sweep condition
12	Out of range
15	Noise in 1 Hz selected while on ZERO SPAN
16	No response from lower unit
18	GPIB: Unidentified block data
20	GPIB: Max line length exceeded
21	GPIB: Unrecognized header
22	GPIB: Unavailable item
23	GPIB: Invalid data field
25	GPIB: Invalid checksum
26	GPIB: Incompatible binary data
27	GPIB: Too many mask limits
28	No tracking generator fitted
30	Lost Track
31	Write protected store NOT cleared
40	Failed to set Osc 0
41	Failed to set Osc 1
42	Failed to set Osc 2
43	Sweep Gain D to A adjust range exceeded
44	Osc 0 not phase locked
45	Osc 1 not phase locked
46	Sweep D to A error on Osc cal
47	Sweep D to A error on Osc cal
49	Osc 1 and 2 out of range
51	Log amp sweep abort failure
52	Log amp sweep abort failure
53	Linearization failure

OI Output instrument status

The Output Instrument Status (OI) command causes the Analyzer to output a status word detailing the state of certain aspects of the instrument as follows:

Bit No.	15	14	13	12	11	10	9	8
	–	–	–	–	–	–	–	Not used
Bit No.	7	6	5	4	3	2	1	0
	Freq std out of lock	Ext freq std	Int freq std uncal	Over-heat	Over-load	Sweep uncal	Valid cal	Calibrated

To decode the number returned by the Analyzer, examine its binary representation. If the corresponding bit in the binary representation is a 1 then the condition applies. If the bit is a 0, the opposite applies. A service request is issued upon the change in state of any of the bits in the status word providing bit 2 of the status byte has been enabled using the SM command.

OPLD, LD Output and Learn display settings

These commands enable the GPIB operator to read out and send back the contents of two of the internal stores – the display annotation stores. There is a display annotation store for the A display and one for the B display, the particular store read from or written to is set by the A B SELECT (SL) command. The bytes sent and received using these commands are literal representations of the 195 byte stores in the memory of the analyzer. These stores contain the data from which the Display unit produces the graticule and annotations for the displays. It should be noted that the stores do not contain the literal annotations, only the data from which the annotations are derived. While a display is live, its display store is updated as changes occur. If the display is saved then the updating is inhibited. However, since the vertical scaling and caption of a saved display may be changed, these variables continue to be updated.

The ability of the GPIB operator to be able to read from and write to the display stores enables the graticule and annotations of a display to be stored in the GPIB Controller and regenerated on the screen of the Analyzer at a later date. When this facility is combined with the facility for reading and writing to the trace stores, using the OPLT and LT commands, complete displays may be saved in the GPIB Controller and regenerated on the screen of the Analyzer at a later date. Whenever data is written to one of the display annotation stores the corresponding display is automatically saved to stop data being overwritten.

Two checks are made on the data returned to the analyzer:

1) The first check ensures that the store version number of the Display Unit that provided the data is the same as the one currently being used to receive it back. The store version number of the source Display Unit is included in the data. It should be noted that the store version number might change if the software version in the Display unit changes.

   The store version number of the current Display Unit is found by pressing 2ND FUNCT RECALL. It is displayed under the heading 'Display store version' together with the current software version numbers.

   Alternatively it can be read over the GPIB using the Identity (ID) command.

2) The second check ensures that the modulo 256 checksum byte is correct.

The returned data is rejected if it fails either of these checks. In the case of failing the first check, the error 'Incompatible binary data' is issued. In the case of data failing the second check, 'Invalid checksum' is issued.

The construction of the data sent out is given below. The display store version number is represented in packed binary coded decimal.

Byte	ASCII	BCD	Description
1	L		Header
2	D		Header
3	#		Unique code indicating block data
4	J		Block data check type letter
5		34	Least significant byte of display store version number (example for store version number 1234)
6		12	Most significant byte of display store version number (example for store version number 1234)
7-201			Data bytes
202			Checksum byte with EOI

OPLI, LI Output and Learn instrument settings

These commands enable you to read out and send back the contents of one of the internal stores – the instrument settings store. This is the store which holds the current live setting of the instrument. The bytes sent and received using these commands are a literal representation of the 233 byte store in the memory of the analyzer. The act of reading the store out to the GPIB Controller using the OPLI command is equivalent to storing the current instrument settings using the STORE function except that the data is held within the GPIB Controller rather than in non-volatile memory inside the analyzer. Correspondingly the act of sending the store content back to the Analyzer using the LI command is equivalent to using the RECALL function; the returned instrument settings are restored.

Three checks are made on data returned to the Analyzer:

1) The first check ensures that the type number of the Analyzer that provided the data is the same as the one currently being used to receive it back. The type number of the source Analyzer is included in the data.

2) The second check ensures that the store version number of the Analyzer that provided the data is the same as the one currently being used to receive it back. The store version number of the source Analyzer is included in the data. It should be noted that the store version number might change if the software version in the Analyzer changes.

   The store version number of the current Analyzer unit is found by pressing 2ND FUNCT RECALL. It is displayed under the heading 'Analyzer store version' together with the current software version numbers.

   Alternatively it can be read over the GPIB using the Identify (ID) command.

3) The third check ensures that the modulo 256 checksum byte is correct.

The returned data is rejected if it fails any of these checks. In the case of data failing the first or second check the error 'Incompatible binary data' is issued. In the case of data failing the third check, 'Invalid checksum' is issued.

The construction of the data sent out is given below. The Analyzer type number and the Analyzer store version number are represented in packed binary coded decimal.

Byte	ASCII	BCD	Description
1	L		Header
2	I		Header
3	#		Unique code indicating block data
4	J		Block data check type letter
5		82	Least significant byte of Analyzer type number (example for Analyzer type 2382)
6		23	Most significant byte of analyzer type number (example for Analyzer type 2382)
7		34	Least significant byte of Analyzer store version number (example for store version number 1234)
8		12	Most significant byte of analyzer store version number (example for store version number 1234)
	9-241		Data bytes
242			Checksum byte with EOI

OPLT, LT Output and learn trace settings

These commands enable you to read out and send back the contents of the two trace stores. There is a trace store for the A display and one for the B display; the particular store read from or written to is set by the A B SELECT (SL) command. These stores contain the data from which the Display Unit produces the traces for the displays. While a display is live, its trace store is updated from the input store as changes occur. When the display is saved then the updating is inhibited. For this reason a display must be saved before writing data to its trace store otherwise the data will be immediately overwritten.

The ability to be able to read from and write to the trace stores enables traces to be analyzed and/or stored in the GPIB Controller and re-displayed on the Analyzer screen at a later date. However, it is important to note that some extra information contained in the corresponding display annotation store is required to display a trace correctly on the screen. The display annotation stores can be read from and written to using the OPLD and LD commands.

The data block contains 1004 bytes. These are interpreted as 502 16-bit 2's complement words. The first byte of each pair is the most significant byte. If the vertical scaling of the Analyzer is in one of the dB modes (dBm,dBV,dBmV,dBμV,dB) then the values output are 100 times their actual value. This preserves accuracy to two decimal places.

In log volts mode the values output are 100 times the equivalent dB values. The following relationship may be used to convert from dBm to volts:

$$20 \log V = P - 30 + 10 \log Z$$

where V = RMS voltage
P = power (dBm)
Z = input impedance of analyzer

In linear volts mode, the values output are 0-1000 on a '1' range, 0-2000 on a '2' range or 0-5000 on a '5' range representing a reading between 0 and top of screen in each case. The actual value of the reading is only known if the top of screen value is also known.

The construction of the data sent out is given below.

Byte	ASCII	BCD	Description
1	L		Header
2	T		Header
3	#		Unique code indicating block data
4	I		Block data check type letter
5-1008			Data bytes, EOI with last byte.

If the controller being used by the GPIB operator is unable to decode and enter this form of data directly into a numeric array, but instead must input the data into a string, then the following form of process must be used to convert from the string to the numeric array. The process assumes a string of length 100B and a numeric array of length 502, indices starting from 1.

BEGIN process – convert String to Numeric array.

String Index = 5.

Numeric Array Index = 1.

WHILE Numeric Array Index $<$ 503.

Numeric Array (Numeric Array Index) = 256 x value of the ASCII code of String (String Index).

String Index = String Index + 1.

Numeric Array (Numeric Array Index) = Numeric Array (Numeric Array Index) + value of the ASCII code of String (String Index).

String index = String Index + 1.

IF Numeric Array (Numeric Array Index) $\geqslant 2^{15}$ then

Numeric Array (Numeric Array Index) = Numeric Array (Numeric Array Index) – 2^{16}.

END IF.

Numeric Array Index = Numeric Array Index + 1.

END WHILE.

End process.

OS Output store status

The status of each of the nine non-volatile stores holding instrument settings or limit masks may be individually interrogated. The following list gives examples of the string returned.

An asterisk indicates a protected store. The reference frequency is returned whenever there is no caption on the display but the output string is padded to the right with spaces to a length of 53 characters to allow for a possible caption.

```
                    Format of store status text

 0         1         2         3         4
 123456789012345678901234567890123456789012345 6

************************************************
* OS   1) [Empty]                              *
* OS   2) [Ref freq = 200MHz]                  *
* OS   3) [Start freq 10kHz   Stop freq 100MHz]  *
* OS   4) 12345678901234567890123456789012345678  *
* OS  *5) CAPTION                              *
* OS M 6) [Left freq 40MHz   Right freq 320MHz ]  *
* OS M*7) [Left freq 40MHz   Right freq 320MHz ]  *
* OS M 8) MASK CAPTION                         *
* OS M 9) [Left freq         Right freq        ]  *
************************************************

                         Summary

 Characters 1 to 3      Header
 Character  4           Limit mask indicator
 Character  5           Write protect indicator
 Character  6 to 8      Store number
 Character  9 to 46     Caption of 38 characters
```

OT Output trace

This command enables you to read out the contents of the trace stores, but in this case data is output from the Analyzer as 502 numbers in ASCII.

The data is in the form of 502 ordinates of amplitude from left to right across the screen of the Analyzer, in either dB or volts depending upon the current vertical mode of the A or B trace. Unit codes are omitted and each reading is terminated by ‹cr› ‹lf›.

SG Switch group enable/disable

The sixteen groups of front panel keys listed below can be independently enabled and disabled. Display Unit key groups are numbered 1 to 8, lower unit key groups are numbered 9 to 16. The command header SG is followed by switch group numbers in the range 1 to ±17 interspersed with separators. A positive number enables that key group and a negative number disables it. The number 17 is used to enable or disable all groups.

The PRESET key is enabled when all other lower unit keys and controls are enabled. As SG17 enables upper and lower unit key groups, the PRESET key is also enabled.

After a change from LOCAL to REMOTE all controls are disabled except for the LOCAL key. This key is disabled by the controller sending the local lock-out command. On changing back from REMOTE to LOCAL, all controls are enabled.

When a lower unit is selected all its controls are enabled if the instrument is in LOCAL and disabled if in REMOTE.

Note that when using the SG command, although the Spectrum Analyzer is under remote operation, it can still be controlled manually. This will only cause a problem in the area of sweep aborts. In the REMOTE state the instrument performs a sweep abort if certain of its operational parameters are altered. It requires a TS or TI command to be sent or alternatively the Sweep mode START button pressed on the instrument to start it sweeping again. You must therefore ensure that the START button is enabled otherwise a manual operator will be unable to restart the sweep.

TABLE 2–4 SWITCH GROUPS

Group	Front Panel Controls
1	Data pad keys 0 to 7
2	Data pad keys 8,9, decimal, –, DELETE
3	Terminator keys
4	2ND FUNCT
5	VIEW, SAVE, MAX HOLD keys – both displays
6	GRAT, OPTNS, A–B→A, A↔B, NORMALIZE, CAL, TEXT
7	VIDEO AVG A and B, REF LEVEL, REF FREQ, SPAN/DIV, INC FREQ, STORE, RECALL
8	SIG TRACK, HORIZ POSN, REF FREQ ↓ and ↑, REF FREQ rotary control
9	VIDEO BANDWIDTH ↓, AUTO and ↑
10	A B SELECT, dB/DIVISION keys, VOLTS/DIV
11	Ref level ↓ and ↑, INTMD IDENT, RF ATTEN ↓, AUTO and ↑, REF LEVEL rotary control, TRACK GEN
12	MKR 1, MKR 2, 1,2 MOVE, MOVE rotary control MKR 1 SETS keys, ΔF SETS key
13	HORIZONTAL mode keys, SPAN/DIV rotary control
14	SWEEP TIME ↓, AUTO and ↑
15	SWEEP MODE keys, TRIGGER SOURCE keys, AUTO TRIG
16	FILTER BANDWIDTH ↓, AUTO and ↑

SM Service request

The bits comprising the status byte are described below:

Status byte Bit No.	Description
0	Not used
1	Error
2	Change in instrument status word
3	Sweep completed
4	Key pressed
5	End of calibration
6	Service request (SRQ)
7	Data ready

Providing it has been enabled using the Service Request (SM) command, an occurrence of any of the above events causes the corresponding bit to be set in the status byte and also sets bit 6. When bit 6 is set a service request is generated. The service request is removed when the instrument is polled (status byte is read). Further information on the meaning of the bits is provided in the following paragraphs.

Error. Refer to the Output Error (OE) command for the method of reading back error data and for a list of errors that will cause this service request.

Change in instrument status word. Refer to the Output Instrument Status Word command (OI) for a description of the status word.

Sweep completed. Only sweeps that were initiated by the Take Sweeps (TS) command, Take Sweeps with initial sweep command (TI) or Normalize (NZ) command will cause a service request. Where the TS or TI commands require more than one sweep only the end of the last sweep will cause a service request.

Key pressed. This only applies to keys on the display unit. Refer to the Output Key (OK) command for the method of reading back a key code and for a list of key codes.

End of calibration. This service request is raised when the automatic self-calibration sequence is completed. Self calibration is started over the GPIB using the Calibrate (CA) command.

Data ready. This service request is raised after the analyzer has been requested for data and the data is ready for transmission. Note that no service request is issued for data in the alternative output buffer.

You can enable and disable service request foeach of the events described independently using the Service Request (SM) command. At power on and after a device clear all events are disabled. The SM command takes a list of status byte bit numbers. If the sense of the bit number is positive then the corresponding event is enabled, if negative then it is disabled. For example the command SM 1, −7 enables service request for 'Error' and disables it for 'Data Ready'.

You can obtain a visual indication on the screen of the analyzer that a service request has been issued. This facility is enabled and disabled using the Service Request Display command (SD); it is disabled at power on and after a device clear. The form of the message displayed is SRQ XXXXXXXX where XXXXXXXX represents the status byte.

SR Switch remote use and key press detection

All keys on the Spectrum Analyzer Display Unit can be disabled using the SR command such that when a key is pressed there is no local effect but bits 4 and 6 of the status byte are set and SRQ is enabled.

The Controller can obtain the internal code for the last key pressed on the Display Unit by using the Output Key command (OK). A zero return indi- cates that no key has been pressed. Relevant keys and their codes are listed below.

Key	Code	Key	Code
0	128	SAVE B	197
1	129	MAX HOLD B	198
2	130	GRAT	208
3	131	OPTNS	209
4	132	A−B→A	210
5	133	A↔B	211
6	134	LOCAL	212
7	135	TEXT	213
8	144	CAL	214
9	145	NORMALIZE	215
.	146	REF FREQ	224
DELETE	147	SPAN/DIV	225
−	148	STORE	226
Hz/μV	160	RECALL	227
kHz/mV	161	VIDEO AVG A	228
MHz/V	162	VIDEO AVG B	229
GHz/dB	163	REF LEVEL	230
2ND FUNCT	176	INC FREQ	231
VIEWA	192	REF FREQ ↓	240
SAVE A	193	REF FREQ ↑	241
MAX HOLD A	194	HORIZ POSN	242
VIEW B	196	SIG TRACK	243

TS, TI Take sweeps

Commands TS [x] – take 'x' sweeps and TI [x] – take 'x' sweeps with initialization, are used to ensure that data has been fully refreshed before being read by the GPIB Controller. This must be done even in meter mode. When TS is sent the reading of data over the GPIB is immediately disabled and an 'S' appears in the top right hand corner of the graticule. Data is acquired by the analyzer when trigger conditions are met and a new sweep is started. The 'S' is deleted from the display to indicate that data has been acquired after x sweeps (if x is given) or after one sweep. Providing it has been enabled by the SM command (page 2–27) bit 3 of the status byte (sweep completed) is set at the end of the last sweep. The GPIB Controller is now able to read the fully refreshed data.

TI functions in a similar way, except that when VIDEO AVG or MAX HOLD is selected one unprocessed sweep is taken before new data is acquired.

2382/2380 GPIB COMMAND MNEMONICS

DISPLAY

A	
View	AV
Save	AS
Max Hold	AH
Video Avg	AA
Infill	AL
B	
View	BV
Save	BS
Max Hold	BH
Video Avg	BA
Infill	BI
Mask	KM KL KU
MISCELLANEOUS	
Grat	GR GM
Annot	AN
GPIB plot	PL PN
Optns	BE BO CL
A — B → A	AB
A — B	DX
Normalize	NZ
Cal	OC CR CA
Text	TX
Secret	SE
Select & Preset	SU PR
Track Gen	TG

VERTICAL

SCALE	
AB Select	SL
10	VS
5	VS
2	VS
1	VS
0.5	VS
Unlock Ref Levels	UL
dBm	VU0
dBv	VU1
dBmV	VU2
dBμV	VU3
dB	VU4
Log Volts	VU5
Volts/Div	VU6
↓	RL/VG/VV
↑	RL/VG/VV
Intmd Ident	IP
RF ATTEN	
↓	RF
Auto	RA
↑	RF

HORIZONTAL

SCALE	
Full Span	HRO
/Div	HR1
Log	HR2/HL
Zero Span	HR3
Meter	HR4
FM Demod	HR5
Audio	AU
SWEEP MODE	
Norm	SW1
Single Arm	SW0
Identity refresh	IR
TRIGGER SOURCE	
Video	TR1
Line	TR2
Ext	TR3
+/− trigger	TE 0/1
Auto Trig	TA
VIDEO BANDWIDTH	
↓	VB
Auto	VA
↑	VB
SWEEP TIME	
↓	ST
Auto	SA
↑	ST
RESOLUTION BANDWIDTH	
↓	WF
Auto	WA
↑	WF

FUNCTION/DATA

Ref Level	RL
Ref Freq	FR
Span	SP
Inc Freq	FI-FA
Set TG	TL
Store	PU/PP/PE/PC
Recall	PO
REF FREQ	
Sig Track	FT
Horiz Posn	HP
↓	FD
↑	FU

MARKERS

AB Select	MS
MKR 1	MK
Freq Count	MC1
Peak Find	PF
Res 1 Hz	MC2
Next Peak	PF
Move	MF/MP
Noise 1 Hz	WN
MKR1 Sets	
Ref Level	ML
Ref Freq	MR
Inc Freq	MI

Display clear	DC
Display select	DS
Display text	DT
Output identification	ID
Learn display settings	LD
Learn instrument settings	LI
Learn trace settings	LT
Output marker amplitude	OA
Select output buffer	OB
Output instrument status	OI
Output error number and header	OE
Output marker frequency	OF
Last key pressed	OK

Output marker amplitude & freq.	OM
Output setting	OP
Output store status	OS
Output trace (ASCII)	OT
Output command string for plotter	PL
Set pen numbers	PN
Set pen speed	PS
SRQ display	SD
Switch group enable/disable	SG
Service request mode	SM
Switch remote enable	SR
Take sweeps with initialization	TI
Take sweeps	TS

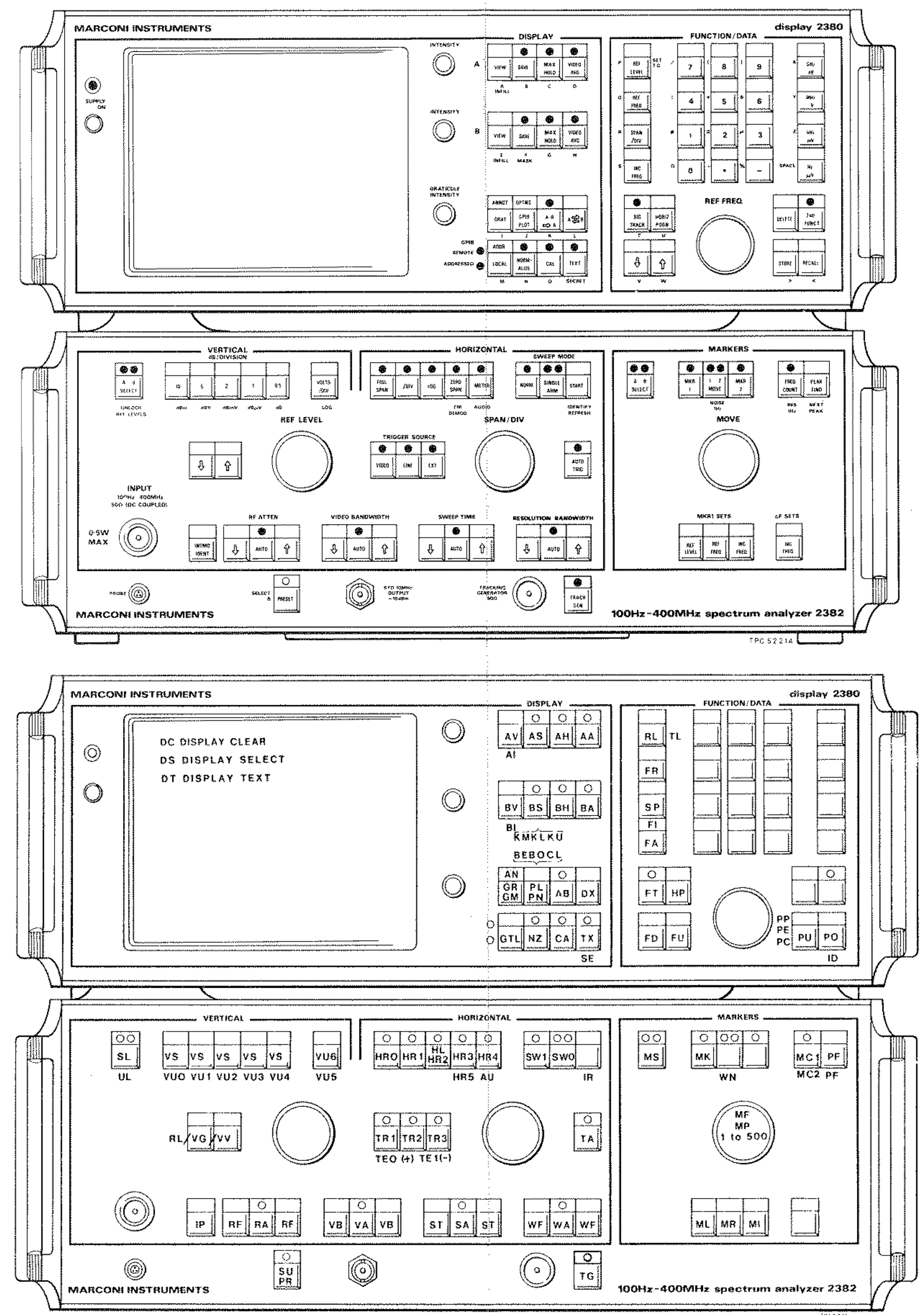


Fig. 2–3 Front panel command mnemonics and associated keys/controls

Chapter 3

PROGRAMMING EXAMPLES

This section contains a library of command subprograms and function subprograms as an aid to writing GPIB Controller programs for the 2382 Spectrum Analyzer. The subprograms are given for the HP Series 200 and, where appropriate, the HP85 desk top controllers and can be incorporated directly into the user's programs.

HP200 Implementation

Currently the series 200 range of controllers includes the following models:

9816 (model 216)
9817 (model 217)
9826 (model 226)
9836 (model 236)
9837 (model 237)
9920 (model 220)

The example programs will run on any of the above models with BASIC 3 installed and language extensions IO and MAT.

The routines should be called where necessary, with the relevant pass parameters. Pass parameters must be of the correct type (real, integer or string).

The main program should contain COM statements to match those in the subprogram and should perform variable initialization, I/O path assignments and dimension array/string variables as necessary. The GPIB device addresses and status byte variables are passed in COM statements.

The following convention has been adopted:

@ Mi2382 Spectrum Analyzer GPIB address

The Controller manuals should be consulted for detailed information on programming syntax.

HP85 Implementation

The HP85 routines are presented as subprograms which require the ADVANCED PROGRAMMING (AP) ROM to be installed. For use without the AP ROM the routines may be easily converted to normal BASIC subroutines with the following procedure.

1) Remove the SUB definition statement.
2) Remove COM statements and replace with DIM statements if necessary.
3) Replace CALL statements with GOSUB statements to the appropriate line number.
4) Replace SUBEXIT and SUBEND statements with RETURN.

For use as subprograms, the routines should be called, where necessary, with the relevant pass parameters. Pass parameters must be of the correct type (real, integer or string).

The main program should contain COM statements to match those in the subprogram and should perform variable initiation and array/string variables as necessary. The GPIB device address is passed in a COM statment.

The following convention has been adopted:

A   Spectrum Analyzer address.

Consult the control manuals for full details of programming semantics.

Description format

Each example contains a title, general description, full listing, subprogram name, pass parameters, subprogram calls and additional comments where necessary.

1 Automatic self-calibration

This example initiates the automatic self-calibration routine. It then monitors the status of the instrument until the self-calibration routine is completed. After self-calibration is completed it returns to the main program.

HP 200

Name:	Cal
Pass parameters:	None
Subprogram and function calls:	None
Returned parameters:	None

```
10     SUB Cal
20       COM /Mi2382/ @Mi2382
30       ON INTR 7 GOTO Service    ! Sets branch on interrupt
40       OUTPUT @Mi2382;"SM5"      ! Unmasks SRQ on end of calibration
50       OUTPUT @Mi2382;"CA"       ! Initiates self-calibration
60       REPEAT                    ! Until calibration SRQ received
70         ENABLE INTR 7;2         ! Enables interrupt on SRQ
80 Wait:   GOTO Wait               ! Waits for interrupt
90 Service:Sbyte=SPOLL(@Mi2382)    ! Reads status byte
100      UNTIL BIT(Sbyte,6) AND BIT(Sbyte,5)! Checks for calibration SRQ
110      OUTPUT @Mi2382;"SM-5"     ! Masks SRQ on end of calibration
120      OFF INTR 7                ! Disables interrupt on SRQ
130    SUBEND
```

HP85

Name:	"CAL"
Pass parameters:	None
Subprogram calls:	None

```
10 SUB "CAL"
20 COM A
30 ON INTR 7 GOTO 80
40 OUTPUT A ;"SM5" !                Unmasks SRQ on End of Calibration
50 OUTPUT A ;"CA" !                 Initiates self calibration routine
60 ENABLE INTR 7;8 !                Enables interrupt on SRQ
70 GOTO 70 !                        Waits for interrupt
80 STATUS 7,1 ; S !                 Resets interface to detect SRQ
90 S1=SPOLL(A) !                    Reads 2382 status byte
100 IF NOT (BIT(S1,6) AND BIT(S1,5)) THEN 60 ! Waits for Cal SRQ
110 OUTPUT A ;"SM-5" !              Disables SRQ on End Of Calibration
120 OFF INTR 7 !                    Disables interrupt on SRQ
130 SUBEND
```

2 Reading instrument status

This example reads a bit encoded word containing information about the status of the instrument:

Bit	0	1
0	Uncalibrated	Calibrated
1	Valid calibration	Invalid calibration
2	Sweep calibrated	Sweep uncalibrated
3	Not overloading	Overloading
4	Not overheating	Overheating
5	Int. freq. std. calibrated	Freq. std. uncalibrated
6	Int. freq. std.	Ext. freq. std.
7	Freq. std. in lock	Freq. std. out of lock
8	No tracking generator	Tracking generator fitted
9 -14	Not used	

HP 200

Name:	FNOutput_status
Pass parameters:	None
Subprogram and function calls:	None
Returned parameters:	Status Status word

```
10     DEF FNOutput_status
20       COM /Mi2382/ @Mi2382
30       OUTPUT @Mi2382;"OB1"     ! Selects alternative buffer
40       OUTPUT @Mi2382;"OI"      ! Outputs status word
50       ENTER @Mi2382;Status     ! Reads Status word into controller
60       OUTPUT @Mi2382;"OB0"     ! Selects main buffer
70       RETURN Status
80     FNEND
```

HP85

Name:	"rINST STATUS"
Pass parameters:	S status word
Subprogram calls:	None

```
10 SUB "rINST STATUS" (S)
20 COM A
30 OUTPUT A ;"OB1" !          Selects alternative buffer
40 OUTPUT A ;"OI" !           Outputs instrument status word
50 ENTER A ; S !              Reads instruments status word into controller

60 OUTPUT A ;"OB0" !          Selects main buffer
70 SUBEND
```

3 Displaying user defined captions

This example displays a user defined caption on the top line of the screen. A maximum of 38 characters (including spaces) can be displayed.

HP200

Name:	Text	
Pass parameters:	Text$	Caption string
Subprogram and function calls:	None	
Returned parameters:	None	

```
10      SUB Text(Text$)
20      COM /Mi2382/ @Mi2382
30        OUTPUT @Mi2382;"TX";CHR$(34)&Text$&CHR$(34) ! Transfers string to 2382
40      SUBEND
```

HP85

Name:	"TEXT"	
Pass parameters:	T$	Caption string
Subprogram calls:	None	

```
10 SUB "TEXT" (T$)
11 COM A
20 OUTPUT A :"TX":CHR$(34)&T$&CHR$(34) ! Char 34 is double quotes character
30 SUBEND
```

4 Returning user defined captions

This example returns a string corresponding to the user-defined caption *from* the Spectrum Analyzer *to* the Controller.

HP200

Name:	FNText$	
Pass parameters:	None	
Subprogram and function calls:	None	
Returned parameters:	T$	Caption string

```
10      DEF FNText$
20        COM /Mi2382/ @Mi2382
30        DIM Text$[40]
40        OUTPUT @Mi2382;"OPTX"     ! Outputs Caption
50        ENTER @Mi2382;Text$       ! Reads Caption into controller
60        RETURN Text$[2,38]
70      FNEND
```

HP85

Name:	"rTEXT"	
Pass parameters:	T$	Caption string
Subprogram calls:	None	

```
10 SUB "rTEXT" (T$)
20 COM A
30 DIM T1$[43]
40 OUTPUT A :"OP":"TX"
50 REM TX "<38 chars>" is returned
60 ENTER A : T1$
70 T$=T1$[5.42]
80 SUBEND
```

5 Displaying text

This routine is used to send user-defined text to the currently selected display. Text is sent line by line and can be displayed anywhere on the screen, starting at the user-defined line and column coordinates. If the column is omitted and the text comprises less than 65 characters then it is automatically centralized on the specified line.

HP200

Name:	Display_text	
Pass parameters:	Text$	Text string
	Line	Start line
	Pcolumn	Start column
Subprogram and function calls:	None	
Returned parameters:	None	

```
10     SUB Display_text(Text$,Line,OPTIONAL Pcolumn)
20       COM /Mi2382/ @Mi2382
30       IF NPAR=2 THEN             ! Centralises if column is not passed
40         IF LEN(Text$)>63 THEN
50           Column=1
60         ELSE
70           Column=INT((67-LEN(Text$))/2)
80         END IF
90       ELSE
100        Column=Pcolumn
110      END IF
120      OUTPUT @Mi2382;"DT";Line,Column;"#I";Text$,END ! Sends text to 2382
130    SUBEND
```

HP85

Name:	"DISP TEXT"	
Pass parameters:	T$	Text string
	L	Start line
	C	Start column
Subprogram calls:	None	

```
10 SUB "DISP TEXT" (T$,L,C)
20 COM A
30 IF NPAR=3 THEN 60 !          Sets branch if start column number is defined

40 IF LEN(T$)>63 THEN C1=1 ELSE C1=INT((67-LEN(T$))/2) ! Centralises text
50 GOTO 70
60 C1=C
70 CONTROL 7,16 ; 129,1 !          Sets End Of Line sequence to EOI and transpar
ent character
80 OUTPUT A USING "K" ; "DT";L;CHR$(44);C1;"#I";T$ !   Displays text
90 CONTROL 7,16 ; 2,13,10 !        Sets End Of Line sequence to carriage return,
 line feed
100 SUBEND
```

6 Returning user defined text

This example returns the user generated text displayed on the screen *from* the Spectrum Analyzer *to* the Controller. Text may be generated on the screen using the method in Example 5.

HP200

Name:	FNDisplay_Text$	
Pass parameters:	Line 1	Start line
	Column 1	Start column
	Line 2	Stop line
	Column 2	Stop column
Subprogram and function calls:	None	
Returned parameters:	Text$	Text string

```
10     DEF FNDisplay_text$(OPTIONAL Line1,Column1,Line2,Column2)
20       COM /Mi23827 @Mi2382
30       DIM Text$[1560]
40       SELECT NPAR
50       CASE 0
60         OUTPUT @Mi2382;"OPDT"        ! Outputs entire screen text
70       CASE 1
80         OUTPUT @Mi2382;"OPDT";Line1 ! Outputs text from Line1
90       CASE 2
100        OUTPUT @Mi2382;"opdt";Line1,Column1 ! Outputs text from Line1,Column1
110      CASE 3
120        OUTPUT @Mi2382;"OPDT";Line1,Column1,Line2 ! Outputs text from Line1,Co
lumn1 to Line2
130      CASE 4
140        OUTPUT @Mi2382;"OPDT";Line1,Column1,Line2,Column2 ! Outputs text from
Line1,Column1 to Line2,Column 2
150      END SELECT
160      ENTER @Mi2382 USING "4X,-K";Text$ ! Reads text into controller
170      RETURN Text$
180    FNEND
```

HP85

Name:	"rDISP TEXT"	
Pass parameters:	T$	Text string
	L1	Start line
	C1	Start column
	L2	Stop line
	C2	Stop column
Subprogram calls:	None	

```
10 SUB "rDISP TEXT" (T$,L1,C1,L2,C2)
20 COM A,S
30 ON NPAR GOTO 40,60,80,100,120 ! Sets branch for parametes passed
40 OUTPUT A USING "K" ; "OPDT" !   Outputs all screen text
50 GOTO 140
60 OUTPUT A USING "K" ; "OPDT";L1 ! Outputs text from C1
70 GOTO 140
80 OUTPUT A USING "K" ; "OPDT";L1;CHR$(44);C1 ! Outputs text from C1,L1

90 GOTO 140
100 OUTPUT A USING "K" ; "OPDT";L1;CHR$(44);C1;CHR$(44);L2 ! Outputs text from L
1,C1 to L2
110 GOTO 140
120 OUTPUT A USING "K" ; "OPDT";L1;CHR$(44);C1;CHR$(44);C2;CHR$(44);L2 ! Outputs
 Text from C1,L1
130 !                              to C2,L2
140 ENTER A USING "%#,4X,%#K" ; T$
150 SUBEND
```

Additional comments

Default case (no parameters passed)	Entire screen read.
Start line only defined	Start line to end of screen inclusive read.
Start line and column only defined	Start coordinate S to end of screen – inclusive read.
Start line/column and stop line defined	Start co-ordinate to stop line – inclusive read.
All pass parameters defined	Start co-ordinate to stop co-ordinate – inclusive read.

7 Learning display annotation

This example transfers the contents of that store containing the data from which the annotation of the currently selected display is generated *from* the Spectrum Analyzer *to* the Controller in a single block data transfer. The data contains all aspects of the display, trace annotation and graticule information but excludes any trace information.

The data is read into a string of length 202 characters. Refer to 'Command functions – LD' for a description of the string format. If the checksum pass parameter is included a value of 1 is returned for a correct checksum and 0 for a failed checksum.

HP200

Name:	Learn_disp_anot
Pass parameters:	String$ Annotation string Checksum
Subprogram and function calls:	FN Checksum
Returned parameters:	None

```
10     SUB Learn_disp_anot(String$,OPTIONAL Checksum)
20       COM /Mi2382/ @Mi2382
30       OUTPUT @Mi2382;"OPLD"     ! Outputs Display Annotation
40       ENTER @Mi2382 USING "-K";String$ ! Reads display annotation into control
ler
50       IF NPAR=2 THEN            ! Validates checksum if required
60         Checksum=FNChecksum(String$)
70       END IF
80     SUBEND
```

HP 85 sub-program

Name:	"rDISP ANNOT" (S$,C)
Pass parameters:	S$ Annotation string C Checksum
Subprogram calls:	Checksum

```
10 SUB "rDISP ANNOT" (S$,C)
20 COM A
30 OUTPUT A ;"OPLD" !                Outputs display annotation
40 ENTER A USING "%#,%#K" ; S$ !     Reads display annotation into controller
50 IF NPAR=2 THEN CALL "CHECKSUM" ( S$,C ) ! Validates checksum if required
60 SUBEND
```

8 Returning display annotation

This example saves the selected display, then transfers a string containing the display annotation *from* the Controller *to* the Spectrum Analyzer. The data must previously have been read into a 202 character string using the method outlined in example 7.

HP200

Name:	Send_disp_annot	
Pass parameters:	String$	Annotation string
Subprogram and function calls:	None	
Returned parameters:	None	

```
10      SUB Send_disp_anot(String$)
20        COM /Mi2382/ @Mi2382
30        OUTPUT @Mi2382;String$;END   ! Transfers display annotation to controller
40      SUBEND
```

HP85

Name:	"DISP ANNOT"	
Pass parameters:	S$	Annotation string
Subprogram calls:	None	

```
10 SUB "DISP ANNOT" (S$)
20 COM A
30 CONTROL 7,16 ; 130,13,10 !       Sends EOI with End Of Line Sequence
40 OUTPUT A USING "K" ; S$ !        Transfers display annotation to Spectrum Anal
yzer
50 CONTROL 7,16 ; 2,13,10 !         Cancels EOI with End Of Line Sequence
60 SUBEND
```

9 Outputting traces in ASCII to the controller

This example transfers the 502 values of amplitude which correspond to the current display *from* the Spectrum Analyzer *to* the Controller. The data is read into a 502 element array and is in explicit point notation.

In log volts mode data is sent in dBm. In linear volts mode the data lies in the range 0 – 1000 if 1 μV, 10 μV . . . 1 V is selected, 0 – 2000 if 2 μV, 20 μV . . . 0.2 V is selected and 0 – 5000 if 5 μV, 50 μV . . . 0.5 V is selected.

HP200

Name:	Output_trace	
Pass parameters:	Trace (*)	Trace array
Subprogram and function calls:	None	
Returned parameters:	None	

```
10    SUB Output_trace(Trace(*))
20      COM /Mi2382/ @Mi2382
30      OUTPUT @Mi2382;"OT"       ! Outputs trace
40      ENTER @Mi2382;Trace(*)     ! Reads trace into controller
50    SUBEND
```

HP85

Name:	"rTR ASCII"	
Pass parameters:	T()	Trace array
Subprogram calls:	None	

```
10 SUB "rTR ASCII" (T())
20 COM A
30 OUTPUT A ;"OT" !              Outputs trace in ASCII
40 FOR I=0 TO 501
50 ENTER A ; T(I) !              Reads trace into controller
60 NEXT I
70 SUBEND
```

10 Learning traces using block data transfer

This example transfers the 502 amplitude values comprising the current trace, *from* the Spectrum Analyzer *to* the Controller, in a single block data transfer. The data is read into a 502 element array and all values are integer.

In dB mode the values are 100 times their actual value. In log volts mode the values are sent in dBm. In linear volts mode the values lie in the range 0 – 1000 for a full-scale of 1 μV, 10 μV, ... 1 V; 0 – 2000 for a full-scale of 2 μV, 20 μV, ... 0.2 V; and 0 – 5000 for a full-scale of 5 μV, 50 . . . . 0.5 V.

HP200

Name:	Learn_trace
Pass parameters:	Trace(*) Trace array
Subprogram and function calls:	None
Returned parameters:	None

```
10     SUB Learn_trace(Trace(*))
20       COM /Mi2382/ @Mi2382
30       OUTPUT @Mi2382;"OPLT"     ! Outputs trace in block binary format
40       ENTER @Mi2382 USING "4(X),502(W)";Trace(*) ! Reads trace data into  trac
e array in controller
50     SUBEND
```

HP85

Name:	"rTrace"
Pass parameters:	T() Trace array
Subprogram calls:	None

```
10 SUB "rTRACE" (T())
20 COM A
30 OUTPUT A ;"OPLT" !               Outputs trace data in block binary format
40 ENTER A USING "#,4X" !           Skips over fist four characters in string
50 FOR I=1 TO 501
60 ENTER A USING "#,W" ; T(I) !     Reads trace data into trace array, the equiva
lent decimal
70 ! value of the binary word formed from the two binary bytes is read into the
array
90 NEXT I
100 SUBEND
```

11 Sending traces to the 2382 using block data transfers

This example transfers and subsequently displays the 502 amplitude values comprising a trace, *from* the Controller *to* the Spectrum Analyzer in a single block data transfer. The data must previously have been read into a 502 element array using the method described in example 10. The subprogram saves the selected display to prevent the new trace data from being overwritten by the analyzer.

HP200

Name:	Send_trace
Pass parameters:	Trace (*) Trace array
Subprogram and function calls:	None
Returned parameters:	None

```
10      SUB Send_trace(Trace(*))
20        COM /Mi2380/ @Mi2380
30        OUTPUT @Mi2380;"OPSL" ! Ask for selected trace
40        ENTER @Mi2380;Selected_trace$ ! Get "SL A" or "SL B"
50        OUTPUT @Mi2380;Selected_trace$[4]&"S1" ! Save trace with AS1 or BS1
60        OUTPUT @Mi2380 USING """LT#I"",502(W)";Trace(*) END ! Binary transfer
70      SUBEND
```

HP85

Name:	"TRACE"
Pass parameters:	T() Trace array
Subprogram calls:	None

```
10 SUB "TRACE" (T())
20 COM A
30 OUTPUT A ;"OPSL" ! Ask for selected trace
40 ENTER A ; T$ ! Get "SL A" or "SL B"
50 OUTPUT A ;T$[4]&"S1" ! Save trace with AS1 or BS1
60 SEND 7 ; UNL MTA LISTEN A-700 ! Configure bus with 2380 as talker, receiver a
s listener
70 OUTPUT 7 USING "#,4A" ; "LT#I" ! LT command header without EOL sequence
80 FOR I=0 TO 500
90 OUTPUT 7 USING "#,W" ; T(I) ! Send each value as a WORD
100 NEXT I
110 CONTROL 7,16 ; 130,13,10 ! Enable EOI
120 OUTPUT 7 USING "W" ; T(I)
130 CONTROL 7,16 ; 2,13,10 ! Cancel EOI
140 SUBEND
```

12 Learning instrument settings

This example transfers data containing the current instrument settings *from* the Spectrum Analyzer *to* the Controller. The data is read into a string of length 242 characters, refer to 'Command functions – OPLI, LI' for a description of the block format. If the checksum parameter is included a value of 1 is returned for a correct checksum and 0 for a failed checksum.

HP200

Name:	Learn_settings
Pass parameters:	String$ Instrument settings string Checksum
Subprogram and function calls:	FN Checksum
Returned parameters:	None

```
10 SUB Learn_settings(String$,OPTIONAL Checksum)
20      COM /Mi2382/ @Mi2382
30      OUTPUT @Mi2382;"OPLI"      ! Outputs instrument settings
40      ENTER @Mi2382 USING "-K";String$ ! Reads instrument settings into contro
ller
50      IF NPAR=2 THEN
60              Checksum=FNChecksum(String$)
70      END IF
80 SUBEND
```

HP85

Name:	"rINST SETTINGS"
Pass parameters:	S$ Instrument settings C Checksum
Subprogram calls	CHECKSUM

```
10 SUB "rINST SETTINGS" (S$,C)
20 COM A
30 OUTPUT A ;"OPLI" !              Outputs insrument settings
40 ENTER A USING "%#,%#K" ; S$ !   Sends instrument settings to controller
50 IF NPAR=2 THEN CALL "CHECKSUM" ( S$,C ) ! Validates checksum if required
60 SUBEND
```

13 Sending instrument settings to the 2382

This example transfers instrument settings data *from* the Controller *to* the Spectrum Analyzer. This data must previously have been read into a character string using the method outlined in example 12.

HP200

Name:	Send_settings
Pass parameters:	String $ Instrument settings
Subprogram and function calls:	None
Returned parameters:	None

```
10    SUB Send_settings(String$)
20      COM /Mi2382/ @Mi2382
30      OUTPUT @Mi2382;String$;END ! Sends settings to Spectrum analyzer in bloc
k binary format
40    SUBEND
```

HP85

Name:	"INST SETTINGS"	
Pass parameters:	S$	Instrument settings
Subprogram calls:	None	

```
10 SUB "INST SETTINGS" (S$)
20 COM A
30 CONTROL 7,16 ; 130,13,10 !       Sends EOI with End Of Line (EOL) sequence
40 OUTPUT A USING "K" ; S$ !        Transfers instrument settings to Spectrum Ana
lyzer
50 CONTROL 7,16 ; 2,13,10 !         Cancels EOI with EOL sequence
60 SUBEND
```

14 Taking sweeps

This example triggers the Spectrum Analyzer to take a specified number of sweeps, or one sweep if the number of sweeps is unspecified. It then monitors the status of the instrument so that it returns to the main program when the correct number of sweeps has been completed.

HP200

Name:	Take_sweep	
Pass parameters:	Count	Number of sweeps
Subprogram and function calls:	None	
Returned parameters:	None	

```
10 SUB Take_sweep(OPTIONAL Count)
20      COM /Mi2382/ @Mi2382
30      ON INTR 7 GOTO Service    ! Sets branch on interrupt
40      OUTPUT @Mi2382;"SM3"      ! Unmasks SRQ on End Of Sweep
50      IF NPAR THEN              ! Takes sweeps if number of sweeps specified
60              OUTPUT @Mi2382;"TS";Count
70      ELSE
80              OUTPUT @Mi2382;"TS"
90      END IF
100     ENABLE INTR 7;3           ! Enable SRQ interrupt
110     REPEAT                    ! Until End Of Sweep SRQ detected
120 Wait:       GOTO Wait         ! Waits for SRQ
121 Service:    Sbyte=SPOLL(@Mi2382)! Reads status byte
130     UNTIL BIT(Sbyte,6) AND BIT(Sbyte,3) ! Checks for End Of Sweep SRQ
140     OFF INTR 7                ! Disables SRQ interrupt
150     OUTPUT @Mi2382;"SM-3"     ! Unmasks SRQ on End Of Sweep
160 SUBEND
```

HP85

Name:	"TAKE SWEEP"
Pass parameters:	C Number of sweeps
Subprogram calls:	None

```
10 SUB "TAKE SWEEP" (C)
20 COM A,S
30 ON INTR 7 GOTO 80 !               Sets branch on interrupt
40 OUTPUT A ;"SM3" !                 Unmasks SRQ for End Of Sweep
50 IF NPAR THEN OUTPUT A ;"TS";C ELSE OUTPUT A ;"TS" ! Initiates sweep(s)
60 ENABLE INTR 7;8 !                 Enables interrupt on SRQ
70 GOTO 70 !                         Waits for SRQ
80 STATUS 7,1 ; S !                  Resets interface to detect SRQ
90 S1=SPOLL(A) !                     Reads status byte
100 IF NOT (BIT(S1,6) AND BIT(S1,3)) THEN 60 ! Checks for sweep SRQ
110 OFF INTR 7 !                     Cancels interrupt on SRQ
120 OUTPUT A ;"SM-3" !               Masks SRQ for End Of Sweep
130 SUBEND
```

15 Setting lower mask limits

This example is used to turn the mask function on, clear current lower mask limits and enter a new set of values if required. The mask limits are stored in (7,1) element array (assuming OPTION BASE 0). The format of the array is:

0	Limit 1 amplitude	Limit 1 freq.
1	Limit 2 amplitude	Limit 2 freq.
2	Limit 3 amplitude	Limit 3 freq.
3	Limit 4 amplitude	Limit 4 freq.
4	Limit 5 amplitude	Limit 5 freq.
5	Limit 6 amplitude	Limit 6 freq.
6	Limit 7 amplitude	Limit 7 freq.
7	Limit 8 amplitude	Limit 8 freq.

HP200

Name:	Lower_mask	
Pass parameters:	Count	Number of mask limits
	Limits (*)	Lower mask limits
Subprogram and function calls:	None	
Returned parameters:	None	

```
10     SUB Lower_mask(Count,OPTIONAL Limits(*))
20       COM /Mi2382/ @Mi2382
30       OUTPUT @Mi2382;"OPKM"     !Outputs status of Mask function
40       ENTER @Mi2382;K           ! Reads Mask function status into controller
50       IF K=1 THEN               ! Turns Mask ON if previously OFF
60         OUTPUT @Mi2382;"KM1"
70       END IF
80       OUTPUT @Mi2382;"KL"       ! Clears Lower Mask limits
90       IF Count>0 THEN
100        REDIM Limits(Count-1,1)
110        OUTPUT @Mi2382;"KL";Limits(*) ! Sends Lower Mask limits to 2382
120        REDIM limits(7,1)
130      END IF
140    SUBEND
```

HP85

Name:	"L MASK"	
Pass parameters:	C	Number of mask limits
	L(,)	Lower mask limits
Subprogram calls:	None	

```
10 SUB "L MASK" (C,L(,))
20 COM A
30 OUTPUT A ;"OPKM" !              Outputs status of MASK function
40 ENTER A ; K !                   Reads status of MASK function into controller

50 IF K=0 THEN OUTPUT A ;"KM1" !   Turns mask function on if previously in off c
ondition
60 OUTPUT A ;"KL" !                Clears lower MASK limits
70 IF C=0 THEN SUBEXIT
80 FOR I=0 TO C-1
90 OUTPUT A ;"KL";L(I,0);CHR$(44);L(I,1)
100 NEXT I
110 SUBEND
```

16 Setting upper mask limits

This example is used to turn the mask function on, clear current upper mask limits and enter a new set of values if required. The mask limits are stored in a (7,1) element array (assuming OPTION BASE 0). The format of the array is the same as example 15 for lower mask limits.

HP200

Name:	Upper_mask	
Pass parameters:	Count	Number of limits
	Limits (*)	Limit array
Subprogram and function calls:	None	
Returned parameters:	None	

As for Example 15 with the exception that the subprogram name is changed to 'Upper_mask' and the command 'KL' is replaced by 'KU'.

HP85

Name:	"U MASK"	
Pass parameters:	C	No. of limits
	L(,)	Limit array
Subprogram calls:	None	

As for Example 15 with the exception that e subprogram name is changed to 'U MASK' and the command 'KL' is replaced by 'KU'.

17 Outputting lower mask limits

This example transfers the values of the lower mask limits *from* the Spectrum Analyzer *to* the Controller. The values are read into a (7,1) element array (assuming option base 0).

HP200

Name:	Op_lower_mask	
Pass parameters:	Number of mask limits	
	Count	
	Limits (*)	Lower limits array
Subprogram and function calls:	None	
Returned parameters:	None	

```
10    SUB Op_lower_mask(Count,Limits(*))
20      COM /Mi2382/ @Mi2382
30      OUTPUT @Mi2382;"OPKL"      ! Outputs Lower Mask limits
40      ENTER @Mi2382;Count        ! Reads number of Lower Mask limits
50      IF Count>0 THEN
60        ENTER @Mi2382 USING "%,K,K,/";Limits(*) ! Reads Lower Mask Limits
70      END IF
80    SUBEND
```

HP85

Name:	"rL MASK"	
Pass parameters:	C	Number of mask limits
	L(,)	Lower limits array
Subprogram calls:	None	

```
10 SUB "rL MASK" (C,L(,))
20 COM A
30 OUTPUT A :"OPKL"
40 ENTER A : C
50 IF C=0 THEN SUBEXIT
60 FOR I=0 TO C-1
70 ENTER A : L(I,0),L(I,1)
80 NEXT I
90 SUBEND
```

18 Outputting upper mask limits

This example transfers the values of the upper mask limits *from* the Spectrum Analyzer *to* the Controller. The values are read into a (7,1) element array (assuming Option base 0).

HP200

Name:	Op_upper mask	
Pass parameters:	Number of mask limits	
	Count	
	Limits (*)	Upper mask limits array
Subprogram and function calls:	None	
Returned parameters:	None	

As for Example 17 with the exception that the subprogram name is changed to 'Op_upper_mask' and the 'OPKL' command is replaced by 'OPKU'.

HP85

Name:	"rU MASK"	
Pass parameters:	C	Number of mask limits
	L(,)	Upper mask limits array
Subprogram calls:	None	

As for Example 17 with the exception that the subprogram name is changed to "rU MASK" and the 'OPKL' command is replaced by 'OPKU'.

19 Waiting for key press

This example monitors the 2382 for an SRQ on key press and returns the code value of the key.

HP200

Name: FNKey_press

Pass parameters: None

Subprogram and function calls: None

Returned parameters: Key Key code

```
10     DEF FNKey_press
20       COM /Mi2382/ @Mi2382
30       ON INTR 7 GOTO Service     ! Sets branch on interrupt
40       OUTPUT @Mi2382;"SM4"       ! Unmasks SRQ on Key Press
50       REPEAT                     ! Untill Key Press SRQ received
60         ENABLE INTR 7;2          ! Enables SRQ interrupt
70 Wait:   GOTO Wait                ! Waits for SRQ
80 Service:Sbyte=SPOLL(@Mi2382)     ! Reads status byte
90       UNTIL BIT(Sbyte,6) AND BIT(Sbyte,4) ! Checks for Key Press SRQ
100      OUTPUT @Mi2382;"SM-4"      ! Masks SRQ on Key Press
110      OFF INTR 7                 ! Disables SRQ interrupt
120      OUTPUT @Mi2382;"OK"        ! Outputs key code
130      ENTER @Mi2382;Key          ! Reads key code
140      RETURN Key
150    FNEND
```

HP85

Name: "rKEY PRESS"

Pass parameters: C Key code

Subprogram calls: None

```
10 SUB "rKEY PRESS" (C)
20 COM A
30 ON INTR 7 GOTO 70 !             Sets branch for SRQ
40 OUTPUT A ;"SM4" !               Unmasks SRQ on Key Press
50 ENABLE INTR 7;8
60 GOTO 60 !                       Waits for SRQ
70 STATUS 7,1 ; S !                Resets interface to detect SRQ

80 S1=SPOLL(A) !                   Reads status byte of Spectrum Analyzer
90 IF NOT (BIT(S1,6) AND BIT(S1,4)) THEN 50 ! Checks for SRQ on Key Press
100 OUTPUT A ;"SM-4" !             Masks SRQ on Key Press
110 OFF INTR 7 !                   Cancels interrupt on SRQ
120 OUTPUT A ;"OK" !               Outputs key code
130 ENTER A ; C !                  Reads key code
140 SUBEND
```

20 Checksum

This example tests the checksum transferred from the spectrum analyzer. It returns a value of 1 for a correct checksum, and 0 for an incorrect checksum.

HP200

Name:	FNChecksum	
Pass parameters:	String$	Checksum string
Subprogram and function calls:	None	
Returned parameters:	Checksum	

```
10     DEF FNChecksum(String$)
20       COM /Mi2382/ @Mi2382
30       Checksum=-255
40       FOR I=5 TO LEN(S$)
50         Checksum=(Checksum+NUM(S$[I])) MOD 256 ! Validates Checksum
60       NEXT I
70       RETURN NOT Checksum      ! Retuns 1 for pass, 0 for fail
80     FNEND
```

HP85

Name:	"CHECKSUM"	
Pass parameters:	String$	Checksum
Subprogram calls:	None	

```
10 SUB "CHECKSUM" (S$,C)
20 COM A,S
30 C=-255
40 FOR I=5 TO LEN(S$)
50 C=(C+NUM(S$[I])) MOD 256 !      Validates checksum
60 NEXT I
70 C=NOT C !                       Sets C=1 for pass, and C=0 for fail
80 SUBEND
```

21 Obtaining a GPIB plot

This example shows the two ways of obtaining a GPIB plot from the 2380 using a controller.

```
10    ! Example of the two ways to obtain a GPIB plot from the 2380 using a
20    ! controller.
30    !
40    Example=2
50    !
60    IF Example=1 THEN
70    !
80    ! Example 1. Simultaineous reception by both 2380 and controller.
90      Mi2380=703
100     Plotter=705
110     ASSIGN @Mi2380 TO Mi2380
120     ASSIGN @Plotter TO Plotter
130     OUTPUT @Plotter;"IN"
140     OUTPUT @Mi2380;"PL1,2,3,4,5,6,7"
150     SEND 7;UNL TALK Mi2380-700 LISTEN Plotter-700 MLA
160     ENTER 7 USING "+,%,20000(X)"
170   ELSE
180   !
190   ! Example 2. Reception by plotter via the 2380.
200   !
210     DIM Buffer$[20000]
220     Mi2380=703
230     Plotter=705
240     ASSIGN @Mi2380 TO Mi2380
250     ASSIGN @Plotter TO Plotter
260     OUTPUT @Plotter;"IN"
270     OUTPUT @Mi2380;"PL1,2,3,4,5,6,7"
280     ENTER @Mi2380;Buffer$
290     OUTPUT @Plotter;Buffer$
300   END IF
310   END
```

Chapter 4

READING CALIBRATION DATA

The automatic internal calibration routine available on the 2382/2380 Spectrum Analyzer corrects for drift in the tuning and gain of the signal path and for its overall frequency response. In this context the term signal path encompasses the path followed by a signal between the front panel input of the 2382 and the point where the level is detected.

It is possible to output the data generated by the calibration routine over the GPIB. This makes it possible to keep a record of how the instrument has performed over a period of time and to spot any sudden changes which may indicate that the instrument has developed a fault.

The microprocessor adjusts the tuning of the signal path by varying the DC voltage applied across a varactor diode thereby altering its capacitance. The gain of the signal path is adjusted in a similar manner by varying the DC voltage applied across a pin diode thereby altering its dynamic resistance. The voltages are obtained from a pair of 8 bit digital-to-analogue converters. Therefore the calibration data consists of the pair of 8 bit numbers (bytes) that the microprocessor writes to the D-to-A converters. There is a pair of numbers for each of the filter bandwidths that may be selected for the signal path. However, the gain of the signal path is also affected by changes in input attenuation. Therefore the calibration data includes a 'gain adjust' number for each of the settings of input attenuation. From the 'gain adjust' numbers measured for the filters and those for the attenuators the required 'gain adjust' number for any filter/attenuator combination can then be found.

The frequency response of the signal path is measured so that compensations may be made. In order to record the frequency response the frequency span of the instrument is notionally divided up into 10 decades and the response is measured at 90 equally spaced ponts across each decade, (the 2382 unit only uses decades up 1 to 7). The 900 resulting measurements are stored in an array as follows:

Decade	Frequency range	Position in array
1	100 Hz – 1 kHz	0 – 89
2	1 kHz – 10 kHz	90 – 179
3	10 kHz – 100 kHz	180 – 269
4	100 kHz – 1 MHz	270 – 359
5	1 MHz – 10 MHz	360 – 449
6	10 MHz – 100 MHz	450 – 539
7	100 MHz – 1 GHz	540 – 629
8	1 GHz – 10 GHz	630 – 719
9	10 GHz – 100 GHz	720 – 809
10	100 GHz – 1 THz	810 – 899

Each point is stored as a signed 16 bit value i.e. it may be in the range 32768 to 32767. A value of 3904 represents a flat response (no correction required) and an increase (decrease) of 1 from this value represents a gain (attenuation) of 0.025 dB.

The calibration data is read over the GPIB using the OC command. The syntax of the command is:

‹listen›OC‹talk›OC‹space›‹B› . . . ‹cr›‹lf›‹eoi›

60 bytes are sent. The first 32 bytes are the correction values for the filters arranged as follows:

CORRECTION VALUE

Filter	Frequency	Level
3 kHz	Byte 00	Unused
3 Hz	Byte 02	Byte 03
10 Hz	Byte 04	Byte 05
30 Hz	Byte 06	Byte 07
100 Hz	Byte 08	Byte 09
300 Hz	Byte 10	Byte 11
1 kHz	Byte 12	Byte 13
3 kHz	Byte 14	Byte 15
10 kHz	Byte 16	Byte 17
30 kHz	Byte 18	Byte 19
100 kHz	Byte 20	Byte 21
300 kHz	Byte 22	Byte 23
1 MHz	Byte 24	Byte 25
Unused	Byte 26	Byte 27
Unused	Byte 28	Byte 29
Unused	Byte 30	Byte 31

The next 8 bytes are the correction values for the attenuators arranged as follows:

Attenuation	Correction Value
00 dB	Byte 32
10 dB	Byte 33
20 dB	Byte 34
30 dB	Byte 35
40 dB	Byte 36
50 dB	Byte 37
60 dB	Byte 38
Unused	Byte 39

The last 20 bytes are actually 10 signed 16 bit values. Each word is the absolute value of the maximum deviation from a flat frequency response (3904) for each of the 10 decades and allows for an unusually large deviation to be noticed quickly. The words are arranged as follows:

Decade		
100 Hz – 1 kHz	Byte 40 – Word 00 (MSB)	Byte 41 – Word 00 (LSB)
1 kHz – 10 kHz	Byte 42 – Word 01 (MSB)	Byte 43 – Word 01 (LSB)
10 kHz – 100 kHz	Byte 44 – Word 02 (MSB)	Byte 45 – Word 02 (LSB)
100 kHz – 1 MHz	Byte 46 – Word 03 (MSB)	Byte 47 – Word 03 (LSB)
1 MHz – 10 MHz	Byte 48 – Word 04 (MSB)	Byte 49 – Word 04 (LSB)
10 MHz – 100 MHz	Byte 50 – Word 05 (MSB)	Byte 51 – Word 05 (LSB)
100 MHz – 1 GHz	Byte 52 – Word 06 (MSB)	Byte 53 – Word 06 (LSB)
1 GHz – 10 GHz	Byte 54 – Word 07 (MSB)	Byte 55 – Word 07 (LSB)
10 GHz – 100 GHz	Byte 56 – Word 08 (MSB)	Byte 57 – Word 08 (LSB)
100 GHz – 1 THz	Byte 58 – Word 09 (MSB)	Byte 59 – Word 09 (LSB)

MSB = Most significant byte. LSB = Least significant byte

Each point in the frequency response array may be read over the GPIB using the OPCR command. The syntax for the command is:

<listen>OPCR<number><talk>CR<space><number><cr><lf><eoi>

The number sent by the Controller is the number of the point required, which must be in the range 0 – 899, and the number returned (ASCII format) is the value of that point.

A complementary command CR permits points in the frequency response array to be entered over the GPIB primarily for development purposes. The syntax for the command is:

<listen>CR<number>, ;<number>

The first number ranges from 0 to 899 and the second from 32768 to 32767.

INDEX

LIST OF TABLES

LIST OF FIGURES