



SPECTRUM ANALYZERS

3280 & 3280A Series



Programming Manual

Document part no. 46892/768



SPECTRUM ANALYZERS

3280 & 3280A SERIES

Programming manual

3281/3281A 3 Hz–3.0 GHz
3282/3282A 3 Hz–13.2 GHz
3283/3283A 3 Hz–26.5 GHz

© Aeroflex International Ltd. 2009

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, or recorded by any information storage or retrieval system, without permission in writing by Aeroflex International Ltd. (hereafter referred to throughout the document as 'Aeroflex').

Manual part no. 46892/768 (PDF version)
Based on Issue 3 of the printed manual

17 September 2009

About this manual

This manual explains how to program the 3280 and 3280A Series Spectrum Analyzers. Programs can provide remote control for all functions of the instrument other than power on/off. Not all of the features detailed in this manual are available in every model.

Intended audience

Persons engaged on work relating to the design and manufacture of RF and microwave sub-systems and modules, or the installation and maintenance of those systems.

It is assumed that the user is familiar with the terms used in RF and microwave measurements.

Structure

Chapter 1

Describes remote control functions and how to select interface ports. Gives examples of configurations, and RS-232C and GPIB specifications.

Chapter 2

Describes how to connect external devices using RS-232C and GPIB cables. Also describes how to set up the instrument's interface.

Chapter 3

Describes the format of the device messages transmitted on the bus between a controller (host computer) and instrument via the RS-232C or GPIB system.

Chapter 4

Full descriptions of all available commands.

Chapter 5

Describes device-status reporting and its data structure.

Chapter 6

This section shows some example codes to transmit messages on the bus between a personal computer and the spectrum analyzer via GPIB.

Appendix A

Listing of remote control commands.

Appendix B

Listing of error codes.

Associated publications

- **3280 Series Operating Manual**
(printed version 46882/766; PDF version 46892/766 on CD-ROM 46886/051)
- **3280A Series Operating Manual**
(printed version 47000/011; PDF version 47000/011 on CD-ROM 46886/075)

Contents

Chapter 1 GENERAL INFORMATION	1-1
Chapter 2 CONNECTING DEVICES.....	2-1
Chapter 3 DEVICE MESSAGE FORMAT	3-1
Chapter 4 DETAILED DESCRIPTION OF COMMANDS.....	4-1
Part 1 Spectrum mode.....	4-9
Part 2 Phase noise mode.....	4-207
Part 3 CCDF mode	4-245
Part 4 GPIB common commands.....	4-286
Chapter 5 STATUS STRUCTURE.....	5-1
Chapter 6 EXAMPLE CODES	6-1
Appendix A REMOTE CONTROL.....	A-1
Appendix B ERROR CODES.....	B-1

Chapter 1

GENERAL INFORMATION

Contents

General	1-1
Remote control functions.....	1-1
Interface port selection functions	1-1
Examples of configurations using RS-232C and GPIB, printer, USB	1-2
Stand-alone type	1-2
Control by the host computer	1-2
Control by the host computer	1-2
RS-232C specification.....	1-3
GPIB specification.....	1-3

General

This spectrum analyzer, when combined with an external controller (host computer, personal computer, etc.) can automate your measurement system. For this purpose, the instrument is equipped with an RS-232C interface port and GPIB interface.

Remote control functions

The remote control functions of the equipment are used to do the following:

- Control most of functions except the power switch and [System] key.
- Read setting value.
- Configure the automatic measurement system when the equipment is combined with a personal computer and other measuring equipments.

* Set the RS-232C interface settings from the front panel.

* Set the GPIB address from the front panel.

Interface port selection functions

The instrument has a standard RS-232C interface and a GPIB interface and parallel (printer) interface and USB interface. Use the panel to select the interface port to be used to connect external devices as shown below.

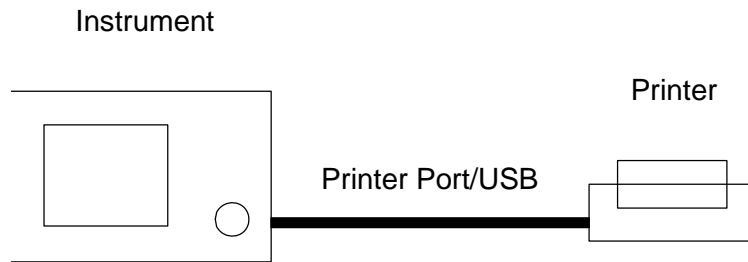
Port for the external controller: select RS-232C or GPIB.

Port for the printer: select parallel port or USB.

Examples of configurations using RS-232C and GPIB, printer, USB

Stand-alone type

Waveforms measured with the instrument are output to the printer.



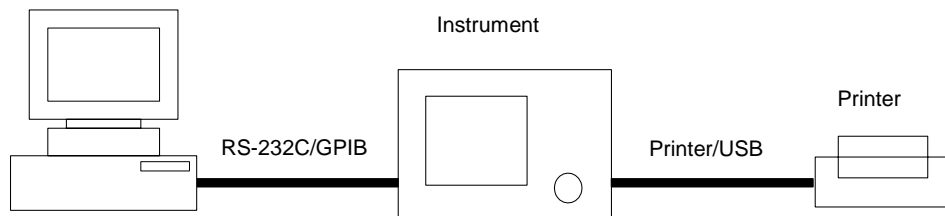
Control by the host computer

The instrument is controlled automatically or remotely from the computer.



Control by the host computer

The waveforms measured by controlling the instrument automatically or remotely are output to the printer.



RS-232C specification

The table below lists the standard specification of RS-232C in the instrument.

ITEM	SPECIFICATION
Function	Control from the external controller (except for power-ON/OFF and [System] key)
Communication system	Asynchronous (start-stop synchronous system), half-duplex
Communication control system	NONE, XON_XOFF, RTS_CTS, DTR_DSR
Baud rate	4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200
Data bits	5, 6, 7, 8 bits
Parity	NONE, ODD, EVEN
Stop bit (bits)	1 or 2 bits
Connector	D-sub 9-pin, male

GPIB specification

The table below lists the specification with the GPIB provided for the instrument.

Item	Specification	Supplementary explanation
Interface function	SH1	All source handshake functions are provided. Synchronizes the timing of data transmission.
	AH1	All acceptor handshake functions are provided. Synchronizes the timing of data reception.
	T6	The basic talk functions and serial poll functions are provided. The talk only functions are not provided. The talker can be canceled by MLA.
	L4	The basic listener functions are provided. The listen only function is not provided. The listener can be canceled by MTA.
	SR1	All service request and status byte functions are provided.
	RL1	All remote/local functions are provided. The local lockout function is provided.
	PP0	The parallel poll functions are not provided.
	DC1	All device clear functions are provided.
	E2	Output is tri-state.
	LE0	No extended listener capabilities.
TE0	No extended talker capabilities.	

Chapter 2 CONNECTING DEVICES

Contents

General	2-1
Connecting an external device with an RS-232C cable.....	2-1
Connection diagram for RS-232C interface signals	2-2
Setting connection port interfaces	2-3
Setting the RS-232C interface conditions.....	2-3
Connecting a device with a GPIB cable	2-4
GPIB constraints.....	2-4
Setting the GPIB address.....	2-5

General

This chapter describes how to connect external devices such as the host computer with RS-232C and GPIB cables. It also describes how to set up the instrument's interface.

Connecting an external device with an RS-232C cable

Connect the RS-232C connector (D-sub 9-pin, male) on the rear panel of the instrument to the RS-232C connector of the external device (host controller, computer) with an RS-232C cable.

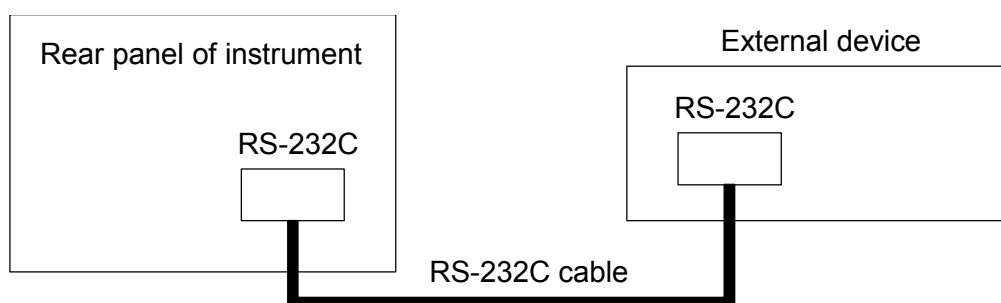


Fig 2-1 RS-232C cable connections

Note: RS-232C connectors with 9 pins are available. When purchasing the RS-232C cable, check the pins on the RS-232C connector of the external device.

The following RS-232C cable is provided with the instrument.

RS-232C cable (for personal computer)

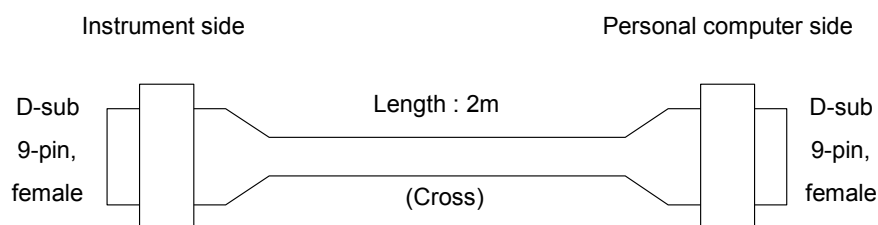


Fig 2-2 RS-232C cable

Connection diagram for RS-232C interface signals

The diagram below shows the RS-232C interface signal connections between the instrument and a host system such as a personal computer.

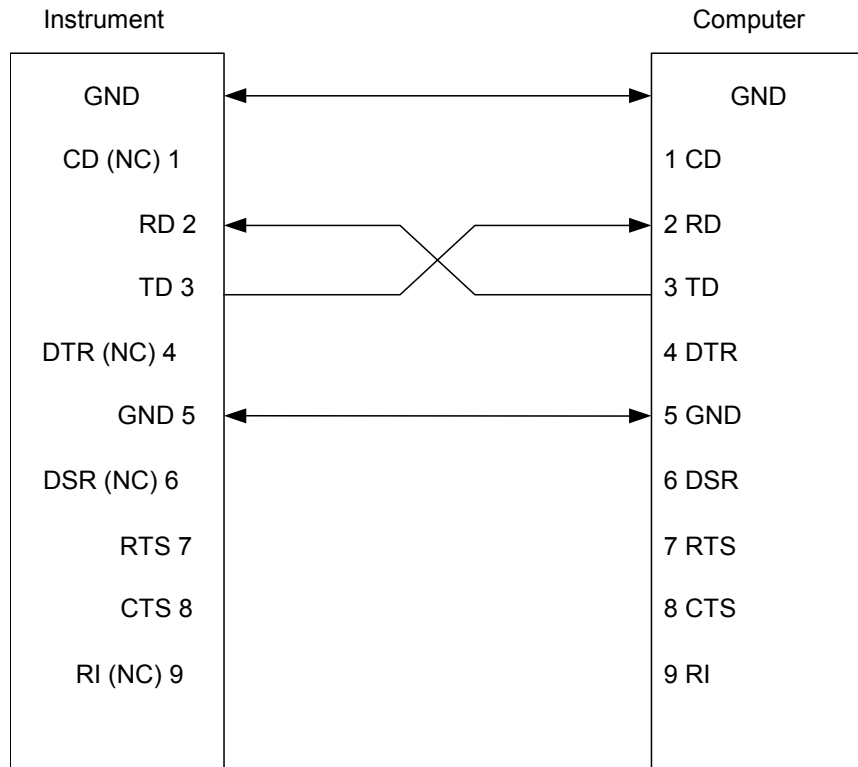
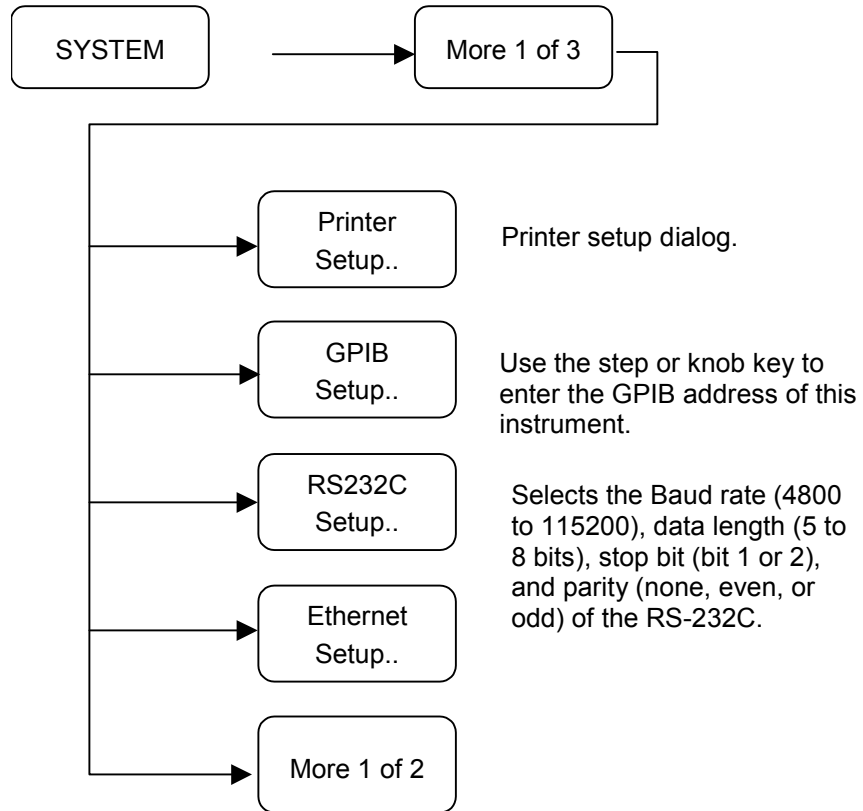


Fig 2-3 Connection to personal computer

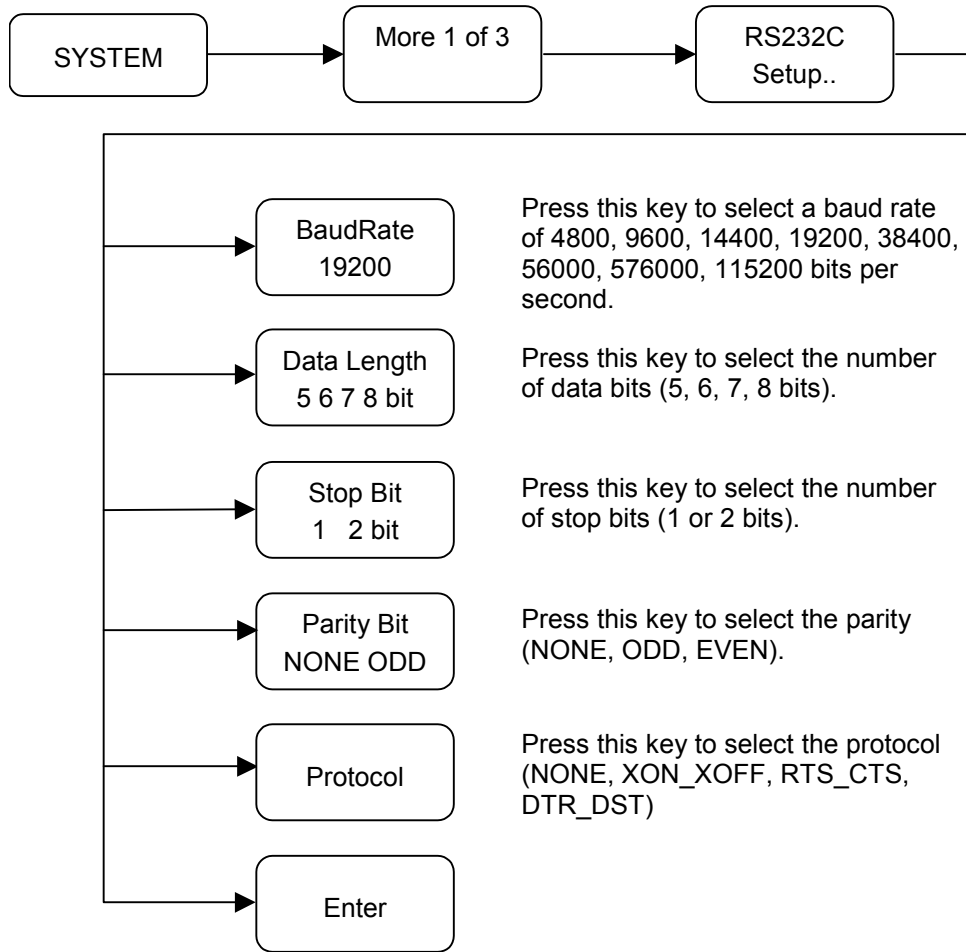
Setting connection port interfaces

Set the interfaces between the connection ports of the instrument and host system.



Setting the RS-232C interface conditions

Set the RS-232C interfaces conditions of the instrument to those of the external device to be connected.



Connecting a device with a GPIB cable

Connect the GPIB connector on the rear panel of the instrument to the GPIB connector of an external device with a GPIB cable.

Be sure to connect the GPIB cable before turning instrument power on.

GPIB constraints

Number of interconnected devices:	15 maximum
Interconnection path maximum cable length:	20 m maximum or 2 m per device (whichever is less).
Message transfer scheme:	Byte serial, bit parallel a synchronous data transfer using a 3-line handshake system.
Data rate:	Maximum of 1 Mb/s over the specified distances with tri-state drivers. Actual data rate depends on the transfer rate of the slowest device connected to the bus.
Address capability:	Primary address: 31 talk, 31 listen. A maximum of 1 talk and 14 listeners can be connected to the interface at given time.

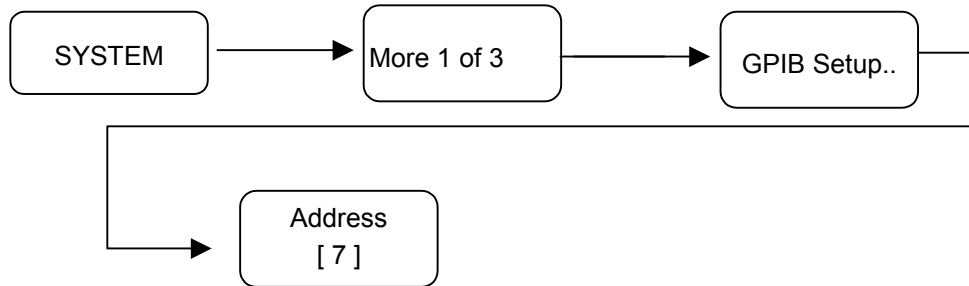
Multiple-controller capability:

In a system with more than one controller, only one controller can be active at any given time.

The active controller can pass control to another controller, but only the system controller can assume unconditional control. Only one system controller is allowed.

Setting the GPIB address

Set the GPIB address of the instrument as follows.



Use the numeric key to enter the GPIB address of this instrument. The initial value is 7.

Chapter 3

DEVICE MESSAGE FORMAT

Contents

General	3-1
Program message format	3-1
Program message terminator	3-2
Program message.....	3-2
Numeric program data.....	3-3
< Integer format (NR1) >	3-3
< Fixed-point format (NR2) >	3-3
Suffix of program data (unit)	3-4
Response message format.....	3-5
Response message terminator.....	3-5
Response message	3-5
Usual response message unit	3-6
Response data.....	3-6
Character response data.....	3-6
Numeric response data	3-7
< Integer format (NR1) >	3-7
< Integer format (NR2) >	3-7

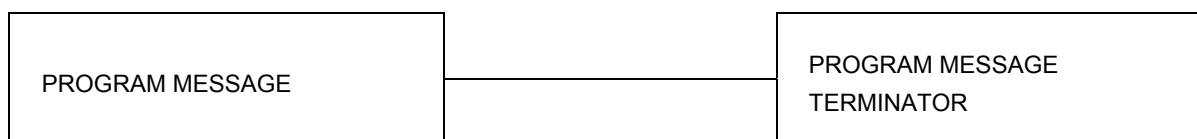
General

This section describes the format of the device messages transmitted on the bus between a controller (host computer) and instrument via the RS-232C or GPIB system.

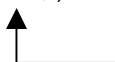
The device messages are data messages transmitted between the controller and devices, program messages transferred from the controller to this instrument (device), and response messages input from this instrument (device) to the controller. There are also two types of program commands and program queries in the program message. The program command is used to set this instrument's parameters and to instruct it to execute processing. The program query is used to query the values of parameters and measured results.

Program message format

To transfer a program message from the controller program to this instrument using the "Send" statement, the program message formats are defined as follows:



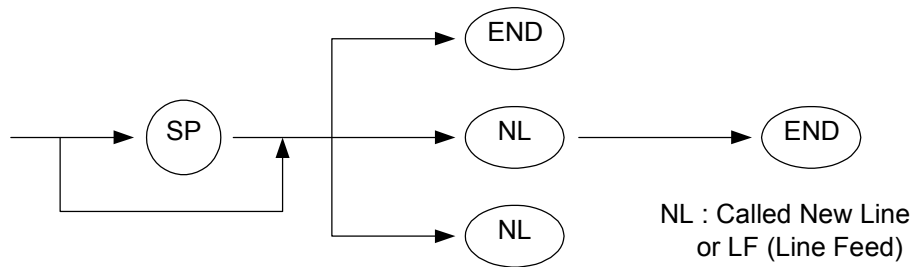
<Example> Send ("CF 1GHz;"):



PROGRAM MESSAGE:

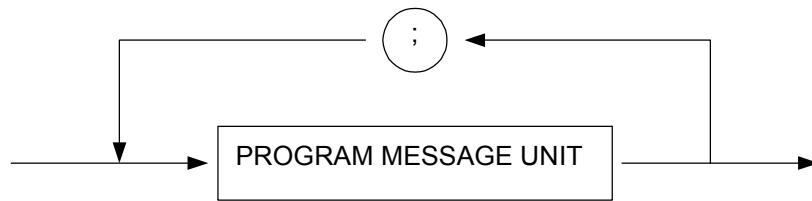
When the program message is transmitted from the controller to this instrument, the specified terminator is attached to the end of the program message to terminate its transmission.

Program message terminator



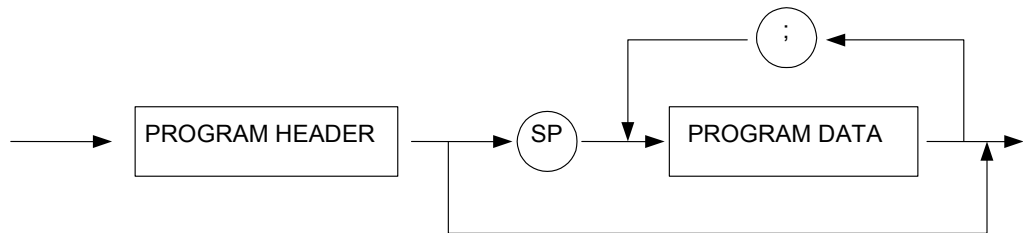
Carriage return (CR) is ignored and is not processed as a terminator.

Program message

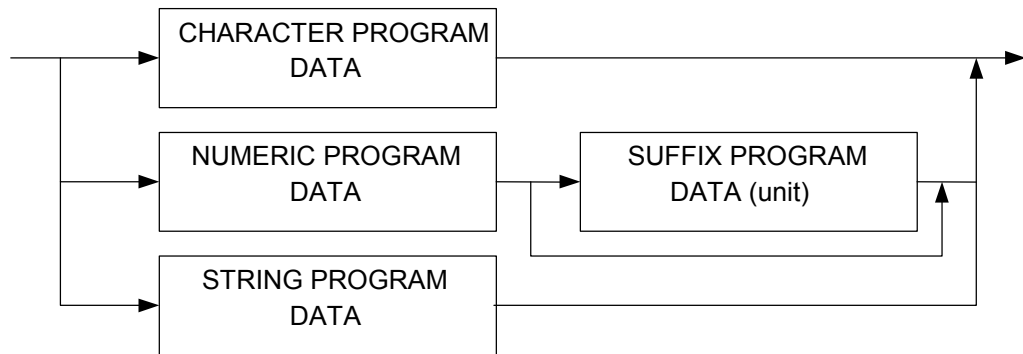


Multiple program message units can be output sequentially by separating them with a semicolon.

< Example > Send (“CF 1GHz; SP 500MHz;”)



A program message consists of a program header and program data. The program header of an IEEE488.2 common command always begins with an asterisk. The program header of a program query always ends with a question mark.



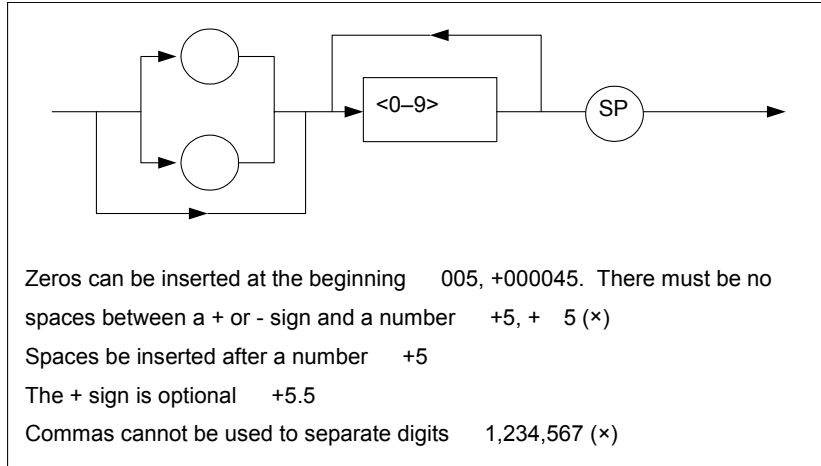
The characters of program data are a specific character string data consisting of the upper-case alphabetic characters from A to Z, numbers 0 to 9, #, *, ?.

< Example > Send (“ST AUTO;”); Sets Sweep Time to AUTO.

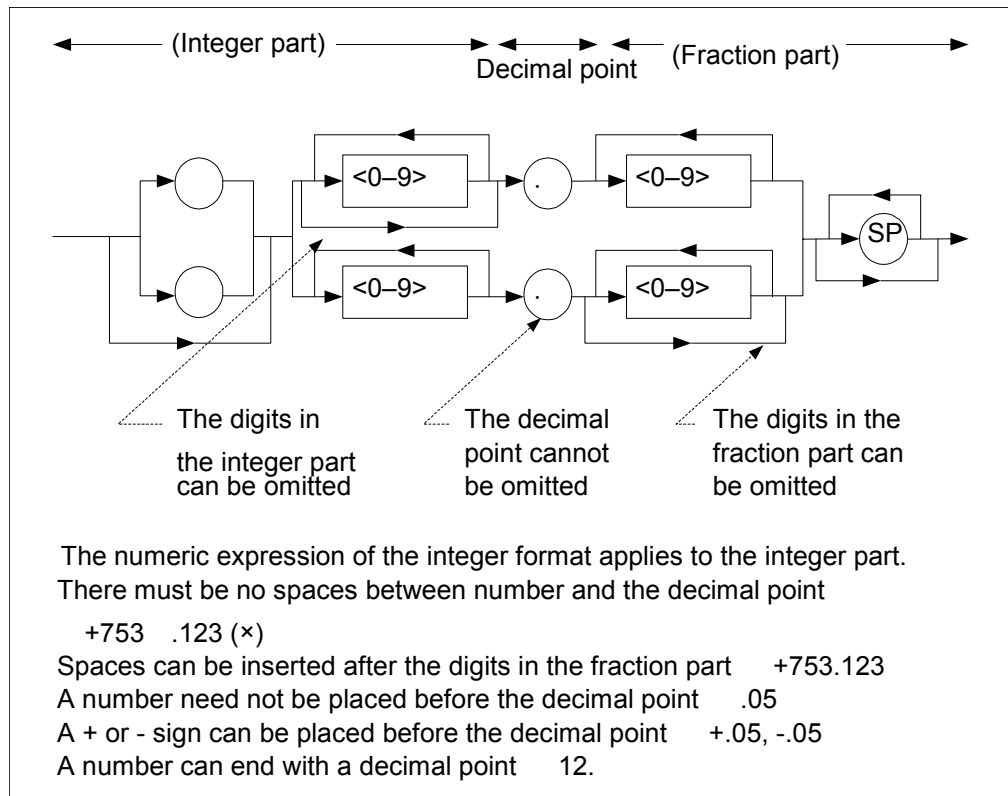
Numeric program data

Numeric program data has two types of format: integer format (NR1) and fixed-point format (NR2).

< Integer format (NR1) >



< Fixed-point format (NR2) >



Suffix of program data (unit)

The table below lists the suffixes used for the signal analyzer.

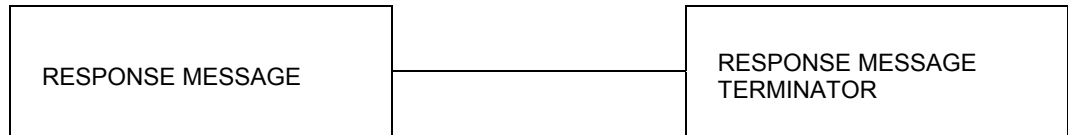
Table 3-1 System suffix codes

Classification	Unit	Suffix Code
Frequency	GHz	GHZ
	MHz	MHZ
	kHz	KHZ
	Hz	KHZ
	Default	HZ
Time	s	SEC S
	ms	MSEC MS
	μs	USEC US
	ns	NSEC NS
	ps	PSEC PS
	Default	SEC S
Level (dB system)	dB	DB
	dBm	DBM
	dBmV	DBMV
	dBμV	DBUV
	dBmA	DBMA
	dBμA	DBUA
	Default	Determined in conformance with the set scale unit.
Level (V system)	V	V
	mV	MV
	μV	UV
	nV	NV
	pV	PV
	Default	Determined in conformance with the set scale unit.
Level (W system)	W	W
	mW	MW
	μW	UW
	nW	NW
	pW	PW
	fW	FW

	Default	Determined in conformance with the set scale unit.
Level (A system)	A	A
	mA	MA
	μ A	UA
	nA	NA
	pA	PA
	Default	Determined in conformance with the set scale unit.

Response message format

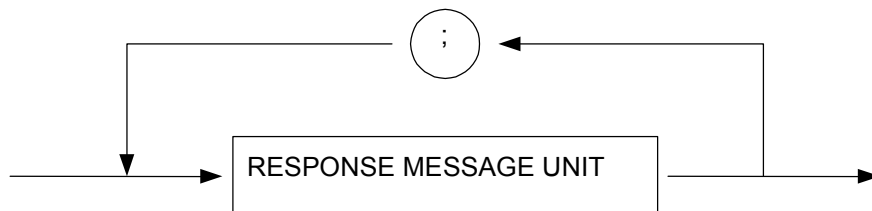
To transfer the response messages from this instrument to the controller using the “Receive” statement, the response message formats are defined as follows.



Response message terminator

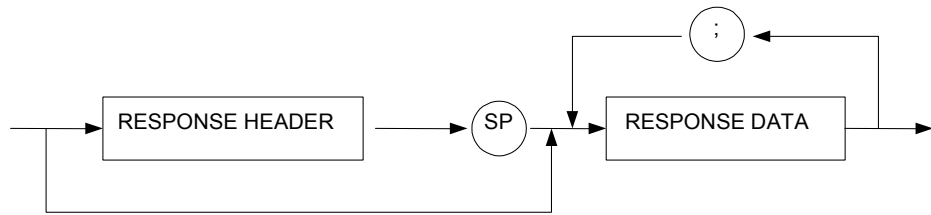


Response message

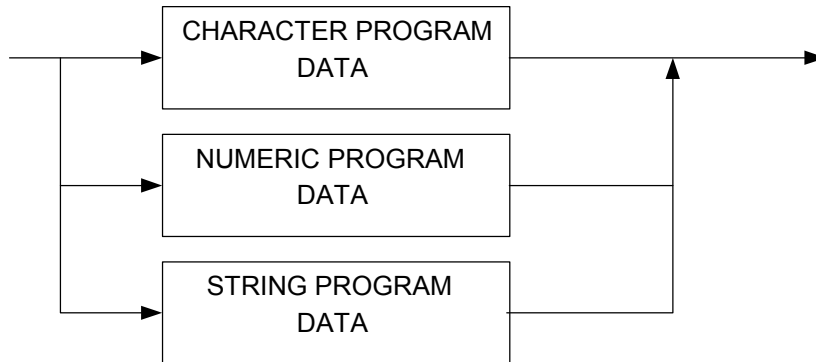


When a query is sent by the “Send” statement with one or more program queries, the response message also consists of one or more response message units.

Usual response message unit



Response data

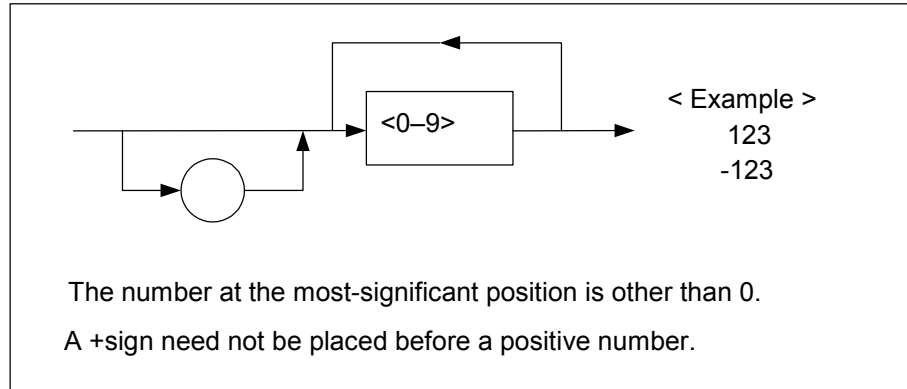


Character response data

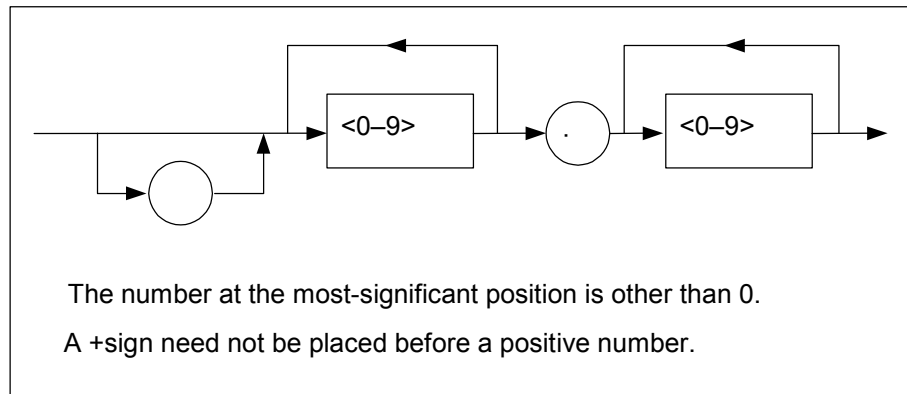
Character response data is specific character string data consisting of the upper-case alphabetic characters from A to Z, lower-case alphabetic characters from a to z, 0 to 9, and [,], dot[.], minus (-), comma (,).

Numeric response data

< Integer format (NR1) >



< Integer format (NR2) >



Chapter 4

DETAILED DESCRIPTION OF COMMANDS

Contents

General	4-8
SA command	4-8
Part 1 Spectrum mode.....	4-9
Amplitude	4-10
RL	4-10
AT	4-11
ATA	4-12
SD	4-13
STY	4-14
AU	4-15
RLO	4-16
IA	4-17
COAS	4-18
COA1 2 3 4	4-19
COAD	4-20
Auxiliary	4-21
AMS	4-21
FMS	4-22
AUDIOS	4-23
AUDIOLEV	4-24
Average	4-25
AVG	4-25
AVGC	4-26
AVGCL	4-27
AVGT	4-28
AVGTA	4-29
Bandwidth	4-30
AB	4-30
RB	4-31
RBA	4-32
VB	4-33
VBA	4-34
VBRB	4-35
SPRB	4-36
SPRBA	4-37
Calibration	4-38
CALALL	4-38
PCAL	4-39
PFCAL	4-40
ZNCCAL	4-41
LVLCAL	4-42
Couple	4-43
ACPL	4-43
DET	4-44
DETA	4-45
Display	4-46
FSCR	4-46
DL	4-47
DLS	4-48
TH	4-49
THS	4-50
TITLE	4-51

GRAT	4-52
ANN	4-53
WH	4-54
File	4-55
FREAD	4-55
FSAVE	4-56
FLOAD	4-57
FDEL	4-58
FCOPY	4-59
FRENAME	4-60
FMOVE	4-61
Frequency	4-62
CF	4-62
FA	4-63
FB	4-64
SS	4-65
SSA	4-66
FO	4-67
STR	4-68
In/Out	4-69
RFC	4-69
Limit Line	4-70
LLCS[1~2]	4-70
LLFC[1~2]	4-71
ALARM	4-72
LLAO	4-73
Marker	4-74
MS[1~9]	4-74
MM[1~9]	4-75
MF[1~9]	4-76
MP[1~9]	4-77
MFS[1~9]	4-78
MPS[1~9]	4-79
MFE[1~9]	4-80
MPE[1~9]	4-81
MFC[1~9]	4-82
MPC[1~9]	4-83
MFSP[1~9]	4-84
MPSP[1~9]	4-85
MA[1~9]	4-86
MT[1~9]	4-87
MR[1~9]	4-88
MTB	4-89
MAO	4-90
Marker Function	4-91
MFN[1~9]	4-91
MFNY	4-92
MFNX	4-93
MFNR	4-94
Marker Shift	4-95
MCF[1~9]	4-95
MSS[1~9]	4-96
MFA[1~9]	4-97
MFB[1~9]	4-98
MRL[1~9]	4-99
MDCF[1~9]	4-100
MDSS	4-101
MDSP	4-102
Measurement	4-103

MEA	4-103
MEAAVG	4-104
MEAAVGM	4-105
MEAAVGR	4-106
MEAAVGN	4-107
MEAO	4-108
Measurement - X dB Down	4-109
XDBP[1~2]	4-109
XDBL[1~2]	4-110
XDBR[1~2]	4-111
XDBRL[1~2]	4-112
XDBS	4-113
Measurement - Adjacent Channel Power	4-114
ACPMC	4-114
ACPAC	4-115
ACPCS	4-116
ACPN	4-117
ACPOUT	4-118
ACPOUTC	4-119
ACPOUTL[1~6]	4-120
ACPOUTR[1~6]	4-121
Measurement - Channel Power	4-122
CHPC	4-122
CHPOUT[C D]	4-123
Measurement - Occupied BandWidth	4-124
OBWP	4-124
Measurement - Harmonic Distortion	4-125
HDN	4-125
HF[1 2 3 4 5]	4-126
HA[1 2 3 4 5]	4-127
HDOUT	4-128
Measurement – Third Order Intermodulation	4-129
TOIOUT	4-129
TOIBLF	4-130
TOIBLL	4-131
TOIBLD	4-132
TOIBUF	4-133
TOIBUL	4-134
TOIBUD	4-135
TOIWCF	4-136
TOIWCL	4-137
TOIWCD	4-138
TOIWCI	4-139
TOI3LF	4-140
TOI3LL	4-141
TOI3LD	4-142
TOI3LI	4-143
TOI3UF	4-144
TOI3UL	4-145
TOI3UD	4-146
TOI3UI	4-147
Measurement – Spurious Emissions	4-148
SEOUT	4-148
Measurement – Spectrum Emission Mask	4-149
SEMMT	4-149
SEMOUT	4-150
:FETCh MEASure READ:SEMAsk	4-150
SEMTPWR	4-151
SEMTPSD	4-152

SEMLF1~6.....	4-153
SEMLPWR1~6.....	4-154
SEMLPSD1~6.....	4-155
SEMUF1~6.....	4-156
SEMUPWR1~6.....	4-157
SEMUPSD1~6.....	4-158
Measurement – Average Power (Burst Power).....	4-159
BPMT.....	4-159
BPTH.....	4-160
BPOUT.....	4-161
BPAVERP.....	4-162
BPMINP.....	4-163
BPMAXP.....	4-164
BPBL.....	4-165
Measurement - Total Power.....	4-166
TPOUT[C D].....	4-166
Peak Search.....	4-167
MPK[1~9].....	4-167
MPKN[1~9].....	4-168
MPKL[1~9].....	4-169
MPKR[1~9].....	4-170
MPKM.....	4-171
MPKP.....	4-172
MPKT.....	4-173
MPKC.....	4-174
MMPKN.....	4-175
MMPK.....	4-176
MMPKT.....	4-177
MPKE.....	4-178
MPKTH.....	4-179
MPKPA.....	4-180
Preset.....	4-181
PRST.....	4-181
Printer.....	4-182
HCOPY.....	4-182
Span.....	4-183
SP.....	4-183
FS.....	4-184
ZS.....	4-185
LS.....	4-186
ZI.....	4-187
ZO.....	4-188
Sweep.....	4-189
ST.....	4-189
STA.....	4-190
CO.....	4-191
SI.....	4-192
STAC.....	4-193
Trace.....	4-194
TRF[1~3].....	4-194
TRD.....	4-195
TDF.....	4-196
Trigger.....	4-197
TSO.....	4-197
TVL.....	4-198
TSL.....	4-199
TD.....	4-200
TDS.....	4-201
Tune.....	4-202

ATN.....	4-202
Tracking Generator (option).....	4-203
TGEN	4-203
TGNORM.....	4-205
PWRSWP.....	4-206
Part 2 Phase noise mode.....	4-207
Amplitude.....	4-208
RL.....	4-208
SD.....	4-209
IA.....	4-210
Display.....	4-211
FSCR.....	4-211
GRAT.....	4-212
ANN.....	4-213
WH.....	4-214
File.....	4-215
FREAD.....	4-215
FSAVE.....	4-216
FLOAD.....	4-217
FDEL.....	4-218
FCOPY.....	4-219
FRENAME.....	4-220
FMOVE.....	4-221
Frequency.....	4-222
CF.....	4-222
CFS.....	4-223
Marker.....	4-224
MS[1~9].....	4-224
MM[1~9].....	4-225
MF[1~9].....	4-226
MA[1~9].....	4-227
MT[1~9].....	4-228
MTB.....	4-229
MAO.....	4-230
Measurement.....	4-231
LP.....	4-231
AVG.....	4-232
AVGC.....	4-233
SM.....	4-234
FT.....	4-235
DTB.....	4-236
Mode.....	4-237
MODE.....	4-237
Preset.....	4-238
PRST.....	4-238
Printer.....	4-239
HCOPY.....	4-239
Span.....	4-240
FA.....	4-240
FB.....	4-241
Sweep.....	4-242
CO.....	4-242
SI.....	4-243
Trigger.....	4-244
TSO.....	4-244
Part 3 CCDF mode.....	4-245
Amplitude.....	4-245
IA.....	4-245
Bandwidth.....	4-246

RB.....	4-246
VB	4-247
VBA	4-248
VBRB	4-249
Display.....	4-250
FSCR.....	4-250
GRAT	4-251
WH	4-252
SREF	4-253
GAUS	4-254
File.....	4-255
FREAD	4-255
FSAVE	4-256
FLOAD.....	4-257
FCOPY	4-258
FRENAME.....	4-259
FMOVE.....	4-260
Frequency	4-261
CF.....	4-261
FO.....	4-262
Marker	4-263
MS[1~9]	4-263
MM[1~9].....	4-264
MA[1~9].....	4-265
MP[1~9]	4-266
MT[1~9].....	4-267
MAO.....	4-268
Measurement	4-269
MEA	4-269
MEAO	4-270
Measurement — CCDF	4-271
CCDFN.....	4-271
CCDFOUT[PWR[PER]]	4-272
CCDFPER10	4-273
CCDFPER1	4-274
CCDFPER01	4-275
CCDFPER001	4-276
CCDFPER0001	4-277
CCDFPER00001	4-278
CCDFCRE.....	4-279
Preset	4-280
PRST	4-280
Printer	4-281
HCOPY	4-281
Span.....	4-282
SD.....	4-282
Sweep	4-283
CO	4-283
SI	4-284
Trigger.....	4-285
TSO	4-285
Part 4 GPIB common commands.....	4-286
*CLS.....	4-287
*ESE.....	4-288
*ESR?.....	4-289
*IDN?.....	4-290
*OPC	4-291
*OPC?	4-292
*RST.....	4-293

*SRE.....	4-294
*STB?.....	4-295
GPIB Common Command - Others.....	4-296
ESE2.....	4-296
ESR2?.....	4-297
ERR.....	4-298

General

This section gives detailed descriptions of the device messages for the spectrum analyzer in functional order. The following example shows the command format.

Note that ' ' = 'blank' throughout this document.

SA command

SCPI command

	Command Name
Function	The explanation of the command.
Remote Command	SA Command sw SA Command f SA Command? SCPI Command sw SCPI Command f SCPI Command?
Response Message	sw or f (Depending on command)
Value of f	Range of sw or f (Depending on command)
Suffix code	Unit of f (Depending on command)
Initial setting	Initial value for SA System
Example	SA Command sw; SA Command f; SA Command?; SCPI Command sw; SCPI Command f; SCPI Command?;

Part 1 Spectrum mode

This section gives detailed descriptions of commands for the instrument when it is used for spectrum analysis.

Amplitude

RL

:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel

	Reference Level
Function	Sets the reference level value.
Remote Command	RL f RL? :DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel f :DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel?
Response Message	Reference Level (dBm)
Value of f	-170 dBm to 30 dBm (Step : 0.01 dBm)
Suffix code	None : dBm DBM : dBm DBMV : dBmV DBUV : dBuV DBMA : dBmA DBUA : dBuA V : V MV : mV (10 ⁻³ V) UV : uV (10 ⁻⁶ V) NV : nV (10 ⁻⁹ V) PV : pV (10 ⁻¹² V) W : W MW : mW (10 ⁻³ W) UW : uW (10 ⁻⁶ W) NW : nW (10 ⁻⁹ W) PW : pW (10 ⁻¹² W) A : A MA : mA (10 ⁻³ A) UA : uA (10 ⁻⁶ A) NA : nA (10 ⁻⁹ A) PA : pA (10 ⁻¹² A)
Initial setting	0 dBm
Example	RL 10; RL 30DBM; RL ?; DISP:WIND:TRAC:Y:RLEV 10; DISP:WIND:TRAC:Y:RLEV 30DBM; DISP:WIND:TRAC:Y:RLEV?;

AT

[[:SENSe]:POWer[:RF]:ATTenuation

	Attenuation
Function	Sets the amount of attenuation for the input attenuator.
Remote Command	AT f AT? [:SENSe]:POWer[:RF]:ATTenuation f [:SENSe]:POWer[:RF]:ATTenuation?
Response Message	the amount of attenuation (dB)
Value of f	0 dB to 55 dB (Step : 5 dB)
Suffix code	None : dB
DB	: dB
Initial setting	10 dB
Example	AT 10; AT 10DB; AT?; POW:ATT 10; POW:ATT 10DB; POW:ATT?;

ATA

[[:SENSe]:POWer[:RF]:ATTenuation:AUTO

	Attenuation Auto
Function	Sets input attenuation mode to the auto mode or the manual mode. If the auto mode is selected, the amount of attenuation is adjusted automatically. If the manual mode is selected, this affects Reference Level.
Remote Command	ATA n ATA sw ATA? [:SENSe]:POWer[:RF]:ATTenuation:AUTO n [:SENSe]:POWer[:RF]:ATTenuation:AUTO sw [:SENSe]:POWer[:RF]:ATTenuation:AUTO?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	1
Example	ATA 1; ATA ON; ATA?; POW:ATT:AUTO 1; POW:ATT:AUTO ON; POW:ATT:AUTO?;

SD

:DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:PDIVision

	Scale/Divide
Function	Sets the scale/divide value.
Remote Command	SD f SD? :DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:PDIVision f :DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:PDIVision?
Response Message	Scale/Divide (dB/div)
Value of f	0.1 dB to 1 dB (step : 0.1 dB) 1dB to 20dB (step : 1dB)
Suffix code	None : dB/div DB : dB/div
Initial setting	10 dB/div
Example	SD 5; SD 10DB; SD?; DISP:LPL:WIND:TRAC:Y:PDIV 5; DISP:LPL:WIND:TRAC:Y:PDIV 10DB; DISP:LPL:WIND:TRAC:Y:PDIV?;

STY

:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing

	Scale Type
Function	Sets the scale type to the linear or the logarithmic mode.
Remote Command	STY sw STY? :DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing sw :DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing?
Response Message	LIN : Linear Mode LOG : Logarithmic Mode
Value of sw	LINear : Linear Mode LOGarithmic : Logarithmic Mode
Initial setting	LOG
Example	STY LOG; STY?; DISP:WIND:TRAC:Y:SPAC LOG; DISp:WIND:TRAC:Y:SPAC?

AU

:UNIT:POWer

	Amplitude Units
Function	Sets the absolute amplitude units for the input signal display.
Remote Command	AU sw AU? :UNIT:POWer sw :UNIT:POWer?
Response Message	DBM : dBm DBMV : dBmV DBUV : dBuV DBMA : dBmA DBUA : dBuA V : V W : W A : A
Value of sw	None : dBm DBM : dBm DBMV : dBmV DBUV : dBuV DBMA : dBmA DBUA : dBuA V : V W : W A : A
Initial setting	DBM
Example	AU DBM; AU? UNIT:POW DBM; UNIT:POW?;

RLO

:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet

	Reference Level Offset
Function	Sets the amount of reference level offset.
Remote Command	RLO f RLO? :DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet f :DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet?
Response Message	Reference Level Offset (dB)
Value of f	-300.00 dB to 300.00 dB (Step : 0.1 dB)
Suffix code	NONE : dB DB : dB
Initial setting	0 dB
Example	RLO 10; RLO -200.5DB; RLO?; DISP:WIND:TRAC:Y:RLEV:OFFS 10; DISP:WIND:TRAC:Y:RLEV:OFFS -200.5DB; DISP:WIND:TRAC:Y:RLEV:OFFS?;

IA

[[:SENSE]:POWER[:RF]:GAIN[:STATE]

	Internal Amplifier
Function	Activates the internal amplifier.
Remote Command	IA n IA sw IA? [:SENSE]:POWER[:RF]:GAIN[:STATE] n [:SENSE]:POWER[:RF]:GAIN[:STATE] sw [:SENSE]:POWER[:RF]:GAIN[:STATE]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	IA 1; IA ON; IA?; POW:GAIN 1; POW:GAIN ON; POW:GAIN?;

COAS

[[:SENSe]:CORRection:CSET:ALL[:STATe]

	All Correction State	
Function	Selects all correction (1:Antenna, 2:Cable, 3:Other, 4:User) states to ON or OFF.	
Remote Command	COAS n COAS sw COAS? [:SENSe]:CORRection:CSET:ALL[:STATe] n [:SENSe]:CORRection:CSET:ALL[:STATe] sw [:SENSe]:CORRection:CSET:ALL[:STATe]?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	COAS 1; COAS ON; COAS?; CORR:CSET:ALL 1; CORR:CSET:ALL ON; CORR:CSET:ALL?;	

COA1|2|3|4

[[:SENSe]:CORRection:CSET1|2|3|4[:STATe]

	Correction State	
Function	Sets individual correction states to ON or OFF. (1:Antenna, 2:Cable, 3:Other, 4:User)	
Remote Command	COA1 2 3 4 n COA1 2 3 4sw COA1 2 3 4? [:SENSe]:CORRection:CSET1 2 3 4[:STATe] n [:SENSe]:CORRection:CSET1 2 3 4[:STATe] sw [:SENSe]:CORRection:CSET1 2 3 4[:STATe]?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	COA1 1; COA2 ON; COA3?; CORR:CSET1 1; CORR:CSET2 ON; CORR:CSET3?;	

COAD

[[:SENSe]:CORRection:CSET:ALL:DELe]

	All Correction OFF
Function	Sets all correction states and data to OFF.
Remote Command	COAD [:SENSe]:CORRection:CSET:ALL:DELe?
Example	COAD; CORR:CSET:ALL:DEL;

Auxiliary

AMS

	AM lation State	
Function	Activates AM demodulation function.	
Remote Command	AMS n AMS sw AMS?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
	Example	AMS 1;
		AMS ON;
		AMS?;

FMS

	FM Demodulation State	
Function	Activates FM demodulation function.	
Remote Command	FMS n	
	FMS sw	
	FMS?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	FMS 1;	
	FMS ON;	
	FMS?;	

AUDIOS

	Audio State	
Function	Sets audio state on or off	
Remote Command	AUDIOS n AUDIOS sw AUDIOS?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	AUDIOS 1; AUDIOS ON; AUDIOS?;	

AUDIOLEV

	Audio Level
Function	Adjusts the level of the audio.
Remote Command	AUDIOLEV n AUDIOLEV?
Response Message	Audio Level
Value of n	0 to 100 (step : 1)
Initial setting	3
Example	AUDIOLEV 5; AUDIOLEV?;

Average

AVG

[[:SENSe]:AVERage[:STATe]]

	Average
Function	Turns the trace average on or off. Depends on the condition of the average count and the average mode.
Remote Command	AVG n AVG sw AVG? [:SENSe]:AVERage[:STATe] n [:SENSe]:AVERage[:STATe] sw [:SENSe]:AVERage[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	AVG 1; AVG ON; AVG?; AVER 1; AVER ON; AVER?;

AVGC

[[:SENSe]:AVERage:COUNT

	Average Count
Function	Sets the averaging count.
Remote Command	AVGC n AVGC? [:SENSe]:AVERage:COUNT n [:SENSe]:AVERage:COUNT?
Response Message	Average count
Value of n	1 to 256
Initial setting	10
Example	AVGC 30; AVGC?; AVER:COUN 30; AVER:COUN?;

AVGCL

[[:SENSe]:AVERage:CLEar

	Average Reset
Function	Resets the trace averaging.
Remote Command	AVGCL [:SENSe]:AVERage:CLEar
Example	AVGCL; AVER:CLE;

AVGT

[[:SENSe]:AVERage:TYPE

	Average/VBW Type
Function	Selects the average type in averaging mode.
Remote Command	AVGT sw AVGT? [:SENSe]:AVERage:TYPE sw [:SENSe]:AVERage:TYPE?
Response Message	RMS : Power (RMS) LOG : Log-Power (video) SCAL : Voltage
Value of sw	RMS : Sets Power (RMS) LOG : Sets Log-Power (video) SCALar : Sets Voltage
Initial setting	LOG
Example	AVGT RMS; AVGC?; AVER:TYPE RMS; AVER:TYPE?;

AVGTA

[[:SENSe]:AVERage:TYPE:AUTO

	Average/VBW Type Auto	
Function	Selects the average type auto.	
Remote Command	AVGTA n AVGTA sw AVGTA? [:SENSe]:AVERage:TYPE:AUTO n [:SENSe]:AVERage:TYPE:AUTO sw [:SENSe]:AVERage:TYPE:AUTO?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	AVGTA 1; AVGTA ON; AVGTA?; AVER:TYPE:AUTO 1; AVER:TYPE:AUTO ON; AVER:TYPE:AUTO?;	

Bandwidth

AB

[[:SENSe]:BANDwidth|BWIDth:AUTO

	Auto Bandwidth
Function	Sets Bandwidth menu to all AUTO.
Remote Command	AB
	[[:SENSe]:BANDwidth BWIDth:AUTO
Example	AB; BAND:AUTO;

RB

[[:SENSe]:BANDwidth|BWIDth[:RESolution]

	Resolution Bandwidth
Function	Sets the RBW value.
Remote Command	RB f RB? [:SENSe]:BANDwidth BWIDth[:RESolution] f [:SENSe]:BANDwidth BWIDth[:RESolution]?
Response Message	Resolution Bandwidth ()
Value of f	1 Hz to 5 MHz (Step : 1, 2, 3, 5)
Suffix code f	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	5 MHz
Example	RB 1000; RB 3KHZ; RB? BAND 1000; BAND 3KHZ; BAND?;

RBA

[[:SENSe]:BANDwidth|BWIDth[:RESolution]:AUTO

	Resolution Bandwidth Auto	
Function	Sets the RBW mode to auto or manual mode.	
Remote Command	RBA n RBA sw RBA? [:SENSe]:BANDwidth BWIDth[:RESolution]:AUTO n [:SENSe]:BANDwidth BWIDth[:RESolution]:AUTO sw [:SENSe]:BANDwidth BWIDth[:RESolution]:AUTO?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	RBA 1; RBA ON; RBA? BAND:AUTO 1; BAND:AUTO ON; BAND:AUTO?;	

VB

[[:SENSE]:BANDwidth|BWIDth:VIDeo

	Video Bandwidth
Function	Sets the VBW value.
Remote Command	VB f VB? [:SENSE]:BANDwidth BWIDth:VIDeo f [:SENSE]:BANDwidth BWIDth:VIDeo?
Response Message	Video Bandwidth (Hz)
Value of f	1 Hz to 3 MHz (Step : 1, 2, 3, 5)
Suffix code f	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	3 MHz
Example	VB 1000; VB 3KHZ; VB? BAND:VID 1000; BAND:VID 3KHZ; BAND:VID?;

VBA

[[:SENSe]:BANDwidth|BWIDth:VIDeo:AUTO

	Video Bandwidth Auto	
Function	Sets the VBW mode to auto or manual mode.	
Remote Command	VBA n VBA sw VBA? [:SENSe]:BANDwidth BWIDth[:RESolution]:VIDeo:AUTO n [:SENSe]:BANDwidth BWIDth[:RESolution]:VIDeo:AUTO sw [:SENSe]:BANDwidth BWIDth[:RESolution]:VIDeo:AUTO?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	VBA 1; VBA ON; VBA? BAND:VID:AUTO 1; BAND:VID:AUTO ON; BAND:VID:AUTO?;	

VBRB

[[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio

	VBW/RBW
Function	Selects the ratio between VBW and RBW.
Remote Command	VBRB f VBRB? [:SENSe]:BANDwidth BWIDth:VIDeo:RATio f [:SENSe]:BANDwidth BWIDth:VIDeo:RATio?
Response Message	Ratio of VBW/RBW
Value of f	0.000001 to 2000000
Initial setting	1
Example	VBRB 1; VBRB? BAND:VID:RAT 1; BAND:VID:RAT?;

SPRB

[[:SENSe]:FREQuency:SPAN:BANDwidth|BWIDth:VIDeo:RATio

	Span/RBW
Function	Selects the ratio between Span and RBW.
Remote Command	SPRB f SPRB? [:SENSe]:FREQuency:SPAN:BANDwidth BWIDth:VIDeo: RATio f [:SENSe]:FREQuency:SPAN:BANDwidth BWIDth:VIDeo: RATio?
Response Message	Ratio of Span/RBW
Value of f	Ratio of Span/RBW
Initial setting	106
Example	SPRB 106; SPRB? FREQ:SPAN:BAND:VID:RAT 106; FREQ:SPAN:BAND:VID:RAT?;

SPRBA

[[:SENSe]:FREQuency:SPAN:BANDwidth|BWIDth:VIDeo:RATio:AUTO

	Ratio Auto of Span/RBW	
Function	Selects the ratio mode between Span and RBW.	
Remote Command	SPRBA n	
	SPRBA sw	
	SPRBA?	
Deo:AUTO n	[:SENSe]:FREQuency:SPAN:BANDwidth BWIDth[:RESolution]:VI	
Deo:AUTO sw	[:SENSe]:FREQuency:SPAN:BANDwidth BWIDth[:RESolution]:VI	
Deo:AUTO?	[:SENSe]:FREQuency:SPAN:BANDwidth BWIDth[:RESolution]:VI	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	SPRBA 1;	
	SPRBA ON;	
	SPRBA?	
	FREQ:SPAN:BAND:VID:AUTO 1;	
	FREQ:SPAN:BAND:VID:AUTO ON;	
	FREQ:SPAN:BAND:VID:AUTO?;	

Calibration

CALALL

	All Calibrations
Function	Executes all calibrations.
Remote Command	CALALL
Example	CALALL;

PCAL

	Periodic Temperature Calibrations	
Function	Initiates periodic temperature calibration execution.	
Remote Command	PCAL n PCAL sw PCAL?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	PCAL 1; PCAL ON; PCAL?;	

PFCAL

	Pre-Filter Calibration
Function	Executes Pre-Filter calibration.
Remote Command	PFCAL
Example	PFCAL;

ZNCCAL

	ZNC Calibration
Function	Executes ZNC Calibration.
Remote Command	ZNCCAL
Example	ZNCCAL;

LVLCAL

	Level Calibration
Function	Executes level calibration.
Remote Command	LVLCAL
Example	LVLCAL;

Couple

ACPL

:COUPle

	Auto Coupled
Function	Sets Couple menu to All AUTO.
Remote Command	ACPL :COUPle
Example	ACPL; COUP;

DET

[[:SENSe]:DETector[:FUNction]]

	Detection Mode
Function	Selects the detection mode.
Remote Command	DET sw DET? [:SENSe]:DETector[:FUNction] sw [:SENSe]:DETector[:FUNction]?
Response Message	NORM : Normal AVER : Average POS : Positive SAMP : Sample NEG : Negative
Value of sw	NORMal : Normal AVERage : Average POSitive : Positive SAMPle : Sample NEGative : Negative
Initial setting	NORM
Example	DET NORM; DET?;

DETA

[[:SENSe]:DETECTOR:AUTO

	Detection Mode Auto	
Function	Sets the detection mode to AUTO or MANUAL.	
Remote Command	DETA n	
	DETA sw	
	DETA?	
	[:SENSe]:DETECTOR:AUTO n	
	[:SENSe]:DETECTOR:AUTO sw	
	[:SENSe]:DETECTOR:AUTO?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	DETA 1;	
	DETA ON;	
	DETA?;	
	DET:AUTO 1;	
	DET:AUTO ON;	
	DET:AUTO?;	

Display

FSCR

:DISPlay:FSCR[en[:STATe]]

	Full Screen
Function	Sets the full screen mode.
Remote Command	FSCR n FSCR sw FSCR? :DISPlay:FSCR[en[:STATe]] n :DISPlay:FSCR[en[:STATe]] sw :DISPlay:FSCR[en[:STATe]]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	1
Example	FSCR 1; FSCR ON; FSCR? DISP:FSCR 1; DISP:FSCR ON; DISP:FSCR?;

DL

:DISPlay:WINDow:TRACe:Y:DLINe

	Display Line Amplitude
Function	Sets the amplitude of the Display Line.
Remote Command	DL f DL? :DISPlay:WINDow:TRACe:Y:DLINe f :DISPlay:WINDow:TRACe:Y:DLINe?
Response Message	Amplitude of Display Line
Value of f	Reference Level to (Reference Level-10*Scale/DIV) (Step : 0.01 dBm)
Suffix code	None : dBm DBM : dBm DBMV : dBmV DBUV : dBuV DBMA : dBmA DBUA : dBuA V : V MV : mV (10 ⁻³ V) UV : uV (10 ⁻⁶ V) NV : nV (10 ⁻⁹ V) PV : pV (10 ⁻¹² V) W : W MW : mW (10 ⁻³ W) UW : uW (10 ⁻⁶ W) NW : nW (10 ⁻⁹ W) PW : pW (10 ⁻¹² W) FW : fW (10 ⁻¹⁵ W) A : A MA : mA (10 ⁻³ A) UA : uA (10 ⁻⁶ A) NA : nA (10 ⁻⁹ A) PA : pA (10 ⁻¹² A)
Initial setting	Reference Level
Example	DL 0; DL -50DBM; DL?; DISP:WIND:TRAC:Y:DLIN 0; DISP:WIND:TRAC:Y:DLIN -50DBM; DISP:WIND:TRAC:Y:DLIN?;

DLS

:DISPlay:WINDow:TRACe:Y:DLINe:STATe

	Display Line State
Function	Turns Display Line ON or OFF.
Remote Command	DLS n DLS sw DLS? :DISPlay:WINDow:TRACe:Y:DLINe:STATe n :DISPlay:WINDow:TRACe:Y:DLINe:STATe sw :DISPlay:WINDow:TRACe:Y:DLINe:STATe?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	DLS 1; DLS ON; DLS? DISP:WIND:TRAC:Y:DLIN:STAT 1; DISP:WIND:TRAC:Y:DLIN:STAT ON; DISP:WIND:TRAC:Y:DLIN:STAT?;

TH

:DISPlay:WINDow:TRACe:Y:TLINe

	Threshold Line Amplitude
Function	Sets the threshold level and ignores data below this value.
Remote Command	TH f TH? :DISPlay:WINDow:TRACe:Y:TLINe f :DISPlay:WINDow:TRACe:Y:TLINe?
Response Message	Threshold line amplitude
Value of f	Reference Level to Reference Level-10*Scale/DIV (Step : 0.01 dBm)
Suffix code	None : dBm DBM : dBm DBMV : dBmV DBUV : dBuV DBMA : dBmA DBUA : dBuA V : V MV : mV (10 ⁻³ V) UV : uV (10 ⁻⁶ V) NV : nV (10 ⁻⁹ V) PV : pV (10 ⁻¹² V) W : W MW : mW (10 ⁻³ W) UW : uW (10 ⁻⁶ W) NW : nW (10 ⁻⁹ W) PW : pW (10 ⁻¹² W) FW : fW (10 ⁻¹⁵ W) A : A MA : mA (10 ⁻³ A) UA : uA (10 ⁻⁶ A) NA : nA (10 ⁻⁹ A) PA : pA (10 ⁻¹² A)
Initial setting	Reference Level-10*Scale/Div
Example	TH 0; TH -50DBM; TH?; DISP:WIND:TRAC:Y:TLIN 0; DISP:WIND:TRAC:Y:TLIN -50DBM; DISP:WIND:TRAC:Y:TLIN?;

THS

:DISPlay:WINDow:TRACe:Y:TLINe:STATe

	Threshold Line State	
Function	Turns Threshold Line ON or OFF.	
Remote Command	THS n	
	THS sw	
	THS?	
	:DISPlay:WINDow:TRACe:Y:TLINe:STATe n	
	:DISPlay:WINDow:TRACe:Y:TLINe:STATe sw	
	:DISPlay:WINDow:TRACe:Y:TLINe:STATe?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	THS 1;	
	THS ON;	
	THS?	
	DISP:WIND:TRAC:Y:TLIN:STAT 1;	
	DISP:WIND:TRAC:Y:TLIN:STAT ON;	
	DISP:WIND:TRAC:Y:TLIN:STAT?;	

TITLE

:DISPlay:ANNotation:TITLe:DATA

	Screen Title
Function	Places the character data in the title area of the display. Available characters are alphanumeric.
Remote Command	TITLE text TITLE? :DISPlay:ANNotation:TITLe:DATA text :DISPlay:ANNotation:TITLe:DATA?
Response Message	Title
Value of text	String
Initial setting	Noname
Example	TITLE SignalAnalyzer; TITLE?; DISP:ANN:TITL:DATA SignalAnalyzer; DISP:ANN:TITL:DATA?;

GRAT

:DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe]

	Graticule
Function	Sets the display graticule to Type1 or Type2 or OFF.
Remote Command	GRAT sw GRAT? :DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe] sw :DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe]?
Response Message	TYPE1 : Type1 TYPE2 : Type2 OFF : OFF
Value of sw	TYPE1 : Type1 TYPE2 : Type2 OFF : OFF
Initial setting	TYPE1
Example	GRAT TYPE1; GRAT? DISP:WIND:TRAC:Y:GRAT:GRID TYPE1; DISP:WIND:TRAC:Y:GRAT:GRID?;

ANN

:DISPlay:WINDow:ANNotation[:ALL]

	Annotation
Function	Turns the annotation on or off
Remote Command	ANN n ANN sw ANN? :DISPlay:WINDow:ANNotation[:ALL] n :DISPlay:WINDow:ANNotation[:ALL] sw :DISPlay:WINDow:ANNotation[:ALL]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	1
Example	ANN 1; ANN ON; ANN? DISP:WIND:ANN 1; DISP:WIND:ANN ON; DISP:WIND:ANN?;

WH

:DISPlay:WINDow:WHITe

	White Mode	
Function	Turns the white mode ON or OFF.	
Remote Command	WH n WH sw WH? :DISPlay:WINDow:WHITe n :DISPlay:WINDow:WHITe sw :DISPlay:WINDow:WHITe?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	WH 1; WH ON; WH? DISP:WIND:WHIT 1; DISP:WIND:WHIT ON; DISP:WIND:WHIT?;	

File

FREAD

:MMEMory:CATalog

	File Read
Function	Reads files in the selected folder.
Remote Command	FREAD? 'file_folder' :MMEMory:CATalog? 'file_folder'
Value of file_folder	File Folder
Response Message	File Name,,File Size.
Example	FREAD? 'C:'; FREAD? 'D:\Temp'; MMEM:CAT? 'C:'; MMEM:CAT? 'D:\Temp';

FSAVE

:MMEMory:STORe

	File Save
Function	Saves the file, type defined by the extension.
Remote Command	FSAVE 'file_name' :MMEMory:STORe 'file_name'
Value of file_name	File Path + File Name
Supported Extension	sts : Status trc : Trace lim : Limit bmp : Bitmap jpg : jpeg png : png ant : Antenna cbl : Cable oth : Other usr : User
Example	FSAVE 'C:\demo1.sts'; FSAVE 'C:\demo2.trc'; MMEM:STRO 'C:\demo1.sts'; MMEM:STRO 'C:\demo2.trc';

FLOAD

:MMEMory:LOAD

	File Load
Function	Loads the selected file.
Remote Command	FLOAD 'file_name' :MMEMory:LOAD 'file_name'
Value of file_name	File Path + File Name
Supported Extension	sts : Status trc : Trace lim : Limit ant : Antenna cbl : Cable oth : Other usr : User
Example	FLOAD 'C:\demo1.sts'; FLOAD 'C:\demo2.trc'; MMEM:LOAD 'C:\demo1.sts'; MMEM:LOAD 'C:\demo2.trc';

FDEL

:MMEMory:DELeTe

	File Delete
Function	Deletes the selected file.
Remote Command	FDEL 'file_name' :MMEMory:DELeTe 'file_name'
Value of file_name	File Path + File Name
Example	FDEL 'C:\demo1.sts'; FDEL 'C:\demo2.trc'; MMEM:DEL 'C:\demo1.sts'; MMEM:DEL 'C:\demo2.trc';

FCOPY

:MMEMory:COPY

	File Copy
Function	Copies the selected file.
Remote Command	FCOPY 'src_file_name', 'dest_file_name' :MMEMory:COPY 'src_file_name', 'dest_file_name'
Value of src_file_name, dest_file_name	File Path + File Name
Example	FCOPY 'C:\demo1.sts', 'D:\demo1.sts'; FCOPY 'C:\demo2.trc', 'D:\demo2.trc'; MMEM:COPY 'C:\demo1.sts', 'D:\demo1.sts'; MMEM:COPY 'C:\demo2.trc', 'D:\demo2.trc';

FRENAME

:MMEMory:MOVE

	File Rename
Function	Renames the selected file.
Remote Command	FRENAME 'src_file_name','dest_file_name' :MMEMory:MOVE 'src_file_name','dest_file_name'
Value of src_file_name, dest_file_name	File Path + File Name
Example	FRENAME 'C:\demo1.sts','C:\demo1_1.sts'; FRENAME 'C:\demo2.trc','C:\demo2_1.trc'; MMEM:MOVE 'C:\demo1.sts','C:\demo1_1.sts'; MMEM:MOVE 'C:\demo2.trc','C:\demo2_1.trc';

FMOVE

MMEMory:DATA

	File Move
Function	Sends or receives binary data of the selected file. The maximum size of the sent file is 2 Mbyte, and the maximum size of the received file is 30 Mbyte.
Remote Command	FMOVE 'file_name',definite_length_block FMOVE? 'file_name' MMEMory:DATA 'file_name',definite_length_block MMEMory:DATA? 'file_name'
Value of file_name	File Path + File Name
Value of definite_length_block	# + number of file size + file size + file data
Example	FMOVE 'C:\Sended_Sample.txt',#14abcd; cf) #+1+4+abcd FMOVE? 'C:\Received_Sample.txt'; MMEM:DATA 'C:\ Sended_Sample.txt',#14abcd; MMEM:DATA? 'C:\ Received_Sample.txt';

Frequency

CF

[[:SENSe]:FREQuency:CENTer

	Center Frequency
Function	Sets the center frequency. If the center frequency is set too near the boundary frequency, the span value is not satisfied. In this case, the span value is adjusted automatically.
Remote Command	CF f CF sw CF? [:SENSe]:FREQuency:CENTer f [:SENSe]:FREQuency:CENTer sw [:SENSe]:FREQuency:CENTer?
Response Message	Center Frequency () (Range : 3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz)
Value of f	3 Hz to 3.0 GHz / / 3 Hz to 13.2 GHz / / 3 Hz to 26.5 GHz /
Value of sw	UP : Center Freq + CF Step DOWN : Center Freq - CF Step
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	1.5 GHz / 6.6 GHz / 13.25 GHz
Example	CF 123456; CF 50MHZ; CF UP; CF?; FREQ:CEN7T 123456; FREQ:CENT 50MHZ; FREQ:CENT UP; FREQ:CENT?;

FA

[[:SENSe]:FREQuency:STARt

	Start Frequency
Function	Sets the start frequency. If the start frequency is set too near the boundary frequency, the span value is not satisfied. In this case, the span value is adjusted automatically
Remote Command	FA f FA? [:SENSe]:FREQuency:STARt f [:SENSe]:FREQuency:STARt?
Response Message	Start Frequency () (Range : 3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz)
Value of f	3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	3 Hz
Example	FA 123456; FA 50MHZ; FA?; FREQ:STAR 123456; FREQ:STAR 50MHZ; FREQ:STAR?;

FB

[[:SENSe]:FREQuency:STOP

	Stop Frequency
Function	Sets the stop frequency. If the stop frequency is set too near the boundary frequency, the span value is not satisfied. In this case, the span value is adjusted automatically.
Remote Command	FB f FB? [:SENSe]:FREQuency:STOP f [:SENSe]:FREQuency:STOP?
Response Message	Stop Frequency () (Range : 3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz)
Value of f	3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	3.0 GHz / 13.2 GHz / 26.5 GHz
Example	FB 123456; FB 50MHZ; FB?; FREQ:STOP 123456; FREQ:STOP 50MHZ; FREQ:STOP?;

SS

[[:SENSe]:FREQuency:CENTer:STEP[:INCRement]

	CF Step
Function	Sets the center frequency step size.
Remote Command	SS f SS? [:SENSe]:FREQuency:CENTer:STEP[:INCRement] f [:SENSe]:FREQuency:CENTer:STEP[:INCRement]?
Response Message	CF Step (Hz)
Value of f	1 Hz to 3.0 GHz / 13.2 GHz / 26.5 GHz
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	10% of Span
Example	SS 123456; SS 50MHZ; SS?; FREQ:CENT:STEP 123456; FREQ:CENT:STEP 50MHZ; FREQ:CENT:STEP?;

SSA

[[:SENSe]:FREQuency:CENTer:STEP:AUTO

	CF Step Auto
Function	Sets the cf step to the auto or the manual mode.
Remote Command	SSA n SSA sw SSA? [:SENSe]:FREQuency:CENTer:STEP:AUTO n [:SENSe]:FREQuency:CENTer:STEP:AUTO sw [:SENSe]:FREQuency:CENTer:STEP:AUTO?
Response Message	1 : Auto 0 : Manual
Value of n	1 : Auto 0 : Manual
Value of sw	ON : Auto OFF : Manual
Initial setting	1
Example	SSA 1; SSA ON; SSA?; FREQ:CENT:STEP:AUTO 1; FREQ:CENT:STEP:AUTO ON; FREQ:CENT:STEP:AUTO?;

FO

[[:SENSe]:FREQuency:OFFSet

	Frequency Offset
Function	Sets the frequency offset value.
Remote Command	FO f FO? [:SENSe]:FREQuency:OFFSet f [:SENSe]:FREQuency:OFFSet?
Response Message	Frequency Offset (Hz)
Value of f	-1000 GHz to 1000 GHz
Suffix code	NONE : Hz HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	0 Hz
Example	FO 123456; FO 3410.7MHZ; FO?; FREQ:OFFS 123456; FREQ:OFFS 3410.7MHZ; FREQ:OFFS?;

STR

:CALCulate:MARKer[1~9]:TRCKing[:STATe]

	Signal Track
Function	Activates the selected marker and sets the active marker to the maximum peak point on the selected trace. Sets the center frequency to the selected marker frequency. This function is done after the sweep, thus maintaining the marker value at the center frequency.
Remote Command	STR[1~9] n STR[1~9] sw STR[1~9]? :CALCulate:MARKer[1~9]:TRCKing[:STATe] n :CALCulate:MARKer[1~9]:TRCKing[:STATe] sw :CALCulate:MARKer[1~9]:TRCKing[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	STR 1; STR5 ON; STR5?; CALC:MARK:TRCK 1; CALC:MARK5:TRCK ON; CALC:MARK5:TRCK?;

In/Out

RFC

:INPut:COUPling

	RF Coupling
Function	Turns the RF Coupling on or off.
Remote Command	RFC sw RFC? :INPut:COUPling sw :INPut:COUPling?
Response Message	AC : AC DC : DC
Value of sw	AC : AC DC : DC
Initial setting	DC
Example	RFC AC; RFC? INP:COUP AC; INP:COUP?

Limit Line

LLCS[1~2]

:CALCulate:LLINe[1~2]:CHECK:STATe

	Limit Line Check State	
Function	Turns the limit line checking on or off.	
Remote Command	LLCS[1~2] n	
	LLCS[1~2] sw	
	LLCS[1~2]?	
	:CALCulate:LLINe[1~2]:CHECK:STATe n	
	:CALCulate:LLINe[1~2]:CHECK:STATe sw	
	:CALCulate:LLINe[1~2]:CHECK:STATe?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	LLCS 1;	
	LLCS2 ON	
	LLCS2?	
	CALC:LLIN:CHEC:STAT 1;	
	CALC:LLIN2:CHEC:STAT ON;	
	CALC:LLIN2:CHEC:STAT?	

LLFC[1~2]

:CALCulate:LLINe[1~2]:FAIL:COUNT

	Limit Line Fail Count
Function	Returns the limit line Fail Count.
Remote Command	LLFC[1~2]? :CALCulate:LLINe[1~2]:FAIL:COUNT?
Response Message	Fail Count
Initial setting	0
Example	LLFC?; LLFC2?; CALC:LLIN:FAIL:COUNT?; CALC:LLIN2:FAIL:COUNT?;

ALARM

:CALCulate:LLINe:ALARM

	Alarm State
Function	Turns the alarm state on or off
Remote Command	ALARM n ALARM sw ALARM? :CALCulate:LLINe:ALARM n :CALCulate:LLINe:ALARM sw :CALCulate:LLINe:ALARM?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	ALARM 1; ALARM ON ALARM? CALC:LLIN:ALARM 1; CALC:LLIN2:ALARM ON; CALC:LLIN2:ALARM?

LLAO

:CALCulate:LLINe:AOff

	Clear Limit Line
Function	Clear Limit Line.
Remote Command	LLAO :CALCulate:LLINe:AOff?
Example	LLAO; CALC:LLIN:AOff?;

[Reference]

You can insert X, Y data of the Limit Line after loading the Limit File(*.lim).

Marker

MS[1~9]

:CALCulate:MARKer[1~9]:STATe

	Marker State
Function	Sets the selected marker state.
Remote Command	MS[1~9] n MS[1~9] sw MS[1~9]? :CALCulate:MARKer[1~9]:STATe n :CALCulate:MARKer[1~9]:STATe sw :CALCulate:MARKer[1~9]:STATe?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	MS 1; MS5 1; MS5?; CALC:MARK:STAT 1; CALC:MARK5:STAT ON; CALC:MARK5:STAT?

MM[1~9]

:CALCulate:MARKer[1~9]:MODE

	Marker Mode
Function	Sets the selected marker to Normal, Delta, Band Pair, Span Pair Mode.
Remote Command	MM[1~9] sw MM[1~9]? :CALCulate:MARKer[1~9]:MODE sw :CALCulate:MARKer[1~9]:MODE?
Response Message	POS : Normal DELT : Delta BAND : Band Pair SPAN : Span Pair OFF : OFF
Value of sw	POSITION : Normal DELTA : Delta BAND : Band Pair
SPAN	: Span Pair OFF : OFF
Initial setting	OFF
Example	MM POS; MM5 BAND; MM5?; CALC:MARK:MODE POS; CALC:MARK5:MODE BAND; CALC:MARK5:MODE?

MF[1~9]

:CALCulate:MARKer[1~9]:X

	Marker Frequency
Function	Sets the marker frequency of the selected marker. If the marker mode is the delta mode, sets the difference value of the marker frequency and the delta marker frequency.
Remote Command	MF[1~9] f MF[1~9]? :CALCulate:MARKer[1~9]:X f :CALCulate:MARKer[1~9]:X?
Response Message	Marker Frequency (Hz)
Value of f	Start Frequency to Stop Frequency
Suffix code	None : Hz (10 ⁰) (when Readout is Freq or ITime) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹) None : s (10 ⁰) (when Readout is Period or Time) kSEC : ks (10 ³) SEC : s (10 ⁰) MSEC : ms (10 ⁻³) USEC : μs (10 ⁻⁶) NSEC : ns (10 ⁻⁹) PSEC : ps (10 ⁻¹²)
Initial setting	Span/2
Example	MF 123456; MF5 1GHZ; MF5 0.01SEC; MF5?; CALC:MARK:X 123456; CALC:MARK5:X 1GHZ; CALC:MARK5:X 0.01SEC; CALC:MARK5:X?

MP[1~9]

:CALCulate:MARKer[1~9]:X:POStion

	Marker Point
Function	Sets the marker point of the selected marker. If the marker mode is delta mode, sets the difference of the marker point and the delta marker point.
Remote Command	MP[1~9] n MP? :CALCulate:MARKer[1~9]:X:POS n :CALCulate:MARKer[1~9]:X:POS?
Response Message	Marker Point
Value of n	0 to points (step : 1)
Initial setting	Points/2
Example	MP 275; MP5 551; MP5?; CALC:MARK:X:POS 275; CALC:MARK5:X:POS 551; CALC:MARK5:X:POS?

MFS[1~9]

:CALCulate:MARKer[1~9]:X:STARt

	Marker Start Frequency
Function	Sets the reference marker frequency of the selected marker.
Remote Command	MFS[1~9] f MFS[1~9]? :CALCulate:MARKer[1~9]:X:STARt f :CALCulate:MARKer[1~9]:X:STARt?
Response Message	Marker Frequency (Hz)
Value of f	Start Frequency to Stop Frequency
Suffix code	None : Hz (10 ⁰) (when Readout is In Freq or ITime) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹) None : s (10 ⁰) (when Readout is In Period or Time) kSEC : ks (10 ³) SEC : s (10 ⁰) MSEC : ms (10 ⁻³) USEC : μs (10 ⁻⁶) NSEC : ns (10 ⁻⁹) PSEC : ps (10 ⁻¹²)
Initial setting	Span/2
Example	MFS 123456; MFS5 1GHZ; MFS5 0.01SEC; MFS5?; CALC:MARK:X:STAR 123456; CALC:MARK5:X:STAR 1GHZ; CALC:MARK5:X:STAR 0.01SEC; CALC:MARK5:X:STAR?

MPS[1~9]

:CALCulate:MARKer[1~9]:X:POSition:STARt

	Marker Start Point
Function	Sets the reference marker point of the selected marker.
Remote Command	MPS[1~9] n MPS[1~9]? :CALCulate:MARKer[1~9]:X:POS:STARt n :CALCulate:MARKer[1~9]:X:POS:STARt?
Response Message	Marker Point
Value of n	0 to Points (Step : 1)
Initial setting	Points/2
Example	MPS 275; MPS5 0; MPS5?; CALC:MARK:X:POS:STAR 275; CALC:MARK5:X:POS:STAR 0; CALC:MARK5:X:POS:STAR?

MFE[1~9]

:CALCulate:MARKer[1~9]:X:STOP

	Marker Stop Frequency
Function	Sets the normal(or delta) marker frequency of the selected marker.
Remote Command	MFE[1~9] f MFE[1~9]? :CALCulate:MARKer[1~9]:X:STOP f :CALCulate:MARKer[1~9]:X:STOP?
Response Message	Marker Frequency (Hz)
Value of f	Start Frequency to Stop Frequency
Suffix code	None : Hz (10 ⁰) (when Readout is In Freq or ITime) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹) None : s (10 ⁰) (when Readout is In Period or Time) kSEC : ks (10 ³) SEC : s (10 ⁰) MSEC : ms (10 ⁻³) USEC : μs (10 ⁻⁶) NSEC : ns (10 ⁻⁹) PSEC : ps (10 ⁻¹²)
Initial setting	Span/2
Example	MFE 123456; MFE5 1GHZ; MFE5 0.01SEC; MFE5?; CALC:MARK:X:STOP 123456; CALC:MARK5:X:STOP 1GHZ; CALC:MARK5:X:STOP 0.01SEC; CALC:MARK5:X:STOP?

MPE[1~9]

:CALCulate:MARKer[1~9]:X:POSition:STOP

	Marker Stop Point
Function	Sets the normal (or delta) marker point of the selected marker.
Remote Command	MPE[1~9] n MPE[1~9]? :CALCulate:MARKer[1~9]:X:POS:STOP n :CALCulate:MARKer[1~9]:X:POS:STOP?
Response Message	Marker Point
Value of n	0 to Points (Step : 1)
Initial setting	Points/2
Example	MPE 275; MPE5 551; MPE5?; CALC:MARK:X:POS:STOP 275; CALC:MARK5:X:POS:STOP 551; CALC:MARK5:X:POS:STOP?

MFC[1~9]

:CALCulate:MARKer[1~9]:X:CENTer

	Marker Center Frequency
Function	Sets the middle frequency of the delta marker and the reference marker.
Remote Command	MFC[1~9] f MFC[1~9]? :CALCulate:MARKer[1~9]:X:CENTer f :CALCulate:MARKer[1~9]:X:CENTer?
Response Message	Middle Frequency of delta marker and reference marker (Hz)
Value of f	Start Frequency to Stop Frequency (Step : Span/(Points-1))
Suffix code	None : Hz (10 ⁰) (when Readout is In Freq or ITime) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹) None : s (10 ⁰) (when Readout is In Period or Time) kSEC : ks (10 ³) SEC : s (10 ⁰) MSEC : ms (10 ⁻³) USEC : μs (10 ⁻⁶) NSEC : ns (10 ⁻⁹) PSEC : ps (10 ⁻¹²)
Initial setting	Span/2
Example	MFC 123456; MFC5 1GHZ; MFC5 0.01SEC; MFC5?; CALC:MARK:X:CENT 123456; CALC:MARK5:X:CENT 1GHZ; CALC:MARK5:X:CENT 0.01SEC; CALC:MARK5:X:CENT?

MPC[1~9]

:CALCulate:MARKer[1~9]:X:POSition:CENTer

	Marker Center Point
Function	Sets the middle point of the delta marker and the reference marker.
Remote Command	MPC[1~9] n MPC[1~9]? :CALCulate:MARKer[1~9]:X:POS:CENTer n :CALCulate:MARKer[1~9]:X:POS:CENTer?
Response Message	Middle Point of delta marker and reference marker
Value of n	0 to Points (Step : 1)
Initial setting	Points/2
Example	MPC 275; MPC5 551; MPC5?; CALC:MARK:X:POS:CENT 275; CALC:MARK5:X:POS:CENT 551; CALC:MARK5:X:POS:CENT?

MFSP[1~9]

:CALCulate:MARKer[1~9]:X:SPAN

	Marker Span Frequency
Function	Sets the difference of the delta marker and the reference marker.
Remote Command	MFSP[1~9] f MFSP[1~9]? :CALCulate:MARKer[1~9]:X:SPAN f :CALCulate:MARKer[1~9]:X:SPAN?
Response Message	Difference of delta marker and reference marker (Hz)
Value of f	Start Frequency to Stop Frequency
Suffix code	None : Hz (10 ⁰) (when Readout is In Freq or ITime) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹) None : s (10 ⁰) (when Readout is In Period or Time) kSEC : ks (10 ³) SEC : s (10 ⁰) MSEC : ms (10 ⁻³) USEC : μs (10 ⁻⁶) NSEC : ns (10 ⁻⁹) PSEC : ps (10 ⁻¹²)
Initial setting	0 Hz
Example	MFSP 123456; MFSP5 1GHZ; MFSP5 0.01SEC; MFSP5?; CALC:MARK:X:SPAN 123456; CALC:MARK5:X:SPAN 1GHZ; CALC:MARK5:X:SPAN 0.01SEC; CALC:MARK5:X:SPAN?

MPSP[1~9]

:CALCulate:MARKer[1~9]:X:POSition:SPAN

	Marker Span Point
Function	Sets the difference between the delta marker and the reference marker.
Remote Command	MPSP[1~9] n MPSP[1~9]? :CALCulate:MARKer[1~9]:X:POS:SPAN n :CALCulate:MARKer[1~9]:X:POS:SPAN?
Response Message	Difference between delta marker and reference marker
Value of n	0 to Points (Step : 1)
Initial setting	0
Example	MPSP 275; MPSP5 551; MPSP5?; CALC:MARK:X:POS:SPAN 275; CALC:MARK5:X:POS:SPAN 551; CALC:MARK5:X:POS:SPAN?

MA[1~9]

:CALCulate:MARKer[1~9]:Y

	Marker Amplitude
Function	Returns amplitude data according to the marker readout.
Remote Command	MA[1~9]? :CALCulate:MARKer[1~9]:Y?
Response Message	Marker Amplitude (Hz in FREQ or ITIME, sec in PER or TIME)
Example	MA?; MA5? CALC:MARK:Y? CALC:MARK5:Y?

MT[1~9]

:CALCulate:MARKer[1~9]:TRACe

	Select Marker Trace	
Function	Selects the marker trace.	
Remote Command	MT[1~9] n MT[1~9]? :CALCulate:MARKer[1~9]:MKT n :CALCulate:MARKer[1~9]:MKT?	
Response Message	1	: Trace A
	2	: Trace B
	3	: Trace C
Value of n	1	: Trace A
	2	: Trace B
	3	: Trace C
Initial setting	1	
Example	MT 2; MT5 2; MT5?; CALC:MARK:TRAC 2; CALC:MARK5:TRAC 2; CALC:MARK5:TRAC?;	

MR[1~9]

:CALCulate:MARKer[1~9]:READout

	Marker Read Out
Function	Defines the marker read-out.
Remote Command	MR[1~9] sw MR[1~9]? :CALCulate:MARKer[1~9]:READout sw :CALCulate:MARKer[1~9]:READout?
Response Message	FREQ : Frequency PER : Period = 1/Frequency TIME : Time ITIME : Inverse Time = 1/Time
Value of sw	FREQ : Frequency PERiod : Period = 1/Frequency TIME : Time ITIME : Inverse Time = 1/Time
Initial setting	FREQ
Example	MR FREQ; MR PER; MR?; CALC:MARK:READ FREQ; CALC:MARK5:READ PER; CALC:MARK5:READ?;

MTB

:CALCulate:MARKer:TABLE:STATe

	Marker Table State	
Function	Sets the marker table state.	
Remote Command	MTB n	
	MTB sw	
	MTB?	
	:CALCulate:MARKer:TABLE:STATe n	
	:CALCulate:MARKer:TABLE:STATe sw	
	:CALCulate:MARKer:TABLE:STATe?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	MTB 1;	
	MTB ON;	
	MTB?;	
	CALC:MARK:TABL:STAT 1;	
	CALC:MARK:TABL:STAT ON;	
	CALC:MARK:TABL:STAT?;	

MAO

:CALCulate:MARKer:AOff

	Marker All OFF
Function	Turns off all markers.
Remote Command	MAO :CALCulate:MARKer:AOff
Example	MAO; CALC:MARK:AOff;

Marker Function

MFN[1~9]

:CALCulate:MARKer[1~9]:FUNction

	Marker Function
Function	Selects the marker function mode.
Remote Command	MFN[1~9] sw MFN[1~9]? :CALCulate:MARKer:FUNC sw :CALCulate:MARKer:FUNC?
Response Message	NOIS : Marker Noise COUN : Marker Counter OFF : Marker Function OFF
Value of sw	NOISe : Marker Noise COUNT : Marker Counter OFF : Marker Function OFF
Example	MFN NOIS; MFN5? CALC:MARK:FUNC NOIS; CALC:MARK5:FUNC?;

MFNY

:CALCulate:MARKer:FUNCTion:Y

	Marker Noise Output
Function	Gets the marker noise output.
Remote Command	MFNY? :CALCulate:MARKer:FUNCTion:Y?
Response Message	Marker Noise Output
Example	MFNY?; CALC:MARK:FUNC:Y?; CALC:MARK:PHAS:OFFS 5KHZ; CALC:MARK:PHAS:OFFS?;

MFNX

:CALCulate:MARKer:FCOunt:X

	Frequency Counter Output
Function	Gets the output of the marker counter function.
Remote Command	MFNX? :CALCulate:MARKer:FCOunt:X?
Response Message	Frequency counter output (Hz)
Example	MFNX?; CALC:MARK:FCO:X?;

MFNR

:CALCulate:MARKer:FCOunt:RESolution

	Frequency Counter Resolution
Function	Sets the resolution of the marker counter function.
Remote Command	MFNR f MFNR? :CALCulate:MARKer:FCOunt:RESolution f :CALCulate:MARKer:FCOunt:RESolution?
Response Message	Frequency counter resolution (Hz)
Value of f	1, 10, 100, 1000 (Hz)
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³)
Initial setting	1000 Hz
Example	MFNR 1000; MFNR 1KHZ; MFNR?;

Marker Shift

MCF[1~9]

:CALCulate:MARKer[1~9][:SET]:CENTer

	Marker to Center Frequency
Function	Sets the center frequency to the frequency value of the normal marker. The normal marker must be active to work.
Remote Command	MCF[1~9] :CALCulate:MARKer[1~9][:SET]:CENTer
Example	MCF; MCF5; CALC:MARK:CENT; CALC:MARK5:CENT;

MSS[1~9]

:CALCulate:MARKer[1~9][:SET]:STEP

	Marker to CF Step
Function	Sets the center frequency step-size equal to the frequency value of the selected marker. The selected marker must be active to work.
Remote Command	MSS[1~9] :CALCulate:MARKer[1~9][:SET]:STEP
Example	MSS; MSS5; CALC:MARK:STEP; CALC:MARK5:STEP;

MFA[1~9]

:CALCulate:MARKer[1~9][:SET]:START

	Marker to Start Frequency
Function	Sets the selected marker frequency to the start frequency. The selected marker must be active to work.
Remote Command	MFA[1~9] :CALCulate:MARKer[1~9][:SET]:START
Example	MFA; MFA5; CALC:MARK:STAR; CALC:MARK5:STAR;

MFB[1~9]

:CALCulate:MARKer[1~9][:SET]:STOP

	Marker to Stop Frequency
Function	Sets the selected marker frequency to the stop frequency. The selected marker must be active to work.
Remote Command	MFB[1~9] :CALCulate:MARKer[1~9][:SET]:STOP
Example	MFB; MFB5; CALC:MARK:STOP; CALC:MARK5:STOP;

MRL[1~9]

:CALCulate:MARKer[1~9][:SET]:RLEVel

	Marker to Reference Level
Function	Sets the reference level to the amplitude of the selected marker. The selected marker must be active to work.
Remote Command	MRL[1~9] :CALCulate:MARKer[1~9][:SET]:RLEVel
Example	MRL; MRL5; CALC:MARK:RLEV; CALC:MARK5:RLEV;

MDCF[1~9]

:CALCulate:MARKer[1~9][:SET]:DELTA:CENTer

	Marker Delta to Center Frequency
Function	Sets the selected delta frequency to center frequency. If both the delta marker and the reference marker are inactive, this function does not work.
Remote Command	MDCF[1~9] :CALCulate:MARKer[1~9][:SET]:DELTA:CENTer
Example	MDCF; MDCF5; CALC:MARK:DELT:CENT; CALC:MARK5:DELT:CENT;

MDSS

:CALCulate:MARKer[1~9][:SET]:DELTA:CENTer

	Marker Delta to CF Step
Function	Sets the selected delta frequency to the center frequency step size. If both the delta marker and the reference marker are inactive, this function does not work.
Remote Command	MDSS[1~9] :CALCulate:MARKer[1~9][:SET]:DELTA:STEP
Example	MDSS; MDSS5; CALC:MARK:DELT:STEP; CALC:MARK5:DELT:STEP;

MDSP

:CALCulate:MARKer[1~9][:SET]:DELTA:SPAN

Marker Delta to Span

Function	Sets the selected delta frequency to the span frequency. If both the delta marker and the reference marker are inactive, this function does not work.
Remote Command	MDSP[1~9] :CALCulate:MARKer[1~9][:SET]:DELTA:SPAN
Example	MDSP; MDSP5; CALC:MARK:DELT:SPAN; CALC:MARK5:DELT:SPAN;

Measurement

MEA

:MEASure:STARt

	Measure Start
Function	Starts the measurement.
Remote Command	MEA sw MEA? :MEASure:STARt sw :MEASure:STARt?
Response Message	XDB : X dB Down ACP : Adjacent Channel Power CHP : Channel Power OBW : Occupied Bandwidth HD : Harmonic Distribution CCDF : CCDF
Value of sw	XDB : X dB Down ACP : Adjacent Channel Power CHP : Channel Power OBW : Occupied Bandwidth HD : Harmonic Distribution CCDF : CCDF
Example	MEA XDB; MEA?; MEAS:STAR XDB; MEAS:STAR?;

MEAAVG

:MEASure:AVERage[:STATe]

	Measurement Average State
Function	Sets the measurement mode to averaging. The measurement value is averaged continuously.
Remote Command	MEAAVG n MEAAVG sw MEAAVG? :MEASure:AVERage[:STATe] n :MEASure:AVERage[:STATe] sw :MEASure:AVERage[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	MEAAVG 1; MEAAVG ON; MEAAVG? MEAS:AVER 1; MEAS:AVER ON; MEAS:AVER?;

MEAAVGM

:MEASure:AVERage:TCONrol

	Measurement Average Mode
Function	Sets the measurement mode to averaging mode.
Remote Command	MEAAVGM sw MEAAVGM? :MEASure:AVERage:TCONrol sw :MEASure:AVERage:TCONrol?
Response Message	EXP : Exponential REP : Repeat
Value of sw	EXponential : Exponential REPeat : Repeat
Initial setting	EXP
Example	MEAAVGM EXP; MEAAVGM? MEAS:AVER:TCON EXP; MEAS:AVER:TCON?;

MEAAVGR

:MEASure:AVERage:RESET

	Measurement Average Reset
Function	Resets the measurement average.
Remote Command	MEAAVGR :MEASure:AVERage:RESET
Example	MEAAVGR; MEAS:AVER:RESET;

MEAAVGN

:MEASure:AVERage:COUNT

	Measurement Average Count
Function	Sets the measurement mode to averaging count.
Remote Command	MEAAVGN n MEAAVGN? :MEASure:AVERage:COUNT n :MEASure:AVERage:COUNT?
Response Message	Average Count
Value of n	1 to 1000
Initial setting	10
Example	MEAAVGN 20; MEAAVGN? MEAS:AVER:COUN 20; MEAS:AVER:COUN?;

MEAO

:MEASure:AOff

	Measure OFF
Function	Stops the measurement.
Remote Command	MEAO :MEASure:AOff
Example	MEAO; MEAS:AOff;

Measurement - X dB Down

XDBP[1~2]

[[:SENSe]:XDB[1~2]:AMPLitude

	X dB Point
Function	Sets the X dB point value.
Remote Command	XDBP[1~2] f XDBP[1~2]? [:SENSe]:XDB[1~2]:AMPLitude f [:SENSe]:XDB[1~2]:AMPLitude?
Response Message	XdB Point (dB)
Value of f	0.01 dB to 210 dB
Suffix code f	None : dB DB : dB
Initial setting	3.00 dB
Example	XDBP 3; XDBP2 10DB; XDBP2?; XDB:AMPL 3; XDB2:AMPL 10DB; XDB2:AMPL?;

XDBL[1~2]

:FETCh|MEASure|READ:XDB[1~2]:FREQuency:LEFT

	X dB Left
Function	Returns left frequency bandwidth that corresponds to X dB below marker level.
Remote Command	XDBL[1~2] f XDBL[1~2]? :FETCh MEASure READ:XDB[1~2]:FREQuency:LEFT f :FETCh MEASure READ:XDB[1~2]:FREQuency:LEFT?
Response Message	Output (Hz)
Example	XDBL?; XDBL2?; FETC:XDB:FREQ:LEFT?; FETC:XDB2:FREQ:LEFT?;

XDBR[1~2]

:FETCh|MEASure|READ:XDB[1~2]:FREQuency:RIGHt

	X dB Right
Function	Returns right frequency bandwidth that corresponds to X dB below marker level.
Remote Command	XDBR[1~2] f XDBR[1~2]? :FETCh MEASure READ:XDB[1~2]:FREQuency:RIGHt f :FETCh MEASure READ:XDB[1~2]:FREQuency:RIGHt?
Response Message	Output (Hz)
Example	XDBR?; XDBR2?; FETC:XDB:FREQ:RIGH?; FETC:XDB2:FREQ:RIGH?;

XDBRL[1~2]

:FETCh|MEASure|READ:XDB[1~2]:FREQuency:BANDwidth|BWIDth

	X dB Relative
Function	Returns both frequency bandwidths that correspond to X dB below marker level.
Remote Command	XDBRL[1~2] f XDBRL[1~2]? :FETCh MEASure READ:XDB[1~2]:FREQuency:BANDwidth BWIDth f :FETCh MEASure READ:XDB[1~2]:FREQuency:BANDwidth BWIDth?
Response Message	Output (Hz)
Example	XDBRL?; XDBRL2?; FETC:XDB:FREQ:BAND?; FETC:XDB2:FREQ:BAND?;

XDBS

:FETCh|MEASure|READ:XDB[1~2]:FREQuency:SHAPe

	Shape Factor
Function	Returns shape factor value.
Remote Command	XDBS f XDBS? :FETCh MEASure READ:XDB[1~2]:FREQuency:SHAPe f :FETCh MEASure READ:XDB[1~2]:FREQuency:SHAPe?
Response Message	Output (Hz)
Example	XDBS?; XDBS?; FETC:XDB:FREQ:SHAP?; FETC:XDB2:FREQ:SHAP?;

Measurement - Adjacent Channel Power

ACPMC

[[:SENSe]:ACPower:BANDwidth|BWIDth:INTegration

	ACP Main Channel Bandwidth
Function	Sets main channel bandwidth in adjacent channel power measurement.
Remote Command	ACPMC f ACPMC? [:SENSe]:ACPower:BANDwidth BWIDth:INTegration f [:SENSe]:ACPower:BANDwidth BWIDth:INTegration?
Response Message	ACP Main Channel Bandwidth (Hz)
Value of f	Depends on Span
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	Span/5
Example	ACPMC 123456; ACPMC 50MHZ; ACPMC?; ACP:BAND:INT 123456; ACP:BAND:INT 50MHZ; ACP:BAND:INT?;

ACPAC

[[:SENSe]:ACPower:ADJacent:BANDwidth|BWIDth:INTegration

	ACP Adjacent Channel Bandwidth
Function	Sets adjacent channel bandwidth in ACP measurement.
Remote Command	ACPAC f ACPAC? [:SENSe]:ACPower:ADJacent:BANDwidth BWIDth:INTegration f [:SENSe]:ACPower:ADJacent:BANDwidth BWIDth:INTegration?
Response Message	ACP Adjacent Channel Bandwidth (Hz)
Value of f	Depends on Span
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : KHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	Span/5
Example	ACPAC 123456; ACPAC 50MHZ; ACPAC?; ACP:ADJ:BAND:INT 123456; ACP:ADJ:BAND:INT 50MHZ; ACP:ADJ:BAND:INT?;

ACPCS

[[:SENSE]:ACPower:SPACe:BANDwidth|BWIDth:INTegration

	ACP Channel Space Bandwidth
Function	Sets channel space, which is the distance from the main channel to the adjacent channel.
Remote Command	ACPCS f ACPCS? [:SENSE]:ACPower:SPACe:BANDwidth BWIDth: INTegration f [:SENSE]:ACPower:SPACe:BANDwidth BWIDth: INTegration?
Response Message	ACP Channel Space Bandwidth (Hz)
Value of f	Depends on Span
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	Span/4
Example	ACPCS 123456; ACPCS 50MHZ; ACPCS?; ACP:SPAC:BAND:INT 123456; ACP:SPAC:BAND:INT 50MHZ; ACP:SPAC:BAND:INT?;

ACPN

[[:SENSe]:ACPower:COUNT

	Adjacent Channel Count
Function	Sets adjacent channel counts from 1 up to 6.
Remote Command	ACPN n ACPN? [:SENSe]:ACPower:COUNT n [:SENSe]:ACPower:COUNT?
Response Message	Adjacent Channel Count
Value of n	1 to 6
Initial setting	1
Example	ACPN 1; ACPN?; ACP:COUN 1; ACP:COUN?;

ACPOUT

:FETCh|MEASure|READ:ACPpower[:CHPower|:LACPpower|:RACPpower]

ACP Measurement Output

Function	Returns the main channel power level value, the lower adjacent channel power level value and the upper adjacent channel power level value.
Remote Command	ACPOUT? :FETCh MEASure READ:ACPpower?
Response Message	Main CHP, Lower ACP1, Upper ACP1, Lower ACP2, Upper ACP2...
Example	ACPOUT?; FETC:ACP?;

ACPOUTC

:FETCh|MEASure|READ:ACPower:CHPower

	Main Channel Power
Function	Returns the main channel power level value.
Remote Command	ACPOUTC? :FETCh MEASure READ:ACPower:CHPower?
Response Message	Main Channel Power
Example	ACPOUTC?; FETC:ACP:CHP?;

ACPOUTL[1~6]

:FETCh|MEASure|READ:ACPower:LACPower[1~6]

	Lower Adjacent Channel Power
Function	Returns the lower adjacent channel power level value.
Remote Command	ACPOUTL[1~6]? :FETCh MEASure READ:ACPower:LACPower[1~6]?
Response Message	Lower Adjacent Channel Power
Example	ACPOUTL?; ACPOUTL6?; FETC:ACP:LACP?; FETC:ACP:LACP6?;

ACPOUTR[1~6]

:FETCh|MEASure|READ:ACPower:RACPower[1~6]

	Upper Adjacent Channel Power
Function	Returns the upper adjacent channel power level value.
Remote Command	ACPOUTR[1~6]? :FETCh MEASure READ:ACPower:RACPower[1~6]?
Response Message	Upper Adjacent Channel Power
Example	ACPOUTR?; ACPOUTR6?; FETC:ACP:RACP?; FETC:ACP:RACP6?;

Measurement - Channel Power

CHPC

[[:SENSe]:CHPower:BANDwidth|BWIDth:INTegration

	CHP Channel BandWidth
Function	Sets the channel power bandwidth for measuring power level in the limited frequency range.
Remote Command	CHPC f CHPC? [:SENSe]:CHPower:BANDwidth BWIDth:INTegration f [:SENSe]:CHPower:BANDwidth BWIDth:INTegration?
Response Message	CHP Bandwidth (Hz)
Value of f	Depend on Span
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	Span/5
Example	SCPBW 123456; SCPBW 50MHZ; SCPBW?; CHP:BAND:INT 123456; CHP:BAND:INT 50MHZ; CHP:BAND:INT?;

CHPOUT[C|D]

FETCh|MEASure|READ:CHPower[:CHPower|DENSity]

	CHP Measurement Output
Function	Returns the channel power level value.
Remote Command	CHPOUT[C D]? FETCh MEASure READ:CHPower[:CHPower DENSity]?
Response Message	Ch. Power, Pwr Spectral Density
Example	CHPOUT?; CHPOUTC?; CHPOUTD?; FETC:CHP?; FETC:CHP:CHP?; FETC:CHP:DENS?;

Measurement - Occupied BandWidth

OBW:FREQ:SPAN 50MHZ;

OBW:FREQ:SPAN ?;

OBWP

[[:SENSe]:OBWidth:PERCent

	OBW Power Percentage
Function	Sets the power percentage for measuring channel power within a specific percentage of total channel power.
Remote Command	OBWP f OBWP? [:SENSe]:OBWidth:PERCent sw [:SENSe]:OBWidth:PERCent?
Response Message	OBW Power Percentage
Value of f	0.01 to 100 (Step : 0.01%)
Initial setting	98.00%
Example	OBWP 90; OBWP? OBW:PERC 90; OBW:PERC?; OBWOUT :FETCh MEASure READ:OBWidth OBW Output
Function	Returns the bandwidth frequency that is limited by OBW.
Remote Command	OBWOUT? :FETCh MEASure READ:OBWidth?
Response Message	OBW Bandwidth ()
Example	OBWOUT?; FETC:OBW?;

Measurement - Harmonic Distortion

HDN

[[:SENSe]:HARMonics:NUMBer

	HD Number
Function	Sets the harmonic order to be measured.
Remote Command	HDN n HDN? [SENSe]:HARMonics:NUMBer n [SENSe]:HARMonics:NUMBer?
Response Message	HD Number
Value of n	2 to 5
Initial setting	5
Example	HDN 5; HDN? HARM:NUMB 90; HARM:NUMB?;

HF[1|2|3|4|5]

:FETCh|MEASure|READ:HARMonics:FREQuency[1|2|3|4|5]

	HD Frequency
Function	Returns the distortion frequency of the harmonic distortion measurement.
Remote Command	HF[1 2 3 4 5] :FETCh MEASure READ:HARMonics:FREQuency[1 2 3 4 5]
Response Message	HD Freq1, HD Freq2, HD Freq3, HD Freq4, HD Freq5
Example	HF?; HF3?; FETC:HARM:FREQ?; FETC:HARM:FREQ3?;

HA[1|2|3|4|5]

:FETCh|MEASure|READ:HARMonics:AMPLitude[1|2|3|4|5]

	HD Amplitude
Function	Returns the first distortion power level of the harmonic distortion measurement.
Remote Command	HA[1 2 3 4 5] :FETCh MEASure READ:HARMonics:AMPLitude[1 2 3 4 5]
Response Message	HD Ampl1, HD Ampl2, HD Ampl3, HD Ampl4, HD Ampl5
Example	HA?; HA3?; FETC:HARM:AMPL?; FETC:HARM:AMPL3?;

HDOUT

	:FETCh MEASure READ:HARMonics[:DISTortion]
	Total Harmonic Distortion
Function	Measures total harmonic distortion.
Remote Command	HDOUT? :FETCh MEASure READ:HARMonics[:DISTortion]?
Response Message	Harmonics Distortion (%)
Example	HDOUT?; FETC:HARM?;

Measurement – Third Order Intermodulation

TOIOUT

:FETCh|MEASure|READ:TOIN

	Third Order Intermodulation
Function	Returns the TOI results table.
Remote Command	TOIOUT? :FETCh MEASure READ:TOIN?
Response Message	Base Lower Freq, Base Lower Level, Base Lower Difference, Base Upper Freq, Base Upper Level, Base Upper Difference, Worst Case Freq, Worst Case Level, Worst Difference, Worst Case IP3, 3-Order Lower Freq, 3-Order Lower Level, 3-Order Lower Difference, 3-Order Lower IP3, 3-Order Upper Freq, 3-Order Upper Level, 3-Order Upper Difference, 3-Order Upper IP3,
Example	TOIOUT?; FETC:TOIN?;

TOIBLF

:FETCh|MEASure|READ:TOIN:BASE:LOWER:FREQuency

	TOI Base Lower Frequency
Function	Returns the lower frequency of two signals.
Remote Command	TOIBLF? :FETCh MEASure READ:TOIN:BASE:LOWER:FREQuency?
Response Message	Base Lower Freq (Hz)
Example	TOIBLF?; FETC:TOIN:BASE:LOWER:FREQ?;

TOIBLL

:FETCh|MEASure|READ:TOIN:BASE:LOWER:LEVel

	TOI Base Lower Level
Function	Returns the lower level of two signals.
Remote Command	TOIBLL? :FETCh MEASure READ:TOIN:BASE:LOWER:LEVel?
Response Message	Base Lower Level (dBm)
Example	TOIBLL?; FETC:TOIN:BASE:LOWER:LEV?;

TOIBLD

:FETCh|MEASure|READ:TOIN:BASE:LOWER:LEVel:DIFFerence

	TOI Base Lower Level Difference
Function	Returns the lower level difference of two signals.
Remote Command	TOIBLD? :FETCh MEASure READ:TOIN:BASE:LOWER:LEVel: DIFFerence?
Response Message	Base Lower Level Difference (dBc)
Example	TOIBLD?; FETC:TOIN:BASE:LOWER:LEV:DIFF?;

TOIBUF

:FETCh|MEASure|READ:TOIN:BASE:UPPER:FREQuency

	TOI Base Upper Frequency
Function	Returns the upper frequency of two signals.
Remote Command	TOIBUF? :FETCh MEASure READ:TOIN:BASE:UPPER:FREQuency?
Response Message	Base Upper Freq (Hz)
Example	TOIBUF?; FETC:TOIN:BASE:UPPER:FREQ?;

TOIBUL

:FETCh|MEASure|READ:TOIN:BASE:UPPER:LEVel

	TOI Base Upper Level
Function	Returns the upper level of two signals.
Remote Command	TOIBUL? :FETCh MEASure READ:TOIN:BASE:UPPER:LEVel?
Response Message	Base Upper Level (dBm)
Example	TOIBUL?; FETC:TOIN:BASE:UPPER:LEV?;

TOIBUD

:FETCh|MEASure|READ:TOIN:BASE:UPPER:LEVel:DIFFerence

	TOI Base Upper Level Difference
Function	Returns the upper level difference of two signals.
Remote Command	TOIBUD? :FETCh MEASure READ:TOIN:BASE:UPPER:LEVel: DIFFerence?
Response Message	Base Upper Level Difference (dBc)
Example	TOIBUD?; FETC:TOIN:BASE:UPPER:LEV:DIFF?;

TOIWCF

:FETCh|MEASure|READ:TOIN:WORST:FREQuency

	TOI Worst Case Frequency
Function	Returns the worst case frequency of TOI results.
Remote Command	TOIWCF? :FETCh MEASure READ:TOIN:WORST:FREQuency?
Response Message	Worst Case Freq (Hz)
Example	TOIWCF?; FETC:TOIN:WORST:FREQ?;

TOIWCL

:FETCh|MEASure|READ:TOIN:WORST:LEVel

	TOI Worst Case Level
Function	Returns the worst case level of TOI results.
Remote Command	TOIWCL? :FETCh MEASure READ:TOIN:WORST:LEVel?
Response Message	Worst Case Level (dBm)
Example	TOIWCL?; FETC:TOIN:WORST:LEV?;

TOIWCD

:FETCh|MEASure|READ:TOIN:WORST:LEVel:DIFFerence

	TOI Worst Case Level Difference
Function	Returns the worst case level difference of TOI results.
Remote Command	TOIWCD? :FETCh MEASure READ:TOIN:WORST:LEVel:DIFFerence?
Response Message	Worst Case Level Difference (dBc)
Example	TOIWCD?; FETC:TOIN:WORST:LEV:DIFF?;

TOIWCI

:FETCh|MEASure|READ:TOIN:WORST:LEVel:IP3

	TOI Worst Case IP3
Function	Returns the worst case IP3 of TOI results.
Remote Command	TOIWCI? :FETCh MEASure READ:TOIN:WORST:LEVel:IP3?
Response Message	Worst Case IP3 (dBm)
Example	TOIWCI?; FETC:TOIN:WORST:LEV:IP3?;

TOI3LF

:FETCh|MEASure|READ:TOIN:THIRD:LOWER:FREQuency

	TOI 3-Order Lower Frequency
Function	Returns the lower frequency of TOI results.
Remote Command	TOI3LF? :FETCh MEASure READ:TOIN:THIRD:LOWER:FREQuency?
Response Message	3-Order Lower Freq (Hz)
Example	TOI3LF?; FETC:TOIN:THIRD:LOWER:FREQ?;

TOI3LL

:FETCh|MEASure|READ:TOIN:THIRD:LOWER:LEVel

	TOI 3-Order Lower Level
Function	Returns the lower level of TOI results.
Remote Command	TOI3LL? :FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel?
Response Message	3-Order Level (dBm)
Example	TOI3LL?; FETC:TOIN:THIRD:LOWER:LEV?;

TOI3LD

:FETCh|MEASure|READ:TOIN:THIRD:LOWER:LEVel:DIFFerence

	TOI 3-Order Level Difference
Function	Returns the lower level difference of TOI results.
Remote Command	TOI3LD? :FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel: DIFFerence?
Response Message	3-Order Level Difference (dBc)
Example	TOI3LD?; FETC:TOIN:THIRD:LOWER:LEV:DIFF?;

TOI3LI

:FETCh|MEASure|READ:TOIN:THIRD:LOWER:LEVel:IP3

	TOI 3-Order Lower IP3
Function	Returns the lower IP3 of TOI results.
Remote Command	TOI3LI? :FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel:IP3?
Response Message	3-Order IP3 (dBm)
Example	TOI3LI?; FETC:TOIN:THIRD:LOWER:LEV:IP3?;

TOI3UF

:FETCh|MEASure|READ:TOIN:THIRD:UPPER:FREQuency

	TOI 3-Order Upper Frequency
Function	Returns the upper frequency of TOI results.
Remote Command	TOI3UF? :FETCh MEASure READ:TOIN:THIRD:UPPER:FREQuency?
Response Message	3-Order Upper Freq (Hz)
Example	TOI3UF?; FETC:TOIN:THIRD:UPPER:FREQ?;

TOI3UL

:FETCh|MEASure|READ:TOIN:THIRD:UPPER:LEVel

	TOI 3-Order Upper Level
Function	Returns the upper level of TOI results.
Remote Command	TOI3UL? :FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel?
Response Message	3-Order Level (dBm)
Example	TOI3UL?; FETC:TOIN:THIRD:UPPER:LEV?;

TOI3UD

:FETCh|MEASure|READ:TOIN:THIRD:UPPER:LEVel:DIFFerence

	TOI 3-Order Level Difference
Function	Returns the upper level difference of TOI results.
Remote Command	TOI3UD? :FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel: DIFFerence?
Response Message	3-Order Level Difference (dBc)
Example	TOI3UD?; FETC:TOIN:THIRD:UPPER:LEV:DIFF?;

TOI3UI

:FETCh|MEASure|READ:TOIN:THIRD:UPPER:LEVel:IP3

	TOI 3-Order Upper IP3
Function	Returns the upper IP3 of TOI results.
Remote Command	TOI3UI? :FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel:IP3?
Response Message	3-Order IP3 (dBm)
Example	TOI3UI?; FETC:TOIN:THIRD:UPPER:LEV:IP3?;

Measurement – Spurious Emissions

SEOUT

:FETCh|MEASure|READ:SPURious

	Spurious Emissions
Function	Returns to SE results table.
Remote Command	SEOUT?
	:FETCh MEASure READ:SPURious?
Response Message	Range, Frequency, Amplitude, Limit Level, Limit State
Example	SEOUT?;
	FETC:SPUR?;

[Caution]

You can insert Range Table data after loading the Range Table file (*.spe).

Measurement – Spectrum Emission Mask

SEMMT

[[:SENSe]:SEMAsk:METhod

	Measure Method
Function	Sets the measurement method to Total Power Ref or PSD Ref.
Remote Command	SEMMT sw SEMMT? [:SENSe]:SEMAsk:METhod sw [:SENSe]:SEMAsk:METhod?
Response Message	TPR : Total Power Ref PSDR : Power Spectral Density
Value of sw	TPRef : Total Power Ref
PSDRef	: Power Spectral Density
Initial setting	PSDR
Example	SEMMT PSDR; SEMMT?; SEM:METh PSDR; SEM:METh?;

SEMOUT

:FETCh|MEASure|READ:SEMAsk

	Spectrum Emission Mask Output
Function	Returns to SEM results table.
Remote Command	SEMOUT? :FETCh MEASure READ:SEMAsk?
Response Message	Total Power Ref, PSD Ref, Lower Pk Freq, Lower CHP, Lower PSD, Lower Limit State, Upper Pk Freq, Upper CHP, Upper PSD, Upper Limit State
Example	SEMOUT?; FETC:SEM?;

SEMPWR

:FETCh|MEASure|READ:SEMask:REFerence:CHPower

Function	Total power reference
Remote Command	SEMPWR? :FETCh MEASure READ:SEMask:REFerence:CHPower?
Response Message	Total Power Ref (dBm)
Example	SEMPWR?; FETC:SEMask:REF:CHP?;

SEMTPSD

:FETCh|MEASure|READ:SEMask:REFerence:DENSity

Function	Total power spectral density reference
Remote Command	SEMTPSD? :FETCh MEASure READ:SEMask:REFerence:DENSity?
Response Message	Total Power Spectral Density Ref (dBm/Hz)
Example	SEMTPSD?; FETC:SEMask:REF:DENS?;

SEMLF1~6

:FETCh|MEASure|READ:SEMAsk:LOWER1~6:FREQuency

	SEM Lower Peak Frequency
Function	Returns to lower peak frequency of SEM results.
Remote Command	SEMLF1~6? :FETCh MEASure READ:SEMAsk:LOWER1~6:FREQuency?
Response Message	Lower Pk Frequency (Hz)
Example	SEMLF1?; FETC:SEMAsk:LOWER1:FREQ?;

SEMLPWR1~6

:FETCh|MEASure|READ:SEMask:LOWER1~6:CHPower

	SEM Lower Channel Power
Function	Returns to lower channel power of SEM results.
Remote Command	SEMLPWR1~6? :FETCh MEASure READ:SEMask:LOWER1~6:CHPower?
Response Message	Lower Channel Power (dBm)
Example	SEMLPWR1?; FETC:SEMask:LOWER1:CHPower?;

SEMLPSD1~6

:FETCh|MEASure|READ:SEMask:LOWER1~6:DENSity

	SEM Lower Power Spectral Density
Function	Returns to lower power spectral density of SEM results.
Remote Command	SEMLPSD1~6? :FETCh MEASure READ:SEMask:LOWER1~6:DENSity?
Response Message	Lower Power Spectral Density (dBm/Hz)
Example	SEMLPSD1?; FETC:SEMask:LOWER1:DENS?;

SEMUF1~6

:FETCh|MEASure|READ:SEMAsk:UPPER1~6:FREQuency

	SEM Upper Peak Frequency
Function	Returns to upper peak frequency of SEM results.
Remote Command	SEMUF1~6? :FETCh MEASure READ:SEMAsk:UPPER1~6:FREQuency?
Response Message	Upper Pk Frequency (Hz)
Example	SEMUF1?; FETC:SEMAsk:UPPER1:FREQ?;

SEMUPWR1~6

:FETCh|MEASure|READ:SEMask:UPPER1~6:CHPower

	SEM Upper Channel Power
Function	Returns to upper channel power of SEM results.
Remote Command	SEMUPWR1~6? :FETCh MEASure READ:SEMask:UPPER1~6:CHPower?
Response Message	Upper Channel Power (dBm)
Example	SEMUPWR1?; FETC:SEMask:UPPER1:CHPower?;

SEMUPSD1~6

:FETCh|MEASure|READ:SEMask:UPPER1~6:DENSity

SEM Upper Power Spectral Density

Function	Returns to upper power spectral density of SEM results.
Remote Command	SEMUPSD1~6? :FETCh MEASure READ:SEMask:UPPER1~6:DENSity?
Response Message	Upper Power Spectral Density (dBm/Hz)
Example	SEMUPSD1?; FETC:SEMask:UPPER1:DENS?;

[Caution]

You can insert Offset Mask Data after loading Offset Mask File (*.sem).

Measurement – Average Power (Burst Power)

BPMT

[:SENSe]:BPOWer:METhod

	Measure Method
Function	Sets measurement method to Threshold or Bandwidth.
Remote Command	BPMT sw BPMT? [:SENSe]:BPOWer:METhod sw [:SENSe]:BPOWer:METhod?
Response Message	THR : Threshold BWID : Bandwidth
Value of sw	THReshold : Threshold BWIDth : Bandwidth
Initial setting	THR
Example	BPMT THR; BPMT?; BPOW:METh THR; BPOW:METh?;

BPTH

[[:SENSe]:BPOWer:THReshold

Function	Threshold
Remote Command	BPTH f BPTH? [:SENSe]:BPOWer:THReshold f [:SENSe]:BPOWer:THReshold?
Response Message	Threshold (dB)
Value of f	–200 to 0.01 (dB)
Suffix	: dB DB : dB
Initial setting	–30 dB
Example	BPTH -30; BPTH –30DB?; BPTH?; BPOW:THR –30; BPOW:THR HR –30DB; BPOW:THR:THR?;

BPOUT

:FETCh|MEASure|READ:BPOWer

	Average Power Output
Function	Returns to BP Result Table.
Remote Command	BPOUT? :FETCh MEASure READ:BPOWer?
Response Message	Average Power, Min Power, Max Power (dBm)
Example	BPOUT?; FETC:BPOW?;

BPAVERP

:FETCh|MEASure|READ:BPOWER:POWER:AVERage

Function	Average power
Remote Command	BPAVERP? :FETCh MEASure READ:BPOWER:POWER:AVERage?
Response Message	Average Power (dBm)
Example	BPAVERP?; FETC:BPOW:POW:AVER?;

BPMINP

:FETCh|MEASure|READ:BPOWER:POWER:MIN

Function	Minimum power
Remote Command	BPMINP? :FETCh MEASure READ:BPOWER:POWER:MIN?
Response Message	Min Power (dBm)
Example	BPMINP?; FETC:BPOW:POW:MIN?;

BPMAXP

:FETCh|MEASure|READ:BPOWER:POWER:MAX

Function	Maximum power
Remote Command	BPMAXP? :FETCh MEASure READ:BPOWER:POWER:MAX?
Response Message	Max Power (dBm)
Example	BPMAXP?; FETC:BPOW:POW:MAX?;

BPBL

:FETCh|MEASure|READ:BPOWer:BWIDth

Function	Burst length
Remote Command	BPBL? :FETCh MEASure READ:BPOWer:BWIDth?
Response Message	Burst Length (sec)
Example	BPBL?; FETC:BPOW:BWID?;

Measurement - Total Power

TPOUT[C|D]

FETCh|MEASure|READ:TPower[:CHPower|DENSity]

	TP Measurement Output
Function	Returns the total power level value
Remote Command	TPOUT[C D]? FETCh MEASure READ:TPower[:CHPower DENSity]?
Response Message	Total Power, Total Power Spectral Density
Example	TPOUT?; TPOUTC?; TPOUTD?; FETC:TP?; FETC:TP:CHP?; FETC:TP:DENS?;

Peak Search

MPK[1~9]

:CALCulate:MARKer[1~9]:MAXimum

	Peak Search
Function	Places the selected marker on the highest point of the marker trace.
Remote Command	MPK[1~9] :CALCulate:MARKer[1~9]:MAXimum
Example	MPK; MPK5; CALC:MARK:MAX; CALC:MARK5:MAX;

MPKN[1~9]

:CALCulate:MARKer[1~9]:MAXimum:NEXT

	Next Peak Search
Function	Places the selected marker on the next highest point of the marker trace.
Remote Command	MPKN[1~9] :CALCulate:MARKer[1~9]:MAXimum:NEXT
Example	MPKN; MPKN5; CALC:MARK:MAX:NEXT; CALC:MARK5:MAX:NEXT;

MPKL[1~9]

:CALCulate:MARKer[1~9]:MAXimum:LEFT

	Next Left Peak Search
Function	Places the selected marker on the next left peak point of the marker trace.
Remote Command	MPKL[1~9] :CALCulate:MARKer[1~9]:MAXimum:LEFT
Example	MPKL; MPKL5; CALC:MARK:MAX:LEFT; CALC:MARK5:MAX:LEFT;

MPKR[1~9]

:CALCulate:MARKer[1~9]:MAXimum:RIGHt

	Next Right Peak Search
Function	Places the selected marker on the next right peak point of the marker trace.
Remote Command	MPKR[1~9] :CALCulate:MARKer[1~9]:MAXimum:RIGHt
Example	MPKR; MPKR5; CALC:MARK:MAX:RIGH; CALC:MARK5:MAX:RIGH;

MPKM

:CALCulate:MARKer[1~9]:MINinum

	Minimum Search
Function	Places the selected marker on the minimum level point of the marker trace.
Remote Command	MPKM[1~9] :CALCulate:MARKer[1~9]:MINinum
Example	MPKM; MPKM5; CALC:MARK:MIN; CALC:MARK5:MIN;

MPKP

:CALCulate:MARKer[1~9]:PTPeak

	Peak to Peak Search
Function	Places the selected reference marker on the minimum level point and places the selected delta marker on the maximum level point.
Remote Command	MPKP[1~9] :CALCulate:MARKer[1~9]:PTPeak
Example	MPKP; MPKP5; CALC:MARK:PTP; CALC:MARK5:PTP;

MPKT

:CALCulate:MARKer[1~9]:TRCKing[:STATe]

	Signal Track
Function	Activates the selected marker and sets the active marker to the maximum peak point on the selected trace. Sets the center frequency to the selected marker frequency. This function is done after sweep, thus maintaining the marker value at the center frequency.
Remote Command	MPKT[1~9] n MPKT[1~9] sw MPKT[1~9]? :CALCulate:MARKer[1~9]:TRCKing[:STATe] n :CALCulate:MARKer[1~9]:TRCKing[:STATe] sw :CALCulate:MARKer[1~9]:TRCKing[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	MPKT 1; MPKT5 ON; MPKT5?; CALC:MARK:TRCK 1; CALC:MARK5:TRCK ON; CALC:MARK5:TRCK?;

MPKC

:CALCulate:MARKer[1~9]:CPEak[:STATe]

	Continuous Peak	
Function	Activates the selected marker and sets the active marker to the maximum peak point on the selected trace. This function is done after sweep, thus maintaining the marker value at the center frequency.	
Remote Command	MPKC[1~9] n MPKC[1~9] sw MPKC[1~9]? :CALCulate:MARKer[1~9]:CPEak[:STATe] n :CALCulate:MARKer[1~9]:CPEak[:STATe] sw :CALCulate:MARKer[1~9]:CPEak[:STATe]?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	MPKC 1; MPKC5 ON; MPKC?; CALC:MARK:CPE 1; CALC:MARK5:CPE ON; CALC:MARK5:CPE?;	

MMPKN

:CALCulate:MARKer:PEAK:MULTi:NUMber

	Marker Multi Peak Number
Function	Sets the multi peak number.
Remote Command	MKPKN n MKPKN? :CALCulate:MARKer:PEAK:MULTi:NUMber n :CALCulate:MARKer:PEAK:MULTi:NUMber?
Response Message	Multi Peak Number
Value of n	1 to 9
Initial setting	9
Example	MMPKN 5; MMPKN?; CALC:MARK:PEAK:MULT:NUM 5; CALC:MARK:PEAK:MULT:NUM?;

MMPK

:CALCulate:MARKer:PEAK:MULTi

	Marker Multi Peak
Function	Searches Multi Peak and places each marker.
Remote Command	MMPK :CALCulate:MARKer:PEAK:MULTi
Example	MMPK; CALC:MARK:PEAK:MULT;

MMPKT

:CALCulate:MARKer:PEAK:MULTi:TRACe

	Marker Multi Peak Trace
Function	Sets the multi peak trace.
Remote Command	MKPKT n MKPKT? :CALCulate:MARKer:PEAK:MULTi:TRACe n :CALCulate:MARKer:PEAK:MULTi:TRACe?
Response Message	Multi Peak Trace
Value of n	1 to 3
Initial setting	1
Example	MMPKT 1; MMPKT?; CALC:MARK:PEAK:MULT:TRAC 1; CALC:MARK:PEAK:MULT:TRAC?;

MPKE

:CALCulate:MARKer:PEAK:EXCursion

	Marker Peak Search Excursion
Function	Sets the peak least amplitude for peak search. It is valid when MPKPA is set to PAR.
Remote Command	MPKE f MPKE? :CALCulate:MARKer:PEAK:EXCursion f :CALCulate:MARKer:PEAK:EXCursion?
Response Message	Marker Peak Search Excursion (dB)
Value of f	0.03 dB to 210 dB
Initial setting	3 dB
Example	MPKE 3; MPKE 6DB?; MPKE? CALC:MARK:PEAK:EXC 3; CALC:MARK:PEAK:EXC 6DB; CALC:MARK:PEAK:EXC?;

MPKTH

:CALCulate:MARKer:PEAK:THReshold

	Marker Peak Search Threshold
Function	Sets the low limit line for peak search. It is valid when MPKPA is set to PAR.
Remote Command	MPKTH f MPKTH? :CALCulate:MARKer:PEAK:THReshold f :CALCulate:MARKer:PEAK:THReshold?
Response Message	Marker Peak Search Threshold (dBm)
Value of f	Ref level to -210 dB
Suffix code	None : dBm DBM : dBm DBMV : dBmV DBUV : dBuV DBMA : dBmA DBUA : dBuA V : V MV : mV (10 ⁻³ V) UV : uV (10 ⁻⁶ V) NV : nV (10 ⁻⁹ V) PV : pV (10 ⁻¹² V) W : W MW : mW (10 ⁻³ W) UW : uW (10 ⁻⁶ W) NW : nW (10 ⁻⁹ W) PW : pW (10 ⁻¹² W) FW : fW (10 ⁻¹⁵ W) A : A MA : mA (10 ⁻³ A) UA : uA (10 ⁻⁶ A) NA : nA (10 ⁻⁹ A) PA : pA (10 ⁻¹² A)
Initial setting	-100 dBm
Example	MPKTH -80; MPKTH -100DBM?; MPKTH?; CALC:MARK:PEAK:THR -80; CALC:MARK:PEAK:THR -100DBM; CALC:MARK:PEAK:THR?;

MPKPA

:CALCulate:MARKer:PEAK:SEARch:MODE

	Marker Peak Mode
Function	Sets peak mode to Parameter Or Maximum.
Remote Command	MPKPA sw MPKPA? :CALCulate:MARKer:PEAK:SEARch:MODE sw :CALCulate:MARKer:PEAK:SEARch:MODE?
Response Message	PAR : Parameter MAX : MAXimum
Value of sw	PARameter : Parameter MAXimum : Maximum
Initial setting	PAR
Example	MPKPA PAR; MPKPA?; CALC:MARK:PEAK:SEAR:MODE PAR; CALC:MARK:PEAK:SEAR:MODE?;

Preset

PRST

:SYSTem:PRESet

	Preset
Function	Executes a preset. All instrument parameters are set to default values.
Remote Command	PRST :SYSTem:PRESet
Example	PRST; SYST:PRES;

Printer

HCOPY

:HCOPY[:IMMEDIATE]

	Hard Copy
Function	Prints entire screen image.
Remote Command	HCOPY :HCOPY[:IMMEDIATE]
Example	HCOPY; HCOP;

Span

SP

[[:SENSe]:FREQuency:SPAN

	Span
Function	Sets the span.
Remote Command	SP f SP? [:SENSe]:FREQuency:SPAN f [:SENSe]:FREQuency:SPAN?
Response Message	Span (Hz)
Value of f	0 Hz, 3 Hz to 3.0 GHz / 0 Hz, 3 Hz to 13.2 GHz / 0 Hz, 3 Hz to 26.5 GHz
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	3.0 GHz / 13.2 GHz / 26.5 GHz
Example	SP 123456; SP 50MHZ; SP ?; FREQ:SPAN 123456; FREQ:SPAN 50MHZ; FREQ:SPAN?;

FS

[[:SENSe]:FREQuency:SPAN:FULL

	Full Span
Function	Sets the full span.
Remote Command	FS
	[[:SENSe]:FREQuency:SPAN:FULL
Example	FS;
	FREQ:SPAN:FULL;

ZS

[[:SENSe]:FREQuency:SPAN:ZERO

	Zero Span
Function	Sets the zero frequency span.
Remote Command	ZS
	[[:SENSe]:FREQuency:SPAN:ZERO
Example	ZS;
	FREQ:SPAN:ZERO;

LS

[[:SENSe]:FREQuency:SPAN:PREVious

	Last Span
Function	Changes to two times the previous span. Span is varied in a range that allows the center frequency to be maintained.
Remote Command	LS [:SENSe]:FREQuency:SPAN:PREVious
Example	LS; FREQ:SPAN:PREV;

ZI

[[:SENSe]:FREQuency:SPAN:ZIN

	Zoom-In
Function	Changes to 50% of the current span.
Remote Command	ZI
	[[:SENSe]:FREQuency:SPAN:ZIN
Example	ZI;
	FREQ:SPAN:ZIN;

ZO

[[:SENSe]:FREQuency:SPAN:ZOUT

	Zoom-Out
Function	Changes to 200% of the current span.
Remote Command	ZO
	[[:SENSe]:FREQuency:SPAN:ZOUT
Example	ZO;
	FREQ:SPAN:ZOUT;

Sweep

ST

[:SENSe]:SWEep:TIME

	Sweep Time
Function	Sets the sweep time.
Remote Command	ST f ST? [:SENSe]:SWEep:TIME f [:SENSe]:SWEep:TIME?
Response Message	Sweep Time (s)
Value of f	5 ms to 2000 s : Sweep mode 1 μ s to 2000 s : Zero Span mode
Suffix code t	None : s (10^0) kSEC : ks (10^3) SEC : s (10^0) MSEC : ms (10^{-3}) USEC : μ s (10^{-6}) NSEC : ns (10^{-9}) PSEC : ps (10^{-12})
Initial setting	100 ms
Example	ST 100; ST 50MSEC ST?; SWE:TIME 100; SWE:TIME 50MSEC; SWE:TIME?;

STA

[[:SENSe]:SWEep:TIME:AUTO

	Sweep Time Auto	
Function	Sets the sweep time mode to the auto or the manual mode.	
Remote Command	STA n STA sw STA? [[:SENSe]:SWEep:TIME:AUTO n [[:SENSe]:SWEep:TIME:AUTO sw [[:SENSe]:SWEep:TIME:AUTO?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	STA 1; STA ON; STA? SWE:TIME:AUTO 1; SWE:TIME:AUTO ON; SWE:TIME:AUTO?;	

CO

:INITiate:CONTInuous

	Continuous Sweep
Function	Sets the continuous sweep mode. Repeats the current sweep
Remote Command	CO :INITiate:CONTInuous
Example	CO; INIT:CONT;

SI

:INITiate[:IMMediate]

	Single Sweep
Function	Sets the single sweep mode. After the current sweep, stops.
Remote Command	SI :INITiate[:Immediate]
Example	SI; INIT;

STAC

:INITiate:ACCuracy

	Sweep Time Accuracy
Function	Sets the sweep time accuracy to the Accuracy mode or the normal mode. Accuracy mode sets Sweep Time to 200% of Normal mode.
Remote Command	STAC sw STAC? :INNiate:ACCuracy sw :INNiate:ACCuracy?
Response Message	ACC : Accuracy NORM : Normal
Value of sw	ACC : Accuracy NORM : Normal
Initial setting	NORM
Example	STAC ACC; STAC? INIT:ACC ACC; INIT:ACC? PO [:SENSe]:SWEep:POINts Points
Function	Sets the points. This value affects the resolution of the spectrum.
Remote Command	PO n PO? [:SENSe]:SWEep:POINts n [:SENSe]:SWEep:POINts?
Response Message	Points
Value of n	101 to 8192 : Sweep Mode 3 to 8192 : Zero Span Mode
Initial setting	551
Example	PO 551; PO?; SWE:POIN 551; SWE:POIN?;

Trace

TRF[1~3]

:TRACe[1~3]:MODE

	Trace Status
Function	Sets the trace status.
Remote Command	TRF sw TRF? :TRACe[1~3]:MODE sw :TRACe[1~3]:MODE?
Response Message	WRIT : Clear & Wirte MAXH : Max Hold MINH : Min Hold VIEW : View BLAN : Blank
Value of sw	WRITe : Clear & Wirte MAXHold : Max Hold MINHold : Min Hold VIEW : View BLANk : Blank
Initial setting	WRIT : Trace A BLAN : Trace B BLAN : Trace C
Example	TRF WRIT; TRF3 MAXH; TRF3? TRAC:MODE WRIT; TRAC3:MODE MAXH; TRAC3:MODE?;

TRD

:TRACe[:DATA]

	Send/Query Trace Data
Function	Sends the trace data or queries the trace data. The number of Send/Query Data changes, depending on the sweep point.
Remote Command	TRD sw,data1,data2,data3..... TRD? sw :TRACe[:DATA] sw :TRACe[:DATA]? sw
Response Message	Data[0],Data[1],Data[2].....(Num : Points)
Value of sw	TRACE1 : Trace A TRACE2 : Trace B TRACE3 : Trace C
Example	TRD TRACE1,-80,-70,-50,-40,-50,-60,-70,-80; TRD? TRACE1; TRAC TRACE1,-80,-70,-50,-40,-50,-60,-70,-80; TRAC? TRACE1;

TDF

:TRACe:FORMat

	Trace Format
Function	Sets the trace format.
Remote Command	TDF sw TDF? :TRACe:FORMat sw :TRACe:FORMat?
Response Message	ASC : ASCii Code REAL,64 : 8 Byte Real INT,32 : 4 Byte Integer
Value of sw	ASCii : Ascii Code REAL,64 : 8 Byte Real INTger,32 : 4 Byte Integer
Initial setting	ASCii Code
Example	TDF ASC; TDF? TRAC:FORM ASC; TRAC:FORM?;

Trigger

TSO

:TRIGger[:SEQuence]:SOURce

	Trigger Source
Function	Sets the trigger switch and source.
Remote Command	TSO sw TSO? :TRIGger[:SEQuence]:SOURce sw :TRIGger[:SEQuence]:SOURce?
Response Message	IMM : Selects the Free-run mode VID : Selects the Video mode LINE : Selects the Line mode EXT : Selects the External mode
Value of sw	IMMediate : Selects the Free-run mode VIDeo : Selects the Video mode LINE : Selects the Line mode EXTernal : Selects the External mode
Initial setting	IMM
Example	TSO IMM; TSO VID; TSO?; TRIG:SOUR IMM; TRIG:SOUR VID; TRIG:SOUR?;

TVL

:TRIGger[:SEQuence]:VIDeo:LEVel

	Trigger Video Level
Function	Sets the threshold level of sweep for the start trigger when the trigger source is video. Sweep trigger level x is vertical position on graticule.
Remote Command	TVL n TVL? :TRIGger[:SEQuence]:VIDeo:LEVel n :TRIGger[:SEQuence]:VIDeo:LEVel?
Response Message	Trigger Level (dBm)
Value of n	Reference Level to Reference Level-10*Scale/Div
Initial setting	Reference Level
Example	TVL 100; TVL?; TRIG:VID:LEV 100; TRIG:VID:LEV?;

TSL

:TRIGger[:SEQuence]:SLOPe

	Trigger Slope
Function	Selects the trigger slope.
Remote Command	TSL sw TSL? :TRIGger[:SEQuence]:SLOPe sw :TRIGger[:SEQuence]:SLOPe?
Response Message	POS NEG
Value of sw	POSitive NEGative
Initial setting	POS
Example	TSL POS; TSL?; TRIG:SLOP POS; TRIG:SLOP?;

TD

:TRIGger[:SEQuence]:DELay

	Trigger Delay
Function	Sets the delay time from the point where trace time triggering occurs. Available only in zero span mode.
Remote Command	TD f TD? TRIGger[:SEQuence]:DELay f TRIGger[:SEQuence]:DELay?
Response Message	Delay Time
Value of t	-150 ms to 500 ms
Suffix code	None : s (10 ⁰) kSEC : ks (10 ³) SEC : s (10 ⁰) MSEC : ms (10 ⁻³) USEC : μs (10 ⁻⁶)
Initial setting	0
Example	TD 50; TD 100MSEC; TD?; TRIG:DEL 50; TRIG:DEL 100MSEC; TRIG:DEL?;

TDS

:TRIGger[:SEQuence]:DELay:STATe

	Trigger Delay State
Function	Turns Delay Time Mode ON or OFF.
Remote Command	TDS n TDS sw TDS? TRIGger[:SEQuence]:DELay:STATe n TRIGger[:SEQuence]:DELay:STATe sw TRIGger[:SEQuence]:DELay:STATe?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	TDS 1; TDS ON; TDS?; TRIG:DEL:STAT 1; TRIG:DEL:STAT ON; TRIG:DEL:STAT?;

Tune

ATN

[[:SENSe]:TUNE:AUTO

	Auto Tune
Function	Detects the maximum peak point in full span, displays its spectrum in the center of the screen, and then changes to a small span width.
Remote Command	ATN [[:SENSe]:TUNE:AUTO
Example	ATN; TUNE:AUTO

Tracking Generator (option)

TGEN

	Tracking Generator State	
Function	Sets the state of the tracking generator output level.	
Remote Command	TGEN n TGEN sw TGEN?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	TGEN 1; TGEN ON; TGEN?; TGLEV	
	Tracking Generator Output Level	
Function	Sets the output level of the tracking generator.	
Remote Command	TGLEV f TGLEV?	
Response Message	Output Level	
Value of f	-70.0 dBm to 0.0 dBm (step : 0.1 dBm)	
Suffix code	None	: dBm
	DBM	: dBm
	DBMV	: dBmV
	DBUV	: dBuV
	DBMA	: dBmA
	DBUA	: dBuA
	V	: V
	MV	: mV (10 ⁻³ V)
	UV	: uV (10 ⁻⁶ V)
	NV	: nV (10 ⁻⁹ V)
	PV	: pV (10 ⁻¹² V)
	W	: W
	MW	: mW (10 ⁻³ W)
	UW	: uW (10 ⁻⁶ W)
	NW	: nW (10 ⁻⁹ W)
	PW	: pW (10 ⁻¹² W)

	FW	: fW (10^{-15} W)
	A	: A
	MA	: mA (10^{-3} A)
	UA	: uA (10^{-6} A)
	NA	: nA (10^{-9} A)
	PA	: pA (10^{-12} A)
	DBUVM	: dBuV/m
	DBUAM	: dBuA/m
Initial setting		-10.0 dBm
Example	TGLEV	-10;
	TGLEV	-12.5DBM;
	TGLEV?	;

TGNORM

Tracking Generator Normalize

Function	Sets the normalize function of the tracking generator.	
Remote Command	TGNORM n TGNORM sw TGNORM?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	TGNORM 1; TGNORM ON; TGNORM?;	

PWRSWP

	Power Sweep
Function	Sets the power sweep to ON or OFF.
Remote Command	PWRSWP n PWRSWP sw PWRSWP?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	PWRSWP 1; PWRSWP ON; PWRSWP?;

Part 2 Phase noise mode

This section gives detailed descriptions of commands for the instrument when it is used for phase noise measurements.

Amplitude

RL

:DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:RLEVel

	Reference Level
Function	Sets the reference level value.
Remote Command	RL f RL? :DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:RLEVel f :DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:RLEVel?
Response Message	Reference Level (dBc/Hz)
Value of f	-200 dBm to 200 dBm (Step : 0.01 dBm)
Suffix code	None : dBc/Hz DB : dBc/Hz
Initial setting	-70 dBc/Hz
Example	RL -50; RL ?; DISP:LPL:WIND:TRAC:Y:RLEV -50; DISP:LPL:WIND:TRAC:Y:RLEV?;

SD

:DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:PDIVision

	Scale/Divide
Function	Sets the scale/divide value.
Remote Command	SD f SD? :DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:PDIVision f :DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:PDIVision?
Response Message	Scale/Divide (dB/div)
Value of f	0.1dB to 20dB (step : 0.01dB)
Suffix code	None : dB/div DB : dB/div
Initial setting	10 dB/div
Example	SD 5; SD 10DB; SD?; DISP:LPL:WIND:TRAC:Y:PDIV 5; DISP:LPL:WIND:TRAC:Y:PDIV 10DB; DISP:LPL:WIND:TRAC:Y:PDIV?;

IA

[[:SENSe]:POWer[:RF]:GAIN[:STATe]

	Internal Amplifier
Function	Activates the internal amplifier.
Remote Command	IA n IA sw IA? [:SENSe]:POWer[:RF]:GAIN[:STATe] n [:SENSe]:POWer[:RF]:GAIN[:STATe] sw [:SENSe]:POWer[:RF]:GAIN[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	IA 1; IA ON; IA?; POW:GAIN 1; POW:GAIN ON; POW:GAIN?;

Display

FSCR

:DISPlay:LPLot:FSCRen[:STATe]

	Full Screen
Function	Sets the full screen mode.
Remote Command	FSCR n FSCR sw FSCR? :DISPlay:LPLot:FSCRen[:STATe] n :DISPlay:LPLot:FSCRen[:STATe] sw :DISPlay:LPLot:FSCRen[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	1
Example	FSCR 1; FSCR ON; FSCR? DISP:LPL:FSCR 1; DISP:LPL:FSCR ON; DISP:LPL:FSCR?;

GRAT

:DISPlay:LPLot:WINDow:TRACe:GRATicule:GRID[:STATe]

	Graticule
Function	Sets the display graticule to ON or OFF.
Remote Command	GRAT n GRAT sw GRAT? :DISPlay:LPLot:WINDow:TRACe:GRATicule:GRID[:STATe] n :DISPlay:LPLot:WINDow:TRACe:GRATicule:GRID[:STATe] sw :DISPlay:LPLot:WINDow:TRACe:GRATicule:GRID[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	1
Example	GRAT 1; GRAT ON; GRAT? DISP:LPLot:WIND:TRAC:Y:GRAT:GRID 1; DISP:LPLot:WIND:TRAC:Y:GRAT:GRID ON; DISP:LPLot:WIND:TRAC:Y:GRAT:GRID?;

ANN

:DISPlay:LPLot:WINDow:ANNotation[:ALL]

	Annotation
Function	Turns the annotation on or off
Remote Command	ANN n ANN sw ANN? :DISPlay:LPLot:WINDow:ANNotation[:ALL] n :DISPlay:LPLot:WINDow:ANNotation[:ALL] sw :DISPlay:LPLot:WINDow:ANNotation[:ALL]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	1
Example	ANN 1; ANN ON; ANN? DISP:LPL:WIND:ANN 1; DISP:LPL:WIND:ANN ON; DISP:LPL:WIND:ANN?;

WH

:DISPlay:LPLot:WINDow:WHITe

	White Mode	
Function	Turns the white mode to ON or OFF.	
Remote Command	WH n	
	WH sw	
	WH?	
	:DISPlay:LPLot:WINDow:WHITe n	
	:DISPlay:LPLot:WINDow:WHITe sw	
	:DISPlay:LPLot:WINDow:WHITe?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	WH 1;	
	WH ON;	
	WH?	
	DISP:LPL:WIND:WHIT 1;	
	DISP:LPL:WIND:WHIT ON;	
	DISP:LPL:WIND:WHIT?;	

File

FREAD

:MMEMory:CATalog

	File Read
Function	Reads files in selected folder.
Remote Command	FREAD? 'file_folder' :MMEMory:CATalog? 'file_folder'
Value of file_folder	File Folder
Response Message	File Name,,File Size.
Example	FREAD? 'C:'; FREAD? 'D:\Temp'; MMEM:CAT? 'C:'; MMEM:CAT? 'D:\Temp';

FSAVE

:MMEMory:STORe

	File Save
Function	Saves the file which type was defined by the extension.
Remote Command	FSAVE 'file_name' :MMEMory:STORe 'file_name'
Value of file_name	File Path + File Name
Supported Extension	sts : Status bmp : Bitmap jpg : jpeg png : png
Example	FSAVE 'C:\demo.sts'; MMEM:STRO 'C:\demo.sts';

FLOAD

:MMEMory:LOAD

	File Load
Function	Loads the selected file.
Remote Command	FLOAD 'file_name' :MMEMory:LOAD 'file_name'
Value of file_name	File Path + File Name
Supported Extension	sts : Status
Example	FLOAD 'C:\demo.sts'; MMEM:LOAD 'C:\demo.sts';

FDEL

:MMEMory:DELeTe

	File Delete
Function	Deletes the selected file.
Remote Command	FDEL 'file_name' :MMEMory:DELeTe 'file_name'
Value of file_name	File Path + File Name
Example	FDEL 'C:\demo.sts'; MMEM:DEL 'C:\demo.sts';

FCOPY

:MMEMory:COPY

	File Copy
Function	Copies the selected file.
Remote Command	FCOPY 'src_file_name', 'dest_file_name' :MMEMory:COPY 'src_file_name', 'dest_file_name'
Value of src_file_name, dest_file_name	File Path + File Name
Example	FCOPY 'C:\demo.sts', 'D:\demo.sts'; MMEM:COPY 'C:\demo.sts', 'D:\demo.sts';

FRENAME

:MMEMory:MOVE

	File Rename
Function	Rename the selected file.
Remote Command	FRENAME 'src_file_name','dest_file_name'
	:MMEMory:MOVE 'src_file_name','dest_file_name'
Value of src_file_name, dest_file_name	File Path + File Name
Example	FRENAME 'C:\demo.sts','C:\demo_1.sts'; MMEM:MOVE 'C:\demo.sts','C:\demo_1.sts';

FMOVE

MMEMory:DATA

	File Move
Function	Sends or Received Binary Data of Selected File. Maximum size of sent file is 2 MB, and maximum size of received file is 30 MB.
Remote Command	FMOVE 'file_name',definite_length_block FMOVE? 'file_name' MMEMory:DATA 'file_name',definite_length_block MMEMory:DATA? 'file_name'
Value of file_name	File Path + File Name
Value of definite_length_block	# + number of file size + file size + file data
Example	FMOVE 'C:\Sended_Sample.txt',#14abcd; cf) #+1+4+abcd FMOVE? 'C:\Received_Sample.txt'; MMEM:DATA 'C:\ Sended_Sample.txt',#14abcd; MMEM:DATA? 'C:\ Received_Sample.txt';

Frequency

CF

[[:SENSE]:LPLot:FREQUENCY:CARRIER]

	Carrier Frequency
Function	Sets the carrier frequency.
Remote Command	CF f CF? [:SENSE]:LPLot:FREQUENCY:CARRIER f [:SENSE]:LPLot:FREQUENCY:CARRIER?
Response Message	Carrier Frequency (Hz) (Range : 3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz)
Value of f	3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	1.5 GHz / 6.6 GHz / 13.25 GHz
Example	CF 123456; CF 50MHZ; CF?; LPL:FREQ:CARR 123456; LPL:FREQ:CARR 50MHZ; LPL:FREQ:CARR?;

CFS

[[:SENSe]:LPLot:FREQuency:CARRier:SEARch

Carrier Frequency Search

Function Searches the carrier frequency.

Remote Command CFS

[[:SENSe]:LPLot:FREQuency:CARRier:SEARch

Example CFS;

LPL:FREQ:CARR:SEAR;

Marker

MS[1~9]

:CALCulate:LPLot:MARKer[1~9]:STATe

	Marker State
Function	Sets the selected marker state.
Remote Command	MS[1~9] n MS[1~9] sw MS[1~9]? :CALCulate:LPLot:MARKer[1~9]:STATe n :CALCulate:LPLot:MARKer[1~9]:STATe sw :CALCulate:LPLot:MARKer[1~9]:STATe?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	MS 1; MS5 1; MS5?; CALC:LPL:MARK:STAT 1; CALC:LPL:MARK5:STAT ON; CALC:LPL:MARK5:STAT?

MM[1~9]

:CALCulate:LPLot:MARKer[1~9]:MODE

	Marker Mode
Function	Sets the selected marker to Normal, Delta, Band Pair Mode.
Remote Command	MM[1~9] sw MM[1~9]? :CALCulate:LPLot:MARKer[1~9]:MODE sw :CALCulate:LPLot:MARKer[1~9]:MODE?
Response Message	POS : Normal DELT : Delta RMSD : RMS Noise (Degree) RMSR : RMS Noise (Radian) RMSJ : RMS Noise (Jitter) RFM : Residual FM
OFF	: OFF
Value of sw	POSition : Normal DELTA : Delta RMSDeree : RMS Noise (Degree) RMSRadian : RMS Noise (Radian) RMSJitter : RMS Noise (Jitter) RFM : Residual FM OFF : OFF
Initial setting	OFF
Example	MM POS; MM5 DELT; MM5?; CALC:LPL:MARK:MODE POS; CALC:LPL:MARK5:MODE DELT; CALC:LPL:MARK5:MODE?

MF[1~9]

:CALCulate:LPLot:MARKer[1~9]:X

	Marker Frequency
Function	Sets the marker frequency of the selected marker. If the marker mode is delta mode, sets the difference value of the marker frequency and the delta marker frequency.
Remote Command	MF[1~9] f MF[1~9]? :CALCulate:LPLot:MARKer[1~9]:X f :CALCulate:LPLot:MARKer[1~9]:X?
Response Message	Marker Frequency ()
Value of f	Start Offset Frequency to Stop Offset Frequency
Suffix code	None : (10 ⁰) HZ : (10 ⁰) KHZ : (10 ³) MHZ : (10 ⁶) GHZ : (10 ⁹)
Initial setting	Center Position of Display Window
Example	MF 123456; MF5 1GHZ; MF5?; CALC:MARK:X 123456; CALC:MARK5:X 1GHZ; CALC:MARK5:X?

MA[1~9]

:CALCulate:LPLot:MARKer[1~9]:Y

	Marker Amplitude
Function	Returns the amplitude data.
Remote Command	MA[1~9]? :CALCulate:LPLot:MARKer[1~9]:Y?
Response Message	Marker Amplitude
Example	MA?; MA5? CALC:LPL:MARK:Y? CALC:LPL:MARK5:Y?

MT[1~9]

:CALCulate:LPLot:MARKer[1~9]:TRACe

	Select Marker Trace	
Function	Selects the marker trace.	
Remote Command	MT[1~9] n MT[1~9]? :CALCulate:LPLot:MARKer[1~9]:MKT n :CALCulate:LPLot:MARKer[1~9]:MKT?	
Response Message	1	: Trace A (Measured)
	2	: Trace B (Smoothed)
Value of n	1	: Trace A (Measured)
	2	: Trace B (Smoothed)
Initial setting	1	
Example	MT 2; MT5 2; MT5?; CALC:LPL:MARK:TRAC 2; CALC:LPL:MARK5:TRAC 2; CALC:LPL:MARK5:TRAC?;	

MTB

:CALCulate:LPLot:MARKer:TABLE:STATe

	Marker Table State	
Function	Sets the marker table state.	
Remote Command	MTB n	
	MTB sw	
	MTB?	
	:CALCulate:LPLot:MARKer:TABLE:STATe n	
	:CALCulate:LPLot:MARKer:TABLE:STATe sw	
Response Message	:CALCulate:LPLot:MARKer:TABLE:STATe?	
	1	: ON
Value of n	0	: OFF
	1	: ON
Value of sw	0	: OFF
	ON	: ON
Initial setting	OFF	: OFF
	0	
Example	MTB 1;	
	MTB ON;	
	MTB?;	
	CALC:LPL:MARK:TABL:STAT 1;	
	CALC:LPL:MARK:TABL:STAT ON;	
CALC:LPL:MARK:TABL:STAT?;		

MAO

:CALCulate:LPLot:MARKer:AOff

	Marker All OFF
Function	Turns off All of the marker.
Remote Command	MAO :CALCulate:LPLot:MARKer:AOff
Example	MAO; CALC:LPL:MARK:AOff;

Measurement

LP

:MEASure:LPLot

	Log Plot
Function	Starts Phase Noise Measurement.
Remote Command	LP :MEASure:LPLot
Example	LP; MEAS:LPL;

AVG

[[:SENSe]:LPLot:AVERage[:STATe]

	Average State	
Function	Sets the measurement mode to averaging.	
Remote Command	AVG n	
	AVG sw	
	AVG?	
	[:SENSe]:LPLot:AVERage[:STATe] n	
	[:SENSe]:LPLot:AVERage[:STATe] sw	
	[:SENSe]:LPLot:AVERage[:STATe]?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	AVG 1;	
	AVG ON;	
	AVG?	
	LPL:AVER 1;	
	LPL:AVER ON;	
	LPL:AVER?;	

AVGC

[[:SENSe]:LPLot:AVERage:COUNT

	Average Count
Function	Sets the measurement mode to averaging count.
Remote Command	AVGC n AVGC? [:SENSe]:LPLot:AVERage:COUNT n [:SENSe]:LPLot:AVERage:COUNT?
Response Message	Average Count
Value of n	1 to 1000
Initial setting	10
Example	AVGN 20; AVGN? LPL:AVER:COUN 20; LPL:AVER:COUN?;

SM

[[:SENSe]:LPLot:SMOoth

	Smoothing
Function	Sets Smoothing Value.
Remote Command	SM f SM? [:SENSe]:LPLot:SMOoth f [:SENSe]:LPLot:SMOoth?
Response Message	Smoothing Value
Value of f	0% to 16%
Initial setting	4 (%)
Example	SM 4; SM? LPL:SMO 4; LPL:SMO?;

FT

[[:SENSE]:LPLot:FILTERing

	Filtering
Function	Sets Filtering Value.
Remote Command	FT f FT? [:SENSE]:LPLot:FILTERing f [:SENSE]:LPLot:FILTERing?
Response Message	Filtering Value
Value of f	1.0, 0.3, 0.1, 0.03
Initial setting	0.1
Example	FT 0.1; FT? LPL:FILT 0.1; LPL:FILT?;

DTB

:CALCulate:LPLot:DECade:TABLE[:STATE]

	Decade Table State	
Function	Sets the decade table state.	
Remote Command	DTB n	
	DTB sw	
	DTB?	
	:CALCulate:LPLot:DECade:TABLE[:STATE] n	
	:CALCulate:LPLot:DECade:TABLE[:STATE] sw	
	:CALCulate:LPLot:DECade:TABLE[:STATE]?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	DTB 1;	
	DTB ON;	
	DTB?;	
	CALC:LPL:DEC:TABL 1;	
	CALC:LPL:DEC:TABL ON;	
	CALC:LPL:DEC:TABL?;	

Mode

MODE

:INSTrument[:SElect]

	Mode
Function	Sets Current Mode.
Remote Command	MODE sw MODE? :INSTrument[:SElect] sw :INSTrument[:SElect]?
Response Message	SA : Spectrum Mode PNOISE : Phase Noise Mode
Value of sw	SA : Spectrum Mode PNOISE : Phase Noise Mode
Initial setting	SA
Example	MODE SA; MODE?; INST SA; INST?;

Preset

PRST

:SYSTem:PRESet

	Preset
Function	Executes preset. All instrument parameters are set to default values.
Remote Command	PRST :SYSTem:PRESet
Example	PRST; SYST:PRES;

Printer

HCOPY

:HCOPY[:IMMEDIATE]

	Hard Copy
Function	Prints entire screen image.
Remote Command	HCOPY :HCOPY[:IMMEDIATE]
Example	HCOPY; HCOP;

Span

FA

[[:SENSE]:LPLot:FREQuency:OFFSet:STARt

	Start Offset Frequency
Function	Sets the start offset frequency.
Remote Command	FA f FA? [:SENSE]:LPLot:FREQuency:OFFSet:STARt f [:SENSE]:LPLot:FREQuency:OFFSet:STARt?
Response Message	Start Offset Frequency (Hz)
Value of f	10 Hz to 1.0 GHz
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	1 kHz
Example	FA 123456; FA 1MHZ; FA?; LPL:FREQ:OFFS:STAR 123456; LPL:FREQ:OFFS:STAR 1MHZ; LPL:FREQ:OFFS:STAR?;

FB

[[:SENSE]:LPLot:FREQuency:OFFSet:STOP

	Stop Offset Frequency
Function	Sets the stop offset frequency.
Remote Command	FB f FB? [:SENSE]:LPLot:FREQuency:OFFSet:STOP f [:SENSE]:LPLot:FREQuency:OFFSet:STOP?
Response Message	Stop Offset Frequency (Hz)
Value of f	10 Hz to 1.0 GHz
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	1 MHz
Example	FB 1234567; FB 10MHZ; FB?; LPL:FREQ:OFFS:STOP 1234567; LPL:FREQ:OFFS:STOP 10MHZ; LPL:FREQ:OFFS:STOP?;

Sweep

CO

:INITiate:LPLot:CONTinuous

	Continuous Sweep
Function	Sets the continuous sweep mode. Repeats acting sweep
Remote Command	CO :INITiate:LPLot:CONTinuous
Example	CO; INIT:LPL:CONT;

SI

:INITiate:LPLot[:IMMEDIATE]

	Single Sweep
Function sweep.	Sets the single sweep mode. After acting sweep, stop repeating
Remote Command	SI :INITiate:LPLot[:IMMEDIATE]
Example	SI; INIT:LPL;

Trigger

TSO

:TRIGger[:SEQuence]:SOURce

	Trigger Source
Function	Sets the trigger switch and the trigger source.
Remote Command	TSO sw TSO? :TRIGger[:SEQuence]:SOURce sw :TRIGger[:SEQuence]:SOURce?
Response Message	IMM : Selects the Free-run mode LINE : Selects the Line mode EXT : Selects the External mode
Value of sw	IMMEDIATE : Selects the Free-run mode LINE : Selects the Line mode EXTernal : Selects the External mode
Initial setting	IMM
Example	TSO IMM; TSO?; TRIG:SOUR IMM; TRIG:SOUR?;

Part 3 CCDF mode

This section gives detailed descriptions of commands for the instrument when it is used for measurements involving complementary cumulative distribution function plots.

Amplitude

IA

[[:SENSe]:POWer[:RF]:GAIN[:STATe]

	Internal Amplifier
Function	Activates the internal amplifier.
Remote Command	IA n IA sw IA? [:SENSe]:POWer[:RF]:GAIN[:STATe] n [:SENSe]:POWer[:RF]:GAIN[:STATe] sw [:SENSe]:POWer[:RF]:GAIN[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	IA 1; IA ON; IA?; POW:GAIN 1; POW:GAIN ON; POW:GAIN?;

Bandwidth

RB

[[:SENSe]:BANDwidth|BWIDth[:RESolution]

	Resolution Bandwidth
Function	Sets the RBW value.
Remote Command	RB f RB? [:SENSe]:BANDwidth BWIDth[:RESolution] f [:SENSe]:BANDwidth BWIDth[:RESolution]?
Response Message	Resolution Bandwidth (Hz)
Value of f	1 Hz to 5 MHz (Step : 1, 2, 3, 5)
Suffix code f	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	5 MHz
Example	RB 1000; RB 3KHZ; RB? BAND 1000; BAND 3KHZ; BAND?;

VB

[[:SENSE]:BANDwidth|BWIDth:VIDeo

	Video Bandwidth
Function	Sets the VBW value.
Remote Command	VB f VB? [:SENSE]:BANDwidth BWIDth:VIDeo f [:SENSE]:BANDwidth BWIDth:VIDeo?
Response Message	Video Bandwidth (Hz)
Value of f	1 Hz to 3 MHz (Step : 1, 2, 3, 5)
Suffix code f	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	3 MHz
Example	VB 1000; VB 3KHZ; VB? BAND:VID 1000; BAND:VID 3KHZ; BAND:VID?;

VBA

[[:SENSe]:BANDwidth|BWIDth:VIDeo:AUTO

	Video Bandwidth Auto	
Function	Sets the VBW mode to the auto mode or the manual mode.	
Remote Command	VBA n VBA sw VBA? [:SENSe]:BANDwidth BWIDth[:RESolution]:VIDeo:AUTO n [:SENSe]:BANDwidth BWIDth[:RESolution]:VIDeo:AUTO sw [:SENSe]:BANDwidth BWIDth[:RESolution]:VIDeo:AUTO?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	VBA 1; VBA ON; VBA? BAND:VID:AUTO 1; BAND:VID:AUTO ON; BAND:VID:AUTO?;	

VBRB

[[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio

	VBW/RBW
Function	Selects the ratio between VBW and RBW.
Remote Command	VBRB f VBRB? [:SENSe]:BANDwidth BWIDth:VIDeo:RATio f [:SENSe]:BANDwidth BWIDth:VIDeo:RATio?
Response Message	Ratio of VBW/RBW
Value of f	0.000001 to 2000000
Initial setting	1
Example	VBRB 1; VBRB? BAND:VID:RAT 1; BAND:VID:RAT?;

Display

FSCR

:DISPlay:CCDF:FSCRen[:STATe]

	Full Screen
Function	Sets the full screen mode.
Remote Command	FSCR n FSCR sw FSCR? :DISPlay:CCDF:FSCRen[:STATe] n :DISPlay:CCDF:FSCRen[:STATe] sw :DISPlay:CCDF:FSCRen[:STATe]?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	1
Example	FSCR 1; FSCR ON; FSCR? DISP:CCDF:FSCR 1; DISP:CCDF:FSCR ON; DISP:CCDF:FSCR?;

GRAT

:DISPlay:CCDF:WINDow:TRACe:GRATicule:GRID[:STATe]

	Graticule	
Function	Sets the display graticule to ON or OFF.	
Remote Command	GRAT n	
	GRAT sw	
	GRAT?	
	:DISPlay:CCDF:WINDow:TRACe:GRATicule:GRID[:STATe] n	
	:DISPlay:CCDF:WINDow:TRACe:GRATicule:GRID[:STATe] sw	
	:DISPlay:CCDF:WINDow:TRACe:GRATicule:GRID[:STATe]?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	1	
Example	GRAT 1;	
	GRAT ON;	
	GRAT?	
	DISP:CCDF:WIND:TRAC:Y:GRAT:GRID 1;	
	DISP:CCDF:WIND:TRAC:Y:GRAT:GRID ON;	
	DISP:CCDF:WIND:TRAC:Y:GRAT:GRID?;	

WH

:DISPlay:CCDF:WINDow:WHITe

	White Mode	
Function	Turns the white mode to ON or OFF.	
Remote Command	WH n	
	WH sw	
	WH?	
	:DISPlay:CCDF:WINDow:WHITe n	
	:DISPlay:CCDF:WINDow:WHITe sw	
	:DISPlay:CCDF:WINDow:WHITe?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	WH 1;	
	WH ON;	
	WH?	
	DISP:CCDF:WIND:WHIT 1;	
	DISP:CCDF:WIND:WHIT ON;	
	DISP:CCDF:WIND:WHIT?;	

SREF

:DISPlay:CCDF:WINDow:RLEVel:STORe

	Store Reference Trace
Function	Store Reference Trace.
Remote Command	SREF
	:DISPlay:CCDF:WINDow:RLEVel:STORe?
Example	SREF;
	DISP:CCDF:WIND:RLEV:STOR;
	REFS
	:DISPlay:CCDF:WINDow:RLEVel[:STATe]
	Reference Trace State
Function	Turns the Reference Trace to ON or OFF.
Remote Command	REFS n
	REFS sw
	REFS?
	:DISPlay:CCDF:WINDow:RLEVel[:STATe] n
	:DISPlay:CCDF:WINDow:RLEVel[:STATe] sw
	:DISPlay:CCDF:WINDow:RLEVel[:STATe]?
Response Message	1 : ON
	0 : OFF
Value of n	1 : ON
	0 : OFF
Value of sw	ON : ON
	OFF : OFF
Initial setting	0
Example	REFS 1;
	REFS ON;
	REFS?
	DISP:CCDF:WIND:RLEV 1;
	DISP:CCDF:WIND:RLEV ON;
	DISP:CCDF:WIND:RLEV?;

GAUS

:DISPlay:CCDF:WINDow:GAUSSian[:STATe]

	Gaussian Trace State	
Function	Turns the Gaussian Trace to ON or OFF.	
Remote Command	GAUS n	
	GAUS sw	
	GAUS?	
	:DISPlay:CCDF:WINDow:GAUSSian[:STATe] n	
	:DISPlay:CCDF:WINDow:GAUSSian[:STATe] sw	
	:DISPlay:CCDF:WINDow:GAUSSian[:STATe]?	
Response Message	1	: ON
	0	: OFF
Value of n	1	: ON
	0	: OFF
Value of sw	ON	: ON
	OFF	: OFF
Initial setting	0	
Example	GAUS 1;	
	GAUS ON;	
	GAUS?	
	DISP:CCDF:WIND:GAUS 1;	
	DISP:CCDF:WIND:GAUS ON;	
	DISP:CCDF:WIND:GAUS?;	

File

FREAD

:MMEMory:CATalog

	File Read
Function	Reads files in selected folder.
Remote Command	FREAD? 'file_folder' :MMEMory:CATalog? 'file_folder'
Value of file_folder	File Folder
Response Message	File Name,,File Size.
Example	FREAD? 'C:'; FREAD? 'D:\Temp'; MMEM:CAT? 'C:'; MMEM:CAT? 'D:\Temp';

FSAVE

:MMEMory:STORe

	File Save
Function	Saves the file which type was defined by the extension.
Remote Command	FSAVE 'file_name' :MMEMory:STORe 'file_name'
Value of file_name	File Path + File Name
Supported Extension	sts : Status bmp : Bitmap jpg : jpeg png : png
Example	FSAVE 'C:\demo.sts'; MMEM:STRO 'C:\demo.sts';

FLOAD

:MMEMory:LOAD

	File Load
Function	Loads the selected file.
Remote Command	FLOAD 'file_name' :MMEMory:LOAD 'file_name'
Value of file_name	File Path + File Name
Supported Extension	sts : Status
Example	FLOAD 'C:\demo.sts'; MMEM:LOAD 'C:\demo.sts'; FDEL :MMEMory:DElete File Delete
Function	Deletes the selected file.
Remote Command	FDEL 'file_name' :MMEMory:DElete 'file_name'
Value of file_name	File Path + File Name
Example	FDEL 'C:\demo.sts'; MMEM:DEL 'C:\demo.sts';

FCOPY

:MMEMory:COPY

	File Copy
Function	Copies the selected file.
Remote Command	FCOPY 'src_file_name', 'dest_file_name' :MMEMory:COPY 'src_file_name', 'dest_file_name' Value of src_file_name, dest_file_name File Path + File Name
Example	FCOPY 'C:\demo.sts', 'D:\demo.sts'; MMEM:COPY 'C:\demo.sts', 'D:\demo.sts';

FRENAME

:MMEMory:MOVE

	File Rename
Function	Rename the selected file.
Remote Command	FRENAME 'src_file_name','dest_file_name' :MMEMory:MOVE 'src_file_name','dest_file_name' Value of src_file_name, dest_file_name File Path + File Name
Example	FRENAME 'C:\demo.sts','C:\demo_1.sts'; MMEM:MOVE 'C:\demo.sts','C:\demo_1.sts';

FMOVE

MMEMory:DATA

	File Move
Function	Sends or receives binary data of selected file. Maximum size of sent file is 2 Mbyte, and maximum size of received file is 30 Mbyte.
Remote Command	FMOVE 'file_name',definite_length_block FMOVE? 'file_name' MMEMory:DATA 'file_name',definite_length_block MMEMory:DATA? 'file_name'
Value of file_name	File Path + File Name
Value of definite_length_block	# + number of file size + file size + file data
Example	FMOVE 'C:\Sended_Sample.txt',#14abcd; cf) #+1+4+abcd FMOVE? 'C:\Received_Sample.txt'; MMEM:DATA 'C:\ Sended_Sample.txt',#14abcd; MMEM:DATA? 'C:\ Received_Sample.txt';

Frequency

CF

[[:SENSe]:FREQuency:CENTer

	Center Frequency
Function	Sets the center frequency.
Remote Command	CF f CF? [:SENSe]:FREQuency:CENTer f [:SENSe]:FREQuency:CENTer?
Response Message	Center Frequency (Hz) (Range : 3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz)
Value of f	3 Hz to 3.0 GHz / 3 Hz to 13.2 GHz / 3 Hz to 26.5 GHz
Suffix code	None : Hz (10 ⁰) HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	1.5 GHz / 6.6 GHz / 13.25 GHz
Example	CF 123456; CF 50MHZ; CF?; FREQ:CEN7T 123456; FREQ:CENT 50MHZ; FREQ:CENT?;

FO

[[:SENSe]:FREQuency:OFFSet

	Frequency Offset
Function	Sets the frequency offset value.
Remote Command	FO f FO? [:SENSe]:FREQuency:OFFSet f [:SENSe]:FREQuency:OFFSet?
Response Message	Frequency Offset (Hz)
Value of f	-1000 GHz to 1000 GHz
Suffix code	NONE : Hz HZ : Hz (10 ⁰) KHZ : kHz (10 ³) MHZ : MHz (10 ⁶) GHZ : GHz (10 ⁹)
Initial setting	0 Hz
Example	FO 123456; FO 3410.7MHZ; FO?; FREQ:OFFS 123456; FREQ:OFFS 3410.7MHZ; FREQ:OFFS?;

Marker

MS[1~9]

:CALCulate:CCDF:MARKer[1~9]:STATe

	Marker State
Function	Sets the selected marker state.
Remote Command	MS[1~9] n MS[1~9] sw MS[1~9]? :CALCulate:CCDF:MARKer[1~9]:STATe n :CALCulate:CCDF:MARKer[1~9]:STATe sw :CALCulate:CCDF:MARKer[1~9]:STATe?
Response Message	1 : ON 0 : OFF
Value of n	1 : ON 0 : OFF
Value of sw	ON : ON OFF : OFF
Initial setting	0
Example	MS 1; MS5 1; MS5?; CALC:CCDF:MARK:STAT 1; CALC:CCDF:MARK5:STAT ON; CALC:CCDF:MARK5:STAT?

MM[1~9]

:CALCulate:CCDF:MARKer[1~9]:MODE

	Marker Mode
Function	Sets the selected marker to Normal, Delta, Band Pair Mode.
Remote Command	MM[1~9] sw MM[1~9]?
:	CALCulate:CCDF:MARKer[1~9]:MODE sw :CALCulate:CCDF:MARKer[1~9]:MODE?
Response Message	POS : Normal DELT : Delta OFF : OFF
Value of sw	POSition : Normal DELTa : Delta OFF : OFF
Initial setting	OFF
Example	MM POS; MM5 DELT; MM5?; CALC:CCDF:MARK:MODE POS; CALC:CCDF:MARK5:MODE DELT; CALC:CCDF:MARK5:MODE?

MA[1~9]

:CALCulate:CCDF:MARKer[1~9]:X

	Marker Amplitude
Function	Sets Marker Level (X).
Remote Command	MA[1~9] f MA[1~9]? :CALCulate:CCDF:MARKer[1~9]:X f :CALCulate:CCDF:MARKer[1~9]:X?
Response Message	Marker Level (dB)
Value of f	0 to Scale/Div*10
Initial setting	Scale/Div*5
Example	MA 10; MA5 10DB; MA5?; CALC:CCDF:MARK:X 10; CALC:CCDF:MARK:X 10DB; CALC:CCDF:MARK5:X?

MP[1~9]

:CALCulate:CCDF:MARKer[1~9]:Y

	Marker Percent
Function	Returns to Marker Percent (Y).
Remote Command	MP[1~9]? :CALCulate:CCDF:MARKer[1~9]:Y?
Response Message	Marker Percent (%)
Example	MP?; MP5?; CALC:CCDF:MARK:Y?; CALC:CCDF:MARK5:Y?

MT[1~9]

:CALCulate:CCDF:MARKer[1~9]:TRACe

	Select Marker Trace	
Function	Selects the marker trace.	
Remote Command	MT[1~9] n	
	MT[1~9]?	
	:CALCulate:CCDF:MARKer[1~9]:MKT n	
	:CALCulate:CCDF:MARKer[1~9]:MKT?	
Response Message	1	: Trace A (Measured)
	2	: Trace B (Gaussian)
	3	: Trace C (Reference)
Value of n	1	: Trace A (Measured)
	2	: Trace B (Gaussian)
	3	: Trace C (Reference)
Initial setting	1	
Example	MT 2; MT5 2; MT5?; CALC:CCDF:MARK:TRAC 2; CALC:CCDF:MARK5:TRAC 2; CALC:CCDF:MARK5:TRAC?;	

MAO

:CALCulate:CCDF:MARKer:AOff

	Marker All OFF
Function	Turns off All of the marker.
Remote Command	MAO :CALCulate:CCDF:MARKer:AOff
Example	MAO; CALC:CCDF:MARK:AOff;

Measurement

MEA

:MEASure:STARt

	Measure Start
Function	Starts the measurement.
Remote Command	MEA sw MEA? :MEASure:STARt sw :MEASure:STARt?
Response Message	XDB : X dB Down ACP : Adjacent Channel Power CHP : Channel Power OBW : Occupied Bandwidth HD : Harmonic Distribution CCDF : CCDF
Value of sw	XDB : X dB Down ACP : Adjacent Channel Power CHP : Channel Power OBW : Occupied Bandwidth HD : Harmonic Distribution CCDF : CCDF
Example	MEA XDB; MEA?; MEAS:STAR XDB; MEAS:STAR?;

MEAO

:MEASure:AOff

	Measure OFF
Function	Stops the measurement.
Remote Command	MEAO :MEASure:AOff
Example	MEAO; MEAS:AOff;

Measurement — CCDF

CCDFN

[[:SENSe]:CCDF:COUNT

	CCDF Count
Function	Sets CCDF Count.
Remote Command	CCDFN f CCDFN? [:SENSe]:CCDF:COUNT f [:SENSe]:CCDF:COUNT?
Response Message	CCDF Count
Value of f	1,000 to 4,000,000
Initial setting	10,000
Example	CCDFN 1000; CCDFN?; CCDF:COUN 1000; CCDF:COUN?;

CCDFOUT[PWR|PER]

FETCh|MEASure|READ:CCDF[::POWer|PERCent]

	CCDF Measurement Output
Function	Returns the Average Power and Average Percent.
Remote Command	CCDFOUT[PWR PER]? FETCh MEASure READ:CCDF[:POWer PERCent]?
Response Message	Average Power, Average Percent
Example	CCDFOUT?; CCDFOUTPWR?; CCDFOUTPER?; FETC:CCDF?; FETC:CCDF:POW?; FETC:CCDF:PER?;

CCDFPER10

FETCh|MEASure|READ:CCDF:PERCent10

	CCDF Level when is 10%
Function	Returns Level when is 10%.
Remote Command	CCDFPER10?
	FETCh MEASure READ:CCDF:PERCent10?
Response Message	Level when is 10%
Example	CCDFPER10?;
	FETC:CCDF:PERC10?;

CCDFPER1

FETCh|MEASure|READ:CCDF:PERCent1

	CCDF Level when is 1%
Function	Returns Level when is 1%.
Remote Command	CCDFPER1?
	FETCh MEASure READ:CCDF:PERCent1?
Response Message	Level when is 1%
Example	CCDFPER1?;
	FETC:CCDF:PERC1?;

CCDFPER01

FETCh|MEASure|READ:CCDF:PERCent01

	CCDF Level when is 0.1%
Function	Returns Level when is 0.1%.
Remote Command	CCDFPER01?
	FETCh MEASure READ:CCDF:PERCent01?
Response Message	Level when is 0.1%
Example	CCDFPER01?;
	FETC:CCDF:PERC01?;

CCDFPER001

FETCh|MEASure|READ:CCDF:PERCent001

	CCDF Level when is 0.01%
Function	Returns Level when is 0.01%.
Remote Command	CCDFPER001?
	FETCh MEASure READ:CCDF:PERCent001?
Response Message	Level when is 0.01%
Example	CCDFPER001?;
	FETC:CCDF:PERC001?;

CCDFPER0001

FETCh|MEASure|READ:CCDF:PERCent0001

	CCDF Level when is 0.001%
Function	Returns Level when is 0.001%.
Remote Command	CCDFPER0001?
	FETCh MEASure READ:CCDF:PERCent0001?
Response Message	Level when is 0.001%
Example	CCDFPER0001?;
	FETC:CCDF:PERC0001?;

CCDFPER00001

FETCh|MEASure|READ:CCDF:PERCent00001

	CCDF Level when is 0.0001%
Function	Returns Level when is 0.0001%.
Remote Command	CCDFPER00001?
	FETCh MEASure READ:CCDF:PERCent00001?
Response Message	Level when is 0.0001%
Example	CCDFPER00001?;
	FETC:CCDF:PERC00001?;

CCDFCRE

FETCh|MEASure|READ:CCDF:CRESt

	CCDF Crest Level
Function	Returns Crest Level.
Remote Command	CCDFCRE?
	FETCh MEASure READ:CCDF:CRESt?
Response Message	Crest Level
Example	CCDFCRE?;
	FETC:CCDF:CRESt?;

Preset

PRST

:SYSTem:PRESet

	Preset
Function	Executes preset. All instrument parameters are set to default values.
Remote Command	PRST :SYSTem:PRESet
Example	PRST; SYST:PRES;

Printer

HCOPY

:HCOPY[:IMMediate]

	Hard Copy
Function	Prints entire screen image.
Remote Command	HCOPY :HCOPY[:IMMediate]
Example	HCOPY; HCOP;

Span

SD

:DISPlay:CCDF:WINDow:TRACe:Y[:SCALe]:PDIVision

	Scale/Div
Function	Sets Scale/Div.
Remote Command	SD f SD? :DISPlay:CCDF:WINDow:TRACe:Y[:SCALe]:PDIVision f :DISPlay:CCDF:WINDow:TRACe:Y[:SCALe]:PDIVision?
Response Message	Scale/Div
Value of f	0.1 to 20
Initial setting	2 dB/Div
Example	SD 2; SD?; DISP:CCDF:WIND:TRAC:Y:PDIV 2; DISP:CCDF:WIND:TRAC:Y:PDIV?;

Sweep

CO

:INITiate:CCDF:CONTInuous

	Continuous Sweep
Function	Sets the continuous sweep mode. Repeats acting sweep
Remote Command	CO :INITiate:CCDF:CONTInuous
Example	CO; INIT:CCDF:CONT;

SI

:INITiate:CCDF[:IMMEDIATE]

	Single Sweep
Function	Sets the single sweep mode. After activating sweep, stops sweep repeating.
Remote Command	SI :INITiate:CCDF[:Immediate]
Example	SI; INIT:CCDF;

Trigger

TSO

:TRIGger[:SEQuence]:SOURce

	Trigger Source
Function	Sets the trigger switch and the trigger source.
Remote Command	TSO sw TSO? :TRIGger[:SEQuence]:SOURce sw :TRIGger[:SEQuence]:SOURce?
Response Message	IMM : Selects the Free-run mode LINE : Selects the Line mode EXT : Selects the External mode
Value of sw	IMMediate : Selects the Free-run mode LINE : Selects the Line mode EXTernal : Selects the External mode
Initial setting	IMM
Example	TSO IMM; TSO?; TRIG:SOUR IMM; TRIG:SOUR?;

Part 4 GPIB common commands

This section gives detailed descriptions of commands that are common for all measurement modes.

***CLS**

	Clear Status Command
Function	Clears the status byte register.
Remote Command	*CLS
Example	*CLS;

***ESE**

	Standard Event Status Enable
Function	Sets the standard event status enable register.
Remote Command	*ESE n *ESE?
Response Message	Register Value
Value of n	0 to 255 : Represents the sum of the bit-weighted values.
Example	*ESE 20: *ESE?;

***ESR?**

Standard Event Status Register Query

Function	Returns the current value in the standard event status register.
Remote Command	*ESR?
Response Message	Register Value
Example	*ESR?;

***IDN?**

	Identification Query
Function	Returns the model name, etc of the equipment
Remote Command	*IDN?
Response Message	Company, Model, Serial, Version
Example	*IDN?;

***OPC**

Operation Complete Command

Function Sets the standard event register bit 0 to 1 when the requested action is complete.

Remote Command *OPC

Example *OPC;

***OPC?**

	Operation Complete Query
Function	Sets the output queue to 1 to generate a MAV summary message when all pending select device operations have completed.
Remote Command	*OPC?
Response Message	1
Example	*OPC?;

***RST**

	Rest Command
Function	Resets the device.
Remote Command	*RST
Example	*RST;

***SRE**

	Service Request Enable Command
Function	Sets the bits in the service request enable register.
Remote Command	*SRE n *SRE?
Response Message	Register Value
Value of n	0 to 255 : Represents the sum of the bit-weighted values.
Example	*SRE 32; *SRE?;

***STB?**

Returns Status Byte Command

Function

Returns the current values of the status bytes including the MSS bit.

Remote Command

*STB?

Response Message

Register Value

Bit	Bit Weight	Bit Name	Condition of status byte register
7	128	----	0 = Not used
6	64	MSS	0 = Service not requested 1 = Service requested
5	32	ESB	0 = Event status not generated 1 = Event status generated
4	16	MAV	0 = No data in output queue 1 = Data in output queue
3	8	ESB2	0 = Event status not generated 1 = Event status generated
2	4	----	0 = Not used
1	2	----	0 = Not used
0	1	----	0 = Not used

Example

*STB?;

GPIB Common Command - Others

ESE2

	Event Status Enable (End)
Function	Allows the End Event Status Enable Register to select which bit in the corresponding Event Register cause a TRUE ESB summary message bit 3 when set.
Remote Command	ESE2 n ESE2?
Response Message	Register Value
Value of n	0 to 255 : Represents the sum of the bit-weighted values.
Example	ESE2 1; ESE2?;

ESR2?

Event Status Register (End) Query

Function Allows the sum of binary-weighted event bit values of the End Event Status Register to be read out by converting them to decimal. After readout, the End Event status Register is reset to 0.

Remote Command ESR2?

Response Message Register Value

Bit	Bit Weight	Event	Description
7	128	Not used	Not used
6	64	Not used	Not used
5	32	Not used	Not used
4	16	Measurement completed	Measurement has completed (Peak search, OBW, X dB, Noise marker, Freq. Counter, Limit Pass/Fail..)
3	8	AUTO TUNE completed	AUTO TUNE has completed.
2	4	Averaging completed	Sweeping according to the specified AVERAGE number has completed.
1	2	Calibration completed	Temp Cal, Pre-Filter Cal, ZNC Cal., Level Cal.. has completed.
0	1	Sweep completed	A single sweep has completed or is in standby.

Example ESR2?;

ERR

	Error Code
Function	Returns the error code of the current function. The error code is cleared.
Remote Command	ERR?
Response Message	Error code
Example	ERR?;

Chapter 5

STATUS STRUCTURE

Contents

General	5-1
Status Byte (STB) Register.....	5-4
ESB and MAV Summary Message	5-4
Device-Dependent Summary Message.....	5-5
Reading and clearing the STB Register.....	5-6
Reading by serial polling (only when the GPIB interface bus is used)	5-6
Reading by the *STB? common query.....	5-6
Definition of MSS (Master Summary Message)	5-6
Clearing the STB register using the *CLS common command.....	5-6
Service Request (SRQ) enabling operation	5-7
Reading the SRE register	5-7
Updating the SRE register.....	5-7
Standard Event Status Register.....	5-8
Bit definition of Standard Event Status Register.....	5-8
Reading, writing, and clearing the Standard Event Status Register	5-9
Reading, writing, and clearing the Standard Event Status Enable Register	5-9
Extended Event Status Register.....	5-10
Bit Definition of END Event Status Register	5-11
Reading, writing, and clearing the Extended Event Status Register	5-12
Reading, writing, and clearing the Extended Status Enable Register.....	5-12

General

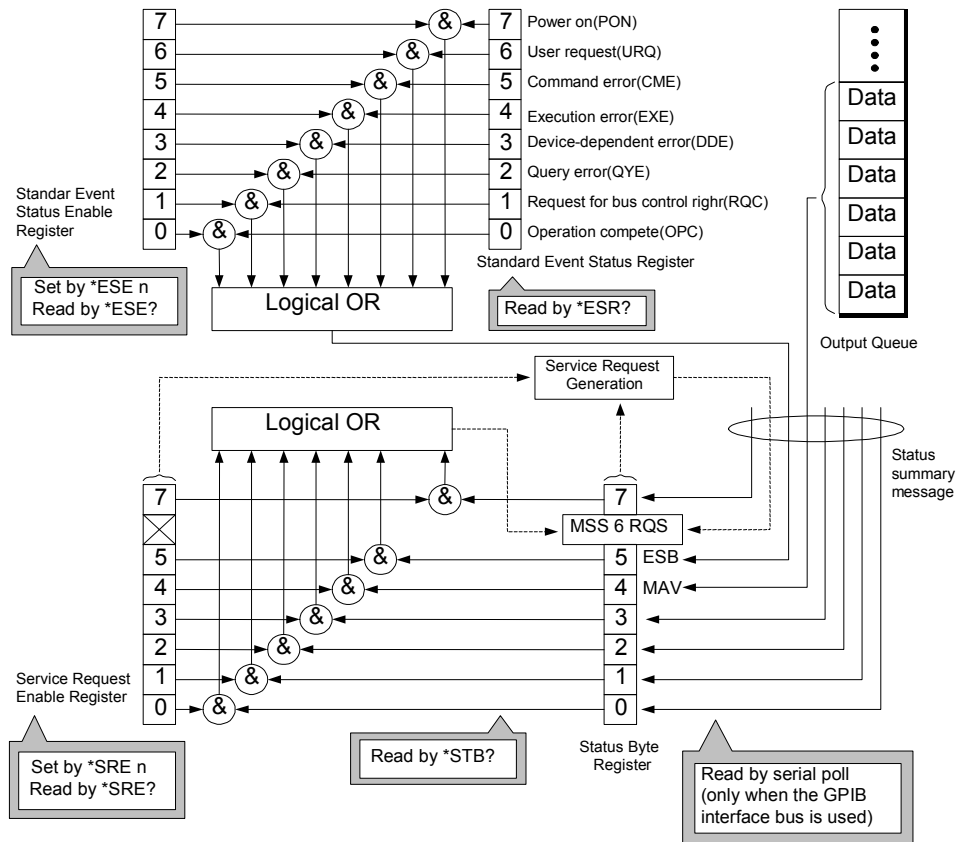
This chapter describes device-status reporting and its data structure defined by IEEE488.2 when a GPIB interface bus is used. This chapter also describes the synchronization techniques between a controller and device.

These functions are used to control a device from an external controller using the GPIB interface bus. Most of these functions can also be used to control a device from an external controller using the RS-232C interface.

The Status Byte (STB) sent to the controller is based on the IEEE488.2 standard. The bits comprising the STB are called status summary messages because they represent a summary of the current data in registers and queues.

IEEE488.2 standard status model

The diagram below shows the standard model for the status data structures stipulated in the IEEE488.2 standard.



In the status model, IEEE488.2 status bytes are used for the lowest grade status. This status byte is composed of seven summary message bits from the higher grade status structure. To create these summary message bits, the status data structure is composed of two types of registers and models.

Register Model	Queue Model
<p>The register model consists of two registers used for recording events and conditions encountered by a device. These two registers are the Event Status Register and Event Status Enable Register. When the results of the AND operation of both register contents are 1, the corresponding bit of the status bit becomes 1. In other cases, the corresponding bit becomes 0. When the result of their Logical OR is 1, the summary message bit also becomes 1. If the Logical OR result is 0, the summary message bit also becomes 0.</p> <p>The other register model, which consists of status Byte Register and Service Request Enable Register, has the same format as above.</p>	<p>The queue in the queue model is used to sequentially record the waiting status values or information. If the queue is not empty, the queue structure summary message becomes 1. If the queue is empty, the message becomes 0.</p>

In IEEE488.2, there are three standard models for the status data structure. Two are register models and one is a queue model based on the register model and queue model described above. The three standard models are:

STATUS STRUCTURE

- Standard Event Status Register and Standard Event Status Enable Register
- Status Byte Register and Service Request Enable Register Output Queue.
- Output queue.

Standard Event Status Register	Status Byte Register	Output Queue
<p>The Standard Event Status Register has the same structure as the previously described register model.</p> <p>In this register, the bits for eight types of standard events encountered by a device are set as follows:</p> <ul style="list-style-type: none"> Power on User request Command error Execution error Device-dependent error Query error Request for bus control right Operation complete <p>The Logical-OR output bit is represented by Status Byte Register bit 5 (DIO6) as a summary message for the Event Status Bit (ESB)</p>	<p>The Status Byte Register is a register in which the RQS bit and the seven summary message bits from the status data structure can be set. This register is used together with the Service Request Enable Register. When the results of the OR operation of both register contents are other than 0, SRQ becomes ON. To indicate this, bit 6 of the Status Byte Register (DIO7) is reserved by the system as the RQS bit. The RQS bit is used to indicate that there is a service request for the external controller. The mechanism of SRQ conforms to the IEEE488.2 standard.</p>	<p>The Output Queue has the structure of the queue model described above.</p> <p>Status Byte Register bit 4 (DIO5) is set as a summary message for Message Available (MAV) to indicate that there is data in the output buffer.</p>

Status Byte (STB) Register

The STB register consists of the STB and RQS (or MSS) messages of the device.

ESB and MAV Summary Message

This paragraph describes the ESB and MAV summary message.

ESB Summary Message

The ESB (Event Summary Bit) is a message defined by IEEE488.2 which uses bit 5 of the STB register. When the setting permits events to occur, the ESB summary message bit becomes 1 if any one of the events recorded in the Standard Status Register becomes 1. Conversely, the ESB summary message bit becomes 0 if one of recorded events occurs, even if events are set to occur.

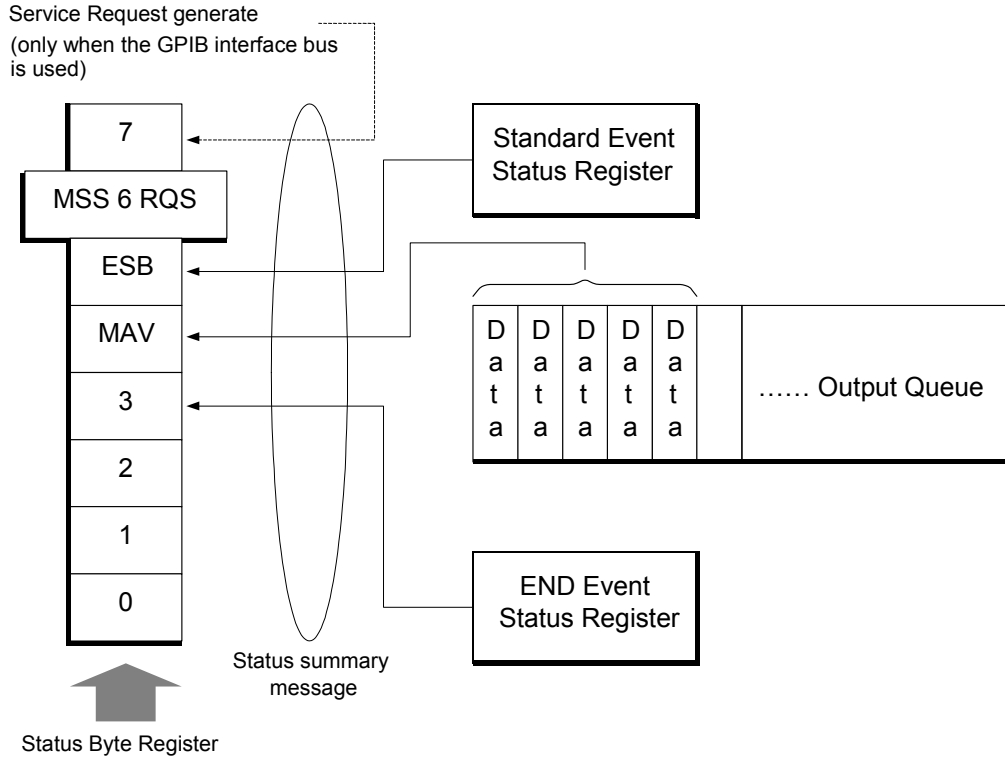
This bit becomes 0 when the ESR register is read by the *ESR? Query or when it is cleared by the *CLS command.

MAV Summary Message

The MAV (Message Available) summary bit is a message defined by IEEE488.2 that uses bit 4 of the STB register. This bit indicates whether the output queue is empty. The MAV summary message bit is set to 1 when a device is ready to receive a request for a response message from the controller. When the output queue is empty, this bit is set to 0. This message is used to synchronize the information exchange with the controller. For example, this message is available when, after the controller sends a query command to a device, the controller waits until MAV becomes 1. While the controller is waiting for a response from the device, other jobs can be processed. Reading the Output Queue without first checking MAV causes all system bus operations to be delayed until the device responds.

Device-Dependent Summary Message

As shown below, the system does not use bits 0, 1, 2 and 7, and it uses bit 3 as the summary bit of the Event Status Register.



Reading and clearing the STB Register

The STB register can be read using serial polling or the *STB? common query.

The IEEE488.2 STB message can be read by either method, but the value sent to bit 6 (position) is different for each method.

The STB register contents can be cleared using the *CLS command.

Reading by serial polling (only when the GPIB interface bus is used)

The IEEE488.2 serial polling allows the device to return a 7-bit status byte and an RQS message bit which conforms to IEEE488.2. The value of the status byte is not changed by serial polling. The device sets the RQS message to 0 immediately after being polled.

Reading by the *STB? common query

The *STB common query requires the devices to send the contents of the STB register and the integer format response message, including the MSS (Master Summary Status) summary message. Therefore, except for bit 6, which represents the MSS summary message, the response to *STB? is identical to that to serial polling.

Definition of MSS (Master Summary Message)

MSS indicates that there is at least one cause for a service request. The MSS message is represented as a bit 6 response to an *STB? Query, but it is not a status byte specified by IEEE488.2.

MSS is configured by the overall logical OR in which the STB register and SRQ enable (SRE) register are combined.

Clearing the STB register using the *CLS common command

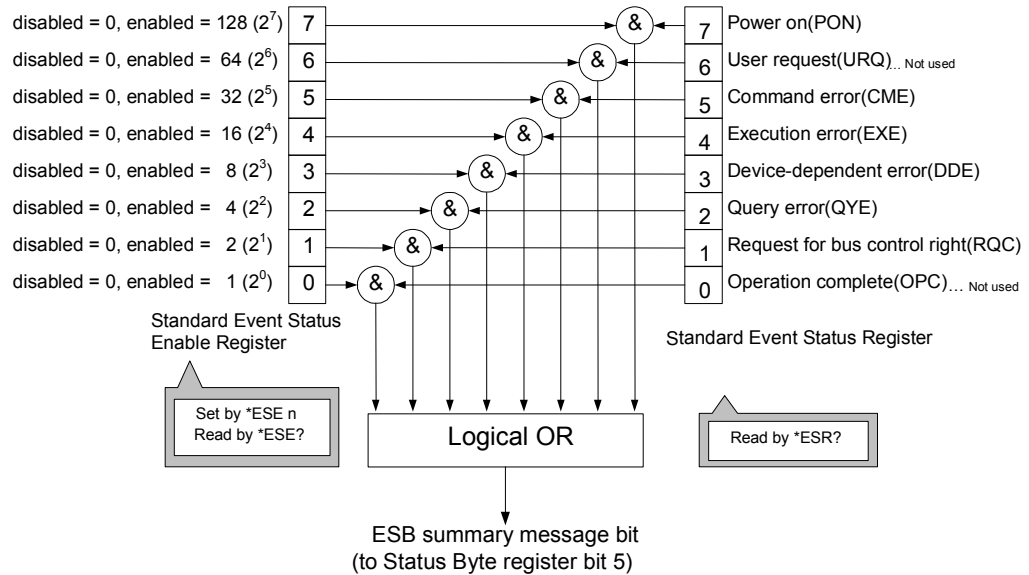
The *CLS common command clears all status data structures as well as the summary message corresponding to them.

The *CLS command does not affect the setting in the Enable Register.

Service Request (SRQ) enabling operation

Bits 0 to 7 of the Service Request Enable Register (SRE) determine which bit of the corresponding STB register can generate SRQ.

The bits in the Service Request Enable Register correspond to the bits in the Status Byte Register. If a bit in the Status Byte Register corresponding to an enabled bit in the Service Request Enable Register is set to 1, the device makes a service request to the controller with the RQS bit set to 1.



Reading the SRE register

The contents of the SRE register are read using the *SRE? Common query.

The response message to this query is an integer from 0 to 255, which is the sum of the bit digit weighted values in the SRE register.

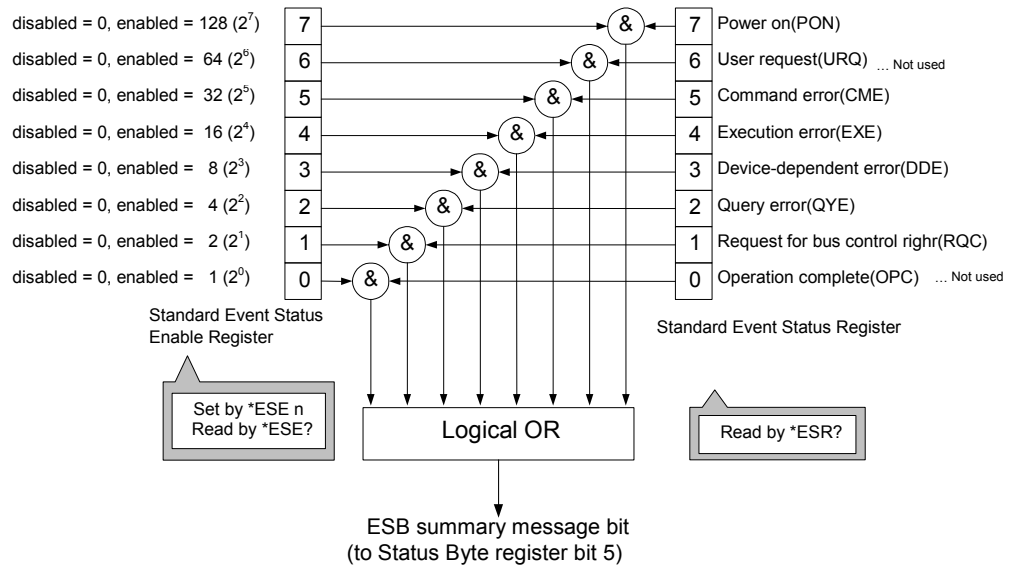
Updating the SRE register

The SRE register is written using the *SRE common command. An integer from 0 to 255 is assigned as a parameter to set the SRE register bit to 0 or 1. The value of bit 6 is ignored.

Standard Event Status Register

Bit definition of Standard Event Status Register

The diagram below shows the operation of the Standard Event Status Register.



The Standard Event Status Enable (ESE) Register on the left is used to select which bits in the corresponding Event Register cause a TRUE summary message when set.

Bit	Event name	Description
7	Power on (PON-Power on)	A transition from power-off to power-on occurred during the power-up procedure.
6	Not used	
5	Command error (CME-Command Error)	An illegal program message or misspelled command was received.
4	Execution error (EXE-Execution Error)	A legal but unexcitable program message was received.
3	Device-dependent error (DDE-Device-dependent Error)	An error not caused by CME, EXE, or QYE occurred (parameter error, etc).
2	Query error (QYE-Query Error)	An attempt was made to read data in the output queue when it was empty. Or, the data in the output queue was lost before it was read.
1	Not used	
0	Operation complete (OPC-Operation Complete)	This bit becomes 1 when this equipment has processed the *OPC command.

Reading, writing, and clearing the Standard Event Status Register

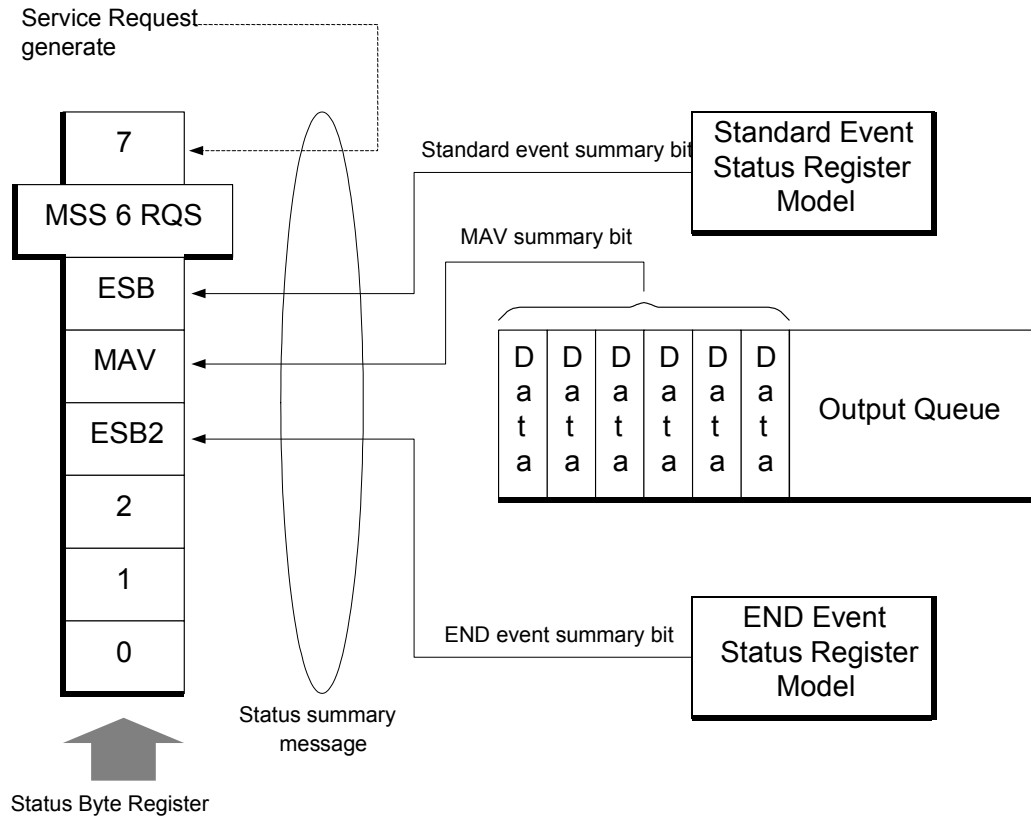
Reading	<p>The register is read using the *ESR? Command query.</p> <p>The register is cleared after being read. The response message is integer - format data with the binary weight added to the event bit and the sum converted to decimal</p>
Writing	<p>With the exception of clearing, data cannot be written to the register from outside.</p>
Clearing	<p>The register is cleared when :</p> <ul style="list-style-type: none"> A *CLS command is received. The power is turned on Bit 7 is set to ON, and the other bits are cleared to 0. An event is read for the *ESR? Query command.

Reading, writing, and clearing the Standard Event Status Enable Register

Reading	<p>The registers is read using the *ESE? Command.</p> <p>The response message is integer-format data with the binary weight added to the event bit and the sum converted to decimal.</p>
Writing	<p>The register is written using the *ESE common command.</p>
Clearing	<p>The register is cleared when :</p> <ul style="list-style-type: none"> An "ESE command with a data value of 0 is received. The power is turned on. <p>The Standard Event Enable Register is not affected when :</p> <ul style="list-style-type: none"> The device clear function status of IEEE488.2 is changed. A *RST common command is received. A *CLS common command is received.

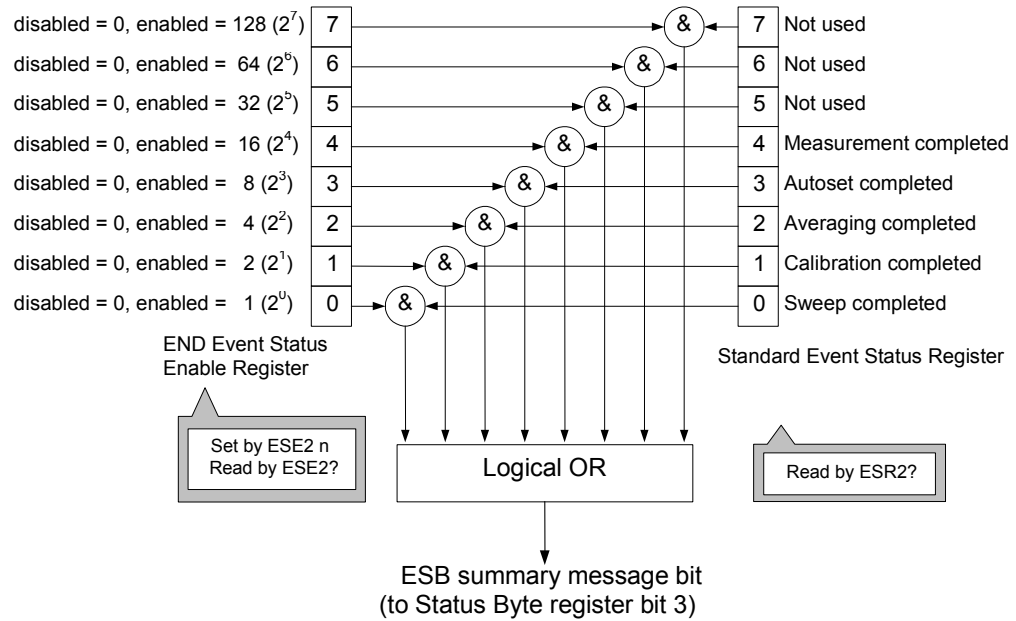
Extended Event Status Register

For the system, bits 7, 2, 1 and 0 are unused. Bit 3 is assigned to the END summary bit as the status-summary bit supplied by the extended register model as shown below.



Bit Definition of END Event Status Register

The diagram below shows the operation and event-bit names of the END Event Status Register.



The END Event Status Enable Register on the left is used to select which bits in the corresponding Event Register will cause a TRUE summary message when set.

Bit	Event name	Description
7	Not used	Not used
6	Not used	Not used
5	Not used	Not used
4	Measurement completed	Calculation processing for measurements (Peak search, OBX, X dB down, Noise marker, Frequency counter Limit pass/fail) has completed.
3	AUTO SET completed	AUTO SET has completed.
2	Averaging completed	Sweeping according to the specified AVERAGE number has completed.
1	Calibration completed	RBW CAL, Power on CAL, All CAL, Temp CAL, Span CAL, Level CAL or LOG CAL has completed.
0	Sweep completed	A single sweep has completed or is in standby.

Reading, writing, and clearing the Extended Event Status Register

Reading	The ESR2? common query is used to read the register. The register is cleared after being read. The response message is integer-format data with the binary weight added to the event bit and the sum converted to decimal.
Writing	With the exception of clearing, data cannot be written to the register from outside.
Clearing	The register is cleared when: A *CLS command is received. The power is turned on when: An event is read for the ESR? query command.

Reading, writing, and clearing the Extended Status Enable Register

Reading	The ESE2? query reads the register. The response message is integer-format data with the binary weight added to the event bit and sum converted to decimals.
Writing	The ESE2 program command is used to write the register. Because bits 0 to 7 of the registers are weighted with values 1, 2, 4, 8, 16, 32, 64 and 128, respectively, the write data is transmitted as integer-format data that is the sum of the required bit digits selected from the weighted value.
Clearing	The register is cleared when: An ESE2 program command with a data value of 0 is received. The power is turned on. The Extended Event Status Enable register is not affected when: The device clear function status of IEEE488.2 is changed. A *RST common command is received. A *CLS common command is received.

Chapter 6

EXAMPLE CODES

Contents

General	6-1
Using MAV bit	6-2
Using ESB bit	6-8
Reading identification	6-14
Reading frequency and level of peak trace	6-20
Reading Delta Marker amplitude	6-27
Measuring XdB Bandwidth	6-34
Measuring Occupied Bandwidth	6-41
Measuring marker noise	6-48
Saving Status File	6-56
Loading Status File	6-62
Reading Real64 data	6-68
Reading Real64 trace data	6-75
Checking Limit Line	6-82
Send File	6-89
Receive File	6-95

General

This section shows some example codes to transmit messages on the bus between a personal computer and the spectrum analyzer via GPIB.

The 'C' language is used with a GPIB card program in the PC to control the spectrum analyzer by using the library produced by the GPIB card manufacturer. For example, here the example cards use the "ni488.h", "gpib-32.obj" library from National Instruments.

Using MAV bit

The MAV bit sets bit 4 of Status Register Enable (SRE) to 1. If you send a query message to the instrument, it prepares a response message and generates the MAV bit in the Status Register (STB). Then the Status Register MSS is generated and the instrument generates an RQS. When you detect the RQS from the instrument, you can read the response message.

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;           /* Device unit descriptor */
int BoardIndex = 0;      /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7;  /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];         /* Read buffer */
char SerialPollResponse = 0; /* Read buffer

void GpibSetup(void);    /* GpibSetup function declaration */
void Send(char *buf);    /* Send function declaration */
void Receive(void);      /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration

void GpibSetup(void)
{
    Device = ibdev(
        BoardIndex,      /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress,  /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s,            /* Timeout setting (T10s = 10 seconds) */
        1,              /* Assert EOI line at end of write */
        0);             /* EOS termination mode
    if (ibsta & ERR)    /* Check for GPIB Error
    {
        GpibError("ibdev Error");
    }

    ibclr(Device);     /* Clear the device
    if (ibsta & ERR)    /* Check for GPIB Error

```

```
{
GpibError("ibclr Error");
}
}

void Send(char *buf)
{
ibwrt(Device, buf, (long)strlen(buf));
/* Send the buffer command */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```
/*
 * Note — you should set Bit4 of Status Register Enable to 1.
 * Send command "*SRE 16" to be able to receive RQS.
 * Tip — if you want to wait a long time, you should eliminate TIMO.
 * Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".
 */
*****

ibwait(Device, (TIMO | RQS));      /* Wait until TIMO or RQS is asserted */
if (ibsta & (TIMO | ERR))         /* Check for GPIB Error */
{
  GpibError("ibwait Error");
}
printf("Detect RQS\n");

ibrsp(Device, &SerialPollResponse);
/* Serial Polling */
if (ibsta & ERR)                 /* Check for GPIB Error */
{
  GpibError("ibrsp Error");
}

ibrd(Device, Buffer, 100);        /* Read up to 100 bytes from the device */
Buffer[ibcntl-1] = '\0';        /* Null terminate the ASCII string */
if (ibsta & ERR)                 /* Check for GPIB Error */
{
  GpibError("ibwrt Error");
}

void main(void)
{
  /*
   * Initialization — done only once at the beginning of your application.
   */
  *****

  GpibSetup();
}
```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Using MAV>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                              /* This enables use of MAV bit of Status Register */
                              /* If MAV bit is 1, MSS is generated. */
                              /* MSS generates RQS. */
                              /* When RQS received, can read message from SA */

Send("CF?");                  /* Identification ? */
printf("Send Command(CF)\n");
Receive();                    /* Read Buffer */
printf("Received Message : %s\n",Buffer);

Send("*CLS");                 /* Clear Register */
printf("<<<End>>>\n");

/*
*****
* Uninitialization — done only once at the end of your application.
*****
*/

ibonl(Device, 0);            /* Take the device offline */
if (ibsta & ERR)              /* Check for GPIB Error */
{
  GpibError("ibonl Error");
}

getchar();                   /* Pause program */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Using ESB bit

Sets bit 5 of the Status Register Enable (SRE) to 1. Sets bit 0 of the Event Status Register Enable (ESE) to 1. If you send an OPC(Operation Complete) message to the instrument, it turns bit 0 of the Event Status Register (ESR) to 1. Then the ESB bit of the Status Register (STB) turns from 0 to 1. An MSS from the Status Register is generated and the instrument generates an RQS. When you detect the RQS from the instrument, you are free to carry out the next action.

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;           /* Device unit descriptor */
int BoardIndex = 0;      /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7;  /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];        /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);    /* GpibSetup function declaration */
void Send(char *buf);    /* Send function declaration */
void Receive(void);      /* Receive function declaration */
bool Wait_RQS(void);     /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{
    Device = ibdev(           /* Create a unit descriptor handle */
        BoardIndex,         /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress,     /* Device primary address */
        SecondaryAddress,   /* Device secondary address */
        T10s,               /* Timeout setting (T10s = 10 seconds) */
        1,                  /* Assert EOI line at end of write */
        0);                 /* EOS termination mode */
    if (ibsta & ERR)         /* Check for GPIB Error */
    {
        GpibError("ibdev Error");
    }
}

```

```
ibclr(Device);                /* Clear the device */
if (ibsta & ERR)              /* Check for GPIB Error */
{
    GpibError("ibclr Error");
}
}

void Send(char *buf)
{
    ibwrt(Device, buf, (long)strlen(buf));
                                /* Send the buffer command */
    if (ibsta & ERR)          /* Check for GPIB Error */
    {
        GpibError("ibwrt Error");
    }
}

void Receive(void)
{
    ibrd(Device, Buffer, 100);  /* Read up to 100 bytes from the device */
    Buffer[ibcntl-1] = '\0';    /* Null terminate the ASCII string */
    if (ibsta & ERR)          /* Check for GPIB Error */
    {
        GpibError("ibwrt Error");
    }
}

bool Wait_RQS(void)
{
```

EXAMPLE CODES

```
/******  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (TIMO | RQS));      /* Wait until TIMO or RQS is asserted   */  
if (ibsta & RQS)                  /* Detect RQS           */  
{  
  ibrsp(Device, &SerialPollResponse);  
                                  /* Serial Polling       */  
  if (ibsta & ERR)                /* Check for GPIB Error  */  
  {  
    GpibError("ibrsp Error");  
  }  
  return true;  
}  
if (ibsta & (TIMO | ERR))         /* Don't detect RQS */  
{  
  return false;  
}  
return false;  
}  
  
void main(void)  
{  
  /******  
  * Initialization — done only once at the beginning of your application.  
  *****/  
  
  GpibSetup();
```

EXAMPLE CODES

```
/******  
* Main Application Body — write the majority of your GPIB code here.  
*****/  
  
printf("\n");  
printf("<<<SECTION 6>>>\n");  
printf("<<<EXAMPLE CODES>>>\n");  
printf("<<<Using ESB>>>\n");  
printf("<<<Start>>>\n");  
  
Send("*CLS;*SRE 32");          /* Clear Register, Status Register Enable : 00100000 */  
                               /* Enables use of the ESB bit of the Status Register */  
                               /* When RQS received, can read a message from the SA*/  
/******  
* Use OPC — if you want to use another bit, you should send correct command.  
* For example, if you detect time of command error,  
* you should send command, "*ESE 32".  
*****/  
  
Send("*ESE 1");              /* Set Event Status Register Enable to 00000001 */  
                               /* Enable user to use OPC bit. */  
  
Send("*OPC");                /* Set Bit0 of Event Status Register to 1 */  
if (Wait_RQS())              /* Success to receive RQS */  
{  
/******  
* Function — when RQS received, send command you want.  
*****/  
  
printf("Detect RQS\n");  
}  
else                          /* Fail to receive RQS */  
{  
printf("No Response\n");  
}  
  
Send("*CLS");                /* Clear Register */  
printf("<<<End>>>\n");
```

EXAMPLE CODES

```

/*****
* Uninitialization — done only once at the end of your application.
*****/

ibonl(Device, 0);          /* Take the device offline */
if (ibsta & ERR)          /* Check for GPIB Error */
{
  GpibError("ibonl Error");
}

getchar();                /* Pause program */

}

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

void GpibError(char *msg)
{
  printf ("%s\n", msg);

  printf ("ibsta = &H%x <", ibsta);
  if (ibsta & ERR ) printf (" ERR");
  if (ibsta & TIMO) printf (" TIMO");
  if (ibsta & END ) printf (" END");
  if (ibsta & SRQI) printf (" SRQI");
  if (ibsta & RQS ) printf (" RQS");
  if (ibsta & CMPL) printf (" CMPL");
  if (ibsta & LOK ) printf (" LOK");
  if (ibsta & REM ) printf (" REM");
}

```

```

if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}

```

Reading identification

Get identification from the instrument.

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;           /* Device unit descriptor */
int BoardIndex = 0;      /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7;  /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];         /* Read buffer */
char SerialPollResponse = 0; /* Read buffer

void GpibSetup(void);    /* GpibSetup function declaration */
void Send(char *buf);    /* Send function declaration */
void Receive(void);      /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration

void GpibSetup(void)
{
    Device = ibdev(
        BoardIndex,      /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress,  /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s,            /* Timeout setting (T10s = 10 seconds) */
        1,               /* Assert EOI line at end of write */
        0);              /* EOS termination mode
    if (ibsta & ERR)      /* Check for GPIB Error
    {
        GpibError("ibdev Error");
    }

    ibclr(Device);      /* Clear the device
    if (ibsta & ERR)      /* Check for GPIB Error
    {

```

```
GpibError("ibclr Error");
}
}

void Send(char *buf)
{
ibwrt(Device, buf, (long)strlen(buf));
/* Send the buffer command */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```
/******  
* Note — you should set bit 4 of Status Register Enable to 1.  
* Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));"  
*****/  
  
ibwait(Device, (/*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */  
if (ibsta & (/*TIMO | */ERR)) /* Check for GPIB Error */  
{  
GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
/* Serial Polling */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibrsp Error");  
}  
  
ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */  
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibwrt Error");  
}  
}  
  
void main(void)  
{  
/******  
* Initialization — done only once at the beginning of your application.  
*****/  
  
GpibSetup();
```

EXAMPLE CODES

```
/******  
* Main Application Body — write the majority of your GPIB code here.  
*****/  
  
printf("\n");  
printf("<<<SECTION 6>>>\n");  
printf("<<<EXAMPLE CODES>>>\n");  
printf("<<<Read Identification>>>\n");  
printf("<<<Start>>>\n");  
  
Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000  */  
                               /* Enables use of MAV bit of Status Register    */  
                               /* When RQS received, can read message from SA    */  
  
Send("*IDN?");                /* Identification ?    */  
Receive();                    /* Read Buffer          */  
printf("%s\n",Buffer);  
  
Send("*CLS");                  /* Clear Register      */  
printf("<<<End>>>\n");  
  
/******  
* Uninitialization — done only once at the end of your application.  
*****/  
  
ibonl(Device, 0);             /* Take the device offline  */  
if (ibsta & ERR)              /* Check for GPIB Error    */  
{  
    GpibError("ibonl Error");  
}  
  
getchar();                    /* Pause program          */  
  
}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Reading frequency and level of peak trace

Sets the normal marker on the peak point at the current waveform and measures the frequency and level of the normal marker.

1. Set
 - a. Span : 20 MHz
 - b. Center Frequency : 1 GHz
 - c. Reference Level : 0 dBm
 - d. VBW, RBW, Input Attenuator : Auto
 - e. Log 10 dB scale, Unit : dBm
 - f. Sweep Time : 5 s
2. Measuring
 - a. Peak Search
 - b. Read the frequency and the amplitude at the peak point

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;           /* Device unit descriptor */
int BoardIndex = 0;      /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7;  /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];        /* Read buffer */
char SerialPollResponse = 0; /* Read buffer

void GpibSetup(void);    /* GpibSetup function declaration */
void Send(char *buf);    /* Send function declaration */
void Receive(void);      /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration

void GpibSetup(void)
{
Device = ibdev(           /* Create a unit descriptor handle */
    BoardIndex,          /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
    PrimaryAddress,      /* Device primary address */
    SecondaryAddress,    /* Device secondary address */
    T10s,                /* Timeout setting (T10s = 10 seconds) */

```

EXAMPLE CODES

```
        1,                /* Assert EOI line at end of write */
        0);              /* EOS termination mode */
if (ibsta & ERR)        /* Check for GPIB Error */
{
    GpibError("ibdev Error");
}

ibclr(Device);         /* Clear the device */
if (ibsta & ERR)        /* Check for GPIB Error */
{
    GpibError("ibclr Error");
}
}

void Send(char *buf)
{
    ibwrt(Device, buf, (long)strlen(buf));
                                /* Send the buffer command */
if (ibsta & ERR)        /* Check for GPIB Error */
{
    GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```

/*****
* Note — you should set bit 4 of Status Register Enable to 1.
* Send command "*SRE 16" to be able to receive RQS.
* Tip — if you want to wait a long time, you should eliminate TIMO.
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".
*****/

ibwait(Device, (/*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */
if (ibsta & (/*TIMO | */ERR)) /* Check for GPIB Error */
{
GpibError("ibwait Error");
}

ibrsp(Device, &SerialPollResponse);

/* Serial Polling */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibrsp Error");
}

ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void main(void)
{
/*****
* Initialization — done only once at the beginning of your application.
*****/

GpibSetup();

```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Reading Frequency and Level of Peak Trace>>>\n");
printf("<<<SA is connected with Signal Generator (CW, 1GHz, -10dBm)>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                              /* Enables use of MAV bit of Status Register */
                              /* When RQS received, can read message from SA */
Send("CF 1GHZ;SP 20MHZ");     /* Center Freq : 1 GHz, Span : 20 MHz */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Center and Span is set)\n",Buffer);
Send("SD 10DB;AU DBM");       /* Scale Divide : 10 dB, Ampl Unit : dBm*/
Send("RL 0DBM;ATA ON");       /* Ref Lev : 0 dBm, Atten : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Scale Divide, Amplitude Unit, Reference Level, Attenuator is
set)\n",Buffer);
Send("RBA ON;VBA ON");        /* RBW : Auto, VBW : Auto */
Send("STA ON");                /* Sweep Time : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Read Buffer */
printf("Response=%s (RBW, VBW, Sweep Time is set)\n",Buffer);

Send("ST 5SEC;*OPC?");        /* Sweep Time : 5 s, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Sweep Time is set)\n",Buffer);
Send("MPK;*OPC?");            /* Peak Search, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Detect Peak Point)\n",Buffer);
Send("MF?");                  /* Marker Frequency ? */
Receive();                    /* Read Buffer */
printf("Marker Frequency = %s Hz\n",Buffer);
Send("MA?");                  /* Marker Amplitude ? */
Receive();                    /* Read Buffer */
printf("Marker Amplitude = %s dBm\n",Buffer);
```

EXAMPLE CODES

```
Send("*CLS");                /* Clear Register */
printf("<<<End>>>\n");

/*****
* Uninitialization — done only once at the end of your application.
*****/

ibonl(Device, 0);           /* Take the device offline */
if (ibsta & ERR)            /* Check for GPIB Error */
{
  GpibError("ibonl Error");
}

getchar();                 /* Pause program */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Reading Delta Marker amplitude

Measuring the difference value of the amplitude between the delta marker and the reference marker.

1. Set
 - a. Span : 20 MHz
 - b. Center Frequency : 1 GHz
 - c. Reference Level : 0 dBm
 - d. VBW, RBW, Input Attenuator : Auto
 - e. Log 10 dB scale, Unit : dBm
 - f. Sweep Time : 5 s
2. Measuring
 - a. Peak Search
 - b. Marker Frequency to Center Frequency
 - c. Marker Level to Reference Level
 - d. Reference Marker : Peak Point, Delta Marker : 5 MHz
 - e. Read the Delta Marker Amplitude

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;           /* Device unit descriptor */
int BoardIndex = 0;      /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];        /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);    /* GpibSetup function declaration */
void Send(char *buf);    /* Send function declaration */
void Receive(void);      /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{
    Device = ibdev(           /* Create a unit descriptor handle */
                  BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */

```

EXAMPLE CODES

```
        PrimaryAddress,      /* Device primary address */
        SecondaryAddress,    /* Device secondary address */
        T10s,                /* Timeout setting (T10s = 10 seconds) */
        1,                   /* Assert EOI line at end of write */
        0);                 /* EOS termination mode */
if (ibsta & ERR)           /* Check for GPIB Error */
{
    GpibError("ibdev Error");
}

ibclr(Device);            /* Clear the device */
if (ibsta & ERR)           /* Check for GPIB Error */
{
    GpibError("ibclr Error");
}
}

void Send(char *buf)
{
    ibwrt(Device, buf, (long)strlen(buf));
                                /* Send the buffer command */
if (ibsta & ERR)           /* Check for GPIB Error */
{
    GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```
/******  
* Note — you should set bit 4 of Status Register Enable to 1.  
*Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (/*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */  
if (ibsta & (/*TIMO | */ERR)) /* Check for GPIB Error */  
{  
GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
/* Serial Polling */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibrsp Error");  
}  
  
ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */  
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibwrt Error");  
}  
}  
  
void main(void)  
{  
/******  
* Initialization — done only once at the beginning of your application.  
*****/  
  
GpibSetup();
```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Reading Delta Marker Amplitude>>>\n");
printf("<<<SA is connected with Signal Generator (CW, 1GHz, -10dBm)>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                               /* Enables use of the MAV bit of Status Register */
                               /* When RQS received, can read message from SA */
Send("CF 1GHZ;SP 20MHZ");     /* Center Freq : 1 GHz, Span : 20 MHz */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Center and Span is set)\n",Buffer);
Send("SD 10DB;AU DBM");       /* Scale Divide : 10dB, Ampl Unit : dBm */
Send("RL 0DBM;ATA ON");       /* Ref Lev : 0dBm, Atten : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Scale Divide, Amplitude Unit, Reference Level, Attenuator is
set)\n",Buffer);
Send("RBA ON;VBA ON");        /* RBW : Auto, VBW : Auto */
Send("STA ON");               /* Sweep Time : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Read Buffer */
printf("Response=%s (RBW, VBW, Sweep Time is set)\n",Buffer);

Send("ST 5SEC;*OPC?");        /* Sweep Time : 5 sec, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Sweep Time is set)\n",Buffer);
Send("MPK;*OPC?");           /* Peak Search, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Detect Peak Point)\n",Buffer);
Send("MCF;*OPC?");           /* Marker->Center Freq, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Center Frequency is set)\n",Buffer);
Send("MRL;*OPC?");           /* Marker->Ref Lev, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Reference Level is set)\n",Buffer);
```

EXAMPLE CODES

```
Send("MF?");          /* Marker Frequency ?    */
Receive();            /* Read Buffer      */
printf("Normal Marker Frequency = %s Hz\n",Buffer);
Send("MA?");          /* Marker Amplitude ?    */
Receive();            /* Read Buffer      */
printf("Normal Marker Amplitude = %s dBm\n",Buffer);
Send("MM DELT");      /* Marker Mode : Delta    */
Send("MF 5MHZ");      /* Delta Mkr Freq : Ref Mkr Freq + 5MHz    */
Send("MA?");          /* Delta Marker Amplitude ?    */
Receive();            /* Read Buffer      */
printf("Delta Marker Amplitude = %s Hz\n",Buffer);

Send("*CLS");         /* Clear Register    */
printf("<<<End>>>\n");

/*****
* Uninitialization — done only once at the end of your application.
*****/

ibonl(Device, 0);     /* Take the device offline    */
if (ibsta & ERR)      /* Check for GPIB Error    */
{
GpibError("ibonl Error");
}

getchar();            /* Pause program    */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Measuring XdB Bandwidth

Searches the X dB point from the normal marker and measures X dB frequency bandwidth. (X is 6 dB on this example code.)

1. Set
 - a. Center Frequency : 1 GHz
 - b. Span : 20 MHz
 - c. Reference Level : 0 dBm
 - d. VBW : 100 KHz
 - e. RBW : 1 KHz
 - f. Sweep Time : 5 sec
 - g. Input Attenuator : Auto
 - h. Log 10 dB scale, Unit : dBm
2. Measuring
 - a. Peak Search
 - b. Marker Frequency to Center Frequency
 - c. Marker Level to Reference Level
 - d. Set 6 dB down point from the normal marker
 - e. Read 6 dB frequency bandwidth
 - f. Stop X dB down

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/
#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;          /* Device unit descriptor   */
int BoardIndex = 0;     /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];       /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);   /* GpibSetup function declaration */
void Send(char *buf);   /* Send function declaration */
void Receive(void);     /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)

```

EXAMPLE CODES

```
{
Device = ibdev(          /* Create a unit descriptor handle */
                BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
                PrimaryAddress, /* Device primary address */
                SecondaryAddress, /* Device secondary address */
                T10s, /* Timeout setting (T10s = 10 seconds) */
                1, /* Assert EOI line at end of write */
                0); /* EOS termination mode */
if (ibsta & ERR) /* Check for GPIB Error */
{
    GpibError("ibdev Error");
}

ibclr(Device); /* Clear the device */
if (ibsta & ERR) /* Check for GPIB Error */
{
    GpibError("ibclr Error");
}
}

void Send(char *buf)
{
    ibwrt(Device, buf, (long)strlen(buf));
    /* Send the buffer command */
if (ibsta & ERR) /* Check for GPIB Error */
{
    GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```
/******  
* Note — you should set bit 4 of Status Register Enable to 1.  
* Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */  
if (ibsta & (*TIMO | */ERR)) /* Check for GPIB Error */  
{  
GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
/* Serial Polling */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibrsp Error");  
}  
  
ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */  
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibwrt Error");  
}  
}  
  
void main(void)  
{  
/******  
* Initialization — done only once at the beginning of your application.  
*****/  
  
GpibSetup();
```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Measuring XdB Bandwidth>>>\n");
printf("<<<SA is connected with Signal Generator (CW, 1GHz, -10dBm)>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                               /* Enables use of the MAV bit of Status Register */
                               /* When RQS received, can read message from SA */
Send("CF 1GHZ;SP 20MHZ");     /* Center Freq : 1 GHz, Span : 20 MHz */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Center and Span is set)\n",Buffer);
Send("SD 10DB;AU DBM");       /* Scale Divide : 10dB, Ampl Unit : dBm */
Send("RL 0DBM;ATA ON");       /* Ref Lev : 0dBm, Atten : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Scale Divide, Amplitude Unit, Reference Level, Attenuator is
set)\n",Buffer);
Send("RBA ON;VBA ON");        /* RBW : Auto, VBW : Auto */
Send("STA ON");                /* Sweep Time : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Read Buffer */
printf("Response=%s (RBW, VBW, Sweep Time is set)\n",Buffer);

Send("ST 5SEC;*OPC?");        /* Sweep Time : 5 sec, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Sweep Time is set)\n",Buffer);
Send("RBA OFF;VBA OFF");      /* RBW : Manual, VBW : Manual */
Send("RB 100kHz;VB 1KHZ");    /* RBW : 100KHz, VBW : 1KHz */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Read Buffer */
printf("Response=%s (RBW, VBW is set)\n",Buffer);
Send("MPK;*OPC?");            /* Peak Search, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Detect Peak Point)\n",Buffer);
Send("MCF;*OPC?");            /* Marker->Center Freq, Operation Complete ?*/
```

EXAMPLE CODES

```
Receive();                /* Waiting for commands to complete */
printf("Response=%s (Center Frequency is set)\n",Buffer);
Send("MRL;*OPC?");       /* Marker->Ref Lev, Operation Complete ? */
Receive();                /* Waiting for commands to complete */
printf("Response=%s (Reference Level is set)\n",Buffer);
Send("XDBP 6DB");        /* 6dB down from Normal Marker */
Send("MEA XDB");         /* Start XdB Measurement */
Send("*OPC?");           /* Operation Complete ? */
Receive();                /* Read Buffer */
printf("Response=%s (XdB Measurement is set)\n",Buffer);
Send("XDBRL?");          /* XdB Bandwidth ? */
Receive();                /* Read Buffer */
printf("Frequency Bandwidth = %s Hz\n",Buffer);
Send("MEAO");            /* Stop XdB Measurement */

Send("*CLS");            /* Clear Register */
printf("<<<End>>>\n");

/*****
* Uninitialization — done only once at the end of your application.
*****/

ibonl(Device, 0);        /* Take the device offline */
if (ibsta & ERR)         /* Check for GPIB Error */
{
GpibError("ibonl Error");
}

getchar();              /* Pause program */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Measuring Occupied Bandwidth

Searches selected percentage power of channel power and then calculates OBW (Occupied Power Bandwidth).

1. Set
 - a. Center Frequency : 1 GHz
 - b. Span : 20 MHz
 - c. Input Attenuator : Auto
 - d. Unit : dBm, Log 10 dB scale
 - e. Reference Level : 0 dBm
 - f. RBW : 100 KHz
 - g. VBW : 1 KHz
 - h. Sweep Time : 5 sec
2. Measuring
 - a. Peak Search
 - b. Marker Frequency to Center Frequency
 - c. Set OBW 50%
 - d. Waiting for OBW calculation completed
 - e. Read OBW

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;           /* Device unit descriptor   */
int BoardIndex = 0;      /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];        /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);    /* GpibSetup function declaration */
void Send(char *buf);    /* Send function declaration */
void Receive(void);      /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{

```

EXAMPLE CODES

```
Device = ibdev(          /* Create a unit descriptor handle */
    BoardIndex,         /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
    PrimaryAddress,     /* Device primary address */
    SecondaryAddress,   /* Device secondary address */
    T10s,               /* Timeout setting (T10s = 10 seconds) */
    1,                  /* Assert EOI line at end of write */
    0);                 /* EOS termination mode */
if (ibsta & ERR)        /* Check for GPIB Error */
{
    GpibError("ibdev Error");
}

ibclr(Device);          /* Clear the device */
if (ibsta & ERR)        /* Check for GPIB Error */
{
    GpibError("ibclr Error");
}

void Send(char *buf)
{
    ibwrt(Device, buf, (long)strlen(buf));
    /* Send the buffer command */
    if (ibsta & ERR)    /* Check for GPIB Error */
    {
        GpibError("ibwrt Error");
    }
}

void Receive(void)
{
```

EXAMPLE CODES

```
/******  
* Note — you should set bit 4 of Status Register Enable to 1.  
* Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (/*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */  
if (ibsta & (/*TIMO | */ERR)) /* Check for GPIB Error */  
{  
GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
/* Serial Polling */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibrsp Error");  
}  
  
ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */  
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibwrt Error");  
}  
}  
  
void main(void)  
{  
/******  
* Initialization — done only once at the beginning of your application.  
*****/  
  
GpibSetup();
```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Measuring Occupied Bandwidth>>>\n");
printf("<<<SA is connected with Signal Generator (CW, 1GHz, -10dBm)>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                              /* This enables use of the MAV bit of Status Register */
                              /* When RQS received, can read message from SA */
Send("CF 1GHZ;SP 20MHZ");     /* Center Freq : 1 GHz, Span : 20 MHz */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Center and Span is set)\n",Buffer);
Send("SD 10DB;AU DBM");       /* Scale Divide : 10dB, Ampl Unit : dBm */
Send("RL 0DBM;ATA ON");       /* Ref Lev : 0dBm, Atten : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Scale Divide, Amplitude Unit, Reference Level, Attenuator is
set)\n",Buffer);
Send("RBA ON;VBA ON");        /* RBW : Auto, VBW : Auto */
Send("STA ON");               /* Sweep Time : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Read Buffer */
printf("Response=%s (RBW, VBW, Sweep Time is set)\n",Buffer);

Send("ST 5SEC;*OPC?");        /* Sweep Time : 5 sec, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Sweep Time is set)\n",Buffer);
Send("RBA OFF;VBA OFF");      /* RBW : Manual, VBW : Manual */
Send("RB 100kHz;VB 1KHZ");    /* RBW : 100KHz, VBW : 1KHz */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Read Buffer */
printf("Response=%s (RBW, VBW is set)\n",Buffer);
Send("MPK;*OPC?");            /* Peak Search, Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Detect Peak Point)\n",Buffer);
Send("MCF;*OPC?");            /* Marker->Center Freq, Operation Complete ?*/
```

EXAMPLE CODES

```
Receive();                /* Waiting for commands to complete */
printf("Response=%s (Center Frequency is set)\n",Buffer);
Send("OBWP 50");         /* Occ BW % Pwr : 50% */
Send("MEA OBW");        /* Start OBW Measurement */
Send("*OPC?");          /* Operation Complete ? */
Receive();               /* Read Buffer */
printf("Response=%s (OBW Measurement is set)\n",Buffer);
Send("OBWOUT?");        /* Occupied BW ? */
Receive();               /* Read Buffer */
printf("Occupied Bandwidth = %s Hz\n",Buffer);
Send("MEAO");           /* Stop OBW Measurement */

Send("*CLS");           /* Clear Register */
printf("<<<End>>>\n");

/*****
* Uninitialization — done only once at the end of your application.
*****/

ibonl(Device, 0);       /* Take the device offline */
if (ibsta & ERR)        /* Check for GPIB Error */
{
  GpibError("ibonl Error");
}

getchar();              /* Pause program */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Measuring marker noise

Sets the reference marker on the signal, and the normal marker on the noise, then measures Marker Noise.

1. Set
 - a. Center Frequency : 1 GHz
 - b. Span : 20 MHz
 - c. Reference Level : 0 dBm
 - d. Input Attenuator : Auto
 - e. Log 10 dB scale, Unit : dBm
 - f. RBW : 100 KHz
 - g. VBW : 1 KHz
 - h. Sweep Time : Auto
2. Measuring
 - a. Reference Marker : Peak point, Delta Marker : 5 MHz
 - b. Read Marker Noise

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/
#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;          /* Device unit descriptor */
int BoardIndex = 0;     /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];       /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);   /* GpibSetup function declaration */
void Send(char *buf);   /* Send function declaration */
void Receive(void);     /* Receive function declaration */
bool Wait_RQS(void);   /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{
    Device = ibdev(          /* Create a unit descriptor handle */
                  BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */

```

EXAMPLE CODES

```
        PrimaryAddress,      /* Device primary address */
        SecondaryAddress,    /* Device secondary address */
        T10s,                /* Timeout setting (T10s = 10 seconds) */
        1,                   /* Assert EOI line at end of write */
        0);                  /* EOS termination mode */
if (ibsta & ERR)             /* Check for GPIB Error */
{
    GpibError("ibdev Error");
}

ibclr(Device);              /* Clear the device */
if (ibsta & ERR)             /* Check for GPIB Error */
{
    GpibError("ibclr Error");
}
}

void Send(char *buf)
{
    ibwrt(Device, buf, (long)strlen(buf));
                                     /* Send the buffer command */
if (ibsta & ERR)             /* Check for GPIB Error */
{
    GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```
/******  
* Note — you should set bit 4 of Status Register Enable to 1.  
* Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (/*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */  
if (ibsta & (/*TIMO | */ERR)) /* Check for GPIB Error */  
{  
GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
/* Serial Polling */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibrsp Error");  
}  
  
ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */  
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibwrt Error");  
}  
}  
  
bool Wait_RQS(void)  
{
```

EXAMPLE CODES

```
/*
*****
* Tip — if you want to wait a long time, you should eliminate TIMO.
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".
*****
*/

ibwait(Device, (TIMO | RQS));      /* Wait until TIMO or RQS is asserted */
if (ibsta & RQS)                  /* Detect RQS */
{
    ibrsp(Device, &SerialPollResponse);

                                /* Serial Polling */
    if (ibsta & ERR)              /* Check for GPIB Error */
    {
        GpibError("ibrsp Error");
    }
    return true;
}
if (ibsta & (TIMO | ERR))         /* Don't detect RQS */
{
    return false;
}
return false;
}

void main(void)
{
/*
*****
* Initialization — done only once at the beginning of your application.
*****
*/

    GpibSetup();
}
```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Measuring Marker Noise>>>\n");
printf("<<<SA is connected with Signal Generator (CW, 1GHz, -10dBm)>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 24");          /* Clear Register, Status Register Enable : 00011000 */
                              /* This enables use of the MAV bit of Status Register */
                              /* When RQS received, can read message from SA */
Send("ESE2 16");             /* End Event Status Register Enable : 00010000 */
                              /* Enables use of ESB2 bit of End Event Status Register*/
                              /* When RQS received, indicates end of measurement */
Send("CF 1GHZ;SP 20MHZ");    /* Center Freq : 1 GHz, Span : 20 MHz */
Send("*OPC?");               /* Operation Complete ? */
Receive();                   /* Waiting for commands to complete */
printf("Response=%s (Center and Span is set)\n",Buffer);
Send("SD 10DB;AU DBM");      /* Scale Divide : 10dB, Ampl Unit : dBm */
Send("RL 0DBM;ATA ON");      /* Ref Lev : 0dBm, Atten : Auto */
Send("*OPC?");               /* Operation Complete ? */
Receive();                   /* Waiting for commands to complete */
printf("Response=%s (Scale Divide, Amplitude Unit, Reference Level, Attenuator is
set)\n",Buffer);
Send("RBA ON;VBA ON");      /* RBW : Auto, VBW : Auto */
Send("STA ON");              /* Sweep Time : Auto */
Send("*OPC?");               /* Operation Complete ? */
Receive();                   /* Read Buffer */
printf("Response=%s (RBW, VBW, Sweep Time is set)\n",Buffer);

Send("ST 5SEC;*OPC?");      /* Sweep Time : 5 sec, Operation Complete ? */
Receive();                   /* Waiting for commands to complete */
Send("RBA OFF;VBA OFF");     /* RBW : Manual, VBW : Manual */
Send("RB 100kHz;VB 1KHZ");   /* RBW : 100KHz, VBW : 1KHz */
Send("*OPC?");               /* Operation Complete ? */
Receive();                   /* Read Buffer */
printf("Response=%s (RBW, VBW is set)\n",Buffer);
Send("MPK;*OPC?");          /* Peak Search, Operation Complete ? */
Receive();                   /* Waiting for commands to complete */
```

EXAMPLE CODES

```
printf("Response=%s (Detect Peak Point)\n",Buffer);
Send("MCF;*OPC?");          /* Marker->Center Freq, Operation Complete ?*/
Receive();                  /* Waiting for commands to complete */
printf("Response=%s (Center Frequency is set)\n",Buffer);
Send("MM DELT");           /* Delta Marker Function */
Send("MF 5MHz");           /* Delta Marker Frequency : 5 MHz */
Send("MFN NOIS");         /* Start Marker Noise Measurement */
if (Wait_RQS())           /* Success to receive RQS */
{
printf("Detect RQS\n");
}
else                       /* Fail to receive RQS */
{
printf("No Response\n");
}
Send("MFNY?");             /* Occupied BW ? */
Receive();                /* Read Buffer */
printf("Marker Noise = %s dBm/Hz\n",Buffer);
Send("MEAO");             /* Stop OBW Measurement */

Send("*CLS");             /* Clear Register */
printf("<<<End>>>\n");

/*****
* Uninitialization — done only once at the end of your application.
*****/

ibonl(Device, 0);         /* Take the device offline */
if (ibsta & ERR)         /* Check for GPIB Error */
{
GpibError("ibonl Error");
}

getchar();               /* Pause program */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Saving Status File

Saves the current system status to current disk

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;          /* Device unit descriptor */
int BoardIndex = 0;     /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];       /* Read buffer */
char SerialPollResponse = 0; /* Read buffer

void GpibSetup(void);   /* GpibSetup function declaration */
void Send(char *buf);   /* Send function declaration */
void Receive(void);     /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration

void GpibSetup(void)
{
    Device = ibdev(
        BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress, /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s, /* Timeout setting (T10s = 10 seconds) */
        1, /* Assert EOI line at end of write */
        0); /* EOS termination mode
    if (ibsta & ERR) /* Check for GPIB Error
    {
        GpibError("ibdev Error");
    }

    ibclr(Device); /* Clear the device
    if (ibsta & ERR) /* Check for GPIB Error
    {

```

```
GpibError("ibclr Error");
}
}

void Send(char *buf)
{
ibwrt(Device, buf, (long)strlen(buf));
/* Send the buffer command */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```

/*****
* Note — you should set bit 4 of Status Register Enable to 1.
* Send command "*SRE 16" to be able to receive RQS.
* Tip — if you want to wait a long time, you should eliminate TIMO.
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".
*****/

ibwait(Device, (*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */
if (ibsta & (*TIMO | */ERR)) /* Check for GPIB Error */
{
GpibError("ibwait Error");
}

ibrsp(Device, &SerialPollResponse);

/* Serial Polling */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibrsp Error");
}

ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void main(void)
{
/*****
* Initialization — done only once at the beginning of your application.
*****/

GpibSetup();

```

EXAMPLE CODES

```
/******  
* Main Application Body — write the majority of your GPIB code here.  
*****/  
  
printf("\n");  
printf("<<<SECTION 6>>>\n");  
printf("<<<EXAMPLE CODES>>>\n");  
printf("<<<Saving Status File>>>\n");  
printf("<<<Start>>>\n");  
  
Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */  
                               /* This enables use of the MAV bit of Status Register */  
                               /* When RQS received, can read message from SA */  
  
Send("FSAVE 'C:\\HSA_Save\\SaveStatus.sts");  
                               /* Save Status File */  
  
printf("Saved Current Status\n");  
  
Send("*CLS");                  /* Clear Register */  
printf("<<<End>>>\n");  
  
/******  
* Uninitialization — done only once at the end of your application.  
*****/  
  
ibonl(Device, 0);             /* Take the device offline */  
if (ibsta & ERR)              /* Check for GPIB Error */  
{  
    GpibError("ibonl Error");  
}  
  
getchar();                    /* Pause program */  
  
}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Loading Status File

Recalls the system status from current disk.

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;          /* Device unit descriptor */
int BoardIndex = 0;     /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];       /* Read buffer */
char SerialPollResponse = 0; /* Read buffer

void GpibSetup(void);   /* GpibSetup function declaration */
void Send(char *buf);   /* Send function declaration */
void Receive(void);     /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration

void GpibSetup(void)
{
    Device = ibdev(
        BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress, /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s, /* Timeout setting (T10s = 10 seconds) */
        1, /* Assert EOI line at end of write */
        0); /* EOS termination mode
    if (ibsta & ERR) /* Check for GPIB Error
    {
        GpibError("ibdev Error");
    }

    ibclr(Device); /* Clear the device
    if (ibsta & ERR) /* Check for GPIB Error
    {

```

```
GpibError("ibclr Error");
}
}

void Send(char *buf)
{
ibwrt(Device, buf, (long)strlen(buf));
/* Send the buffer command */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```
/******  
* Note — You should set bit 4 of Status Register Enable to 1.  
* Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */  
if (ibsta & (*TIMO | */ERR)) /* Check for GPIB Error */  
{  
GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
/* Serial Polling */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibrsp Error");  
}  
  
ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */  
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibwrt Error");  
}  
}  
  
void main(void)  
{  
/******  
* Initialization — done only once at the beginning of your application.  
*****/  
  
GpibSetup();
```

EXAMPLE CODES

```
/******  
* Main Application Body — write the majority of your GPIB code here.  
*****/  
  
printf("\n");  
printf("<<<SECTION 6>>>\n");  
printf("<<<EXAMPLE CODES>>>\n");  
printf("<<<Loading Status File>>>\n");  
printf("<<<Start>>>\n");  
  
Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */  
                               /* Enables use of MAV bit of Status Register */  
                               /* When received RQS, can read message from SA */  
  
Send("FLOAD 'C:\\HSA_Save\\SaveStatus.sts");  
                               /* Load Status File */  
  
printf("Loaded Status File\n");  
  
Send("*CLS");                  /* Clear Register */  
printf("<<<End>>>\n");  
  
/******  
* Uninitialization — done only once at the end of your application.  
*****/  
  
ibonl(Device, 0);             /* Take the device offline */  
if (ibsta & ERR)              /* Check for GPIB Error */  
{  
    GpibError("ibonl Error");  
}  
  
getchar();                    /* Pause program */  
  
}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Reading Real64 data

Get real64 data from the instrument.

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;          /* Device unit descriptor */
int BoardIndex = 0;     /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];       /* Read buffer */
double dBuffer;        /* Read buffer */
double* pdBuffer=&dBuffer; /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void); /* GpibSetup function declaration */
void Send(char *buf); /* Send function declaration */
void Receive(void); /* Receive function declaration */
void Receive_double(void); /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{
    Device = ibdev( /* Create a unit descriptor handle */
        BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress, /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s, /* Timeout setting (T10s = 10 seconds) */
        1, /* Assert EOI line at end of write */
        0); /* EOS termination mode */
    if (ibsta & ERR) /* Check for GPIB Error */
    {
        GpibError("ibdev Error");
    }
}

```

EXAMPLE CODES

```
ibclr(Device);                /* Clear the device */
if (ibsta & ERR)              /* Check for GPIB Error */
{
    GpibError("ibclr Error");
}
}

void Send(char *buf)
{
    ibwrt(Device, buf, (long)strlen(buf));
                                /* Send the buffer command */
    if (ibsta & ERR)          /* Check for GPIB Error */
    {
        GpibError("ibwrt Error");
    }
}

void Receive(void)
{
```

EXAMPLE CODES

```
/******  
* Note — you should set bit 4 of Status Register Enable to 1.  
* Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (/*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */  
if (ibsta & (/*TIMO | */ERR)) /* Check for GPIB Error */  
{  
GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
/* Serial Polling */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibrsp Error");  
}  
  
ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */  
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibwrt Error");  
}  
}  
  
void Receive_double(void)  
{
```

EXAMPLE CODES

```
/******  
* Note — you should set bit 4 of Status Register Enable to 1.  
* Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (TIMO | RQS));      /* Wait until TIMO or RQS is asserted  */  
if (ibsta & (TIMO | ERR))        /* Check for GPIB Error    */  
{  
    GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
                                /* Serial Polling      */  
if (ibsta & ERR)                /* Check for GPIB Error    */  
{  
    GpibError("ibrsp Error");  
}  
  
ibrd (Device, pBuffer, sizeof(double));  
                                /* Read double value from the device  */  
if (ibsta & ERR)                /* Check for GPIB Error    */  
{  
    GpibError("ibwrt Error");  
}  
}  
  
void main(void)  
{  
    /******  
    * Initialization — done only once at the beginning of your application.  
    *****/  
  
    GpibSetup();
```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Reading Real64 Data>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                              /* Enables use of MAV bit of Status Register */
                              /* When RQS received, can read message from SA */

Send("TDF REAL64");          /* Set data type to Real64 */
Send("CF?");                 /* Center Frequency ? */
Receive_double();           /* Read Center Frequency */
printf("Center Frequency : %.0f Hz\n",dBuffer);
Send("TDF ASC");            /* Set data type to ASCII */

Send("*CLS");                /* Clear Register */
printf("<<<End>>>\n");

/*
*****
* Uninitialization — done only once at the end of your application.
*****
*/

ibonl(Device, 0);           /* Take the device offline */
if (ibsta & ERR)            /* Check for GPIB Error */
{
  GpibError("ibonl Error");
}

getchar();                 /* Pause program */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Reading Real64 trace data

Get real64 trace data from the instrument.

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;          /* Device unit descriptor */
int BoardIndex = 0;     /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];       /* Read buffer */
double dBuffer[8192];  /* Read buffer */
double* pdBuffer=&dBuffer[0]; /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);   /* GpibSetup function declaration */
void Send(char *buf);   /* Send function declaration */
void Receive(void);     /* Receive function declaration */
void Receive_double(int cnt); /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{
    Device = ibdev(
        BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress, /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s, /* Timeout setting (T10s = 10 seconds) */
        1, /* Assert EOI line at end of write */
        0); /* EOS termination mode */
    if (ibsta & ERR) /* Check for GPIB Error */
    {
        GpibError("ibdev Error");
    }
}

```

EXAMPLE CODES

```
ibclr(Device);                /* Clear the device */
if (ibsta & ERR)              /* Check for GPIB Error */
{
    GpibError("ibclr Error");
}
}

void Send(char *buf)
{
    ibwrt(Device, buf, (long)strlen(buf));
                                /* Send the buffer command */
    if (ibsta & ERR)          /* Check for GPIB Error */
    {
        GpibError("ibwrt Error");
    }
}

void Receive(void)
{
```

EXAMPLE CODES

```
/******  
* Note — you should set bit 4 of Status Register Enable to 1.  
* Send command "*SRE 16" to be able to receive RQS.  
* Tip — if you want to wait a long time, you should eliminate TIMO.  
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".  
*****/  
  
ibwait(Device, (*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */  
if (ibsta & (*TIMO | */ERR)) /* Check for GPIB Error */  
{  
GpibError("ibwait Error");  
}  
  
ibrsp(Device, &SerialPollResponse);  
/* Serial Polling */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibrsp Error");  
}  
  
ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */  
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */  
if (ibsta & ERR) /* Check for GPIB Error */  
{  
GpibError("ibwrt Error");  
}  
}  
  
void Receive_double(int cnt)  
{
```

EXAMPLE CODES

```
/*
* Note — you should set bit 4 of Status Register Enable to 1.
* Send command "*SRE 16" to be able to receive RQS.
* Tip — if you want to wait a long time, you should eliminate TIMO.
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));"
*/
*/

ibwait(Device, (TIMO | RQS));      /* Wait until TIMO or RQS is asserted */
if (ibsta & (TIMO | ERR))         /* Check for GPIB Error */
{
    GpibError("ibwait Error");
}

ibrsp(Device, &SerialPollResponse);
                                /* Serial Polling */
if (ibsta & ERR)                 /* Check for GPIB Error */
{
    GpibError("ibrsp Error");
}

ibrd (Device, pdBuffer, cnt*sizeof(double));
                                /* Read double value from the device */
if (ibsta & ERR)                 /* Check for GPIB Error */
{
    GpibError("ibwrt Error");
}
}

void main(void)
{
    int count, i;

/*
* Initialization — done only once at the beginning of your application.
*/
*/

    GpibSetup();
```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Reading Real64 Trace Data>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                              /* Enables use of the MAV bit of Status Register */
                              /* When RQS received, can read message from SA */

Send("TDF REAL64");          /* Set data type to Real64 */
Send("PO?"); /* Points ? */
Receive_double(1);          /* Read Points */
count=(int)dBuffer[0];
Send("TRD? TRACE1");        /* Trace Data ? */
Receive_double(count);      /* Read Trace Data */
for (i=0;i<count;i++)
{
printf("%d : %.2fdBm\n",i+1,dBuffer[i]);
}
Send("TDF ASC");            /* Set data type to ASCII */

Send("*CLS");               /* Clear Register */
printf("<<<End>>>\n");

/*
*****
* Uninitialization — done only once at the end of your application.
*****
*/

ibonl(Device, 0);           /* Take the device offline */
if (ibsta & ERR)            /* Check for GPIB Error */
{
GpibError("ibonl Error");
}

getchar();                  /* Pause program */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Checking Limit Line

Check PASS or FAIL by comparing the current waveform with the upper limit line or the lower limit line.

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;          /* Device unit descriptor */
int BoardIndex = 0;     /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];       /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);   /* GpibSetup function declaration */
void Send(char *buf);   /* Send function declaration */
void Receive(void);     /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{
    Device = ibdev(
        BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress, /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s, /* Timeout setting (T10s = 10 seconds) */
        1, /* Assert EOI line at end of write */
        0); /* EOS termination mode */
    if (ibsta & ERR) /* Check for GPIB Error */
    {
        GpibError("ibdev Error");
    }

    ibclr(Device); /* Clear the device */
    if (ibsta & ERR) /* Check for GPIB Error */

```

```
{
GpibError("ibclr Error");
}
}

void Send(char *buf)
{
ibwrt(Device, buf, (long)strlen(buf));
/* Send the buffer command */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void Receive(void)
{
```

EXAMPLE CODES

```

/*****
* Note — you should set bit 4 of Status Register Enable to 1.
* Send command "*SRE 16" to be able to receive RQS.
* Tip — if you want to wait a long time, you should eliminate TIMO.
* Instead of "ibwait(Device, (TIMO | RQS));", use "ibwait(Device, (RQS));".
*****/

ibwait(Device, (/*TIMO | */RQS)); /* Wait until TIMO or RQS is asserted */
if (ibsta & (/*TIMO | */ERR)) /* Check for GPIB Error */
{
GpibError("ibwait Error");
}

ibrsp(Device, &SerialPollResponse);

/* Serial Polling */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibrsp Error");
}

ibrd(Device, Buffer, 100); /* Read up to 100 bytes from the device */
Buffer[ibcntl-1] = '\0'; /* Null terminate the ASCII string */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void main(void)
{
/*****
* Initialization — done only once at the beginning of your application.
*****/

GpibSetup();

```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Checking Limit Line>>>\n");
printf("<<<SA is connected with Signal Generator (CW, 1GHz, -10dBm)>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                              /* This enables use of the MAV bit of Status Register */
                              /* When RQS received, can read message from

instrument*/
Send("CF 1GHZ;SP 20MHZ");     /* Center Freq : 1 GHz, Span : 20 MHz */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Center and Span is set)\n",Buffer);
Send("SD 10DB;AU DBM");       /* Scale Divide : 10dB, Ampl Unit : dBm */
Send("RL -5DBM;ATA ON");     /* Ref Lev : -5dBm, Atten : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Waiting for commands to complete */
printf("Response=%s (Scale Divide, Amplitude Unit, Reference Level, Attenuator is
set)\n",Buffer);
Send("RBA ON;VBA ON");       /* RBW : Auto, VBW : Auto */
Send("STA ON");               /* Sweep Time : Auto */
Send("*OPC?");                /* Operation Complete ? */
Receive();                    /* Read Buffer */
printf("Response=%s (RBW, VBW, Sweep Time is set)\n",Buffer);
```

EXAMPLE CODES

```
/*
* The Limit mask data is saved in the current disk.
* Recall the limit mask data from the current disk.
* When loading is complete, the configuration is replaced by the saved data.
*/

Send("FLOAD 'C:\\HSA_Save\\SaveLimit.lim");
/* Load the limit mask data */

Send("LLCS ON"); /* Start Checking Upper Limit */
Send("LLCS2 ON"); /* Start Checking Lower Limit */
printf("Running for 10 sec\n",Buffer);
Sleep(10000); /* Waiting 10 sec */
Send("LLFC?"); /* Upper Limit Line Fail Count ? */
Receive(); /* Read Upper Limit Line Fail Count */
printf("Fail Count of Upper Limit Line = %s\n",Buffer);
Send("LLFC2?"); /* Lower Limit Line Fail Count ? */
Receive(); /* Read Lower Limit Line Fail Count */
printf("Fail Count of Lower Limit Line = %s\n",Buffer);
Send("LLCS OFF"); /* Stop Checking Upper Limit */
Send("LLCS2 OFF"); /* Stop Checking Lower Limit */
Send("LLAO"); /* Clear Limit Line */

Send("*CLS"); /* Clear Register */
printf("<<<End>>>\n");

/*
* Uninitialization — done only once at the end of your application.
*/

ibonl(Device, 0); /* Take the device offline */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibonl Error");
}

getchar(); /* Pause program */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Send File

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

int Device = 0;           /* Device unit descriptor */
int BoardIndex = 0;      /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7;  /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];         /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);    /* GpibSetup function declaration */
void Send(char *buf);    /* Send function declaration */
void Receive(void);      /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{
    Device = ibdev(
        BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress, /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s, /* Timeout setting (T10s = 10 seconds) */
        1, /* Assert EOI line at end of write */
        0); /* EOS termination mode */
    if (ibsta & ERR) /* Check for GPIB Error */
    {
        GpibError("ibdev Error");
    }

    ibclr(Device); /* Clear the device */
    if (ibsta & ERR) /* Check for GPIB Error */
    {
        GpibError("ibclr Error");
    }
}

```

```
}

void main(void)
{
FILE *pF;
fpos_t posStart, posEnd;
long lFileSize,lReadSize,lWriteSize;
char *cReadBuffer,*cWriteBuffer;
char cReadSize[20],cMessage[200];
int nReadSize;

/*****
* Initialization — done only once at the beginning of your application.
*****/

GpibSetup();
```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Send File>>>\n");
printf("<<<PC has the File named '\SendedSample.jpg' in C Drive.>>>\n");
printf("<<<Start>>>\n");

/* Open File */
pF=fopen("C:\\SendedSample.jpg","rb");
if (!pF) return;

fseek(pF,0,SEEK_END); /* Get File Size */
fgetpos(pF,&posEnd);
fseek(pF,0,SEEK_SET);
fgetpos(pF,&posStart);
lFileSize=(long)(posEnd-posStart);

/* Read File */
cReadBuffer=new char[lFileSize];
lReadSize=(long)fread(cReadBuffer,sizeof(char),lFileSize,pF);
fclose(pF); /* Close File */

/* Make Command */
sprintf(cReadSize,"%d",lReadSize);
nReadSize=(int)strlen(cReadSize);
sprintf(cMessage,"MMEM:DATA 'C:\\ReceivedSample.jpg',#%d%d",nReadSize,lReadSize);
lWriteSize=(long)strlen(cMessage);
cWriteBuffer=new char[lWriteSize+lReadSize];
memcpy(cWriteBuffer,cMessage,lWriteSize);
memcpy(cWriteBuffer+lWriteSize,cReadBuffer,lReadSize);

/* Send Command */
ibwrt(Device,cWriteBuffer,lWriteSize+lReadSize);
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}
```

```
delete cWriteBuffer;
delete cReadBuffer;          /* Clear Buffer    */
printf("Sended File\n");

printf("<<<End>>>\n");

/*****
* Uninitialization — done only once at the end of your application.
*****/

ibonl(Device, 0);           /* Take the device offline */
if (ibsta & ERR)           /* Check for GPIB Error    */
{
  GpibError("ibonl Error");
}

getchar();                 /* Pause program          */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Receive File

```

/*****
* Refer to the language interface documentation for details on
* which header and .obj files to include in your project.
*****/

#include <windows.h>
#include "ni488.h"
#include <stdio.h>

#define MAX_FILE_SIZE 30000

int Device = 0;          /* Device unit descriptor */
int BoardIndex = 0;     /* Interface Index (GPIB0=0,GPIB1=1,etc.) */
int PrimaryAddress = 7; /* Primary address of the device */
int SecondaryAddress = 0; /* Secondary address of the device */
char Buffer[101];       /* Read buffer */
char SerialPollResponse = 0; /* Read buffer */

void GpibSetup(void);   /* GpibSetup function declaration */
void Send(char *buf);   /* Send function declaration */
void Receive(void);     /* Receive function declaration */
void GpibError(char *msg); /* Error function declaration */

void GpibSetup(void)
{
    Device = ibdev(
        BoardIndex, /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
        PrimaryAddress, /* Device primary address */
        SecondaryAddress, /* Device secondary address */
        T10s, /* Timeout setting (T10s = 10 seconds) */
        1, /* Assert EOI line at end of write */
        0); /* EOS termination mode */
    if (ibsta & ERR) /* Check for GPIB Error */
    {
        GpibError("ibdev Error");
    }

    ibclr(Device); /* Clear the device */
    if (ibsta & ERR) /* Check for GPIB Error */
    {

```

```

GpibError("ibclr Error");
}
}

void Send(char *buf)
{
ibwrt(Device, buf, (long)strlen(buf));
/* Send the buffer command */
if (ibsta & ERR) /* Check for GPIB Error */
{
GpibError("ibwrt Error");
}
}

void main(void)
{
FILE *pF;
char cReadBuffer[MAX_FILE_SIZE];
long lFileSize;

/*****
* Initialization — done only once at the beginning of your application.
*****/

GpibSetup();

```

EXAMPLE CODES

```
/*
*****
* Main Application Body — write the majority of your GPIB code here.
*****
*/

printf("\n");
printf("<<<SECTION 6>>>\n");
printf("<<<EXAMPLE CODES>>>\n");
printf("<<<Receive File>>>\n");
printf("<<<SA has the File named '\SendedSample.jpg' in C Drive.>>>\n");
printf("<<<Start>>>\n");

Send("*CLS;*SRE 16");          /* Clear Register, Status Register Enable : 00010000 */
                              /* Enables use of MAV bit of Status Register */
                              /* When RQS received, can read message from SA */

                              /* Send Command */
Send("MMEM:DATA? \C:\SendedSample.jpg");

ibwait(Device, (TIMO | RQS)); /* Receive Data */
if (ibsta & ERR)              /* Check for GPIB Error */
{
    GpibError("ibwait Error");
}
ibrsp(Device, &SerialPollResponse);
if (ibsta & ERR)              /* Check for GPIB Error */
{
    GpibError("ibrsp Error");
}
ibrd (Device, cReadBuffer, (long)MAX_FILE_SIZE);
if (ibsta & ERR)              /* Check for GPIB Error */
{
    GpibError("ibrd Error");
}

lFileSize=ibcntl;             /* Get Length of Received Data */

                              /* Open File */
pF=fopen("C:\\ReceivedSample.jpg","wb");
if (!pF) return;

                              /* Write File */
size_t WriteSize=fwrite(cReadBuffer,sizeof(char),lFileSize,pF);
```

EXAMPLE CODES

```
fclose(pF);                /* Close File      */

printf("Received File\n");

Send("*CLS");              /* Clear Register */
printf("<<<End>>>\n");

/*****
* Uninitialization — done only once at the end of your application.
*****/

ibonl(Device, 0);          /* Take the device offline */
if (ibsta & ERR)           /* Check for GPIB Error    */
{
  GpibError("ibonl Error");
}

getchar();                 /* Pause program          */

}
```

```

/*****
* Function GPIBERROR
* This function notifies you that a NI-488 function failed by
* printing an error message. The status variable IBSTA is also
* printed in hexadecimal along with the mnemonic meaning of the bit
* position. The status variable IBERR is printed in decimal
* along with the mnemonic meaning of the decimal value. The status
* variable IBCNTL is printed in decimal.
*
* The NI-488 function IBONL is called to disable the hardware and
* software.
*
* The EXIT function terminates this program.
*****/

```

```

void GpibError(char *msg)
{
printf ("%s\n", msg);

printf ("ibsta = &H%x <", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" >\n");

printf ("iberr = %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");

```

```
if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
if (iberr == EABO) printf (" EABO <Operation aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO <File system error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

printf ("ibcntl = %ld\n", ibcntl);
printf ("\n");
getchar();

/* Call ibonl to take the device and interface offline      */
ibonl (Device,0);

exit(1);
}
```

Appendix A

REMOTE CONTROL

< Catalog Order (Spectrum Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Amplitude	Reference Level	RL	:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel	<amplitude> ?
Amplitude	Attenuation	AT	[:SENSe]:POWer[:RF]:ATTenuation	<amplitude> ?
Amplitude	Attenuation Auto	ATA	[:SENSe]:POWer[:RF]:ATTenuation:AUTO	OFF ON 0 1 ?
Amplitude	Scale/Divide	SD	:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision	<amplitude> ?
Amplitude	Scale Type	STY	:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing	LINear LOGarithmic ?
Amplitude	Amplitude Units	AU	:UNIT:POWer	DBM DBMV DBMA V W A DBUV DBUA ?
Amplitude	Ref Level Offset	RLO	:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet	<amplitude> ?
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[:RF]:GAIN[:STATe]	OFF ON 0 1 ?
Amplitude	Apply Corrections	COAS	[:SENSe]:CORRection:CSET:ALL[:STATe]	OFF ON 0 1 ?
Amplitude	Correction State	COS1 2 3 4	[:SENSe]:CORRection:CSET1 2 3 4[:STATe]	OFF ON 0 1 ?
Amplitude	Delete All Corrections	COAD	[:SENSe]:CORRection:CSET:ALL:DELete	none
Auxiliary	AM Demodulation State	AMS		OFF ON 0 1 ?
Auxiliary	FM Demodulation State	FMS		OFF ON 0 1 ?
Auxiliary	Audio State	AUDIOS		OFF ON 0 1 ?
Auxiliary	Audio Level	AUDIOLEV		<integer> ?
Average	Average	AVG	[:SENSe]:AVERAge[:STATe]	OFF ON 0 1 ?
Average	Average Count	AVGC	[:SENSe]:AVERAge:COUNT	<integer> ?
Average	Average Clear	AVGCL	[:SENSe]:AVERAge:CLEar	none
Average	Avg/VBW Type	AVGT	[:SENSe]:AVERAge:TYPE	RMS LOG SCALar ?
Average	Avg/VBW Type Auto	AVGTA	[:SENSe]:AVERAge:TYPE:AUTO	OFF ON 0 1 ?
Bandwidth	Auto Bandwidth	AB	[:SENSe]:BANDwidth BWIDth:AUTO	none
Bandwidth	Resolution Bandwidth	RB	[:SENSe]:BANDwidth BWIDth[:RESolution]	<frequency> ?
Bandwidth	Resolution Bandwidth Auto	RBA	[:SENSe]:BANDwidth BWIDth[:RESolution]:AUTO	OFF ON 0 1 ?
Bandwidth	Video Bandwidth	VB	[:SENSe]:BANDwidth BWIDth:VIDeo	<frequency> ?
Bandwidth	Video Bandwidth Auto	VBA	[:SENSe]:BANDwidth BWIDth:VIDeo:AUTO	OFF ON 0 1 ?
Bandwidth	VBW/RBW	VBRB	[:SENSe]:BANDwidth BWIDth:VIDeo:RATio	<ratio> ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Bandwidth	Span/RBW	SPRB	[:SENSe]:FREQUency:SPAN:BANDwidth BWIDth:VIDeo:RATio	<ratio> ?
Bandwidth	Span/RBW Auto	SPRBA	[:SENSe]:FREQUency:SPAN:BANDwidth BWIDth:VIDeo:RATio:AUTO	OFF ON 0 1 ?
Calibration	Align All Now	CALALL	:CALibration[:ALL]	none ?
Calibration	Periodic Temperature Cal	PCAL	:CALibration:PTEMPerature	OFF ON 0 1 ?
Calibration	Pre-Filter Cal	PFCAL	:CALibration:YIG	none ?
Calibration	ZNC Cal	ZNCCAL	:CALibration:RBW	none ?
Calibration	Level Cal	LVLCAL	:CALibration:LEVel	none ?
Couple	Auto Couple	ACPL	:COUPle	none
Couple	Detector	DET	[:SENSe]:DETector[:FUNction]	NORMal AVERage POSitive SAMPle NEGative ?
Couple	Detector Auto	DETA	[:SENSe]:DETector:AUTO	OFF ON 0 1 ?
Display	Full Screen	FSCR	:DISPlay:FSCReen[:STATe]	OFF ON 0 1 ?
Display	Display Line Ampl	DL	:DISPlay:WINDow:TRACe:Y:DLINe	<amplitude> ?
Display	Display Line State	DLS	:DISPlay:WINDow:TRACe:Y:DLINe:STATe	OFF ON 0 1 ?
Display	Threshold Line Ampl	TH	:DISPlay:WINDow:TRACe:Y:TLINe	<amplitude> ?
Display	Threshold Line State	THS	:DISPlay:WINDow:TRACe:Y:TLINe:STATe	OFF ON 0 1 ?
Display	Title	TITLE	:DISPlay:ANNotation:TITLe:DATA	<string> ?
Display	Graticule	GRAT	:DISPlay:WINDow:TRACe:GRATICule:GRID[:STATe]	TYPE1 TYPE OFF ?
Display	Annotation	ANN	:DISPlay:WINDow:ANNotation[:ALL]	OFF ON 0 1 ?
Display	White Mode	WH	:DISPlay:WINDow:WHITe	OFF ON 0 1 ?
File	Read	FREAD	:MMEMory:CATalog	? <'directory_name'>
File	Save	FAVE	:MMEMory:STORe	<'file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<'file_name'>
File	Delete	FDEL	:MMEMory:DELete	<'file_name'>
File	Copy	FCOPY	:MMEMory:COPIY	<'file_name1'>,<'file_name2'>
File	Rename	FRENAME	:MMEMory:MOVE	<'file_name1'>,<'file_name2'>
File	Move	FMOVE	:MMEMory:DATA	<'file_name'>,definite_length ? <'file_name'>
Frequency	Center Frequency	CF	[:SENSe]:FREQUency:CENTer	<frequency> ?
Frequency	Start Frequency	FA	[:SENSe]:FREQUency:STARt	<frequency> ?
Frequency	Stop Frequency	FB	[:SENSe]:FREQUency:STOP	<frequency> ?
Frequency	CF Step	SS	[:SENSe]:FREQUency:CENTer:STEP[:INCRement]	<frequency> ?
Frequency	CF Step Auto	SSA	[:SENSe]:FREQUency:CENTer:STEP:AUTO	OFF ON 0 1 ?
Frequency	Frequency Offset	FO	[:SENSe]:FREQUency:OFFSet	<frequency> ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Frequency	Signal Track	STR[1~9]	:CALCulate:MARKer[1~9]:TRCKing[:STATe]	OFF ON 0 1 ?
In/Out	RF Coupling	RFC	:INPut:COUPling	AC DC ?
Limit Line	Limit Line Check State	LLCS[1~2]	:CALCulate:LLINe[1~2]:CHECK:STATe	OFF ON 0 1 ?
Limit Line	Limit Line Fail Count	LLFC[1~2]	:CALCulate:LLINe[1~2]:FAIL:COUNT	?
Limit Line	Pass/Fail Alarm	ALARM	:CALCulate:LLINe:ALARM	OFF ON 0 1 ?
Limit Line	Clear Limit Line	LLAO	:CALCulate:LLINe:AOFF	none
Marker	Marker State	MS[1~9]	:CALCulate:MARKer[1~9]:STATe	OFF ON 0 1 ?
Marker	Marker Mode	MM[1~9]	:CALCulate:MARKer[1~9]:MODE	POSition DELTA BAND SPAN OFF ?
Marker	Marker Freq	MF[1~9]	:CALCulate:MARKer[1~9]:X	<frequency> ?
Marker	Marker Point	MP[1~9]	:CALCulate:MARKer[1~9]:X:POSition	<integer> ?
Marker	Marker Start Freq	MFS[1~9]	:CALCulate:MARKer[1~9]:X:START	<frequency> ?
Marker	Marker Start Point	MPS[1~9]	:CALCulate:MARKer[1~9]:X:POSition:START	<integer> ?
Marker	Marker Stop Freq	MFE[1~9]	:CALCulate:MARKer[1~9]:X:STOP	<frequency> ?
Marker	Marker Stop Point	MPE[1~9]	:CALCulate:MARKer[1~9]:X:POSition:STOP	<integer> ?
Marker	Marker Center Freq	MFC[1~9]	:CALCulate:MARKer[1~9]:X:CENTer	<frequency> ?
Marker	Marker Center Point	MPC[1~9]	:CALCulate:MARKer[1~9]:X:POSition:CENTer	<integer> ?
Marker	Marker Span Freq	MFSP[1~9]	:CALCulate:MARKer[1~9]:X:SPAN	<frequency> ?
Marker	Marker Span Point	MPSP[1~9]	:CALCulate:MARKer[1~9]:X:POSition:SPAN	<integer> ?
Marker	Marker Amplitude	MA[1~9]	:CALCulate:MARKer[1~9]:Y	?
Marker	Marker Trace	MT[1~9]	:CALCulate:MARKer[1~9]:TRACe	1 2 3 ?
Marker	Marker Readout	MR[1~9]	:CALCulate:MARKer[1~9]:READout	FREQuency TIME ITIME PERiod ?
Marker	Marker Table	MTB	:CALCulate:MARKer:TABLE:STATe	OFF ON 0 1 ?
Marker	Marker All Off	MAO	:CALCulate:MARKer:AOFF	none
Marker Fctn	Marker Function	MFN[1~9]	:CALCulate:MARKer[1~9]:FUNCTion	NOISe COUNT OFF ?
Marker Fctn	Marker Noise Output	MFNY	:CALCulate:MARKer:FUNCTion:Y	?
Marker Fctn	Marker Counter Output	MFNX	:CALCulate:MARKer:FCOunt:X	?
Marker Fctn	Marker Counter Res	MFNR	:CALCulate:MARKer:FCOunt:RESolution	<frequency> ?
Marker Shift	Marker to Center Freq	MCF[1~9]	:CALCulate:MARKer[1~9][:SET]:CENTer	none
Marker Shift	Marker to CF Step	MSS[1~9]	:CALCulate:MARKer[1~9][:SET]:STEP	none
Marker Shift	Marker to Start Freq	MFA[1~9]	:CALCulate:MARKer[1~9][:SET]:START	none
Marker Shift	Marker to Stop Freq	MFB[1~9]	:CALCulate:MARKer[1~9][:SET]:STOP	none
Marker Shift	Marker to Ref Level	MRL[1~9]	:CALCulate:MARKer[1~9][:SET]:RLEVel	none

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Marker Shift	Marker Delta to CF	MDCF[1~9]	:CALCulate:MARKer[1~9][:SET]:DELTA:CENTer	none
Marker Shift	Marker Delta to CF Step	MDSS[1~9]	:CALCulate:MARKer[1~9][:SET]:DELTA:STEP	none
Marker Shift	Marker Delta to Span	MDSP[1~9]	:CALCulate:MARKer[1~9][:SET]:DELTA:SPAN	none
Measurement	Meas. Start	MEA	:MEASure:START	XDB ACP CHP OBW HD CCDF TOI SE SEM BP TP ?
Measurement	Meas. Average State	MEAAVG	:MEASure:AVERage[:STATe]	OFF ON 0 1 ?
Measurement	Meas. Average Mode	MEAAVGM	:MEASure:AVERage:TCONrol	EXPonential REPeat ?
Measurement	Meas. Average Reset	MEAAVGR	:MEASure:AVERage:RESET	none
Measurement	Meas. Average Count	MEAAVGN	:MEASure:AVERage:COUNT	<integer> ?
Measurement	Meas. OFF	MEAO	:MEASure:AOFF	none
Meas - XDB	X dB Point	XDBP[1~2]	[:SENSe]:XDB[1~2]:AMPLitude	<Amplitude> ?
Meas - XDB	X dB Left	XDBL[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:LEFT	?
Meas - XDB	X dB Right	XDBR[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:RIGHT	?
Meas - XDB	X dB Bandwidth	XDBRL[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:BANDwidth BWIDth	?
Meas - XDB	X dB Shape Factor	XDBS	:FETCh MEASure READ:XDB[1~2]:FREQuency:SHAPE	?
Meas - ACP	ACP Main Channel BW	ACPMC	[:SENSe]:ACPpower:BANDwidth BWIDth:INTegration	<frequency> ?
Meas - ACP	ACP Adjacent Channel BW	ACPAC	[:SENSe]:ACPpower:ADJacent:BANDwidth BWIDth:INTegration	<frequency> ?
Meas - ACP	ACP Channel Space BW	ACPCS	[:SENSe]:ACPpower:SPACE:BANDwidth BWIDth:INTegration	<frequency> ?
Meas - ACP	ACP Channel Space BW	ACPN	[:SENSe]:ACPpower:COUNT	<integer> ?
Meas - ACP	ACP Measurement Output	ACPOUT	:FETCh MEASure READ:ACPpower	?
Meas - ACP	ACP Measurement Output	ACPOUTC	:FETCh MEASure READ:ACPpower:CHPower	?
Meas - ACP	ACP Measurement Output	ACPOUTL[1~6]	:FETCh MEASure READ:ACPpower:LACPpower[1~6]	?
Meas - ACP	ACP Measurement Output	ACPOUTR[1~6]	:FETCh MEASure READ:ACPpower:RACPpower[1~6]	?
Meas - CHP	CHP Channel BW	CHPC	[:SENSe]:CHPower:BANDwidth BWIDth:INTegration	<frequency> ?
Meas - CHP	CHP Measurement Output	CHPOUT	:FETCh MEASure READ:CHPower	?
Meas - CHP	CHP Channel Power	CHPOUTC	:FETCh MEASure READ:CHPower:CHPower	?
Meas - CHP	CHP Pwr Spectral Density	CHPOUTD	:FETCh MEASure READ:CHPower:DENSity	?
Meas - OBW	OBW Percentage	OBWP	[:SENSe]:OBWidth:PERCent	<percent> ?
Meas - OBW	OBW Measurement Output	OBWOUT	:FETCh MEASure READ:OBWidth	?
Meas - HD	HD Number	HDN	[:SENSe]:HARMonics:NUMBer	<integer> ?
Meas - HD	HD Frequency	HF[1~5]	:FETCh MEASure READ:HARMonics:FREQuency[1~5]	?
Meas - HD	HD Amplitude	HA[1~5]	:FETCh MEASure READ:HARMonics:AMPLitude[1~5]	?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Meas - HD	Total Harm Distortion	HDOUT	:FETCh MEASure READ:HARMonics[:DISTortion]	?
Meas - TOI	TOI Measurement Output	TOIOUT	:FETCh MEASure READ:TOIN	?
Meas - TOI	TOI Base Lower Freq	TOIBLF	:FETCh MEASure READ:TOIN:BASE:LOWER:FREQuency	?
Meas - TOI	TOI Base Lower Level	TOIBLL	:FETCh MEASure READ:TOIN:BASE:LOWER:LEVel	?
Meas - TOI	TOI Base Lower Diff	TOIBLD	:FETCh MEASure READ:TOIN:BASE:LOWER:LEVel:DIFFerence	?
Meas - TOI	TOI Base Upper Freq	TOIBUF	:FETCh MEASure READ:TOIN:BASE:UPPER:FREQuency	?
Meas - TOI	TOI Base Upper Level	TOIBUL	:FETCh MEASure READ:TOIN:BASE:UPPER:LEVel	?
Meas - TOI	TOI Base Upper Diff	TOIBUD	:FETCh MEASure READ:TOIN:BASE:UPPER:LEVel:DIFFerence	?
Meas - TOI	TOI Worst Case Freq	TOIWCF	:FETCh MEASure READ:TOIN:WORST:FREQuency	?
Meas - TOI	TOI Worst Case Level	TOIWCL	:FETCh MEASure READ:TOIN:WORST:LEVel	?
Meas - TOI	TOI Worst Case Diff	TOIWCD	:FETCh MEASure READ:TOIN:WORST:LEVel:DIFFerence	?
Meas - TOI	TOI Worst Case IP3	TOIWCI	:FETCh MEASure READ:TOIN:WORST:LEVel:IP3	?
Meas - TOI	TOI Third Order Lower Freq	TOI3LF	:FETCh MEASure READ:TOIN:THIRD:LOWER:FREQuency	?
Meas - TOI	TOI Third Order Lower Level	TOI3LL	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel	?
Meas - TOI	TOI Third Order Lower Diff	TOI3LD	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel:DIFFerence	?
Meas - TOI	TOI Third Order Lower IP3	TOI3LI	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel:IP3	?
Meas - TOI	TOI Third Order Upper Freq	TOI3UF	:FETCh MEASure READ:TOIN:THIRD:UPPER:FREQuency	?
Meas - TOI	TOI Third Order Upper Level	TOI3UL	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel	?
Meas - TOI	TOI Third Order Upper Diff	TOI3UD	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel:DIFFerence	?
Meas - TOI	TOI Third Order Upper IP3	TOI3UI	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel:IP3	?
Meas - SE	SE Measurement Output	SEOUT	:FETCh MEASure READ:SPURious	?
Meas - SEM	SEM Meas Type	SEMMT	[:SENSe]:SEMAsk:METhod	TPRef PSDRef ?
Meas - SEM	SEM Measurement Output	SEMOUT	:FETCh MEASure READ:SEMAsk	?
Meas - SEM	SEM Ref. Channel CHP	SEMPWR	:FETCh MEASure READ:SEMAsk:REFerence:CHPower	?
Meas - SEM	SEM Ref. Channel PSD	SEMPSD	:FETCh MEASure READ:SEMAsk:REFerence:DENSity	?
Meas - SEM	SEM Lower Peak Freq	SEMLF1~6	:FETCh MEASure READ:SEMAsk:LOWER1~6:FREQuency	?
Meas - SEM	SEM Lower CHP	SEMLPWR1~6	:FETCh MEASure READ:SEMAsk:LOWER1~6:CHPower	?
Meas - SEM	SEM Lower PSD	SEMLPSD1~6	:FETCh MEASure READ:SEMAsk:LOWER1~6:DENSity	?
Meas - SEM	SEM Upper Peak Freq	SEMUF1~6	:FETCh MEASure READ:SEMAsk:UPPER1~6:FREQuency	?
Meas - SEM	SEM Upper CHP	SEMUPWR1~6	:FETCh MEASure READ:SEMAsk:UPPER1~6:CHPower	?
Meas - SEM	SEM Upper PSD	SEMUPSD1~6	:FETCh MEASure READ:SEMAsk:UPPER1~6:DENSity	?
Meas - BP	BP Meas Type	BPMT	[:SENSe]:BPOWer:METhod	THReshold BWIDth ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Meas - BP	BP Threshold	BPTH	[:SENSe]:BPOWer:THReshold	<amplitude> ?
Meas - BP	BP Measurement Output	BPOUT	:FETCh MEASure READ:BPOWer	?
Meas - BP	BP Average Power	BPAVERP	:FETCh MEASure READ:BPOWer:POWer:AVERAge	?
Meas - BP	BP Min Power	BPMINP	:FETCh MEASure READ:BPOWer:POWer:MIN	?
Meas - BP	BP Max Power	BPMAXP	:FETCh MEASure READ:BPOWer:POWer:MAX	?
Meas - BP	BP Burst Length	BPBL	:FETCh MEASure READ:BPOWer:WIDTH	?
Meas - TP	TP Measurement Output	TPOUT	:FETCh MEASure READ:TPower	?
Meas - TP	TP Channel Power	TPOUTC	:FETCh MEASure READ:TPower:CHPower	?
Meas - TP	TP Pwr Spectral Density	TPOUTD	:FETCh MEASure READ:TPower:DENSity	?
Peak Search	Peak Search	MPK[1~9]	:CALCulate:MARKer[1~9]:MAXimum	none
Peak Search	Next Peak Search	MPKN[1~9]	:CALCulate:MARKer[1~9]:MAXimum:NEXT	none
Peak Search	Next Left Peak Search	MPKL[1~9]	:CALCulate:MARKer[1~9]:MAXimum:LEFT	none
Peak Search	Next Right Peak Search	MPKR[1~9]	:CALCulate:MARKer[1~9]:MAXimum:RIGHT	none
Peak Search	Minimum Search	MPKM[1~9]	:CALCulate:MARKer[1~9]:MINimum	none
Peak Search	Peak to Peak Search	MPKP[1~9]	:CALCulate:MARKer[1~9]:PTPeak	none
Peak Search	Signal Track	MPKT[1~9]	:CALCulate:MARKer[1~9]:TRCKing[:STATe]	OFF ON 0 1 ?
Peak Search	Continuous Peak	MPKC[1~9]	:CALCulate:MARKer[1~9]:CPEak[:STATe]	OFF ON 0 1 ?
Peak Search	Multi Peak Number	MMPKN	:CALCulate:MARKer:PEAK:MULTi:NUMber	<integer> ?
Peak Search	Multi Peak	MMPK	:CALCulate:MARKer:PEAK:MULTi	none
Peak Search	Multi Peak Trace	MMPKT	:CALCulate:MARKer:PEAK:MULTi:TRACe	<integer> ?
Peak Search	Peak Excursion	MPKE	:CALCulate:MARKer:PEAK:EXCursion	<amplitude> ?
Peak Search	Peak Threshold	MPKTH	:CALCulate:MARKer:PEAK:THReshold	<amplitude> ?
Peak Search	Peak Parameter	MPKPA	:CALCulate:MARKer:PEAK:SEARch:MODE	MAX PARAmeter ?
Preset	Preset	PRST	:SYSTem:PRESet	none
Printer	Hard Copy	HCOPIY	:HCOPIy[:IMMediate]	none
Span	Span	SP	[:SENSe]:FREQUency:SPAN	<frequency> ?
Span	Full Span	FS	[:SENSe]:FREQUency:SPAN:FULL	none
Span	Zero Span	ZS	[:SENSe]:FREQUency:SPAN:ZERO	none
Span	Last Span	LS	[:SENSe]:FREQUency:SPAN:PREVious	none
Span	Zoom In	ZI	[:SENSe]:FREQUency:SPAN:ZIN	none
Span	Zoom Out	ZO	[:SENSe]:FREQUency:SPAN:ZOUT	none
Sweep	Sweep Time	ST	[:SENSe]:SWEep:TIME	<time> ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Sweep	Sweep Time Auto	STA	[:SENSe]:SWEep:TIME:AUTO	OFF ON 0 1 ?
Sweep	Continuous	CO	:INITiate:CONTinuous	OFF ON 0 1 ?
Sweep	Single	SI	:INITiate[:IMMediate]	none
Sweep	Sweep Time Accuracy	STAC	:INITiate:ACCuracy	NORMal ACCuracy ?
Sweep	Point	PO	[:SENSe]:SWEep:POINts	<integer> ?
Trace	Trace Function	TRF[1~3]	:TRACe[1~2] 3:MODE	WRITE MAXHold MINHold VIEW BLANK ?
Trace	Send Trace Data	TRD	:TRACe[:DATA]	TRACE1 TRACE2 TRACE3,<ASCII_data>
Trace	Query Trace Data	TRD	:TRACe[:DATA]	? TRACE1 TRACE2 TRACE3
Trace	Trace Data Format	TDF	:TRACe:FORMat	ASCIi REAL,64 INT,32 ?
Trigger	Trigger Source	TSO	:TRIGger[:SEQuence]:SOURce	IMMediate VIDeo LINE EXTernal ?
Trigger	Trigger Video Level	TVL	:TRIGger[:SEQuence]:VIDeo:LEVel	<amplitude> ?
Trigger	Trigger Slope	TSL	:TRIGger[:SEQuence]:SLOPe	POSitive NEGative ?
Trigger	Trigger Delay	TD	:TRIGger[:SEQuence]:DELay	<time> ?
Trigger	Trigger Delay State	TDS	:TRIGger[:SEQuence]:DELay:STATe	OFF ON 0 1 ?
Tune	Auto Tune	ATN	[:SENSe]:TUNE:AUTO	none
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Others	ESE2	ESE2		<integer> ?
Others	ESR2	ESR2		?
Others	Error Code	ERR		?
TG (Option)	TG State	TGEN		OFF ON 0 1 ?
TG (Option)	TG Output Level	TGLEV		<amplitude> ?
TG (Option)	TG Normalize	TGNORM		OFF ON 0 1 ?
TG (Option)	Power Sweep	PWRSWP		OFF ON 0 1 ?

REMOTE CONTROL

< Catalog Order (CCDF Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[:RF]:GAIN[:STATe]	OFF ON 0 1 ?
Bandwidth	Resolution Bandwidth	RB	[:SENSe]:BANDwidth BWIDth[:RESolution]	<frequency> ?
Bandwidth	Video Bandwidth	VB	[:SENSe]:BANDwidth BWIDth:VIDeo	<frequency> ?
Bandwidth	Video Bandwidth Auto	VBA	[:SENSe]:BANDwidth BWIDth:VIDeo:AUTO	OFF ON 0 1 ?
Bandwidth	VBW/RBW	VBRB	[:SENSe]:BANDwidth BWIDth:VIDeo:RATio	<ratio> ?
Display	Full Screen	FSCR	:DISPlay:CCDF:FSCReen[:STATe]	OFF ON 0 1 ?
Display	Graticule	GRAT	:DISPlay:CCDF:WINDow:TRACe:GRATicule:GRID[:STATe]	OFF ON 0 1 ?
Display	White Mode	WH	:DISPlay:CCDF:WINDow:WHITe	OFF ON 0 1 ?
Display	Store Ref Trace	SREF	:DISPlay:CCDF:WINDow:RLEVel:STORe	none
Display	Ref Trace State	REFS	:DISPlay:CCDF:WINDow:RLEVel[:STATe]	OFF ON 0 1 ?
Display	Gaussian Trace State	GAUS	:DISPlay:CCDF:WINDow:GAUSSian[:STATe]	OFF ON 0 1 ?
File	Read	FREAD	:MMEMory:CATalog	? <'directory_name'>
File	Save	FSAVE	:MMEMory:STORe	<'file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<'file_name'>
File	Delete	FDEL	:MMEMory:DELeTe	<'file_name'>
File	Copy	FCOPY	:MMEMory:COPIY	<'file_name1'>,<'file_name2'>
File	Rename	FRENAME	:MMEMory:MOVE	<'file_name1'>,<'file_name2'>
File	Move	FMOVE	:MMEMory:DATA	<'file_name'>,definite_length ? <'file_name'>
Frequency	Center Frequency	CF	[:SENSe]:FREQuency:CENTer	<frequency> ?
Frequency	Frequency Offset	FO	[:SENSe]:FREQuency:OFFSet	<frequency> ?
Marker	Marker State	MS[1~9]	:CALCulate:CCDF:MARKer[1~9]:STATe	OFF ON 0 1 ?
Marker	Marker Mode	MM[1~9]	:CALCulate:CCDF:MARKer[1~9]:MODE	POSition DELTA BAND SPAN OFF ?
Marker	Marker Amplitude	MA[1~9]	:CALCulate:CCDF:MARKer[1~9]:X	<amplitude> ?
Marker	Marker Percent	MP[1~9]	:CALCulate:CCDF:MARKer[1~9]:Y	?
Marker	Marker Trace	MT[1~9]	:CALCulate:CCDF:MARKer[1~9]:TRACe	1 2 3 ?
Marker	Marker All Off	MAO	:CALCulate:CCDF:MARKer:AOFF	none
Measurement	Meas. Start	MEA	:MEASure:STARt	XDB ACP CHP OBW HD CCDF ?
Measurement	Meas. OFF	MEAO	:MEASure:AOFF	none
Meas - CCDF	Count	CCDFN	[:SENSe]:CCDF:COUNT	<integer> ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Meas - CCDF	Output	CCDFOUT[PWR PER]	:FETCh MEASure READ:CCDF[:POWER PERCent]	?
Meas - CCDF	Output	CCDFPER10	:FETCh MEASure READ:CCDF:PERCent10	?
Meas - CCDF	Output	CCDFPER1	:FETCh MEASure READ:CCDF:PERCent1	?
Meas - CCDF	Output	CCDFPER01	:FETCh MEASure READ:CCDF:PERCent01	?
Meas - CCDF	Output	CCDFPER001	:FETCh MEASure READ:CCDF:PERCent001	?
Meas - CCDF	Output	CCDFPER0001	:FETCh MEASure READ:CCDF:PERCent0001	?
Meas - CCDF	Output	CCDFPER00001	:FETCh MEASure READ:CCDF:PERCent00001	?
Meas - CCDF	Output	CCDFCRE	:FETCh MEASure READ:CCDF:CRESt	?
Preset	Preset	PRST	:SYSTem:PRESet	none
Printer	Hard Copy	HCOPY	:HCOPY[:IMMEDIATE]	none
Span	Scale/Divide	SD	:DISPlay:CCDF:WINDow:TRACe:Y[:SCALE]:PDIVision	<amplitude> ?
Sweep	Continuous	CO	:INITiate:CONTinuous	OFF ON 0 1 ?
Sweep	Single	SI	:INITiate:IMMEDIATE	none
Trigger	Trigger Source	TSO	:TRIGger[:SEQUence]:SOURce	IMMEDIATE VIDeo LINE EXTernal ?
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Others	ESE2	ESE2		<integer> ?
Others	ESR2	ESR2		?
Others	Error Code	ERR		?

REMOTE CONTROL

< Catalog Order (Phase Noise Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Amplitude	Ref. Level	RL	:DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:RLEVel	<amplitude> ?
Amplitude	Scale/Div	DS	:DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:PDIVision	<amplitude> ?
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[:RF]:GAIN[:STATe]	OFF ON 0 1 ?
Display	Full Screen	FSCR	:DISPlay:LPLot:FSCReen[:STATe]	OFF ON 0 1 ?
Display	Graticule	GRAT	:DISPlay:LPLot:WINDow:TRACe:GRATICule:GRID[:STATe]	OFF ON 0 1 ?
Display	Annotation	ANN	:DISPlay:LPLot:WINDow:ANNotation[:ALL]	OFF ON 0 1 ?
Display	White Mode	WH	:DISPlay:LPLot:WINDow:WHITe	OFF ON 0 1 ?
File	Read	FREAD	:MMEMory:CATalog	? <'directory_name'>
File	Save	FSAVE	:MMEMory:STORe	<'file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<'file_name'>
File	Delete	FDEL	:MMEMory:DELeTe	<'file_name'>
File	Copy	FCOPY	:MMEMory:COPIY	<'file_name1'>,<'file_name2'>
File	Rename	FRENAME	:MMEMory:MOVE	<'file_name1'>,<'file_name2'>
File	Move	FMOVE	:MMEMory:DATA	<'file_name'>,<definite_length_block ? <'file_name'>
Frequency	Carrier Freq	CF	[:SENSe]:LPLot:FREQuency:CARRier	<frequency> ?
Frequency	Carrier Search	CFS	[:SENSe]:LPLot:FREQuency:CARRier:SEARCh	none
Marker	Marker State	MS[1~9]	:CALCulate:LPLot:MARKer[1~9]:STATe	OFF ON 0 1 ?
Marker	Marker Mode	MM[1~9]	:CALCulate:LPLot:MARKer[1~9]:MODE	POSition DELTA RMSDegree RMSRadian RMSJitter RFM OFF ?
Marker	Marker Frequency	MF[1~9]	:CALCulate:LPLot:MARKer[1~9]:X	<frequency> ?
Marker	Marker Amplitude	MA[1~9]	:CALCulate:LPLot:MARKer[1~9]:Y	?
Marker	Marker Trace	MT[1~9]	:CALCulate:LPLot:MARKer[1~9]:TRACe	1 2 ?
Marker	Marker Table	MTB	:CALCulate:LPLot:MARKer:TABLE[:STATe]	OFF ON 0 1 ?
Marker	Marker All Off	MAO	:CALCulate:LPLot:MARKer:AOFF	none
Measurement	Log Plot	LP	:MEASure:LPLot	none
Measurement	Average State	AVG	[:SENSe]:LPLot:AVERAge[:STATe]	OFF ON 0 1 ?
Measurement	Avg. Count	AVGC	[:SENSe]:LPLot:AVERAge:COUNT	<integer> ?
Measurement	Smoothing	SM	[:SENSe]:LPLot:SMOoth	<percentage> ?
Measurement	Filtering	FT	[:SENSe]:LPLot:FILTering	<value> ?
Measurement	Decade Table	DTB	:CALCulate:LPLot:DECade:TABLE[:STATe]	OFF ON 0 1 ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Mode	Mode	MODE	:INSTrument[:SElect]	SA PNOISE ?
Preset	Preset	PRST	:SYSTem:PRESet	none
Printer	Hard Copy	HCOPI	:HCOPI[:IMMediate]	none
Span	Start Offset	FA	[:SENSe]:LPLot:FREQuency:OFFSet:STARt	<frequency> ?
Span	Stop Offset	FB	[:SENSe]:LPLot:FREQuency:OFFSet:STOP	<frequency> ?
Sweep	Single	SI	:INITiate:LPLot[:IMMediate]	none
Sweep	Continuous	CO	:INITiate:LPLot:CONTInuous	OFF ON 0 1 ?
Trigger	Trigger Source	TSO	:TRIGger[:SEQuence]:SOURce	IMMediate VIDeo LINE EXTernal ?
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Others	ESE2	ESE2		<integer> ?
Others	ESR2	ESR2		?
Others	Error Code	ERR		?

< SA Command Order (Spectrum Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Bandwidth	Auto Bandwidth	AB	[:SENSe]:BANDwidth BWIDth:AUTO	none
Meas - ACP	ACP Adjacent Channel BW	ACPAC	[:SENSe]:ACPower:ADJacent:Bandwidth BWIDth:INTEgration	<frequency> ?
Meas - ACP	ACP Channel Space BW	ACPCS	[:SENSe]:ACPower:SPACe:Bandwidth BWIDth:INTEgration	<frequency> ?
Couple	Auto Couple	ACPL	:COUPle	none
Meas - ACP	ACP Main Channel BW	ACPMC	[:SENSe]:ACPower:Bandwidth BWIDth:INTEgration	<frequency> ?
Meas - ACP	ACP Channel Space BW	ACPN	[:SENSe]:ACPower:COUNT	<integer> ?
Meas - ACP	ACP Measurement Output	ACPOUT	:FETCh MEASure READ:ACPower	?
Meas - ACP	ACP Measurement Output	ACPOUTC	:FETCh MEASure READ:ACPower:CHPower	?
Meas - ACP	ACP Measurement Output	ACPOUTL[1-6]	:FETCh MEASure READ:ACPower:LACPower[1-6]	?
Meas - ACP	ACP Measurement Output	ACPOUTR[1-6]	:FETCh MEASure READ:ACPower:RACPower[1-6]	?
Limit Line	Pass/Fail Alarm	ALARM	:CALCulate:LLINe:ALARM	OFF ON 0 1 ?
Auxiliary	AM Demodulation State	AMS		OFF ON 0 1 ?
Display	Annotation	ANN	:DISPlay:WINDow:ANNotation[:ALL]	OFF ON 0 1 ?
Amplitude	Attenuation	AT	[:SENSe]:POWer[:RF]:ATTenuation	<amplitude> ?
Amplitude	Attenuation Auto	ATA	[:SENSe]:POWer[:RF]:ATTenuation:AUTO	OFF ON 0 1 ?
Tune	Auto Tune	ATN	[:SENSe]:TUNE:AUTO	none
Amplitude	Amplitude Units	AU	:UNIT:POWer	DBM DBMV DBMA V W A DBUV DBUA ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Auxiliary	Audio Level	AUDIOLEV		<integer> ?
Auxiliary	Audio State	AUDIOS		OFF ON 0 1 ?
Average	Average	AVG	[:SENSe]:AVERAge[:STATe]	OFF ON 0 1 ?
Average	Average Count	AVGC	[:SENSe]:AVERAge:COUNT	<integer> ?
Average	Average CLear	AVGCL	[:SENSe]:AVERAge:CLear	none
Average	Avg/VBW Type	AVGT	[:SENSe]:AVERAge:TYPE	RMS LOG SCALar ?
Average	Avg/VBW Type Auto	AVGTA	[:SENSe]:AVERAge:TYPE:AUTO	OFF ON 0 1 ?
Meas - BP	BP Average Power	BPAVERP	:FETCh MEASure READ:BPOWer:POWer:AVERAge	?
Meas - BP	BP Burst Length	BPBL	:FETCh MEASure READ:BPOWer:WIDTh	?
Meas - BP	BP Max Power	BPMAXP	:FETCh MEASure READ:BPOWer:POWer:MAX	?
Meas - BP	BP Min Power	BPMINP	:FETCh MEASure READ:BPOWer:POWer:MIN	?
Meas - BP	BP Meas Type	BPMT	[:SENSe]:BPOWer:METhod	THReshold BWIDth ?
Meas - BP	BP Measurement Output	BPOUT	:FETCh MEASure READ:BPOWer	?
Meas - BP	BP Threshold	BPTH	[:SENSe]:BPOWer:THReshold	<amplitude> ?
Calibration	Align All Now	CALALL	:CALibration[:ALL]	none ?
Frequency	Center Frequency	CF	[:SENSe]:FREQuency:CENTer	<frequency> ?
Meas - CHP	CHP Channel BW	CHPC	[:SENSe]:CHPower:Bandwidth BWIDth:INTEgration	<frequency> ?
Meas - CHP	CHP Measurement Output	CHPOUT	:FETCh MEASure READ:CHPower	?
Meas - CHP	CHP Channel Power	CHPOUTC	:FETCh MEASure READ:CHPower:CHPower	?
Meas - CHP	CHP Pwr Spectral Density	CHPOUTD	:FETCh MEASure READ:CHPower:DENSity	?
Sweep	Continuous	CO	:INITiate:CONTInuous	OFF ON 0 1 ?
Amplitude	Delete All Corrections	COAD	[:SENSe]:CORRection:CSET:ALL:DELeTe	none
Amplitude	Apply Corrections	COAS	[:SENSe]:CORRection:CSET:ALL[:STATe]	OFF ON 0 1 ?
Amplitude	Correction State	COS1 2 3 4	[:SENSe]:CORRection:CSET1 2 3 4[:STATe]	OFF ON 0 1 ?
Couple	Detector	DET	[:SENSe]:DETector[:FUNction]	NORMal AVERAge POSitive SAMPle NEGative ?
Couple	Detector Auto	DETA	[:SENSe]:DETector:AUTO	OFF ON 0 1 ?
Display	Display Line Ampl	DL	:DISPlay:WINDow:TRACe:Y:DLIne	<amplitude> ?
Display	Display Line State	DLS	:DISPlay:WINDow:TRACe:Y:DLIne:STATe	OFF ON 0 1 ?
Others	Error Code	ERR		?
Others	ESE2	ESE2		<integer> ?
Others	ESR2	ESR2		?
Frequency	Start Frequency	FA	[:SENSe]:FREQuency:STARt	<frequency> ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Frequency	Stop Frequency	FB	[:SENSe]:FREQUency:STOP	<frequency> ?
File	Copy	FCOPY	:MMEMory:COpy	<`file_name1'>,<`file_name2'>
File	Delete	FDEL	:MMEMory:DELeTe	<`file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<`file_name'>
File	Move	FMOVE	:MMEMory:DATA	<`file_name'>,<definite_length ? <'file_name'>
Auxiliary	FM Demodulation State	FMS		OFF ON 0 1 ?
Frequency	Frequency Offset	FO	[:SENSe]:FREQUency:OFFSet	<frequency> ?
File	Read	FREAD	:MMEMory:CATalog	? <`directory_name'>
File	Rename	FRENAME	:MMEMory:MOVE	<`file_name1'>,<`file_name2'>
Span	Full Span	FS	[:SENSe]:FREQUency:SPAN:FULL	none
File	Save	FSAVE	:MMEMory:STORE	<`file_name'>
Display	Full Screen	FSCR	:DISPlay:FSCReen[::STATe]	OFF ON 0 1 ?
Display	Graticule	GRAT	:DISPlay:WINDow:TRACe:GRATicule:GRID[::STATe]	TYPE1 TYPE OFF ?
Meas - HD	HD Amplitude	HA[1~5]	:FETCh MEASure READ:HARMonics:AMPLitude[1~5]	?
Printer	Hard Copy	HCOPY	:HCOPy[:IMMediate]	none
Meas - HD	HD Number	HDN	[:SENSe]:HARMonics:NUMBer	<integer> ?
Meas - HD	Total Harm Distortion	HDOUT	:FETCh MEASure READ:HARMonics[::DISTortion]	?
Meas - HD	HD Frequency	HF[1~5]	:FETCh MEASure READ:HARMonics:FREQUency[1~5]	?
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[::RF]:GAIN[::STATe]	OFF ON 0 1 ?
Limit Line	Clear Limit Line	LLAO	:CALCulate:LLINe:AOff	none
Limit Line	Limit Line Check State	LLCS[1~2]	:CALCulate:LLINe[1~2]:CHECK:STATe	OFF ON 0 1 ?
Limit Line	Limit Line Fail Count	LLFC[1~2]	:CALCulate:LLINe[1~2]:FAIL:COUNT	?
Span	Last Span	LS	[:SENSe]:FREQUency:SPAN:PREVious	none
Calibration	Level Cal	LVLCAL	:CALibration:LEVel	none ?
Marker	Marker Amplitude	MA[1~9]	:CALCulate:MARKer[1~9]:Y	?
Marker	Marker All Off	MAO	:CALCulate:MARKer:AOff	none
Marker Shift	Marker to Center Freq	MCF[1~9]	:CALCulate:MARKer[1~9][::SET]:CENTer	none
Marker Shift	Marker Delta to CF	MDCF[1~9]	:CALCulate:MARKer[1~9][::SET]:DELTA:CENTer	none
Marker Shift	Marker Delta to Span	MDSP[1~9]	:CALCulate:MARKer[1~9][::SET]:DELTA:SPAN	none
Marker Shift	Marker Delta to CF Step	MDSS[1~9]	:CALCulate:MARKer[1~9][::SET]:DELTA:STEP	none
Measurement	Meas. Start	MEA	:MEASure:START	XDB ACP CHP OBW HD CCDF TOI SE SEM BP TP ?
Measurement	Meas. Average State	MEAAVG	:MEASure:AVERage[::STATe]	OFF ON 0 1 ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Measurement	Meas. Average Mode	MEAAVGM	:MEASure:AVERage:TCONrol	EXPonential REPeat ?
Measurement	Meas. Average Count	MEAAVGN	:MEASure:AVERage:COUNT	<integer> ?
Measurement	Meas. Average Reset	MEAAVGR	:MEASure:AVERage:RESET	none
Measurement	Meas. OFF	MEAO	:MEASure:AOFF	none
Marker	Marker Freq	MF[1~9]	:CALCulate:MARKer[1~9]:X	<frequency> ?
Marker Shift	Marker to Start Freq	MFA[1~9]	:CALCulate:MARKer[1~9]:[SET]:START	none
Marker Shift	Marker to Stop Freq	MFB[1~9]	:CALCulate:MARKer[1~9]:[SET]:STOP	none
Marker	Marker Center Freq	MFC[1~9]	:CALCulate:MARKer[1~9]:X:CENTer	<frequency> ?
Marker	Marker Stop Freq	MFE[1~9]	:CALCulate:MARKer[1~9]:X:STOP	<frequency> ?
Marker Fctn	Marker Function	MFN[1~9]	:CALCulate:MARKer[1~9]:FUNCTion	NOISe COUNT OFF ?
Marker Fctn	Marker Counter Res	MFNR	:CALCulate:MARKer:FCOunt:RESolution	<frequency> ?
Marker Fctn	Marker Counter Output	MFNX	:CALCulate:MARKer:FCOunt:X	?
Marker Fctn	Marker Noise Output	MFNY	:CALCulate:MARKer:FUNCTion:Y	?
Marker	Marker Start Freq	MFS[1~9]	:CALCulate:MARKer[1~9]:X:START	<frequency> ?
Marker	Marker Span Freq	MFSP[1~9]	:CALCulate:MARKer[1~9]:X:SPAN	<frequency> ?
Marker	Marker Mode	MM[1~9]	:CALCulate:MARKer[1~9]:MODE	POSition DELTA BAND SPAN OFF ?
Peak Search	Multi Peak	MMPK	:CALCulate:MARKer:PEAK:MULTi	none
Peak Search	Multi Peak Number	MMPKN	:CALCulate:MARKer:PEAK:MULTi:NUMber	<integer> ?
Peak Search	Multi Peak Trace	MMPKT	:CALCulate:MARKer:PEAK:MULTi:TRACe	<integer> ?
Marker	Marker Point	MP[1~9]	:CALCulate:MARKer[1~9]:X:POSition	<integer> ?
Marker	Marker Center Point	MPC[1~9]	:CALCulate:MARKer[1~9]:X:POSition:CENTer	<integer> ?
Marker	Marker Stop Point	MPE[1~9]	:CALCulate:MARKer[1~9]:X:POSition:STOP	<integer> ?
Peak Search	Peak Search	MPK[1~9]	:CALCulate:MARKer[1~9]:MAXimum	none
Peak Search	Continuous Peak	MPKC[1~9]	:CALCulate:MARKer[1~9]:CPEak[:STATe]	OFF ON 0 1 ?
Peak Search	Peak Excursion	MPKE	:CALCulate:MARKer:PEAK:EXCURsion	<amplitude> ?
Peak Search	Next Left Peak Search	MPKL[1~9]	:CALCulate:MARKer[1~9]:MAXimum:LEFT	none
Peak Search	Minimum Search	MPKM[1~9]	:CALCulate:MARKer[1~9]:MINimum	none
Peak Search	Next Peak Search	MPKN[1~9]	:CALCulate:MARKer[1~9]:MAXimum:NEXT	none
Peak Search	Peak to Peak Search	MPKP[1~9]	:CALCulate:MARKer[1~9]:PTPeak	none
Peak Search	Peak Parameter	MPKPA	:CALCulate:MARKer:PEAK:SEARch:MODE	MAX PARAmeter ?
Peak Search	Next Right Peak Search	MPKR[1~9]	:CALCulate:MARKer[1~9]:MAXimum:RIGHT	none
Peak Search	Signal Track	MPKT[1~9]	:CALCulate:MARKer[1~9]:TRCKing[:STATe]	OFF ON 0 1 ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Peak Search	Peak Threshold	MPKTH	:CALCulate:MARKer:PEAK:THReshold	<amplitude> ?
Marker	Marker Start Point	MPS[1~9]	:CALCulate:MARKer[1~9]:X:POSition:STARt	<integer> ?
Marker	Marker Span Point	MPSP[1~9]	:CALCulate:MARKer[1~9]:X:POSition:SPAN	<integer> ?
Marker	Marker Readout	MR[1~9]	:CALCulate:MARKer[1~9]:READout	FREQuency TIME ITIME PERiod ?
Marker Shift	Marker to Ref Level	MRL[1~9]	:CALCulate:MARKer[1~9][:SET]:RLEVel	none
Marker	Marker State	MS[1~9]	:CALCulate:MARKer[1~9]:STATe	OFF ON 0 1 ?
Marker Shift	Marker to CF Step	MSS[1~9]	:CALCulate:MARKer[1~9][:SET]:STEP	none
Marker	Marker Trace	MT[1~9]	:CALCulate:MARKer[1~9]:TRACe	1 2 3 ?
Marker	Marker Table	MTB	:CALCulate:MARKer:TABLE:STATe	OFF ON 0 1 ?
Meas - OBW	OBW Measurement Output	OBWOUT	:FETCh MEASure READ:OBWidth	?
Meas - OBW	OBW Percentage	OBWP	[:SENSe]:OBWidth:PERCent	<percent> ?
Calibration	Periodic Temperature Cal	PCAL	:CALibration:PTEMPerature	OFF ON 0 1 ?
Calibration	Pre-Filter Cal	PFCAL	:CALibration:YIG	none ?
Sweep	Point	PO	[:SENSe]:SWEep:POINts	<integer> ?
Preset	Preset	PRST	:SYSTem:PRESet	none
TG (Option)	Power Sweep	PWRSWP		OFF ON 0 1 ?
Bandwidth	Resolution Bandwidth	RB	[:SENSe]:BANDwidth BWIDth[:RESolution]	<frequency> ?
Bandwidth	Resolution Bandwidth Auto	RBA	[:SENSe]:BANDwidth BWIDth[:RESolution]:AUTO	OFF ON 0 1 ?
In/Out	RF Coupling	RFC	:INPut:COUPling	AC DC ?
Amplitude	Reference Level	RL	:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel	<amplitude> ?
Amplitude	Ref Level Offset	RLO	:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet	<amplitude> ?
Amplitude	Scale/Divide	SD	:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision	<amplitude> ?
Meas - SEM	SEM Lower Peak Freq	SEMLF1~6	:FETCh MEASure READ:SEMAsk:LOWER1~6:FREQuency	?
Meas - SEM	SEM Lower PSD	SEMLPSD1~6	:FETCh MEASure READ:SEMAsk:LOWER1~6:DENSity	?
Meas - SEM	SEM Lower CHP	SEMLPWR1~6	:FETCh MEASure READ:SEMAsk:LOWER1~6:CHPower	?
Meas - SEM	SEM Meas Type	SEMMT	[:SENSe]:SEMAsk:METHod	TPRef PSDRef ?
Meas - SEM	SEM Measurement Output	SEMOUT	:FETCh MEASure READ:SEMAsk	?
Meas - SEM	SEM Ref. Channel PSD	SEMTPSD	:FETCh MEASure READ:SEMAsk:REFerence:DENSity	?
Meas - SEM	SEM Ref. Channel CHP	SEMTPWR	:FETCh MEASure READ:SEMAsk:REFerence:CHPower	?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Meas - SEM	SEM Upper Peak Freq	SEMUF1~6	:FETCh MEASure READ:SEMAsk:UPPER1~6:FREQuency	?
Meas - SEM	SEM Upper PSD	SEMUPSD1~6	:FETCh MEASure READ:SEMAsk:UPPER1~6:DENSity	?
Meas - SEM	SEM Upper CHP	SEMUPWR1~6	:FETCh MEASure READ:SEMAsk:UPPER1~6:CHPpower	?
Meas - SE	SE Measurement Output	SEOUT	:FETCh MEASure READ:SPURious	?
Sweep	Single	SI	:INITiate[:IMMediate]	none
Span	Span	SP	[:SENSe]:FREQuency:SPAN	<frequency> ?
Bandwidth	Span/RBW	SPRB	[:SENSe]:FREQuency:SPAN:BANDwidth BWIDth:VIDeo:RATio	<ratio> ?
Bandwidth	Span/RBW Auto	SPRBA	[:SENSe]:FREQuency:SPAN:BANDwidth BWIDth:VIDeo:RATio:AUTO	OFF ON 0 1 ?
Frequency	CF Step	SS	[:SENSe]:FREQuency:CENTer:STEP[:INCRement]	<frequency> ?
Frequency	CF Step Auto	SSA	[:SENSe]:FREQuency:CENTer:STEP:AUTO	OFF ON 0 1 ?
Sweep	Sweep Time	ST	[:SENSe]:SWEep:TIME	<time> ?
Sweep	Sweep Time Auto	STA	[:SENSe]:SWEep:TIME:AUTO	OFF ON 0 1 ?
Sweep	Sweep Time Accuracy	STAC	:INITiate:ACCuracy	NORMal ACCuracy ?
Frequency	Signal Track	STR[1~9]	:CALCulate:MARKer[1~9]:TRCKing[:STATe]	OFF ON 0 1 ?
Amplitude	Scale Type	STY	:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing	LINear LOGarithmic ?
Trigger	Trigger Delay	TD	:TRIGger[:SEQuence]:DELay	<time> ?
Trace	Trace Data Format	TDF	:TRACe:FORMat	ASCIi REAL,64 INT,32 ?
Trigger	Trigger Delay State	TDS	:TRIGger[:SEQuence]:DELay:STATe	OFF ON 0 1 ?
TG (Option)	TG State	TGEN		OFF ON 0 1 ?
TG (Option)	TG Output Level	TGLEV		<amplitude> ?
TG (Option)	TG Normalize	TGNORM		OFF ON 0 1 ?
Display	Threshold Line Ampl	TH	:DISPlay:WINDow:TRACe:Y:TLINe	<amplitude> ?
Display	Threshold Line State	THS	:DISPlay:WINDow:TRACe:Y:TLINe:STATe	OFF ON 0 1 ?
Display	Title	TITLE	:DISPlay:ANNotation:TITLe:DATA	<string> ?
Meas - TOI	TOI Third Order Lower Diff	TOI3LD	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel:DIFFerence	?
Meas - TOI	TOI Third Order Lower Freq	TOI3LF	:FETCh MEASure READ:TOIN:THIRD:LOWER:FREQuency	?
Meas - TOI	TOI Third Order Lower IP3	TOI3LI	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel:IP3	?
Meas - TOI	TOI Third Order Lower Level	TOI3LL	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel	?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Meas - TOI	TOI Third Order Upper Diff	TOI3UD	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel:DIFFerence	?
Meas - TOI	TOI Third Order Upper Freq	TOI3UF	:FETCh MEASure READ:TOIN:THIRD:UPPER:FREQuency	?
Meas - TOI	TOI Third Order Upper IP3	TOI3UI	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel:IP3	?
Meas - TOI	TOI Third Order Upper Level	TOI3UL	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel	?
Meas - TOI	TOI Base Lower Diff	TOIBLD	:FETCh MEASure READ:TOIN:BASE:LOWER:LEVel:DIFFerence	?
Meas - TOI	TOI Base Lower Freq	TOIBLF	:FETCh MEASure READ:TOIN:BASE:LOWER:FREQuency	?
Meas - TOI	TOI Base Lower Level	TOIBLL	:FETCh MEASure READ:TOIN:BASE:LOWER:LEVel	?
Meas - TOI	TOI Base Upper Diff	TOIBUD	:FETCh MEASure READ:TOIN:BASE:UPPER:LEVel:DIFFerence	?
Meas - TOI	TOI Base Upper Freq	TOIBUF	:FETCh MEASure READ:TOIN:BASE:UPPER:FREQuency	?
Meas - TOI	TOI Base Upper Level	TOIBUL	:FETCh MEASure READ:TOIN:BASE:UPPER:LEVel	?
Meas - TOI	TOI Measurement Output	TOIOUT	:FETCh MEASure READ:TOIN	?
Meas - TOI	TOI Worst Case Diff	TOIWCD	:FETCh MEASure READ:TOIN:WORST:LEVel:DIFFerence	?
Meas - TOI	TOI Worst Case Freq	TOIWCF	:FETCh MEASure READ:TOIN:WORST:FREQuency	?
Meas - TOI	TOI Worst Case IP3	TOIWCI	:FETCh MEASure READ:TOIN:WORST:LEVel:IP3	?
Meas - TOI	TOI Worst Case Level	TOIWCL	:FETCh MEASure READ:TOIN:WORST:LEVel	?
Meas - TP	TP Measurement Output	TPOUT	:FETCh MEASure READ:TPower	?
Meas - TP	TP Channel Power	TPOUTC	:FETCh MEASure READ:TPower:CHPower	?
Meas - TP	TP Pwr Spectral Density	TPOUTD	:FETCh MEASure READ:TPower:DENSity	?
Trace	Send Trace Data	TRD	:TRACe[:DATA]	TRACE1 TRACE2 TRACE3,<ASCII_data>
Trace	Query Trace Data	TRD	:TRACe[:DATA]	? TRACE1 TRACE2 TRACE3
Trace	Trace Function	TRF[1~3]	:TRACe[1~2][3:MODE	WRITE MAXHold MINHold VIEW BLANK ?
Trigger	Trigger Slope	TSL	:TRIGger[:SEQuence]:SLOPe	POSitive NEGative ?
Trigger	Trigger Source	TSO	:TRIGger[:SEQuence]:SOURce	IMMediate VIDeo LINE EXTernal ?
Trigger	Trigger Video Level	TVL	:TRIGger[:SEQuence]:VIDeo:LEVel	<amplitude> ?
Bandwidth	Video Bandwidth	VB	[:SENSe]:BANDwidth BWIDth:VIDeo	<frequency> ?
Bandwidth	Video Bandwidth Auto	VBA	[:SENSe]:BANDwidth BWIDth:VIDeo:AUTO	OFF ON 0 1 ?
Bandwidth	VBW/RBW	VBRB	[:SENSe]:BANDwidth BWIDth:VIDeo:RATio	<ratio> ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Display	White Mode	WH	:DISPlay:WINDow:WHITe	OFF ON 0 1 ?
Meas - XDB	X dB Left	XDBL[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:LEFT	?
Meas - XDB	X dB Point	XDBP[1~2]	[:SENSe]:XDB[1~2]:AMPLitude	<Amplitude> ?
Meas - XDB	X dB Right	XDBR[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:RIGHT	?
Meas - XDB	X dB Bandwidth	XDBRL[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:BANDwidth BWIDth	?
Meas - XDB	X dB Shape Factor	XDBS	:FETCh MEASure READ:XDB[1~2]:FREQuency:SHAPE	?
Span	Zoom In	ZI	[:SENSe]:FREQuency:SPAN:ZIN	none
Calibration	ZNC Cal	ZNCCAL	:CALibration:RBW	none ?
Span	Zoom Out	ZO	[:SENSe]:FREQuency:SPAN:ZOUT	none
Span	Zero Span	ZS	[:SENSe]:FREQuency:SPAN:ZERO	none

< SA Command Order (CCDF Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Meas - CCDF	Output	CCDFCRE	:FETCh MEASure READ:CCDF:CRESt	?
Meas - CCDF	Count	CCDFN	[:SENSe]:CCDF:COUNt	<integer> ?
Meas - CCDF	Output	CCDFOUT[PWR PER]	:FETCh MEASure READ:CCDF[:POWER PERCent]	?
Meas - CCDF	Output	CCDFPER00001	:FETCh MEASure READ:CCDF:PERCent00001	?
Meas - CCDF	Output	CCDFPER0001	:FETCh MEASure READ:CCDF:PERCent0001	?
Meas - CCDF	Output	CCDFPER001	:FETCh MEASure READ:CCDF:PERCent001	?
Meas - CCDF	Output	CCDFPER01	:FETCh MEASure READ:CCDF:PERCent01	?
Meas - CCDF	Output	CCDFPER1	:FETCh MEASure READ:CCDF:PERCent1	?
Meas - CCDF	Output	CCDFPER10	:FETCh MEASure READ:CCDF:PERCent10	?
Frequency	Center Frequency	CF	[:SENSe]:FREQuency:CENTer	<frequency> ?
Sweep	Continuous	CO	:INITiate:CONTinuous	OFF ON 0 1 ?
Others	Error Code	ERR		?
Others	ESE2	ESE2		<integer> ?
Others	ESR2	ESR2		?
File	Copy	FCOPY	:MMEMory:COPIY	<'file_name1'>,<'file_name2'>
File	Delete	FDEL	:MMEMory:DELete	<'file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<'file_name'>
File	Move	FMOVE	:MMEMory:DATA	<'file_name'>,definite_length ? <'file_name'>
Frequency	Frequency Offset	FO	[:SENSe]:FREQuency:OFFSet	<frequency> ?
File	Read	FREAD	:MMEMory:CATalog	? <'directory_name'>
File	Rename	FRENAME	:MMEMory:MOVE	<'file_name1'>,<'file_name2'>

REMOTE CONTROL

File	Save	FSAVE	:MMEMory:STORe	<'file_name'>
Display	Full Screen	FSCR	:DISPlay:CCDF:FSCReen[:STATe]	OFF ON 0 1 ?
Display	Gaussian Trace State	GAUS	:DISPlay:CCDF:WINDow:GAUSSian[:STATe]	OFF ON 0 1 ?
Display	Graticule	GRAT	:DISPlay:CCDF:WINDow:TRACe:GRATICule:GRID[:STATe]	OFF ON 0 1 ?
Printer	Hard Copy	HCOPY	:HCOPY[:IMMEdiate]	none
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[:RF]:GAIN[:STATe]	OFF ON 0 1 ?
Marker	Marker Amplitude	MA[1~9]	:CALCulate:CCDF:MARKer[1~9]:X	<amplitude> ?
Marker	Marker All Off	MAO	:CALCulate:CCDF:MARKer:AOFF	none
Measurement	Meas. Start	MEA	:MEASure:STARt	XDB ACP CHP OBW HD CCDF ?
Measurement	Meas. OFF	MEAO	:MEASure:AOFF	none
Marker	Marker Mode	MM[1~9]	:CALCulate:CCDF:MARKer[1~9]:MODE	POSition DELTA BAND SPAN OFF ?
Marker	Marker Percent	MP[1~9]	:CALCulate:CCDF:MARKer[1~9]:Y	?
Marker	Marker State	MS[1~9]	:CALCulate:CCDF:MARKer[1~9]:STATe	OFF ON 0 1 ?
Marker	Marker Trace	MT[1~9]	:CALCulate:CCDF:MARKer[1~9]:TRACe	1 2 3 ?
Preset	Preset	PRST	:SYSTem:PRESet	none
Bandwidth	Resolution Bandwidth	RB	[:SENSe]:BANDwidth BWIDth[:RESolution]	<frequency> ?
Display	Ref Trace State	REFS	:DISPlay:CCDF:WINDow:RLEVel[:STATe]	OFF ON 0 1 ?
Span	Scale/Divide	SD	:DISPlay:CCDF:WINDow:TRACe:Y[:SCALe]:PDIVision	<amplitude> ?
Sweep	Single	SI	:INITiate[:IMMEdiate]	none
Display	Store Ref Trace	SREF	:DISPlay:CCDF:WINDow:RLEVel:STORe	none
Trigger	Trigger Source	TSO	:TRIGger[:SEQUence]:SOURce	IMMEdiate VIDeo LINE EXTernal ?
Bandwidth	Video Bandwidth	VB	[:SENSe]:BANDwidth BWIDth:VIDeo	<frequency> ?
Bandwidth	Video Bandwidth Auto	VBA	[:SENSe]:BANDwidth BWIDth:VIDeo:AUTO	OFF ON 0 1 ?
Bandwidth	VBW/RBW	VBRB	[:SENSe]:BANDwidth BWIDth:VIDeo:RATIo	<ratio> ?
Display	White Mode	WH	:DISPlay:CCDF:WINDow:WHITe	OFF ON 0 1 ?

< SA Command Order (Phase Noise Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Display	Annotation	ANN	:DISPlay:LPLot:WINDow:ANNotation[:ALL]	OFF ON 0 1 ?
Measurement	Average State	AVG	[:SENSe]:LPLot:AVERAge[:STATe]	OFF ON 0 1 ?
Measurement	Avg. Count	AVGC	[:SENSe]:LPLot:AVERAge:COUNT	<integer> ?
Frequency	Carrier Freq	CF	[:SENSe]:LPLot:FREQuency:CARRier	<frequency> ?
Frequency	Carrier Search	CFS	[:SENSe]:LPLot:FREQuency:CARRier:SEARCh	none
Sweep	Continuous	CO	:INITiate:LPLot:CONTInuous	OFF ON 0 1 ?
Amplitude	Scale/Div	DS	:DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:PDIVision	<amplitude> ?
Measurement	Decade Table	DTB	:CALCulate:LPLot:DECade:TABLE[:STATe]	OFF ON 0 1 ?
Others	Error Code	ERR		?
Others	ESE2	ESE2		<integer> ?
Others	ESR2	ESR2		?
Span	Start Offset	FA	[:SENSe]:LPLot:FREQuency:OFFSet:START	<frequency> ?
Span	Stop Offset	FB	[:SENSe]:LPLot:FREQuency:OFFSet:STOP	<frequency> ?
File	Copy	FCOPY	:MMEMory:COPIY	<'file_name1'>,<'file_name2'>
File	Delete	FDEL	:MMEMory:DELeTe	<'file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<'file_name'>
File	Move	FMOVE	:MMEMory:DATA	<'file_name'>,<definite_length_block?> <'file_name'>
File	Read	FREAD	:MMEMory:CATalog	? <'directory_name'>
File	Rename	FRENAME	:MMEMory:MOVE	<'file_name1'>,<'file_name2'>
File	Save	FSAVE	:MMEMory:STORe	<'file_name'>
Display	Full Screen	FSCR	:DISPlay:LPLot:FSCReen[:STATe]	OFF ON 0 1 ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Measurement	Filtering	FT	[:SENSe]:LPLot:FiLTeRiNg	<value> ?
Display	Graticule	GRAT	:DISPlay:LPLot:WINDow:TRACe:GRATicule:GRID[:STATe]	OFF ON 0 1 ?
Printer	Hard Copy	HCOPY	:HCOPY[:IMMEDIATE]	none
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[:RF]:GAIN[:STATe]	OFF ON 0 1 ?
Measurement	Log Plot	LP	:MEASure:LPLot	none
Marker	Marker Amplitude	MA[1~9]	:CALCulate:LPLot:MARKer[1~9]:Y	?
Marker	Marker All Off	MAO	:CALCulate:LPLot:MARKer:AOff	none
Marker	Marker Frequency	MF[1~9]	:CALCulate:LPLot:MARKer[1~9]:X	<frequency> ?
Marker	Marker Mode	MM[1~9]	:CALCulate:LPLot:MARKer[1~9]:MODE	POSition DELTA RMSDegree RMSRadian RMSJitter RFM OFF ?
Mode	Mode	MODE	:INSTrument[:SElect]	SA PNOISE ?
Marker	Marker State	MS[1~9]	:CALCulate:LPLot:MARKer[1~9]:STATe	OFF ON 0 1 ?
Marker	Marker Trace	MT[1~9]	:CALCulate:LPLot:MARKer[1~9]:TRACe	1 2 ?
Marker	Marker Table	MTB	:CALCulate:LPLot:MARKer:TABLE[:STATe]	OFF ON 0 1 ?
Preset	Preset	PRST	:SYSTem:PRESet	none
Amplitude	Ref. Level	RL	:DISPlay:LPLot:WINDow:TRACe:Y[:SCALe]:RLEVel	<amplitude> ?
Sweep	Single	SI	:INITiate:LPLot[:IMMEDIATE]	none
Measurement	Smoothing	SM	[:SENSe]:LPLot:SMOoth	<percentage> ?
Trigger	Trigger Source	TSO	:TRIGger[:SEQUence]:SOURce	IMMEDIATE VIDeo LINE EXTernal ?
Display	White Mode	WH	:DISPlay:LPLot:WINDow:WHITe	OFF ON 0 1 ?

< SCPI Command Order (Spectrum Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Limit Line	Pass/Fail Alarm	ALARM	:CALCulate:LLINe:ALARM	OFF ON 0 1 ?
Limit Line	Clear Limit Line	LLAO	:CALCulate:LLINe:AOFF	none
Limit Line	Limit Line Check State	LLCS[1~2]	:CALCulate:LLINe[1~2]:CHECK:STATe	OFF ON 0 1 ?
Limit Line	Limit Line Fail Count	LLFC[1~2]	:CALCulate:LLINe[1~2]:FAIL:COUNT	?
Marker	Marker All Off	MAO	:CALCulate:MARKer:AOFF	none
Marker Fctn	Marker Counter Res	MFNR	:CALCulate:MARKer:FCOunt:RESolution	<frequency> ?
Marker Fctn	Marker Counter Output	MFNX	:CALCulate:MARKer:FCOunt:X	?
Marker Fctn	Marker Noise Output	MFNY	:CALCulate:MARKer:FUNCTion:Y	?
Peak Search	Peak Excursion	MPKE	:CALCulate:MARKer:PEAK:EXCURsion	<amplitude> ?
Peak Search	Multi Peak	MMPK	:CALCulate:MARKer:PEAK:MULTi	none
Peak Search	Multi Peak Number	MMPKN	:CALCulate:MARKer:PEAK:MULTi:NUMber	<integer> ?
Peak Search	Multi Peak Trace	MMPKT	:CALCulate:MARKer:PEAK:MULTi:TRACe	<integer> ?
Peak Search	Peak Parameter	MPKPA	:CALCulate:MARKer:PEAK:SEARCh:MODE	MAX PARAmeter ?
Peak Search	Peak Threshold	MPKTH	:CALCulate:MARKer:PEAK:THReshold	<amplitude> ?
Marker	Marker Table	MTB	:CALCulate:MARKer:TABLe:STATe	OFF ON 0 1 ?
Peak Search	Continuous Peak	MPKC[1~9]	:CALCulate:MARKer[1~9]:CPEak[:STATe]	OFF ON 0 1 ?
Marker Fctn	Marker Function	MFN[1~9]	:CALCulate:MARKer[1~9]:FUNCTion	NOISe COUNT OFF ?
Peak Search	Peak Search	MPK[1~9]	:CALCulate:MARKer[1~9]:MAXimum	none
Peak Search	Next Left Peak Search	MPKL[1~9]	:CALCulate:MARKer[1~9]:MAXimum:LEFT	none

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Peak Search	Next Peak Search	MPKN[1~9]	:CALCulate:MARKer[1~9]:MAXimum:NEXT	none
Peak Search	Next Right Peak Search	MPKR[1~9]	:CALCulate:MARKer[1~9]:MAXimum:RIGHT	none
Peak Search	Minimum Search	MPKM[1~9]	:CALCulate:MARKer[1~9]:MINimum	none
Marker	Marker Mode	MM[1~9]	:CALCulate:MARKer[1~9]:MODE	POSITION DELTA BAND SPAN OFF ?
Peak Search	Peak to Peak Search	MPKP[1~9]	:CALCulate:MARKer[1~9]:PTPeak	none
Marker	Marker Readout	MR[1~9]	:CALCulate:MARKer[1~9]:READout	FREQUENCY TIME ITIME PERiod ?
Marker	Marker State	MS[1~9]	:CALCulate:MARKer[1~9]:STATE	OFF ON 0 1 ?
Marker	Marker Trace	MT[1~9]	:CALCulate:MARKer[1~9]:TRACE	1 2 3 ?
Frequency	Signal Track	STR[1~9]	:CALCulate:MARKer[1~9]:TRCKing[:STATE]	OFF ON 0 1 ?
Peak Search	Signal Track	MPKT[1~9]	:CALCulate:MARKer[1~9]:TRCKing[:STATE]	OFF ON 0 1 ?
Marker	Marker Freq	MF[1~9]	:CALCulate:MARKer[1~9]:X	<frequency> ?
Marker	Marker Center Freq	MFC[1~9]	:CALCulate:MARKer[1~9]:X:CENTer	<frequency> ?
Marker	Marker Point	MP[1~9]	:CALCulate:MARKer[1~9]:X:POSITION	<integer> ?
Marker	Marker Center Point	MPC[1~9]	:CALCulate:MARKer[1~9]:X:POSITION:CENTer	<integer> ?
Marker	Marker Span Point	MPSP[1~9]	:CALCulate:MARKer[1~9]:X:POSITION:SPAN	<integer> ?
Marker	Marker Start Point	MPS[1~9]	:CALCulate:MARKer[1~9]:X:POSITION:START	<integer> ?
Marker	Marker Stop Point	MPE[1~9]	:CALCulate:MARKer[1~9]:X:POSITION:STOP	<integer> ?
Marker	Marker Span Freq	MFSP[1~9]	:CALCulate:MARKer[1~9]:X:SPAN	<frequency> ?
Marker	Marker Start Freq	MFS[1~9]	:CALCulate:MARKer[1~9]:X:START	<frequency> ?
Marker	Marker Stop Freq	MFE[1~9]	:CALCulate:MARKer[1~9]:X:STOP	<frequency> ?
Marker	Marker Amplitude	MA[1~9]	:CALCulate:MARKer[1~9]:Y	?
Marker Shift	Marker to Center Freq	MCF[1~9]	:CALCulate:MARKer[1~9][:SET]:CENTer	none
Marker Shift	Marker Delta to CF	MDCF[1~9]	:CALCulate:MARKer[1~9][:SET]:DELTA:CENTer	none
Marker Shift	Marker Delta to Span	MDSP[1~9]	:CALCulate:MARKer[1~9][:SET]:DELTA:SPAN	none
Marker Shift	Marker Delta to CF Step	MDSS[1~9]	:CALCulate:MARKer[1~9][:SET]:DELTA:STEP	none
Marker Shift	Marker to Ref Level	MRL[1~9]	:CALCulate:MARKer[1~9][:SET]:RLEVEL	none
Marker Shift	Marker to Start Freq	MFA[1~9]	:CALCulate:MARKer[1~9][:SET]:START	none

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Marker Shift	Marker to CF Step	MSS[1~9]	:CALCulate:MARKer[1~9][:SET]:STEP	none
Marker Shift	Marker to Stop Freq	MFB[1~9]	:CALCulate:MARKer[1~9][:SET]:STOP	none
Calibration	Level Cal	LVLCAL	:CALibration:LEVel	none ?
Calibration	Periodic Temperature Cal	PCAL	:CALibration:PTEMPerature	OFF ON 0 1 ?
Calibration	ZNC Cal	ZNCCAL	:CALibration:RBW	none ?
Calibration	Pre-Filter Cal	PFCAL	:CALibration:YIG	none ?
Calibration	Align All Now	CALALL	:CALibration[:ALL]	none ?
Couple	Auto Couple	ACPL	:COUPle	none
Display	Title	TITLE	:DISPlay:ANNotation:TITLe:DATA	<string> ?
Display	Full Screen	FSCR	:DISPlay:FSCReen[:STATe]	OFF ON 0 1 ?
Display	Annotation	ANN	:DISPlay:WINDow:ANNotation[:ALL]	OFF ON 0 1 ?
Display	Graticule	GRAT	:DISPlay:WINDow:TRACe:GRATICule:GRID[:STATe]	TYPE1 TYPE OFF ?
Display	Display Line Ampl	DL	:DISPlay:WINDow:TRACe:Y:DLINe	<amplitude> ?
Display	Display Line State	DLS	:DISPlay:WINDow:TRACe:Y:DLINe:STATe	OFF ON 0 1 ?
Display	Threshold Line Ampl	TH	:DISPlay:WINDow:TRACe:Y:TLINe	<amplitude> ?
Display	Threshold Line State	THS	:DISPlay:WINDow:TRACe:Y:TLINe:STATe	OFF ON 0 1 ?
Amplitude	Scale/Divide	SD	:DISPlay:WINDow:TRACe:Y[:SCALE]:PDIVision	<amplitude> ?
Amplitude	Reference Level	RL	:DISPlay:WINDow:TRACe:Y[:SCALE]:RLEVel	<amplitude> ?
Amplitude	Ref Level Offset	RLO	:DISPlay:WINDow:TRACe:Y[:SCALE]:RLEVel:OFFSet	<amplitude> ?
Amplitude	Scale Type	STY	:DISPlay:WINDow:TRACe:Y[:SCALE]:SPACing	LINear LOGarithmic ?
Display	White Mode	WH	:DISPlay:WINDow:WHITe	OFF ON 0 1 ?
Meas - ACP	ACP Measurement Output	ACPOUT	:FETCh MEASure READ:ACPowEr	?
Meas - ACP	ACP Measurement Output	ACPOUTC	:FETCh MEASure READ:ACPowEr:CHPowEr	?
Meas - ACP	ACP Measurement Output	ACPOUTL[1~6]	:FETCh MEASure READ:ACPowEr:LACPowEr[1~6]	?
Meas - ACP	ACP Measurement Output	ACPOUTR[1~6]	:FETCh MEASure READ:ACPowEr:RACPowEr[1~6]	?
Meas - BP	BP Measurement Output	BPOUT	:FETCh MEASure READ:BPowEr	?
Meas - BP	BP Average Power	BPAVERP	:FETCh MEASure READ:BPowEr:POWEr:AVERAge	?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Meas - BP	BP Max Power	BPMAXP	:FETCh MEASure READ:BPOWer:POWer:MAX	?
Meas - BP	BP Min Power	BPMINP	:FETCh MEASure READ:BPOWer:POWer:MIN	?
Meas - BP	BP Burst Length	BPBL	:FETCh MEASure READ:BPOWer:WIDTh	?
Meas - CHP	CHP Measurement Output	CHPOUT	:FETCh MEASure READ:CHPower	?
Meas - CHP	CHP Channel Power	CHPOUTC	:FETCh MEASure READ:CHPower:CHPower	?
Meas - CHP	CHP Pwr Spectral Density	CHPOUTD	:FETCh MEASure READ:CHPower:DENSity	?
Meas - HD	HD Amplitude	HA[1~5]	:FETCh MEASure READ:HARMonics:AMPLitude[1~5]	?
Meas - HD	HD Frequency	HF[1~5]	:FETCh MEASure READ:HARMonics:FREQuency[1~5]	?
Meas - HD	Total Harm Distortion	HDOUT	:FETCh MEASure READ:HARMonics[:DISTortion]	?
Meas - OBW	OBW Measurement Output	OBWOUT	:FETCh MEASure READ:OBWwidth	?
Meas - SEM	SEM Measurement Output	SEMOUT	:FETCh MEASure READ:SEMask	?
Meas - SEM	SEM Lower CHP	SEMLPWR1~6	:FETCh MEASure READ:SEMask:LOWER1~6:CHPower	?
Meas - SEM	SEM Lower PSD	SEMLPSD1~6	:FETCh MEASure READ:SEMask:LOWER1~6:DENSity	?
Meas - SEM	SEM Lower Peak Freq	SEMLF1~6	:FETCh MEASure READ:SEMask:LOWER1~6:FREQuency	?
Meas - SEM	SEM Ref. Channel CHP	SEMPWR	:FETCh MEASure READ:SEMask:REFerence:CHPower	?
Meas - SEM	SEM Ref. Channel PSD	SEMPD	:FETCh MEASure READ:SEMask:REFerence:DENSity	?
Meas - SEM	SEM Upper CHP	SEMUPWR1~6	:FETCh MEASure READ:SEMask:UPPER1~6:CHPower	?
Meas - SEM	SEM Upper PSD	SEMUPSD1~6	:FETCh MEASure READ:SEMask:UPPER1~6:DENSity	?
Meas - SEM	SEM Upper Peak Freq	SEMUF1~6	:FETCh MEASure READ:SEMask:UPPER1~6:FREQuency	?
Meas - SE	SE Measurement Output	SEOUT	:FETCh MEASure READ:SPURious	?
Meas - TOI	TOI Measurement Output	TOIOUT	:FETCh MEASure READ:TOIN	?
Meas - TOI	TOI Base Lower Freq	TOIBLF	:FETCh MEASure READ:TOIN:BASE:LOWER:FREQuency	?
Meas - TOI	TOI Base Lower Level	TOIBLL	:FETCh MEASure READ:TOIN:BASE:LOWER:LEVel	?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Meas - TOI	TOI Base Lower Diff	TOIBLD	:FETCh MEASure READ:TOIN:BASE:LOWER:LEVel:DIFFerence	?
Meas - TOI	TOI Base Upper Freq	TOIBUF	:FETCh MEASure READ:TOIN:BASE:UPPER:FREQuency	?
Meas - TOI	TOI Base Upper Level	TOIBUL	:FETCh MEASure READ:TOIN:BASE:UPPER:LEVel	?
Meas - TOI	TOI Base Upper Diff	TOIBUD	:FETCh MEASure READ:TOIN:BASE:UPPER:LEVel:DIFFerence	?
Meas - TOI	TOI Third Order Lower Freq	TOI3LF	:FETCh MEASure READ:TOIN:THIRD:LOWER:FREQuency	?
Meas - TOI	TOI Third Order Lower Level	TOI3LL	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel	?
Meas - TOI	TOI Third Order Lower Diff	TOI3LD	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel:DIFFerence	?
Meas - TOI	TOI Third Order Lower IP3	TOI3LI	:FETCh MEASure READ:TOIN:THIRD:LOWER:LEVel:IP3	?
Meas - TOI	TOI Third Order Upper Freq	TOI3UF	:FETCh MEASure READ:TOIN:THIRD:UPPER:FREQuency	?
Meas - TOI	TOI Third Order Upper Level	TOI3UL	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel	?
Meas - TOI	TOI Third Order Upper Diff	TOI3UD	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel:DIFFerence	?
Meas - TOI	TOI Third Order Upper IP3	TOI3UI	:FETCh MEASure READ:TOIN:THIRD:UPPER:LEVel:IP3	?
Meas - TOI	TOI Worst Case Freq	TOIWCF	:FETCh MEASure READ:TOIN:WORST:FREQuency	?
Meas - TOI	TOI Worst Case Level	TOIWCL	:FETCh MEASure READ:TOIN:WORST:LEVel	?
Meas - TOI	TOI Worst Case Diff	TOIWCD	:FETCh MEASure READ:TOIN:WORST:LEVel:DIFFerence	?
Meas - TOI	TOI Worst Case IP3	TOIWCI	:FETCh MEASure READ:TOIN:WORST:LEVel:IP3	?
Meas - TP	TP Measurement Output	TPOUT	:FETCh MEASure READ:TPower	?
Meas - TP	TP Channel Power	TPOUTC	:FETCh MEASure READ:TPower:CHPower	?
Meas - TP	TP Pwr Spectral Density	TPOUTD	:FETCh MEASure READ:TPower:DENSity	?
Meas - XDB	X dB Bandwidth	XDBRL[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:BANDwidth BWIDth	?
Meas - XDB	X dB Left	XDBL[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:LEFT	?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Meas - XDB	X dB Right	XDBR[1~2]	:FETCh MEASure READ:XDB[1~2]:FREQuency:RIGHt	?
Meas - XDB	X dB Shape Factor	XDBS	:FETCh MEASure READ:XDB[1~2]:FREQuency:SHAPe	?
Printer	Hard Copy	HCOPY	:HCOPY[:IMMediate]	none
Sweep	Sweep Time Accuracy	STAC	:INITiate:ACCuracy	NORMal ACCuracy ?
Sweep	Continuous	CO	:INITiate:CONTinuous	OFF ON 0 1 ?
Sweep	Single	SI	:INITiate[:IMMediate]	none
In/Out	RF Coupling	RFC	:INPut:COUPling	AC DC ?
Measurement	Meas. OFF	MEAO	:MEASure:AOFF	none
Measurement	Meas. Average Count	MEAAVGN	:MEASure:AVERage:COUNT	<integer> ?
Measurement	Meas. Average Reset	MEAAVGR	:MEASure:AVERage:RESET	none
Measurement	Meas. Average Mode	MEAAVGM	:MEASure:AVERage:TCONrol	EXPonential REPeat ?
Measurement	Meas. Average State	MEAAVG	:MEASure:AVERage[:STATe]	OFF ON 0 1 ?
Measurement	Meas. Start	MEA	:MEASure:STARt	XDB ACP CHP OBW HD CCDF TOI SE SEM BP TP ?
File	Read	FREAD	:MMEMory:CATalog	? <directory_name>
File	Copy	FCOPY	:MMEMory:COPIY	<'file_name1'>,<'file_name2'>
File	Move	FMOVE	:MMEMory:DATA	<'file_name'>,definite_length ? <'file_name'>
File	Delete	FDEL	:MMEMory:DELeTe	<'file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<'file_name'>
File	Rename	FRENAME	:MMEMory:MOVE	<'file_name1'>,<'file_name2'>
File	Save	FSAVE	:MMEMory:STORe	<'file_name'>
Preset	Preset	PRST	:SYSTem:PRESet	none
Trace	Trace Data Format	TDF	:TRACe:FORMat	ASCIi REAL,64 INT,32 ?
Trace	Send Trace Data	TRD	:TRACe[:DATA]	TRACE1 TRACE2 TRACE3,<ASCII_data>
Trace	Query Trace Data	TRD	:TRACe[:DATA]	? TRACE1 TRACE2 TRACE3
Trace	Trace Function	TRF[1~3]	:TRACe[1~2] 3:MODE	WRITe MAXHold MINHold VIEW BLANK ?
Trigger	Trigger Delay	TD	:TRIGger[:SEQuence]:DELay	<time> ?
Trigger	Trigger Delay State	TDS	:TRIGger[:SEQuence]:DELay:STATe	OFF ON 0 1 ?
Trigger	Trigger Slope	TSL	:TRIGger[:SEQuence]:SLOPe	POSitive NEGative ?
Trigger	Trigger Source	TSO	:TRIGger[:SEQuence]:SOURce	IMMediate VIdeo LINE EXTernal ?
Trigger	Trigger Video Level	TVL	:TRIGger[:SEQuence]:VIdeo:LEVel	<amplitude> ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Amplitude	Amplitude Units	AU	:UNIT:POWer	DBM DBMV DBMA V W A DBUV DBUA ?
Meas - ACP	ACP Adjacent Channel BW	ACPAC	[:SENSe]:ACPower:ADJacent:BA NDwidth BWI Dth:INTegration	<frequency> ?
Meas - ACP	ACP Main Channel BW	ACPMC	[:SENSe]:ACPower:BA NDwidth BWI Dth:INTegration	<frequency> ?
Meas - ACP	ACP Channel Space BW	ACPN	[:SENSe]:ACPower:COUNt	<integer> ?
Meas - ACP	ACP Channel Space BW	ACPCS	[:SENSe]:ACPower:SPACe:BA NDwidth BWI Dth:INTegration	<frequency> ?
Average	Average CLear	AVGCL	[:SENSe]:AVERAge:CLEar	none
Average	Average Count	AVGC	[:SENSe]:AVERAge:COUNt	<integer> ?
Average	Avg/VBW Type	AVGT	[:SENSe]:AVERAge:TYPE	RMS LOG SCALar ?
Average	Avg/VBW Type Auto	AVGTA	[:SENSe]:AVERAge:TYPE:AUTO	OFF ON 0 1 ?
Average	Average	AVG	[:SENSe]:AVERAge[:STATe]	OFF ON 0 1 ?
Bandwidth	Auto Bandwidth	AB	[:SENSe]:BA NDwidth BWI Dth:AUTO	none
Bandwidth	Video Bandwidth	VB	[:SENSe]:BA NDwidth BWI Dth:VIDeo	<frequency> ?
Bandwidth	Video Bandwidth Auto	VBA	[:SENSe]:BA NDwidth BWI Dth:VIDeo:AUTO	OFF ON 0 1 ?
Bandwidth	VBW/RBW	VBRB	[:SENSe]:BA NDwidth BWI Dth:VIDeo:RATio	<ratio> ?
Bandwidth	Resolution Bandwidth	RB	[:SENSe]:BA NDwidth BWI Dth[:RESolution]	<frequency> ?
Bandwidth	Resolution Bandwidth Auto	RBA	[:SENSe]:BA NDwidth BWI Dth[:RESolution]:AUTO	OFF ON 0 1 ?
Meas - BP	BP Meas Type	BPMT	[:SENSe]:BPOWer:METHod	THReshold BWI Dth ?
Meas - BP	BP Threshold	BPTH	[:SENSe]:BPOWer:THReshold	<amplitude> ?
Meas - CHP	CHP Channel BW	CHPC	[:SENSe]:CHPower:BA NDwidth BWI Dth:INTegration	<frequency> ?
Amplitude	Delete All Corrections	COAD	[:SENSe]:CORRection:CSET:ALL:DELeTe	none
Amplitude	Apply Corrections	COAS	[:SENSe]:CORRection:CSET:ALL[:STATe]	OFF ON 0 1 ?
Amplitude	Correction State	COS1 2 3 4	[:SENSe]:CORRection:CSET1 2 3 4[:STATe]	OFF ON 0 1 ?
Couple	Detector Auto	DETA	[:SENSe]:DETEctor:AUTO	OFF ON 0 1 ?
Couple	Detector	DET	[:SENSe]:DETEctor[:FUNCTion]	NORMal AVERAge POSitive SAMPle NEGative ?
Frequency	Center Frequency	CF	[:SENSe]:FREQuency:CENTer	<frequency> ?
Frequency	CF Step Auto	SSA	[:SENSe]:FREQuency:CENTer:STEP:AUTO	OFF ON 0 1 ?
Frequency	CF Step	SS	[:SENSe]:FREQuency:CENTer:STEP[:INCRement]	<frequency> ?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Frequency	Frequency Offset	FO	[:SENSe]:FREQuency:OFFSet	<frequency> ?
Span	Span	SP	[:SENSe]:FREQuency:SPAN	<frequency> ?
Bandwidth	Span/RBW	SPRB	[:SENSe]:FREQuency:SPAN:BANDwidth BWIDth:VIDeo:RATio	<ratio> ?
Bandwidth	Span/RBW Auto	SPRBA	[:SENSe]:FREQuency:SPAN:BANDwidth BWIDth:VIDeo:RATio:AUTO	OFF ON 0 1 ?
Span	Full Span	FS	[:SENSe]:FREQuency:SPAN:FULL	none
Span	Last Span	LS	[:SENSe]:FREQuency:SPAN:PREVious	none
Span	Zero Span	ZS	[:SENSe]:FREQuency:SPAN:ZERO	none
Span	Zoom In	ZI	[:SENSe]:FREQuency:SPAN:ZIN	none
Span	Zoom Out	ZO	[:SENSe]:FREQuency:SPAN:ZOUT	none
Frequency	Start Frequency	FA	[:SENSe]:FREQuency:STARt	<frequency> ?
Frequency	Stop Frequency	FB	[:SENSe]:FREQuency:STOP	<frequency> ?
Meas - HD	HD Number	HDN	[:SENSe]:HARMonics:NUMBer	<integer> ?
Meas - OBW	OBW Percentage	OBWP	[:SENSe]:OBWidth:PERCent	<percent> ?
Amplitude	Attenuation	AT	[:SENSe]:POWer[:RF]:ATTenuation	<amplitude> ?
Amplitude	Attenuation Auto	ATA	[:SENSe]:POWer[:RF]:ATTenuation:AUTO	OFF ON 0 1 ?
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[:RF]:GAIN[:STATe]	OFF ON 0 1 ?
Meas - SEM	SEM Meas Type	SEMMT	[:SENSe]:SEMAsk:METHod	TPRef PSDRef ?
Sweep	Point	PO	[:SENSe]:SWEep:POINts	<integer> ?
Sweep	Sweep Time	ST	[:SENSe]:SWEep:TIME	<time> ?
Sweep	Sweep Time Auto	STA	[:SENSe]:SWEep:TIME:AUTO	OFF ON 0 1 ?
Tune	Auto Tune	ATN	[:SENSe]:TUNE:AUTO	none
Meas - XDB	X dB Point	XDBP[1~2]	[:SENSe]:XDB[1~2]:AMPLitude	<Amplitude> ?

REMOTE CONTROL

< SCPI Command Order (CCDF Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Marker	Marker All Off	MAO	:CALCulate:CCDF:MARKer:AOff	none
Marker	Marker Mode	MM[1~9]	:CALCulate:CCDF:MARKer[1~9]:MODE	POSition DELTA BAND SPAN OFF ?
Marker	Marker State	MS[1~9]	:CALCulate:CCDF:MARKer[1~9]:STATe	OFF ON 0 1 ?
Marker	Marker Trace	MT[1~9]	:CALCulate:CCDF:MARKer[1~9]:TRACe	1 2 3 ?
Marker	Marker Amplitude	MA[1~9]	:CALCulate:CCDF:MARKer[1~9]:X	<amplitude> ?
Marker	Marker Percent	MP[1~9]	:CALCulate:CCDF:MARKer[1~9]:Y	?
Display	Full Screen	FSCR	:DISPlay:CCDF:FSCReem[:STATe]	OFF ON 0 1 ?
Display	Gaussian Trace State	GAUS	:DISPlay:CCDF:WINDow:GAUSSian[:STATe]	OFF ON 0 1 ?
Display	Store Ref Trace	SREF	:DISPlay:CCDF:WINDow:RLEVel:STORe	none
Display	Ref Trace State	REFS	:DISPlay:CCDF:WINDow:RLEVel[:STATe]	OFF ON 0 1 ?
Display	Graticule	GRAT	:DISPlay:CCDF:WINDow:TRACe:GRATICule:GRID[:STATe]	OFF ON 0 1 ?
Span	Scale/Divide	SD	:DISPlay:CCDF:WINDow:TRACe:Y[:SCALe]:PDIVision	<amplitude> ?
Display	White Mode	WH	:DISPlay:CCDF:WINDow:WHITe	OFF ON 0 1 ?
Meas - CCDF	Output	CCDFCRE	:FETCh MEASure READ:CCDF:CRESt	?
Meas - CCDF	Output	CCDFPER0001	:FETCh MEASure READ:CCDF:PERCent0001	?
Meas - CCDF	Output	CCDFPER0001	:FETCh MEASure READ:CCDF:PERCent0001	?
Meas - CCDF	Output	CCDFPER001	:FETCh MEASure READ:CCDF:PERCent001	?
Meas - CCDF	Output	CCDFPER01	:FETCh MEASure READ:CCDF:PERCent01	?
Meas - CCDF	Output	CCDFPER1	:FETCh MEASure READ:CCDF:PERCent1	?
Meas - CCDF	Output	CCDFPER10	:FETCh MEASure READ:CCDF:PERCent10	?
Meas - CCDF	Output	CCDFOUT[PWR PER]	:FETCh MEASure READ:CCDF[:POWER PERCent]	?

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
Printer	Hard Copy	HCOPY	:HCOPY[:IMMediate]	none
Sweep	Continuous	CO	:INITiate:CONTinuous	OFF ON 0 1 ?
Sweep	Single	SI	:INITiate[:IMMediate]	none
Measurement	Meas. OFF	MEAO	:MEASure:AOFF	none
Measurement	Meas. Start	MEA	:MEASure:STARt	XDB ACP CHP OBW HD CCDF ?
File	Read	FREAD	:MMEMory:CATalog	? <'directory_name'>
File	Copy	FCOPY	:MMEMory:COpy	<'file_name1'>,<'file_name2'>
File	Move	FMOVE	:MMEMory:DATA	<'file_name'>,definite_length ? <'file_name'>
File	Delete	FDEL	:MMEMory:DELeTe	<'file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<'file_name'>
File	Rename	FRENAME	:MMEMory:MOVE	<'file_name1'>,<'file_name2'>
File	Save	FSAVE	:MMEMory:STORe	<'file_name'>
Preset	Preset	PRST	:SYSTem:PRESet	none
Trigger	Trigger Source	TSO	:TRIGger[:SEQuence]:SOURce	IMMediate VIDeo LINE EXTernal ?
Bandwidth	Video Bandwidth	VB	[:SENSe]:BANDwidth BWIDth:VIDeo	<frequency> ?
Bandwidth	Video Bandwidth Auto	VBA	[:SENSe]:BANDwidth BWIDth:VIDeo:AUTO	OFF ON 0 1 ?
Bandwidth	VBW/RBW	VBRB	[:SENSe]:BANDwidth BWIDth:VIDeo:RATio	<ratio> ?
Bandwidth	Resolution Bandwidth	RB	[:SENSe]:BANDwidth BWIDth[:RESolution]	<frequency> ?
Meas - CCDF	Count	CCDFN	[:SENSe]:CCDF:COUNt	<integer> ?
Frequency	Center Frequency	CF	[:SENSe]:FREQuency:CENTer	<frequency> ?
Frequency	Frequency Offset	FO	[:SENSe]:FREQuency:OFFSet	<frequency> ?
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[:RF]:GAIN[:STATe]	OFF ON 0 1 ?

< SCPI Command Order (Phase Noise Mode) >

Index	Description	SA Command	SCPI Command	Suffix
Common	*CLS	*CLS	*CLS	none
Common	*ESE	*ESE	*ESE	<integer> ?
Common	*ESR	*ESR	*ESR	?
Common	*IDN	*IDN	*IDN	?
Common	*OPC	*OPC	*OPC	?
Common	*RST	*RST	*RST	none
Common	*SRE	*SRE	*SRE	<integer> ?
Common	*STB	*STB	*STB	?
Measurement	Decade Table	DTB	:CALCulate:LPLot:DECade:TABLE[:STATe]	OFF ON 0 1 ?
Marker	Marker All Off	MAO	:CALCulate:LPLot:MARKer:AOFF	none
Marker	Marker Table	MTB	:CALCulate:LPLot:MARKer:TABLE[:STATe]	OFF ON 0 1 ?
Marker	Marker Mode	MM[1~9]	:CALCulate:LPLot:MARKer[1~9]:MODE	POSITION DELTA RMSDegree RMSRadian RMSJitter RFM OFF ?
Marker	Marker State	MS[1~9]	:CALCulate:LPLot:MARKer[1~9]:STATe	OFF ON 0 1 ?
Marker	Marker Trace	MT[1~9]	:CALCulate:LPLot:MARKer[1~9]:TRACe	1 2 ?
Marker	Marker Frequency	MF[1~9]	:CALCulate:LPLot:MARKer[1~9]:X	<frequency> ?
Marker	Marker Amplitude	MA[1~9]	:CALCulate:LPLot:MARKer[1~9]:Y	?
Display	Full Screen	FSCR	:DISPlay:LPLot:FSCReem[:STATe]	OFF ON 0 1 ?
Display	Annotation	ANN	:DISPlay:LPLot:WINDow:ANNotation[:ALL]	OFF ON 0 1 ?
Display	Graticule	GRAT	:DISPlay:LPLot:WINDow:TRACe:GRATICule:GRID[:STATe]	OFF ON 0 1 ?
Amplitude	Scale/Div	DS	:DISPlay:LPLot:WINDow:TRACe:Y[:SCALE]:PDIVision	<amplitude> ?
Amplitude	Ref. Level	RL	:DISPlay:LPLot:WINDow:TRACe:Y[:SCALE]:RLEVel	<amplitude> ?
Display	White Mode	WH	:DISPlay:LPLot:WINDow:WHITe	OFF ON 0 1 ?
Printer	Hard Copy	HCOPY	:HCOPY[:IMMEDIATE]	none
Sweep	Continuous	CO	:INITiate:LPLot:CONTInuous	OFF ON 0 1 ?
Sweep	Single	SI	:INITiate:LPLot[:IMMEDIATE]	none
Mode	Mode	MODE	:INSTrument[:SElect]	SA PNOISE ?
Measurement	Log Plot	LP	:MEASure:LPLot	none
File	Read	FREAD	:MMEMory:CATalog	? <'directory_name'>
File	Copy	FCOPY	:MMEMory:COPY	<'file_name1'>,<'file_name2'>

REMOTE CONTROL

Index	Description	SA Command	SCPI Command	Suffix
File	Move	FMOVE	:MMEMory:DATA	<'file_name'>,definite_length_block ?<'file_name'>
File	Delete	FDEL	:MMEMory:DELeTe	<'file_name'>
File	Load	FLOAD	:MMEMory:LOAD	<'file_name'>
File	Rename	FRENAME	:MMEMory:MOVE	<'file_name1'>,<'file_name2'>
File	Save	FSAVE	:MMEMory:STORe	<'file_name'>
Preset	Preset	PRST	:SYSTem:PRESet	none
Trigger	Trigger Source	TSO	:TRIGger[:SEQuence]:SOURce	IMMediate VIDeo LINE EXTernal ?
Measurement	Avg. Count	AVGC	[:SENSe]:LPLot:AVERAge:COUNT	<integer> ?
Measurement	Average State	AVG	[:SENSe]:LPLot:AVERAge[:STATe]	OFF ON 0 1 ?
Measurement	Filtering	FT	[:SENSe]:LPLot:FILTering	<value> ?
Frequency	Carrier Freq	CF	[:SENSe]:LPLot:FREQuency:CARRier	<frequency> ?
Frequency	Carrier Search	CFS	[:SENSe]:LPLot:FREQuency:CARRier:SEARCh	none
Span	Start Offset	FA	[:SENSe]:LPLot:FREQuency:OFFSet:STARt	<frequency> ?
Span	Stop Offset	FB	[:SENSe]:LPLot:FREQuency:OFFSet:STOP	<frequency> ?
Measurement	Smoothing	SM	[:SENSe]:LPLot:SMOoth	<percentage> ?
Amplitude	Internal Amplifier	IA	[:SENSe]:POWer[:RF]:GAIN[:STATe]	OFF ON 0 1 ?

Appendix B ERROR CODES

CODE	DESCRIPTION
990	Not supported in current mode
991	Not installed (option)
992	System is busy
993	Execution error (EXE)
994	Query error (QYE)
995	Suffix error
996	Input Data Size Over Error
997	Undefined command
998	Unnecessary suffix insertion
999	Undefined suffix

AEROFLEX INTERNATIONAL LTD. SOFTWARE LICENCE AND WARRANTY

This document is an Agreement between the user of this Licensed Software, the Licensee, and Aeroflex International Limited, the Licensor. By opening this Software package or commencing to use the software you accept the terms of this Agreement. If you do not agree to the terms of this Agreement please return the Software package unopened to Aeroflex International Limited or do not use the software.

1. DEFINITIONS

The following expressions will have the meanings set out below for the purposes of this Agreement:

Add-In Application Software	Licensed Software that may be loaded separately from time to time into the Equipment to improve or modify its functionality
Computer Application Software	Licensed Software supplied to run on a standard PC or workstation
Designated Equipment	the single piece of Equipment upon which the licensed software is installed
Downloaded Software	any software downloaded from an Aeroflex web site
Embedded Software	Licensed Software that forms part of the Equipment supplied by Aeroflex and without which the Equipment cannot function
Licence Fee	the consideration ruling at the date of this Agreement for the use of one copy of the Licensed Software on the Designated Equipment
Licensed Software	All and any programs, listings, flow charts and instructions in whole or in part including Add-in, Computer Application, Downloaded and Embedded Software supplied to work with Designated Equipment

2. LICENCE FEE

The Licensee shall pay the Licence Fee to Aeroflex in accordance with the terms of the contract between the Licensee and Aeroflex.

3. TERM

This Agreement shall be effective from the date hereof and shall continue in force until terminated under the provisions of Clause 9.

4. LICENCE

- 4.1 Unless and until terminated, this Licence confers upon the Licensee the non-transferable and non-exclusive right to use the Licensed Software on the Designated Equipment.
- 4.2 The Licensee may not use the Licensed Software on other than the Designated Equipment, unless written permission is first obtained from Aeroflex and until the appropriate additional Licence Fee has been paid to Aeroflex.
- 4.3 The Licensee may not amend or alter the Licensed Software and shall have no right or licence other than that stipulated herein.
- 4.4 The Licensee may make not more than two copies of the Licensed Software (but not the Authoring and Language Manuals) in machine-readable form for operational security and shall ensure that all such copies include Aeroflex's copyright notice, together with any features which disclose the name of the Licensed Software and the Licensee. Furthermore, the Licensee shall not permit the Licensed Software or any part to be disclosed in any form to any third party and shall maintain the Licensed Software in secure premises to prevent any unauthorised disclosure. The Licensee shall notify Aeroflex immediately if the Licensee has knowledge that any unlicensed party possesses the Licensed Software. The Licensee's obligation to maintain confidentiality shall cease when the Licensed Software and all copies have been destroyed or returned. The copyright in the Licensed Software shall remain with Aeroflex. The Licensee will permit Aeroflex at all reasonable times to audit the use of the Licensed Software.
- 4.5 The Licensee will not disassemble or reverse engineer the Licensed Software, nor sub-licence, lease, rent or part with possession or otherwise transfer the whole or any part of the Licensed Software.

5. WARRANTY

- 5.1 Aeroflex certifies that the Licensed Software supplied by Aeroflex will at the time of delivery function substantially in accordance with the applicable Software Product Descriptions, Data Sheets or Product Specifications published by Aeroflex.
 - 5.2 The warranty period (unless an extended warranty for Embedded Software has been purchased) from date of delivery in respect of each type of Licensed Software is:

Embedded Software	12 months
Add-In Application Software	90 days
Computer Application Software	90 days
Downloaded Software	No warranty
 - 5.3 If during the appropriate Warranty Period the Licensed Software does not conform substantially to the Software Product Descriptions, Data Sheets or Product Specifications Aeroflex will provide:
 - 5.3.1 In the case of Embedded Software and at Aeroflex's discretion either a fix for the problem or an effective and efficient work-around.
 - 5.3.2 In the case of Add-In Application Software and Computer Application Software and at Aeroflex's discretion replacement of the software or a fix for the problem or an effective and efficient work-around.
 - 5.4 Aeroflex does not warrant that the operation of any software will be uninterrupted or error free.
-

6 The above Warranty does not apply to:

- 6.1 Defects resulting from software not supplied by Aeroflex, from unauthorised modification or misuse or from operation outside of the specification.
- 6.2 Third party produced Proprietary Software which Aeroflex may deliver with its products, in such case the third party Software Licence Agreement including its warranty terms shall apply.
- 7 The remedies offered above are sole and exclusive remedies and to the extent permitted by applicable law are in lieu of any implied conditions, guarantees or warranties whatsoever and whether statutory or otherwise as to the software all of which are hereby expressly excluded.

8. INDEMNITY

- 8.1 Aeroflex shall defend, at its expense, any action brought against the Licensee alleging that the Licensed Software infringes any patent, registered design, trademark or copyright, and shall pay all Licensor's costs and damages finally awarded up to an aggregate equivalent to the Licence fee provided the Licensee shall not have done or permitted to be done anything which may have been or become any such infringement and shall have exercised reasonable care in protecting the same failing which the Licensee shall indemnify Aeroflex against all claims costs and damages incurred and that Aeroflex is given prompt written notice of such claim and given information, reasonable assistance and sole authority to defend or settle such claim on behalf of the Licensee. In the defence or settlement of any such claim, Aeroflex may obtain for the Licensee the right to continue using the Licensed Software or replace it or modify it so that it becomes non-infringing.
- 8.2 Aeroflex shall not be liable if the alleged infringement:
 - 8.2.1 is based upon the use of the Licensed Software in combination with other software not furnished by Aeroflex, or
 - 8.2.2 is based upon the use of the Licensed Software alone or in combination with other software in equipment not functionally identical to the Designated Equipment, or
 - 8.2.3 arises as a result of Aeroflex having followed a properly authorised design or instruction of the Licensee, or
 - 8.2.4 arises out of the use of the Licensed Software in a country other than the one disclosed to Aeroflex as the intended country of use of the Licensed Software at the commencement of this Agreement.
- 8.3 Aeroflex shall not be liable to the Licensee for any loss of use or for loss of profits or of contracts arising directly or indirectly out of any such infringement of patent, registered design, trademark or copyright.

9. TERMINATION

- 9.1 Notwithstanding anything herein to the contrary, this Licence shall forthwith determine if the Licensee:
 - 9.1.1 As an individual has a Receiving Order made against him or is adjudicated bankrupt or compounds with creditors or as a corporate body, compounds with creditors or has a winding-up order made against it or
 - 9.1.2 Parts with possession of the Designated Equipment.
- 9.2 This Licence may be terminated by notice in writing to the Licensee if the Licensee shall be in breach of any of its obligations hereunder and continue in such breach for a period of 21 days after notice thereof has been served on the Licensee.
- 9.3 On termination of this Agreement for any reason, Aeroflex may require the Licensee to return to Aeroflex all copies of the Licensed Software in the custody of the Licensee and the Licensee shall, at its own cost and expense, comply with such requirement within 14 days and shall, at the same time, certify to Aeroflex in writing that all copies of the Licensed Software in whatever form have been obliterated from the Designated Equipment.

10. THIRD PARTY LICENCES

The software or part thereof may be the proprietary property of third party licensors. In such an event such third party licensors (as referenced on the package or the Order Acknowledgement) and/or Aeroflex may directly enforce the terms of this Agreement and may terminate the Agreement if the Licensee is in breach of the conditions contained herein.

11. EXPORT REGULATIONS

The Licensee undertakes that where necessary the Licensee will conform with all relevant export regulations imposed by the Governments of the United Kingdom and/or the United State of America.

12. NOTICES

Any notice to be given by the Licensee to Aeroflex shall be addressed to:

Aeroflex International Limited, Longacres House, Six Hills Way, Stevenage, SG1 2AN, UK.

13. LAW AND JURISDICTION

This Agreement shall be governed by the laws of England and shall be subject to the exclusive jurisdiction of the English courts. This agreement constitutes the whole Contract between the parties and may be changed only by memorandum signed by both parties.

© AEROFLEX INTERNATIONAL LTD 2004

**CHINA Beijing**

Tel: [+86] (10) 6539 1166
Fax: [+86] (10) 6539 1778

CHINA Shanghai

Tel: [+86] (21) 5109 5128
Fax: [+86] (21) 5150 6112

FINLAND

Tel: [+358] (9) 2709 5541
Fax: [+358] (9) 804 2441

FRANCE

Tel: [+33] 1 60 79 96 00
Fax: [+33] 1 60 77 69 22

GERMANY

Tel: [+49] 8131 2926-0
Fax: [+49] 8131 2926-130

HONG KONG

Tel: [+852] 2832 7988
Fax: [+852] 2834 5364

INDIA

Tel: [+91] 80 5115 4501
Fax: [+91] 80 5115 4502

KOREA

Tel: [+82] (2) 3424 2719
Fax: [+82] (2) 3424 8620

SCANDINAVIA

Tel: [+45] 9614 0045
Fax: [+45] 9614 0047

SPAIN

Tel: [+34] (91) 640 11 34
Fax: [+34] (91) 640 06 40

UK Burnham

Tel: [+44] (0) 1628 604455
Fax: [+44] (0) 1628 662017

UK Stevenage

Tel: [+44] (0) 1438 742200
Fax: [+44] (0) 1438 727601
Freephone: 0800 282388

USA

Tel: [+1] (316) 522 4981
Fax: [+1] (316) 522 1360
Toll Free: (800) 835 2352

As we are always seeking to improve our products, the information in this document gives only a general indication of the product capacity, performance and suitability, none of which shall form part of any contract. We reserve the right to make design changes without notice.

web www.aeroflex.com

Email info-test@aeroflex.com

November 2005