Programmer's Guide

Agilent Technologies EMC Series Spectrum Analyzers

This guide documents firmware revision A.08.xx

This manual provides documentation for the following instruments:

E7401A (9 kHz - 1.5 GHz)
E7402A (9 kHz - 3.0 GHz)
E7403A (9 kHz - 6.7 GHz)
E7404A (9 kHz - 13.2 GHz)
E7405A (9 kHz - 26.5 GHz)

Agilent Technologies

Manufacturing Part Number: E7401-90053
Supersedes: E7401-90028

Printed in USA

December 2001

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Safety Information

The following safety symbols are used throughout this manual. Familiarize yourself with the symbols and their meaning before operating this instrument.

WARNING ***Warning* denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in injury or loss of life. Do not proceed beyond a warning note until the indicated conditions are fully understood and met.**

CAUTION *Caution* denotes a hazard. It calls attention to a procedure that, if not correctly performed or adhered to, could result in damage to or destruction of the instrument. Do not proceed beyond a caution sign until the indicated conditions are fully understood and met.

NOTE *Note* calls out special information for the user's attention. It provides operational information or additional instructions of which the user should be aware.

⚠	The instruction documentation symbol. The product is marked with this symbol when it is necessary for the user to refer to the instructions in the documentation.
|	This symbol is used to mark the on position of the power line switch.
⏻	This symbol is used to mark the standby position of the power line switch.
∼	This symbol indicates that the input power required is AC.

WARNING **This is a Safety Class 1 Product (provided with a protective earth ground incorporated in the power cord). The mains plug shall be inserted only in a socket outlet provided with a protected earth contact. Any interruption of the protective conductor inside or outside of the product is likely to make the product dangerous. Intentional interruption is prohibited.**

WARNING **No operator serviceable parts inside. Refer servicing to qualified personnel. To prevent electrical shock do not remove covers.**

WARNING **If this product is not used as specified, the protection provided by the equipment could be impaired. This product must be used in a normal condition (in which all means for protection are intact) only.**

CAUTION Always use the three-prong AC power cord supplied with this product. Failure to ensure adequate grounding may cause product damage.

Warranty

This Agilent Technologies instrument product is warranted against defects in material and workmanship for a period of three years from date of shipment. During the warranty period, Agilent Technologies will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Agilent Technologies. Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent Technologies from another country.

Agilent Technologies warrants that its software and firmware designated by Agilent Technologies for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent Technologies does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error-free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. AGILENT TECHNOLOGIES SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Should Agilent have a negotiated contract with the User and should any of the contract terms conflict with these terms, the contract terms shall control.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER’S SOLE AND EXCLUSIVE REMEDIES. AGILENT TECHNOLOGIES SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

Where to Find the Latest Information

Documentation is updated periodically. For the latest information about Agilent Technologies EMC Analyzers, including firmware upgrades and application information, please visit the following Internet URL:

http://www.agilent.com/find/emc

Contents

1. Programming Fundamentals
Creating Valid Commands 39
Command Notation Syntax 40
Special Characters in Commands 41
Parameters in Commands 42
Improving Measurement Speed 45
Turn off the display updates 45
Disable auto alignment 45
Use a fixed IF Gain range 45
Disable the IF/Video/Sweep output ports 45
Select phase noise performance 46
Use binary data format instead of ASCII 46
Minimize the number of GPIB transactions. 46
Avoid unnecessary use of *RST. 47
Minimize DUT/instrument setup changes. 47
Putting Multiple Commands on the Same Line 48
SCPI Termination and Separator Syntax 48
Overview of GPIB (Option A4H) 50
GPIB Instrument Nomenclature 50
GPIB Command Statements 50
Overview of RS-232 (Option 1AX) 52
Settings for the Serial Interface 52
Handshake and Baud Rate 52
Character Format Parameters 52
Modem Line Handshaking 53
Data Transfer Errors 53
Printer Setup and Operation 54
Equipment 54
Interconnection and Setup 54
Testing Printer Operation 56

2. Status Registers
Use Status Registers to Determine the State of Analyzer Events and Conditions 58
What are the Status Registers? 59
How Do You Access the Status Registers? 61
Using the Service Request (SRQ) Method 63
Generating a Service Request 63
Setting and Querying the Status Register 65
Details of Bits in All Registers 65
Status Byte Register 66
Service Request Enable Register 67
Standard Event Status Register 68

Standard Event Status Event Enable Register 70
STATus:OPERation Register 70
STATus:OPERation Condition and Event Enable Registers 73
STATus:QUEStionable Registers 73
STATus:QUEStionable:POWer Register 76
Questionable Status Event Enable Register 77
Questionable Status Power Condition and Event Registers 79
STATus:QUEStionable:FREQuency Register 80
Questionable Status Frequency Condition and Event Enable Registers 82
STATus:QUEStionable:CALibration Register 83
STATus:QUEStionable:INTegrity:UNCalibrated Register 86
Questionable Status Calibration Condition and Event Enable Registers 87
Questionable Status Integrity Uncalibrated Condition and Event Enable Registers 88
STATus:QUEStionable:INTegrity Register 89
Questionable Status Integrity Event Condition and Enable Registers 91

3. Programming Examples
List of Programming Examples 94
Programming Examples System Requirements 95
C Programming Examples using VTL 96
Typical Example Program Contents 96
Linking to VTL Libraries 97
Compiling and Linking a VTL Program 97
Example Program 99
Including the VISA Declarations File 99
Opening a Session 100
Device Sessions 100
Addressing a Session 102
Closing a Session 103
Using Marker Peak Search and Peak Excursion 104
Example: 104
Using Marker Delta Mode and Marker Minimum Search 108
Performing Internal Self-alignment 112
Reading Trace Data using ASCII Format (GPIB) 116
Reading Trace Data Using 32-bit
Real Format (GPIB) 120
Reading Trace Data Using ASCII Format (RS-232) 125
Reading Trace Data Using 32-bit Real Format (RS-232) 130
Using Limit Lines 135
Measuring Noise 141
Entering Amplitude Correction Data 145
Status Register–Determine When a Measurement is Done 149

Determine if an Error has Occurred . . . 155
Measuring Harmonic Distortion (GPIB) . . . 161
Measuring Harmonic Distortion (RS-232) . . . 169
Making Faster Measurements (multiple measurements) . . . 177

4. Programming Command Cross-References
Functional Index to SCPI Subsection . . . 186

5. Language Reference
SCPI Sections and Subsections . . . 191
IEEE Common Commands . . . 193
Calibration Query . . . 193
Clear Status . . . 193
Standard Event Status Enable . . . 193
Standard Event Status Register Query . . . 194
Identification Query . . . 194
Instrument State Query . . . 194
Operation Complete . . . 195
Query Instrument Options . . . 195
Recall . . . 195
Reset . . . 196
Save . . . 196
Service Request Enable . . . 196
Read Status Byte Query . . . 196
Trigger . . . 197
Self Test Query . . . 197
Wait-to-Continue . . . 197
ABORt Subsystem . . . 198
Abort . . . 198
CALCulate Subsystem . . . 199
NdBpoints . . . 199
NdBresults . . . 199
NdBstate . . . 200
Calculate Correction at Frequency . . . 200
Test Current Trace Data Against all Limit Lines . . . 200
CALCulate:EMI:SLISt Subsection . . . 201
Add Measure to List . . . 201
Add Marker to List . . . 201
Append Signal Data to List . . . 201
Clear Marks . . . 204
Set Comment for Signals . . . 205
Set Comment for Current Signal . . . 205

- Set Comment for Marked Signals . . . 205
- Delete Signal . . . 205
- Turn Signal List On or Off . . . 206
- Retrieve Signal as a String . . . 206
- Retrieve Signals in List . . . 206
- Mark Selected Signals . . . 206
- Position Cursor to Signal List . . . 207
- Specify Sort Key . . . 207
- Specify Signal List Display Parameters . . . 208

CALCulate:LLINe Subsection . . . 209

- Delete All Correction Sets in Memory . . . 209
- Control Limit Line Amplitude Interpolation . . . 209
- Set Fixed or Relative Limit Lines . . . 209
- Set Limit Line X-axis Units . . . 210
- Control Limit Line Frequency Interpolation . . . 211
- Define Limit Line Values . . . 211
- Merge Additional Values into the Existing Limit Line . . . 213
- Delete Limit Line . . . 213
- Display the Limit Line . . . 213
- Test the Data Against the Limit Line . . . 214
- Set the Margin Size . . . 214
- Display the Limit Margin . . . 214
- Control Limit Line Testing . . . 215
- Select the Type of Limit Line . . . 215

CALCulate:MARKer Subsection . . . 216

- Markers All Off on All Traces . . . 216
- Continuous Peaking Marker Function . . . 216
- Frequency Counter Marker Resolution . . . 216
- Frequency Counter Marker Automatic Resolution . . . 217
- Frequency Counter Marker . . . 217
- Frequency Counter Marker Query . . . 217
- Marker Function . . . 217
- Marker Peak (Maximum) Search . . . 218
- Marker Peak (Maximum) Left Search . . . 218
- Marker Next Peak (Maximum) Search . . . 218
- Marker Peak (Maximum) Right Search . . . 219
- Marker Peak (Minimum) Search . . . 219
- Marker Mode . . . 219
- Define Peak Excursion . . . 220
- Define Peak Search . . . 220
- Define Peak Threshold . . . 221
- Peak to Peak Delta Markers . . . 221

Set Center Frequency to the Marker Value . . . 221
Set Reference Level to the Marker Value . . . 221
Set Span to the Marker Value . . . 222
Set Start Frequency to the Marker Value . . . 222
Set Center Frequency Step Size to the Marker Value . . . 222
Set Stop Frequency to the Marker Value . . . 222
Marker On/Off . . . 222
Marker Table On/Off . . . 223
Marker to Trace . . . 223
Marker to Trace Auto . . . 223
Continuous Signal Tracking Function . . . 223
Marker X Value . . . 224
Span Markers Center Frequency X Value . . . 224
Marker X Position . . . 224
Span Markers Center Frequency X Position . . . 225
Span Markers Span X Position . . . 225
Delta Pair Markers Start Frequency X Position . . . 225
Delta Pair Markers Stop Frequency X Position . . . 226
Marker X-Axis Readout . . . 226
Span Markers Span X Value . . . 226
Delta Pair Markers Start Frequency X Value . . . 227
Delta Pair Markers Stop Frequency X Value . . . 227
Marker Read Y Value . . . 227
CALCulate:NTData Subsection . . . 228
Normalize the Trace Data . . . 228
CALibration Subsystem . . . 229
Align All Instrument Assemblies . . . 229
Set Auto Align Mode All or Not RF . . . 229
Automatic Alignment . . . 230
Return to the Default Alignment Data . . . 230
Align FM Demodulation . . . 230
Query the Internal or External Frequency Reference . . . 230
Coarse Adjust the Frequency Reference . . . 231
Fine Adjust the Frequency Reference . . . 231
Select the Frequency Corrections . . . 231
Align the RF Circuitry . . . 231
Select the Source State for Calibration . . . 232
Calibrate the Tracking Generator . . . 232
COUPle Subsystem . . . 233
COUPle the Function to Other Settings . . . 233
DISPlay Subsystem . . . 235
Display Viewing Angle . . . 235

Date and Time Display Format 235
Date and Time Display 235
Display Annotation Title Data 236
Turn the Entire Display On/Off 236
Turn the Full Screen Display On/Off 236
Window Annotation 237
Trace Graticule Display 237
Trace X-Axis Scale Offset 237
Display Line Amplitude 238
Display Line On/Off 238
IF Gain Auto/Reference Level Auto Ranging 238
Normalized Reference Level 239
Normalized Reference Level Position 239
Trace Y-Axis Amplitude Scaling 240
Trace Y-Axis Frequency Scaling 240
Trace Y-Axis Reference Level 240
Trace Y-Axis Reference Level Offset 241
Vertical Axis Scaling 242
FORMat Subsystem 243
Byte Order 243
Numeric Data format 243
HCOPy Subsystem 245
Abort the Print 245
Printer Type 245
Select Report Content 245
Print Report 245
Select a Signal List to Include in a Report 246
Select a Signal List to Include in a Report (Delta) .246
Color Hard Copy 246
Print a Hard Copy 247
Form Feed the Print Item 247
Page Orientation 247
Number of Items Printed on a Page 247
Printed Page Size 248
Select Report Type 248
INITiate Subsystem 249
Continuous or Single Measurements 249
Take New Data Acquisitions 250
Abort Measurement 250
Pause the Measurement 250
Restart the Measurement 251
Resume the Measurement 251

INPut Subsystem . . . 252
Input Port Coupling . . . 252
Clear the Input Overload . . . 253
MEASure Group of Commands . . . 254
Configure Commands . . . 254
Fetch Commands . . . 255
Measure Commands . . . 255
Read Commands . . . 256
Get Measurement Results . . . 257
Read Command . . . 257
Configure for Measuring Frequency . . . 257
Measure at Marker . . . 258
Measure at Marker and Add to List . . . 258
Setting Max/Min On or Off . . . 258
Max/Min View ->Max . . . 258
Measure Peaks . . . 259
Remeasure Current Signal . . . 259
MMEMory Subsystem . . . 260
Catalog the Selected Memory Location . . . 260
Copy a File . . . 260
Move Data to File . . . 261
Delete a File . . . 261
Load a Corrections Table from a File . . . 261
Load a Limit Line from Memory to the Instrument . . . 261
Load an Instrument State from a File . . . 262
Load a Trace From a File to the Instrument . . . 262
Make a Directory . . . 263
Store (Load/Save) a Signal List . . . 263
Delete a Directory . . . 264
Store a Corrections Table to a File . . . 264
Store a Limit Line in a File . . . 265
Store Measurement Results in a File . . . 265
Store a Screen Image in a Graphic File . . . 265
Store an Instrument State in a File . . . 265
Store a Trace in a File . . . 266
OUTPut Subsystem . . . 267
Turn Output On/Off . . . 267
SENSe Subsystem . . . 268
[:SENSe]:AVERage Subsection . . . 269
Clear the Current Average . . . 269
Set the Average Count . . . 269
Turn Averaging On/Off . . . 269

Turn Automatic Averaging On/Off 270
Type of Averaging for Measurements 271
[:SENSe]:BANDwidth Subsection 272
Resolution Bandwidth 272
Resolution Bandwidth Automatic 272
Resolution Bandwidth Mode 272
Video Bandwidth 273
Video Bandwidth Automatic 273
Video to Resolution Bandwidth Ratio 273
Video to Resolution Bandwidth Ratio Mode Select 273
Resolution Bandwidth Type 275
[:SENSe]:CORRection Subsection 276
Delete All Corrections 276
Perform Amplitude Correction 276
Set Amplitude Correction Data 276
Merge Additional Values into the Existing Amplitude Correction Data 277
Delete Amplitude Correction 278
Set Amplitude Correction Frequency Interpolation 278
Perform Amplitude Correction 278
Input Impedance Correction 279
External Amplifier Correction 279
[:SENSe]:DEMod Subsection 280
Type of Demodulation 280
FM Deviation 280
Squelch 280
Demodulation Control 281
Demod Time 281
Demod View 281
[:SENSe]:DETector Subsection 282
Automatic Detection Type Selected 282
Type of Detection 284
Type of EMI Detector 285
EMI View 286
Range Immediate 286
Unrange 286
[:SENSe]:EMI Subsection 287
Auto Measure Average On or Off 287
Auto Measure Peak On or Off 287
Auto Measure Quasi Peak On or Off 287
Setting the Dwell Time for Peak 288
Setting the Dwell Time for Quasi Peak 288
Setting the Dwell Time for Average Peak 288

Preselector Centering On or Off (E7403A, E7404A, E7405A only) 288
Setting the Dwell Time for Range 289
Auto Measure Margin On or Off 289
[:SENSe]:FREQuency Subsection 290
Center Frequency 290
Center Frequency Step Size Automatic 290
Center Frequency Step Size 291
Frequency Span 291
Full Frequency Span 292
Last Frequency Span 292
Start Frequency 292
Stop Frequency 293
Frequency Synthesis Mode 293
Frequency Synthesis State 294
[:SENSe]:POWer Subsection 295
Enable/Disable QPD X10 Gain 295
Input Attenuation 295
Input Port Attenuator Auto 295
Input Port Power Gain 296
Input Port Maximum Mixer Power 296
Optimize Preselector Frequency 296
Preselector Center 297
[:SENSe]:SWEep Subsection 298
Sweep Points 298
Set Frequency Domain Scale Type 298
Sweep Time 299
Automatic Sweep Time 299
Sweep Time Mode 300
Time Gating Delay (Option 1D6 Only) 300
Time Gate Length (Option 1D6 Only) 300
Time Gate Level (Option 1D6 Only) 301
Time Gate Polarity (Option 1D6 Only) 301
Preset Time Gate (Option 1D6 Only) 301
Control Time Gate (Option 1D6 Only) 301
Time Gate Trigger Type (Option 1D6 Only) 302
SOURce Subsystem 303
Sets the Output Power Offset Correction 303
Source Attenuation 303
Automatic Source Attenuation 304
Sets the Output Power 304
Sets the Source Output Power Mode 305
Set the Source Sweep Power Range 305

Set the Output Power at the Start of the Sweep 306
Set the Output Power to Step Automatically 306
Set the Output Power Step Size 306
Set the Source Sweep Power Range 307
Output Power Tracking 307
Output Power Tracking Peak 307
STATus Subsystem 308
Operation Condition Query 308
Operation Enable 308
Operation Event Query 308
Operation Negative Transition 309
Operation Positive Transition 309
Preset the Status Byte 309
STATus:QUEStionable Subsection 310
Questionable Calibration Condition 310
Questionable Calibration Enable 310
Questionable Calibration Event Query 310
Questionable Calibration Negative Transition 311
Questionable Calibration Positive Transition 311
Questionable Condition 311
Questionable Enable 312
Questionable Event Query 312
Questionable Frequency Condition 312
Questionable Frequency Enable 313
Questionable Frequency Event Query 313
Questionable Frequency Negative Transition 313
Questionable Frequency Positive Transition 314
Questionable Integrity Condition 314
Questionable Integrity Enable 314
Questionable Integrity Event Query 314
Questionable Integrity Negative Transition 315
Questionable Integrity Positive Transition 315
Questionable Integrity Uncalibrated Enable 315
Questionable Integrity Uncalibrated Event Query 316
Questionable Integrity Uncalibrated Negative Transition 316
Questionable Integrity Uncalibrated Positive Transition 316
Questionable Negative Transition 317
Questionable Power Condition 317
Questionable Power Enable 317
Questionable Power Event Query 317
Questionable Power Negative Transition 318
Questionable Power Positive Transition 318

Questionable Positive Transition 318
SYSTem Subsystem 319
GPIB Address 319
Serial Port DTR Setup 319
Serial Port RTS Setup 320
Serial Port Baud Rate Setup 320
Serial Port Receive Pace Setup 321
Serial Port Transmit Pace Setup 321
Hardware Configuration Query 321
Display the Hardware Configuration 321
System Configuration Query 322
Display System Configuration 322
Set Date 322
Error Information Query 322
Locate SCPI Command Errors 323
Host Identification Query 323
License Key – Install Application/Option 324
Delete a License Key 324
Query Instrument Options 324
Power On Elapsed Time 325
Power On Time 325
Power On Type 325
Enable IF/Video/Sweep Output Ports 325
Preset 326
Persistent State Reset 326
Preset Type 326
Save User Preset 327
Speaker Control 327
Set Time 327
SCPI Version Query 327
TRACe Subsystem 328
Copy Trace 328
Transfer Trace Data 328
Exchange Traces 329
Trace Math Add 329
Mean Trace Data 330
Query the Signal Peaks 330
Query Number of Peaks Found 330
Peak Sorting 330
Smooth Trace Data 330
Number of Points for Smoothing 332
Trace Math Subtract 332

Trace Math Subtract From Display Line 333
Select Trace Display Mode 333
TRIGger Subsystem 334
External Trigger, Line Trigger Delay Value 334
External Trigger, Line Trigger Delay Enable 334
External Trigger Slope 334
Trigger Offset 335
Trigger Source 336
Video Trigger Level Amplitude 336
Video Trigger Level Frequency 337
UNIT Subsystem 338
Select Power Units of Measure 338

6. Agilent 8590/EMC Analyzers Programming Conversion Guide

:SOURce:POWer:ATTenuation <ampl> 303
*CAL? 193
*CLS 193
*CLS 196
*ESE <number> 193
*ESE? 193
*ESR? 194
*IDN? 194
*LRN? 194
*OPC 195
*OPC? 195
*RCL <register> 195
*RST 196
*RST 196
*RST 196
*SAV <register> 196
*SRE <integer> 196
*SRE? 196
*STB? 196
*TRG 197
*TST? 197
*WAI 197
:ABORt 198
:CALCulate:BWIDth|BANDwidth:NDB <rel_ampl> 199
:CALCulate:BWIDth|BANDwidth:NDB? 199
:CALCulate:BWIDth|BANDwidth:RESult? 199
:CALCulate:BWIDth|BANDwidth:RESult? 199
:CALCulate:BWIDth|BANDwidth[:STATe] OFF|ON|0|1 200
:CALCulate:BWIDth|BANDwidth[:STATe]? 200
:CALCulate:CLIMits:FAIL? 200
:CALCulate:CORRection:ATFREquency? <freq> 200

:CALCulate:EMI:SLISt:ADD:MARKer [1]|2|3|4 201
:CALCulate:EMI:SLISt:ADD:MEASured 201
:CALCulate:EMI:SLISt:CLEar ALL|CURRent|<integer> 204
:CALCulate:EMI:SLISt:COMMent:ALL <string> 205
:CALCulate:EMI:SLISt:COMMent:CURRent <string> 205
:CALCulate:EMI:SLISt:COMMent:MARKed <string> 205
:CALCulate:EMI:SLISt:DELete ALL|CURRent|MARKed| <integer> 205
:CALCulate:EMI:SLISt:DISPlay:STATe? 206
:CALCulate:EMI:SLISt:DISPlay[:STATe] OFF|ON|0|1 206
:CALCulate:EMI:SLISt:FETCh? CURRent|<integer> 206
:CALCulate:EMI:SLISt:LENGth? 206
:CALCulate:EMI:SLISt:MARK ALL|CURRent|TOEnd|DUPLicate|LOWer|COMPlement <integer> 206
:CALCulate:EMI:SLISt:SELect FIRSt|LAST|NEXT|PREVious| <integer> 207
:CALCulate:EMI:SLISt:SORT FREQ, ASC? 207
:CALCulate:EMI:SLISt:SORT
FREQuency|PEAK|QPEak|AVERage|LLINE1|LLINE2 ASCending|DESending 207
:CALCulate:EMI:SLISt:View
COMMent|PEAK|QPEak|AVERage 208
:CALCulate:EMI:SLISt:View? 208
:CALCulate:LLINe:ALL:DELete 209
:CALCulate:LLINe:CMODe FIXed|RELative 209
:CALCulate:LLINe:CMODe? 209
:CALCulate:LLINe:CONTrol:DOMain FREQuency|TIME 210
:CALCulate:LLINe:CONTrol:DOMain? 210
:CALCulate:LLINe[1]|2:AMPLitude:INTerpolate:TYPE LOGarithmic|LINear 209
:CALCulate:LLINe[1]|2:AMPLitude:INTerpolate:TYPE? 209
:CALCulate:LLINe[1]|2:CONTrol:INTerpolate:TYPE LOGarithmic| 211
:CALCulate:LLINe[1]|2:CONTrol:INTerpolate:TYPE? 211
:CALCulate:LLINe[1]|2:DATA <x-axis>,<ampl>,<connected>{,<x-axis>,<ampl>,<connected>} 211
:CALCulate:LLINe[1]|2:DATA:MERGe <x-axis>,<ampl>,<connected>{,<x-axis>,<ampl>,<connected>} . . . 213
:CALCulate:LLINe[1]|2:DATA? 211
:CALCulate:LLINe[1]|2:DELete 213

:CALCulate:LLINe[1]|2:DISPlay OFF|ON|0|1 213
:CALCulate:LLINe[1]|2:DISPlay? 213
:CALCulate:LLINe[1]|2:FAIL? 214
:CALCulate:LLINe[1]|2:MARGin <rel_ampl> 214
:CALCulate:LLINe[1]|2:MARGin:STATe OFF|ON|0|1 214
:CALCulate:LLINe[1]|2:MARGin:STATe? 214
:CALCulate:LLINe[1]|2:MARGin? 214
:CALCulate:LLINe[1]|2:STATe OFF|ON|0|1 215
:CALCulate:LLINe[1]|2:STATe? 215
:CALCulate:LLINe[1]|2:TYPE UPPer|LOWer 215
:CALCulate:LLINe[1]|2:TYPE? 215
:CALCulate:MARKer:AOFF 216
:CALCulate:MARKer:FCOunt:RESolution <real> 216
:CALCulate:MARKer:FCOunt:RESolution:AUTO OFF|ON|0|1 217
:CALCulate:MARKer:FCOunt:RESolution:AUTO? 217
:CALCulate:MARKer:FCOunt:RESolution? 216
:CALCulate:MARKer:MODE SPAN 226
:CALCulate:MARKer:PEAK: 220
:CALCulate:MARKer:PEAK:EXCursion <rel_ampl> 220
:CALCulate:MARKer:PEAK:EXCursion? 220
:CALCulate:MARKer:PEAK:SEARch:MODE PARameter|MAXimum 220
:CALCulate:MARKer:PEAK:THReshold <ampl> 221
:CALCulate:MARKer:PEAK:THReshold? 221
:CALCulate:MARKer:TABLe:STATe OFF|ON|0|1 223
:CALCulate:MARKer:TABLe:STATe? 223
:CALCulate:MARKer[1]:FCOunt:X? 217
:CALCulate:MARKer[1]|2|3|4:CPEak[:STATe] OFF|ON|0|1 216
:CALCulate:MARKer[1]|2|3|4:CPEak[:STATe]? 216
:CALCulate:MARKer[1]|2|3|4:FCOunt:X? 217
:CALCulate:MARKer[1]|2|3|4:FCOunt[:STATe] OFF|ON|0|1 217
:CALCulate:MARKer[1]|2|3|4:FCOunt[:STATe]? 217

:CALCulate:MARKer[1]|2|3|4:FUNCtion BPOWer|NOISe|OFF 217
:CALCulate:MARKer[1]|2|3|4:FUNCtion? 217
:CALCulate:MARKer[1]|2|3|4:MAXimum 218
:CALCulate:MARKer[1]|2|3|4:MAXimum:LEFT 218
:CALCulate:MARKer[1]|2|3|4:MAXimum:NEXT 218
:CALCulate:MARKer[1]|2|3|4:MAXimum:RIGHt 219
:CALCulate:MARKer[1]|2|3|4:MINimum. 219
:CALCulate:MARKer[1]|2|3|4:MODE POSition|DELTa|BAND|SPAN 219
:CALCulate:MARKer[1]|2|3|4:MODE? 219
:CALCulate:MARKer[1]|2|3|4:PTPeak 221
:CALCulate:MARKer[1]|2|3|4:STATe OFF|ON|0|1 222
:CALCulate:MARKer[1]|2|3|4:STATe? 222
:CALCulate:MARKer[1]|2|3|4:TRACe <integer>. 223
:CALCulate:MARKer[1]|2|3|4:TRACe:AUTO OFF|ON|0|1. 223
:CALCulate:MARKer[1]|2|3|4:TRACe:AUTO? 223
:CALCulate:MARKer[1]|2|3|4:TRACe? 223
:CALCulate:MARKer[1]|2|3|4:TRCKing[:STATe] OFF|ON|0|1. 223
:CALCulate:MARKer[1]|2|3|4:TRCKing[:STATe]? 223
:CALCulate:MARKer[1]|2|3|4:X <param> 224
:CALCulate:MARKer[1]|2|3|4:X:CENTer <param>. 224
:CALCulate:MARKer[1]|2|3|4:X:CENTer?. 224
:CALCulate:MARKer[1]|2|3|4:X:POSition <integer> 224
:CALCulate:MARKer[1]|2|3|4:X:POSition:CENTer <param> 225
:CALCulate:MARKer[1]|2|3|4:X:POSition:CENTer?. 225
:CALCulate:MARKer[1]|2|3|4:X:POSition:SPAN <param>. 225
:CALCulate:MARKer[1]|2|3|4:X:POSition:SPAN? 225
:CALCulate:MARKer[1]|2|3|4:X:POSition:STARt <param> 225
:CALCulate:MARKer[1]|2|3|4:X:POSition:STARt? 225
:CALCulate:MARKer[1]|2|3|4:X:POSition:STOP <param>. 226
:CALCulate:MARKer[1]|2|3|4:X:POSition:STOP? 226
:CALCulate:MARKer[1]|2|3|4:X:POSition? 224

:CALCulate:MARKer[1]|2|3|4:X:READout FREQuency|TIME|ITIMe|PERiod . . . 226
:CALCulate:MARKer[1]|2|3|4:X:READout? . . . 226
:CALCulate:MARKer[1]|2|3|4:X:SPAN <param> . . . 226
:CALCulate:MARKer[1]|2|3|4:X:SPAN? . . . 226
:CALCulate:MARKer[1]|2|3|4:X:STARt <param> . . . 227
:CALCulate:MARKer[1]|2|3|4:X:STARt? . . . 227
:CALCulate:MARKer[1]|2|3|4:X:STOP <param> . . . 227
:CALCulate:MARKer[1]|2|3|4:X:STOP? . . . 227
:CALCulate:MARKer[1]|2|3|4:X? . . . 224
:CALCulate:MARKer[1]|2|3|4:Y? . . . 217
:CALCulate:MARKer[1]|2|3|4:Y? . . . 227
:CALCulate:MARKer[1]|2|3|4[:SET]:CENTer . . . 221
:CALCulate:MARKer[1]|2|3|4[:SET]:RLEVel . . . 221
:CALCulate:MARKer[1]|2|3|4[:SET]:SPAN . . . 222
:CALCulate:MARKer[1]|2|3|4[:SET]:STARt . . . 222
:CALCulate:MARKer[1]|2|3|4[:SET]:STEP . . . 222
:CALCulate:MARKer[1]|2|3|4[:SET]:STOP . . . 222
:CALCulate:NTData[:STATe] OFF|ON|0|1 . . . 228
:CALCulate:NTData[:STATe]? . . . 228
:CALibration:AUTO OFF|ON|0|1 . . . 230
:CALibration:AUTO:MODE ALL|NRF . . . 229
:CALibration:AUTO:MODE? . . . 229
:CALibration:AUTO? . . . 230
:CALibration:DATA:DEFault . . . 230
:CALibration:FMDemod . . . 230
:CALibration:FMDemod? . . . 230
:CALibration:FREQuency:REFerence:COARse <setting> . . . 231
:CALibration:FREQuency:REFerence:COARse? . . . 231
:CALibration:FREQuency:REFerence:FINE <setting> . . . 231
:CALibration:FREQuency:REFerence:FINE? . . . 231
:CALibration:FREQuency:REFerence? . . . 230

:CALibration:FREQuency[:STATe] OFF|ON|0|1 . . . 231
:CALibration:FREQuency[:STATe]? . . . 231
:CALibration:RF . . . 231
:CALibration:RF? . . . 231
:CALibration:SOURce:STATe OFF|ON|0|1 . . . 232
:CALibration:SOURce:STATe? . . . 232
:CALibration:TG . . . 232
:CALibration:TG? . . . 232
:CALibration[:ALL] . . . 229
:CALibration[:ALL]? . . . 229
:CONFigure:<measurement> . . . 254
:CONFigure:EMI:AFRequency <freq><unit> . . . 257
:CONFigure:EMI:MARKer[1|2|3|4]? . . . 258
:CONFigure:EMI:PEAKs . . . 259
:CONFigure:EMI:SLISt[CURRent]|MARKed|ALL . . . 259
:COUPle ALL|NONE . . . 233
:COUPle? . . . 233
:DISPlay:ANGLe <integer> . . . 235
:DISPlay:ANGLe? . . . 235
:DISPlay:ANNotation:CLOCk:DATE:FORMat MDY|DMY . . . 235
:DISPlay:ANNotation:CLOCk:DATE:FORMat? . . . 235
:DISPlay:ANNotation:CLOCk[:STATe] OFF|ON|0|1 . . . 235
:DISPlay:ANNotation:CLOCk[:STATe]? . . . 235
:DISPlay:ANNotation:TITLe:DATA <string> . . . 236
:DISPlay:ANNotation:TITLe:DATA? . . . 236
:DISPlay:ENABle OFF|ON|0|1 . . . 236
:DISPlay:MENU:STATe OFF|ON|0|1 . . . 236
:DISPlay:MENU:STATe? . . . 236
:DISPlay:WINDow:ANNotation[:ALL] OFF|ON|0|1 . . . 237
:DISPlay:WINDow:ANNotation[:ALL]? . . . 237
:DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe] OFF|ON|0|1 . . . 237

:DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe]? 237
:DISPlay:WINDow:TRACe:X[:SCALe]:OFFSet <freq> 237
:DISPlay:WINDow:TRACe:X[:SCALe]:OFFSet? 237
:DISPlay:WINDow:TRACe:Y:DLINe <ampl> 238
:DISPlay:WINDow:TRACe:Y:DLINe:STATe OFF|ON|0|1 238
:DISPlay:WINDow:TRACe:Y:DLINe:STATe? 238
:DISPlay:WINDow:TRACe:Y:DLINe? 238
:DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel <rel_ampl> 239
:DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel? 239
:DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition <integer> 239
:DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition? 239
:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision <rel_ampl> 240
:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision:FREQuency 240
:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision:FREQuency? 240
:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision? 240
:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel <ampl> 240
:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet <rel_ampl> 241
:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet? 241
:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel? 240
:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing LINear|LOGarithmic 242
:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing? 242
:FETCh:<measurement>[n]? 255
:FETCH? 257
:FORMat:BORDer NORMal|SWAPped 243
:FORMat:BORDer? 243
:FORMat[:TRACe][:DATA]? 243
:FORMat[:TRACe][:DATA]ASCii|INTeger,32|REAL,32|
REAL,64|UINTeger,16 243
:HCOPy:ABORt 245
:HCOPy:DEVice:TYPE AUTO|CUSTom|NONE 245
:HCOPy:DEVice:TYPE? 245

:HCOPy:EMI:ITEM:REPort:SCReem|SLISt|ISETups[:State]? 245

:HCOPy:EMI:ITEM:REPort:SCReen|SLISt|ISETups[:State] |ON|OFF|1|0 245

:HCOPy:EMI:ITEM:SLISt:DELTa1|DELTa2:PPEak|QPEak|AVERage
:STATe ON|OFF|1|0 246

:HCOPy:EMI:ITEM:SLISt:PPEak|QPEak|AVERage|LLINe1|LLINe2| CORRection|UNCertainty|COMMent:STATe ON|OFF|1|0 246

:HCOPy:EMI:ITEM:SLISt:PPEak|QPEak|AVERage|LLINe1|LLINe2| CORRection|UNCertainty|COMMent:STATe? 246

:HCOPy:EMI:REPort:IMMediate 245

:HCOPy:IMAGe:COLor[:STATe] OFF|ON|0|1 246

:HCOPy:IMAGe:COLor[:STATe]? 246

:HCOPy:ITEM:FFEed[:IMMediate] 247

:HCOPy:PAGE:ORIentation LANDscape|PORTrait 247

:HCOPy:PAGE:ORIentation? 247

:HCOPy:PAGE:PRINts <integer> 247

:HCOPy:PAGE:PRINts? 247

:HCOPy:PAGE:SIZE A|B|A3|A4|LETTer|LEGal|EXECutive|LEDGer 248

:HCOPy:PAGE:SIZE? 248

:HCOPy:REPOrt:TYPE SCREen|REPort 248

:HCOPy[:IMMediate] 247

:INITiate:ABort 250

:INITiate:CONTinuous OFF|ON|0|1 249

:INITiate:CONTinuous?. 249

:INITiate:PAUSe 250

:INITiate:RESTart 251

:INITiate:RESume 251

:INITiate[:IMMediate] 250

:INPut:COUPling AC|DC 252

:INPut:COUPling? 252

:INPut:PROTection:CLEar 253

:MEASure:<measurement>[n]? 255

:MEASure:EMI:AFRequency? <freq><unit> 257

:MEASure:EMI:MARKer[1|2|3|4]? 258
:MEASure:EMI:MARKer[1|2|3|4]ADD?. . . . 258
:MEASure:EMI:MMIN:STATe OFF|ON|0|1 258
:MEASure:EMI:MMIN:STATe? 258
:MEASure:EMI:SLISt?[CURRent]MARKed|ALL 259
:MEASure:MARKer?. . . . 258
:MMEMory:CATalog? <drive> 260
:MMEMory:COPY <file_name1>,<file_name2> 260
:MMEMory:DATA <file_name>,<definite_length_block> 261
:MMEMory:DATA? <file_name> 261
:MMEMory:DELete <file_name> 261
:MMEMory:LOAD:CORRection ANTenna|CABLe|OTHer|USER,<file_name> 261
:MMEMory:LOAD:LIMit LLINE1|LLINE2,<file_name> 261
:MMEMory:LOAD:SIGNallist <file_name>. 263
:MMEMory:LOAD:STATe 1,<file_name> 262
:MMEMory:LOAD:TRACe <file_name> 262
:MMEMory:MDIRectory <dir_path> 263
:MMEMory:RDIRectory <dir_name> 264
:MMEMory:STORe:CORRection ANTenna|CABLe|OTHer|USER,<file_name> 264
:MMEMory:STORe:LIMit LLINE1|LLINE2,<file_name> 265
:MMEMory:STORe:RESults <file_name> 265
:MMEMory:STORe:SCReen <file_name> 265
:MMEMory:STORe:SIGNallist <file_name> 263
:MMEMory:STORe:STATe 1,<file_name> 265
:MMEMory:STORe:TRACe <label>,<file_name> 266
:OUTPut[:STATe] OFF|ON|0|1 267
:OUTPut[:STATe]? 267
:READ:<measurement>[n]?. 256
:SOURce:CORRection:OFFSet <rel_ampl> 303
:SOURce:CORRection:OFFSet? 303
:SOURce:POWer:ATTenuation <ampl>. 303

:SOURce:POWer:ATTenuation:AUTO OFF|ON|0|1 304
:SOURce:POWer:ATTenuation:AUTO? 304
:SOURce:POWer:ATTenuation? 303
:SOURce:POWer:MODE FIXed|SWEep 305
:SOURce:POWer:MODE? 305
:SOURce:POWer:SPAN <rel_ampl> 305
:SOURce:POWer:SPAN? 305
:SOURce:POWer:STARt <ampl> 306
:SOURce:POWer:STARt? 306
:SOURce:POWer:STEP:AUTO OFF|ON|0|1 306
:SOURce:POWer:STEP:AUTO? 306
:SOURce:POWer:STEP[:INCRement] <ampl> 306
:SOURce:POWer:STEP[:INCRement]? 306
:SOURce:POWer:SWEep <rel_ampl> 307
:SOURce:POWer:SWEep? 307
:SOURce:POWer:TRCKing <integer> 307
:SOURce:POWer:TRCKing:PEAK 307
:SOURce:POWer:TRCKing? 307
:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude] <ampl> 304
:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude] UP|DOWN 304
:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]? 304
:STATus:OPERation:CONDition? 308
:STATus:OPERation:ENABle<integer> 308
:STATus:OPERation:ENABle? 308
:STATus:OPERation:NTRansition <integer> 309
:STATus:OPERation:NTRansition? 309
:STATus:OPERation:PTRansition <integer> 309
:STATus:OPERation:PTRansition? 309
:STATus:OPERation[:EVENt]? 308
:STATus:PRESet 309
:STATus:QUEStionable:CALibration:CONDition? 310

:STATus:QUEStionable:CALibration:ENABle <integer> . . . 310
:STATus:QUEStionable:CALibration:ENABle? . . . 310
:STATus:QUEStionable:CALibration:NTRansition <integer> . . . 311
:STATus:QUEStionable:CALibration:NTRansition? . . . 311
:STATus:QUEStionable:CALibration:PTRansition <integer> . . . 311
:STATus:QUEStionable:CALibration:PTRansition? . . . 311
:STATus:QUEStionable:CALibration[:EVENt]? . . . 310
:STATus:QUEStionable:CONDition? . . . 311
:STATus:QUEStionable:ENABle <integer> . . . 312
:STATus:QUEStionable:ENABle? . . . 312
:STATus:QUEStionable:FREQuency:CONDition? . . . 312
:STATus:QUEStionable:FREQuency:ENABle <integer> . . . 313
:STATus:QUEStionable:FREQuency:ENABle? . . . 313
:STATus:QUEStionable:FREQuency:NTRansition <integer> . . . 313
:STATus:QUEStionable:FREQuency:NTRansition? . . . 313
:STATus:QUEStionable:FREQuency:PTRansition <integer> . . . 314
:STATus:QUEStionable:FREQuency:PTRansition? . . . 314
:STATus:QUEStionable:FREQuency[:EVENt]? . . . 313
:STATus:QUEStionable:INTegrity:CONDition? . . . 314
:STATus:QUEStionable:INTegrity:ENABle <integer> . . . 314
:STATus:QUEStionable:INTegrity:ENABle? . . . 314
:STATus:QUEStionable:INTegrity:NTRansition <integer> . . . 315
:STATus:QUEStionable:INTegrity:NTRansition? . . . 315
:STATus:QUEStionable:INTegrity:PTRansition <integer> . . . 315
:STATus:QUEStionable:INTegrity:PTRansition? . . . 315
:STATus:QUEStionable:INTegrity[:EVENt]? . . . 314
:STATus:QUEStionable:NTRansition <integer> . . . 317
:STATus:QUEStionable:NTRansition? . . . 317
:STATus:QUEStionable:POWer:CONDition? . . . 317
:STATus:QUEStionable:POWer:ENABle <integer> . . . 317
:STATus:QUEStionable:POWer:ENABle?> . . . 317

:STATus:QUEStionable:POWer:NTRansition <integer> 318
:STATus:QUEStionable:POWer:NTRansition? 318
:STATus:QUEStionable:POWer:PTRansition <integer> 318
:STATus:QUEStionable:POWer:PTRansition? 318
:STATus:QUEStionable:POWer[:EVENt]? 317
:STATus:QUEStionable:PTRansition <integer> 318
:STATus:QUEStionable:PTRansition? 318
:STATus:QUEStionable[:EVENt]? 312
:SYSTem:COMMunicate:GPIB[1][:SELF]:ADDRess <integer> 319
:SYSTem:COMMunicate:GPIB[1][:SELF]:ADDRess? 319
:SYSTem:COMMunicate:SERial[1]:CONTrol:DTR OFF|ON|IBFull 319
:SYSTem:COMMunicate:SERial[1]:CONTrol:DTR? 319
:SYSTem:COMMunicate:SERial[1]:CONTrol:RTS OFF|ON|IBFull 320
:SYSTem:COMMunicate:SERial[1]:CONTrol:RTS? 320
:SYSTem:COMMunicate:SERial[1]:TRANsmit:PACE XON|NONE 321
:SYSTem:COMMunicate:SERial[1]:TRANsmit:PACE? 321
:SYSTem:COMMunicate:SERial[1][:RECeive]:BAUD <baud_rate> 320
:SYSTem:COMMunicate:SERial[1][:RECeive]:BAUD? 320
:SYSTem:COMMunicate:SERial[1][:RECeive]:PACE XON|NONE 321
:SYSTem:COMMunicate:SERial[1][:RECeive]:PACE? 321
:SYSTem:CONFigure:HARDware:STATe OFF|ON|0|1 321
:SYSTem:CONFigure:HARDware:STATe? 321
:SYSTem:CONFigure:HARDware? 321
:SYSTem:CONFigure[:SYSTem]:STATe OFF|ON|0|1 322
:SYSTem:CONFigure[:SYSTem]:STATe? 322
:SYSTem:CONFigure[:SYSTem]? 322
:SYSTem:DATE <year>,<month>,<day> 322
:SYSTem:DATE? 322
:SYSTem:ERRor:VERBose OFF|ON|0|1 323
:SYSTem:ERRor:VERBose? 323
:SYSTem:ERRor[:NEXT]? 322

:SYSTem:HID? 323
:SYSTem:LKEY <“option”>, <“license key”> 324
:SYSTem:LKEY:DELete <“option”> 324
:SYSTem:LKEY? <“option”> 324
:SYSTem:OPTions? 324
:SYSTem:PON:ETIMe? 325
:SYSTem:PON:TIME? 325
:SYSTem:PON:TYPE PRESet|LAST 325
:SYSTem:PON:TYPE? 325
:SYSTem:PRESet 326
:SYSTem:PRESet:PERSistent 319
:SYSTem:PRESet:PERSistent 326
:SYSTem:PRESet:TYPE FACTory|USER 325
:SYSTem:PRESet:TYPE FACTory|USER|MODE 326
:SYSTem:PRESet[:USER]:SAVE 327
:SYSTem:SPEaker[:STATe] OFF|ON|0|1 327
:SYSTem:SPEaker[:STATe]? 327
:SYSTem:TIME <hour>,<minute>,<second> 327
:SYSTem:TIME? 327
:SYSTem:VERSion? 327
:TRACe:COPY <source_trace>,<dest_trace> 328
:TRACe:EXCHange <trace_1>,<trace_2> 329
:TRACe:MATH:ADD <destination_trace>,<source_trace1>,<source_trace2> 329
:TRACe:MATH:MEAN? <trace> 330
:TRACe:MATH:PEAK:POINts? 330
:TRACe:MATH:PEAK:SORT AMPLitude|FREQuency 330
:TRACe:MATH:PEAK:SORT? 330
:TRACe:MATH:PEAK[:DATA]? 330
:TRACe:MATH:SMOoth <trace> 330
:TRACe:MATH:SMOoth:POINts <integer> 332
:TRACe:MATH:SMOoth:POINts? 332

:TRACe:MATH:SUBTract <destination_trace>,<source_trace1>,<source_trace2> 332
:TRACe:MATH:SUBTract:DLINe <trace>. 333
:TRACe[:DATA] <trace_name>|RAWTRACE,<definite_length_
block>|<comma_separated_ASCII_data> 328
:TRACe[:DATA]? <trace_name> 328
:TRACe1|2|3:MODE WRITe|MAXHold|MINHold|VIEW|BLANk 333
:TRACe1|2|3:MODE? 333
:TRIG:SEQ:OFFS 1.0s 335
:TRIGger[:SEQuence]:DELay. 334
:TRIGger[:SEQuence]:DELay:STATe OFF|ON|0|1 334
:TRIGger[:SEQuence]:DELay:STATe? 334
:TRIGger[:SEQuence]:DELay?. 334
:TRIGger[:SEQuence]:EXTernal[1]:SLOPe POSitive|NEGative 334
:TRIGger[:SEQuence]:EXTernal[1]:SLOPe? 334
:TRIGger[:SEQuence]:OFFSet <64 bit floating point value>. 335
:TRIGger[:SEQuence]:OFFSet? 335
:TRIGger[:SEQuence]:SOURce IMMediate|VIDeo|LINE|EXTernal 336
:TRIGger[:SEQuence]:SOURce? 336
:TRIGger[:SEQuence]:VIDeo:LEVel <ampl>. 336
:TRIGger[:SEQuence]:VIDeo:LEVel:FREQuency 337
:TRIGger[:SEQuence]:VIDeo:LEVel:FREQuency? 337
:TRIGger[:SEQuence]:VIDeo:LEVel? 336
:UNIT:POWer DBM|DBMV|DBUV|DBUA|V|W|A 338
:UNIT:POWer?. 338
[:SENSe]:AVERage:CLEar 269
[:SENSe]:AVERage:COUNt <integer> 269
[:SENSe]:AVERage:COUNt? 269
[:SENSe]:AVERage:TYPE VIDeo|RMS 271
[:SENSe]:AVERage:TYPE:AUTO OFF|ON|0|1 270
[:SENSe]:AVERage:TYPE:AUTO? 270
[:SENSe]:AVERage:TYPE? 271

[:SENSe]:AVERage:TYPE? 271
[:SENSe]:AVERage[:STATe] OFF|ON|0|1 269
[:SENSe]:AVERage[:STATe]? 269
[:SENSe]:BANDwidth:TYPE IMPulse|DB6|DB3 275
[:SENSe]:BANDwidth:TYPE? 275
[:SENSe]:BANDwidth|BWIDth:VIDeo <freq> 273
[:SENSe]:BANDwidth|BWIDth:VIDeo:AUTO OFF|ON|0|1 273
[:SENSe]:BANDwidth|BWIDth:VIDeo:AUTO? 273
[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio <number> 273
[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio:AUTO OFF|ON|0|1 273
[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio:AUTO? 273
[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio? 273
[:SENSe]:BANDwidth|BWIDth:VIDeo? 273
[:SENSe]:BANDwidth|BWIDth[:RESolution] <freq> 272
[:SENSe]:BANDwidth|BWIDth[:RESolution]:AUTO OFF|ON|0|1 272
[:SENSe]:BANDwidth|BWIDth[:RESolution]:AUTO? 272
[:SENSe]:BANDwidth|BWIDth[:RESolution]:MODE EMI|SAN|OFF 272
[:SENSe]:BANDwidth|BWIDth[:RESolution]:MODE? 272
[:SENSe]:BANDwidth|BWIDth[:RESolution]? 272
[:SENSe]:CORRection:CSET:ALL:DELete 276
[:SENSe]:CORRection:CSET:ALL[:STATe] OFF|ON|0|1 276
[:SENSe]:CORRection:CSET:ALL[:STATe]? 276
[:SENSe]:CORRection:CSET[1]|2|3|4:DATA <freq>,<rel_ampl>{,<freq>,<rel_ampl>} 276
[:SENSe]:CORRection:CSET[1]|2|3|4:DATA:MERGe <freq>,<rel_ampl>{,<freq>,<rel_ampl>} 277
[:SENSe]:CORRection:CSET[1]|2|3|4:DATA? 276
[:SENSe]:CORRection:CSET[1]|2|3|4:DELete 278
[:SENSe]:CORRection:CSET[1]|2|3|4:X:SPACing LINear|LOGarithmic 278
[:SENSe]:CORRection:CSET[1]|2|3|4[:STATe] OFF|ON|0|1 278
[:SENSe]:CORRection:CSET[1]|2|3|4[:STATe]? 278
[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude] <number> 279
[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude]? 279

[:SENSe]:CORRection:OFFSet[:MAGNitude] 279
[:SENSe]:CORRection:OFFSet[:MAGNitude]? 279
[:SENSe]:DEMod AM|FM 280
[:SENSe]:DEMod:FMDeviation <freq> 280
[:SENSe]:DEMod:FMDeviation? 280
[:SENSe]:DEMod:SQUelch <integer> 280
[:SENSe]:DEMod:STATe OFF|ON|0|1 281
[:SENSe]:DEMod:STATe?. 281
[:SENSe]:DEMod:TIME <time>. 281
[:SENSe]:DEMod:TIME? 281
[:SENSe]:DEMod?. 280
[:SENSe]:DETector:AUTO OFF|ON|0|1 282
[:SENSe]:DETector:AUTO? 282
[:SENSe]:DETector:RANGe:IMMediate 286
[:SENSe]:DETector:RANGe:IMMediate 286
[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle|AVERage|RMS 284
[:SENSe]:DETector[:FUNCtion]: EMI QPEak|AVERage|OFF 285
[:SENSe]:DETector[:FUNCtion]:EMI?. 285
[:SENSe]:DETector[:FUNCtion]? 284
[:SENSe]:DETector[:FUNCtion]EMI:VIEW POSitive|EMI. 286
[:SENSe]:DETector[:FUNCtion]EMI:VIEW? 286
[:SENSe]:DETector[:UNRange] 286
[:SENSe]:EMI:MEASure:DETector:AVERage:DWELl<time> 288
[:SENSe]:EMI:MEASure:DETector:AVERage:DWELl? 288
[:SENSe]:EMI:MEASure:DETector:AVERage? 287
[:SENSe]:EMI:MEASure:DETector:AVERage[STATe] OFF|ON|0|1 287
[:SENSe]:EMI:MEASure:DETector:PPEak:DWELl<time>. 288
[:SENSe]:EMI:MEASure:DETector:PPEak:DWELl?. 288
[:SENSe]:EMI:MEASure:DETector:PPEak? 287
[:SENSe]:EMI:MEASure:DETector:PPEak[STATe]OFF|ON|0|1| 287
[:SENSe]:EMI:MEASure:DETector:QPEak:DWELl<time> 288

[:SENSe]:EMI:MEASure:DETector:QPEak:DWELl? 288
[:SENSe]:EMI:MEASure:DETector:QPEak? 287
[:SENSe]:EMI:MEASure:DETector:QPEak[STATe]OFF|ON|0|1| 287
[:SENSe]:EMI:MEASure:PCENter[:STATe] OFF|ON|0|1 288
[:SENSe]:EMI:MEASure:PCENter[:STATe]? 288
[:SENSe]:EMI:MEASure:PEAKs:SGTMargin? 289
[:SENSe]:EMI:MEASure:PEAKs:SGTMargin[:STATe]ON|OFF|1|0 289
[:SENSe]:EMI:MEASure:RANGe:DWELl<time> 289
[:SENSe]:EMI:MEASure:RANGe:DWELl? 289
[:SENSe]:FREQuency:CENTer <freq> 290
[:SENSe]:FREQuency:CENTer UP|DOWN 290
[:SENSe]:FREQuency:CENTer:STEP:AUTO OFF|ON|0|1 290
[:SENSe]:FREQuency:CENTer:STEP:AUTO? 290
[:SENSe]:FREQuency:CENTer:STEP[:INCRement] <freq> 291
[:SENSe]:FREQuency:CENTer:STEP[:INCRement]? 291
[:SENSe]:FREQuency:CENTer? 290
[:SENSe]:FREQuency:SPAN <freq> 291
[:SENSe]:FREQuency:SPAN:FULL 292
[:SENSe]:FREQuency:SPAN:PREVious 292
[:SENSe]:FREQuency:SPAN? 291
[:SENSe]:FREQuency:STARt <freq> 292
[:SENSe]:FREQuency:STARt? 292
[:SENSe]:FREQuency:STOP <freq> 293
[:SENSe]:FREQuency:STOP? 293
[:SENSe]:FREQuency:SYNThesis 1|2|3 293
[:SENSe]:FREQuency:SYNThesis:AUTO OFF|ON|0|1 294
[:SENSe]:FREQuency:SYNThesis:AUTO? 294
[:SENSe]:FREQuency:SYNThesis?. 293
[:SENSe]:POWer:QPGain[:STATe]? 295
[:SENSe]:POWer:QPGain[:STATe]ON|OFF|1|0 295
[:SENSe]:POWer[:RF]:ATTenuation <rel_ampl> 295

[:SENSe]:POWer[:RF]:ATTenuation:AUTO OFF|ON|0|1 . . . 295
[:SENSe]:POWer[:RF]:ATTenuation:AUTO? . . . 295
[:SENSe]:POWer[:RF]:ATTenuation? . . . 295
[:SENSe]:POWer[:RF]:GAIN[:STATe] OFF|ON|0|1 . . . 296
[:SENSe]:POWer[:RF]:GAIN[:STATe]? . . . 296
[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer] <ampl> . . . 296
[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer]? . . . 296
[:SENSe]:POWer[:RF]:PADJust <freq> . . . 296
[:SENSe]:POWer[:RF]:PADJust? . . . 296
[:SENSe]:POWer[:RF]:PCENter . . . 297
[:SENSe]:SWEep:POINts <number of points> . . . 298
[:SENSe]:SWEep:POINts? . . . 298
[:SENSe]:SWEep:SPACing LINear|LOGarithmic . . . 237
[:SENSe]:SWEep:SPACing LINear|LOGarithmic . . . 298
[:SENSe]:SWEep:SPACing? . . . 298
[:SENSe]:SWEep:TIME <time> . . . 299
[:SENSe]:SWEep:TIME:AUTO OFF|ON|0|1 . . . 299
[:SENSe]:SWEep:TIME:AUTO:MODE SRESponse|SANalyzer . . . 300
[:SENSe]:SWEep:TIME:AUTO:MODE? . . . 300
[:SENSe]:SWEep:TIME:AUTO? . . . 299
[:SENSe]:SWEep:TIME:GATE:DELay <time> . . . 300
[:SENSe]:SWEep:TIME:GATE:DELay? . . . 300
[:SENSe]:SWEep:TIME:GATE:LENGth <time> . . . 300
[:SENSe]:SWEep:TIME:GATE:LENGth? . . . 300
[:SENSe]:SWEep:TIME:GATE:LEVel HIGH|LOW . . . 301
[:SENSe]:SWEep:TIME:GATE:LEVel? . . . 301
[:SENSe]:SWEep:TIME:GATE:POLarity NEGative|POSitive . . . 301
[:SENSe]:SWEep:TIME:GATE:POLarity? . . . 301
[:SENSe]:SWEep:TIME:GATE:PRESet . . . 301
[:SENSe]:SWEep:TIME:GATE:TYPE LEVel|EDGE . . . 302
[:SENSe]:SWEep:TIME:GATE:TYPE? . . . 302

[:SENSe]:SWEep:TIME:GATE[:STATe] OFF|ON|0|1 301
[:SENSe]:SWEep:TIME:GATE[:STATe]? 301
[:SENSe]:SWEep:TIME? 299

1 Programming Fundamentals

The purpose of this chapter is to serve as a reminder of SCPI (Standard Commands for Programmable Instruments) fundamentals to those who have previous experience in programming SCPI. This chapter is not intended to teach you everything about the SCPI programming language.

The SCPI Consortium or IEEE can provide detailed information on the subject of SCPI programming. Refer to IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation.* New York, NY, 1987, or to IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

Valid EMC Analyzer SCPI commands are used for examples in this chapter. Topics included in this chapter are:

- "Creating Valid Commands"
- "Command Notation Syntax"
- "Special Characters in Commands"
- "Parameters in Commands"
- "Improving Measurement Speed"
- "Putting Multiple Commands on the Same Line"
- "Overview of GPIB (Option A4H)"
- "Overview of RS-232 (Option 1AX)"
- "Printer Setup and Operation"

Creating Valid Commands

Commands are not case sensitive and there are often many different ways of writing a particular command. These are examples of valid commands for a given command syntax:

Command Syntax	Sample Valid Commands
`[:SENSe]:BANDwidth[:RESolution] <freq>`	The following sample commands are all identical. They will all cause the same result. • `:Sense:Band:Res 1700` • `:BANDWIDTH:RESOLUTION 1.7e3` • `:sens:band 1.7KHZ` • `:SENS:band 1.7E3Hz` • `:band 1.7kHz` • `:bandwidth:RES 1.7e3Hz`
`:CALCulate:MARKer[1]|2|3|4:Y?`	The last command below returns different results than the commands above it. The number 3 in the command causes this. See the command description for more information. • `:CALC:MARK:Y?` • `:calc:mark:y?` • `:CALC:MARK2:Y?`
`[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle`	• `DET:FUNC NEG` • `:Sense:Detector:Function Sample`
`:INITiate:CONTinuous OFF|ON|0|1`	The sample commands below are identical. • `:INIT:CONT ON` • `:init:continuous 1`

Command Notation Syntax

A typical command is made up of key words set off by colons. The key words are followed by parameters that can be followed by optional units.

Example: `:TRIGger:SEQuence:VIDeo:LEVel 2.5V`

The instrument does not distinguish between upper and lower case letters. In the documentation, upper case letters indicate the short form of the key word. The upper and lower case letters, together, indicate the long form of the key word. Either form may be used in the command.

Example: `:Trig:Seq:Vid:Lev 2.5V` is the same as `trigger:sequence:video:level 2.5V`.

NOTE The command `:TRIGG:Sequence:Video:Level 2.5V` is not valid because `:TRIGG` is neither the long, nor the short form of the command.

Special Characters in Commands

Special Character	Meaning	Example
|	A vertical stroke between **parameters** indicates alternative choices. The effect of the command is different depending on which parameter is selected.	Command: `[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle` The choices are neg, pos, and samp. `:SENSe:DETector:FUNCtion SAMPle` is one possible command choice.

	A vertical stroke between **key words** indicates identical effects exist for several key words. Only one of these key words is used at a time. The command functions the same for either key word.	Command:
		`[:SENSe]:CHPower:BANDwidth|BWIDth:INTegrati on`
		Two identical commands are:
		`:SENSe:CHPower:BANDwidth:INTegration`
		`:SENSe:CHPower:BWIDth:INTegration`

[ ]	Key words in square brackets are optional when composing the command. These implied key words will be executed even if they are omitted.	Command:
		`[SENSe:]BANDwidth[:RESolution]:AUTO`
		The following commands are all valid and have identical effects:
		`:bandwidth:auto`
		`:bandwidth:resolution:auto`
		`:sense:bandwidth:auto`

< >	Angle brackets around a word, or words, indicates they are not to be used literally in the command. They represent the needed item.	Command:
		`:SENSe:FREQ <freq>`
		In this command example the word <freq> should be replaced by an actual frequency:
		`:SENSe:FREQ 9.7 MHz`

{ }	Parameters in braces can optionally be used in the command either not at all, once, or several times.	Command:
		`[SENSe:]CORRection:CSET[1]|2|3|4:DATA:MERGe <freq>,<rel_ampl>{,<freq>,<rel_ampl>}`
		A valid form of this command is:
		`[SENSe:]CORRection:CSET1:DATA:MERGe 740000,.94 1250000,.31 3320000,1.7`

Parameters in Commands

There are four basic types of parameters: boolean, key words, variables and arbitrary block program data.

Boolean

The expression OFF|ON|0|1 is a two state boolean-type parameter. The numeric value 0 is equivalent to OFF. Any numeric value other than 0 is equivalent to ON. The numeric values of 0 or 1 are commonly used in the command instead of OFF or ON, and queries of the parameter always return a numeric value of 0 or 1.

Key Word

The parameter key words that are allowed for a particular command are defined in the command description and are separated with a vertical slash.

Units

Numerical variables may include units. The valid units for a command depends on the variable type being used. See the following variable descriptions. If no units are sent, the indicated default units will be used. Units can follow the numerical value with, or without, a space.

Variable

A variable can be entered in exponential format as well as standard numeric format. The appropriate variable range and its optional units are defined in the command description.

In addition to these values, the following key words may also be used in commands where they are applicable.

MINimum - sets the parameter to the smallest possible value.

MAXimum - sets the parameter to the largest possible value.

UP - increments the parameter.

DOWN- decrements the parameter.

Include the key word MINimum or MAXimum after the question mark in a query in order to return the numeric value of the key word.

Example query: **`[:SENSE]:FREQuency:CENTer? MAXimum`**

Variable Parameters

<ampl>, <rel_ampl>	The <ampl> (amplitude) parameter and the <rel_ampl> (relative amplitude) parameter consist of a rational number followed by optional units. Acceptable units for <ampl> include: V, mV, µV, dBm, dBmV, dBµV, Watts, W. <rel_ampl> units are given in dB.

<angle>	An angle parameter is a rational number followed by optional units. The default units are degrees. Acceptable units include: DEG, RAD.
<file_name>	A file name parameter is the name of your file including the full path. The back slash that follows the drive colon (C:\), usually used in computer paths, is not used in the SCPI command string.
<freq>	A frequency parameter is a positive rational number followed by optional units. The default unit is Hz. Acceptable units include: Hz, kHz, MHz, GHz.
<integer>	There are no units associated with an integer parameter.
<number>	A number parameter is a member of the set of positive or negative intriguers and including zero. Fractional numbers are included in the number parameter. There are no units associated with a number parameter.
<percent>	A percent parameter is a rational number between 0 and 100, with no units.
<rel_power>	A relative power parameter is a positive rational number followed by optional units. The default units are dB. Acceptable units are dB only.
<string>	A string parameter includes a series of alpha numeric characters.
<time>	A time parameter is a rational number followed by optional units. The default units are seconds. Acceptable units include: S, MS, US.

Block Program Data

Definite length arbitrary block response data is defined in section 8.7.9.2 of IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

<definite_length_block>

Allows data to be transmitted over the system interface as a series of 8 bit data bytes. This element is particularly useful for sending large quantities of data, 8 bit extended ASCII codes, or other data that are not able to be directly displayed.

A definite length block of data starts with an ASCII header that begins with # and indicates how many additional data points are following in the block. For example, if the header is #512320, then interpret the header as follows:

- The first digit in the header (5) represents how many additional digits/bytes there are in the header.

- The numbers 12320 indicates 12 thousand, 3 hundred, 20 data bytes follow the header.
- To determine how may points in the block, divide 12320 by your selected data format bytes/point. Divide by 8 for real 64, or 4 for real 32. In this example there are 1540 points in the block if your selected data format is real 64.

Improving Measurement Speed

There are a number of things you can do in your programs to make them run faster:

"Turn off the display updates" on page 45.

"Use binary data format instead of ASCII" on page 46.

"Minimize the number of GPIB transactions." on page 46.

"Avoid unnecessary use of *RST." on page 47.

"Minimize DUT/instrument setup changes." on page 47.

Turn off the display updates

`:DISPlay:ENABle OFF` turns off the display. Updating the display slows down the measurement. For remote testing, since the computer is processing the data rather than a person, there is no need to display the data on the analyzer screen.

Disable auto alignment

`:CALibration:AUTO OFF` disables the automatic alignment process of the instrument. Automatic alignment processing occurs at the end of each sweep. In a stable operating environment, automatic alignment consumes very little instrument resources. However, in a high throughput application, any demand upon instrument resources affects measurement update rate.

NOTE When auto alignment is off, the **Align Now**, **All** function should be performed periodically. Refer to the appropriate "Specifications and Characteristics" chapter in the *Agilent Technologies EMC Analyzers Specifications Guide* for more information on how often to perform **Align Now**, **All** when the auto alignment is off.

Use a fixed IF Gain range

In applications where narrow resolution bandwidths (< 1 kHz) are required and a high dynamic range is not required, **`:DISPlay:WINDow:TRACe:Y[SCALe]:LOG:RANGe:AUTO OFF`** disables auto ranging and results in increased measurement update rate.

Disable the IF/Video/Sweep output ports

If the analyzer has Options A4J (IF, Video and Sweep Ports) or AYX (Fast Time Domain Sweeps), various output signals with rear-panel ports are controlled by instrument processing. If these ports are not used in a particular application, **`:SYSTem:PORTs:IFVSweep:ENABle OFF`** can be used to disable the ports and conserve instrument resources.

Select phase noise performance

`[:SENSe]:FREQuency:SYNThesis` can be used to optimize either phase noise performance or tuning speed. In some settings optimizing for tuning speed reduces sweep time and the “re-tune” time between sweeps. In other settings only the re-tune time is improved.

Use binary data format instead of ASCII

The ASCII data format is the instrument default since it is easier for people to understand and is required by SCPI for *RST. However, data input/output is faster using the binary formats.

`:FORMat:DATA REAL,64` selects the 64-bit binary data format for all your numerical data queries. You may need to swap the byte order if you are using a PC rather than UNIX. `NORMal` is the default byte order. Use `:FORMat:BORDer SWAP` to change the byte order so that the least significant byte is sent first.

When using the binary format, data is sent in a block of bytes with an ASCII header. A data query would return the block of data in the following format: `#DNNN<nnn binary data bytes>`

To parse the data:

- Read two characters (#D), where D tells you how many N characters follow the D character.
- Read D characters, the resulting integer specifies the number of data bytes sent.
- Read the bytes into a real array.

For example, suppose the header is #512320.

- The first character/digit in the header (5) tells you how many additional digits there are in the header.
- The 12320 means 12 thousand, 3 hundred, 20 data bytes follow the header.
- Divide this number of bytes by your current data format (bytes/data point), 8 for real, 64. For this example, there are 1540 data points in the block of data.

Minimize the number of GPIB transactions.

When you are using the GPIB for control of your instrument, each transaction requires driver overhead and bus handshaking, so minimizing these transactions reduces the time used.

You can reduce bus transactions by sending multiple commands per transaction. See the information on “Putting Multiple Commands on the Same Line” in the SCPI Language Basics section.

If you are using the pre-configured **MEASURE** key measurements and are making the same measurement multiple times with small changes in the measurement setup, use the single READ command. It is faster then using INITiate and FETCh.

Avoid unnecessary use of *RST.

Remember that *RST presets all the measurements and settings to their factory defaults and my also change the mode. This forces you to reset the measurement settings of the analyzer even if they use similar mode setup or measurement settings. See Minimize DUT/instrument setup changes. below.

Minimize DUT/instrument setup changes.

- Some instrument setup parameters are common to multiple measurements. You should look at your measurement process with a focus on minimizing setup changes. If your test process involves nested loops, make sure that the inner-most loop is the fastest. Also, check if the loops could be nested in a different order to reduce the number of parameter changes as you step through the test.
- Are you are using the pre-configured Measurements (**MEASURE** key)? Remember that if you have already set your Meas Setup parameters for a measurement, and you want to make another one of these measurements later, use READ:<meas>?. The MEASure:<meas>?. command resets all the settings to the defaults, while READ changes back to that measurement without changing the setup parameters from the previous use.
- Are you are using the pre-configured Measurements (**MEASURE** key)? Also remember that Mode Setup parameters remain constant across all the measurements (such as: center/channel frequency, amplitude, radio standard, input selection, trigger setup). You don’t have to re-initialize them each time you change to a different measurement.

Putting Multiple Commands on the Same Line

Multiple commands can be written on the same line, reducing your code space requirement. To do this:

- Commands must be separated with a semicolon (;).
- If the commands are in different subsystems, the key word for the new subsystem must be preceded by a colon (:).
- If the commands are in the same subsystem, the full hierarchy of the command key words need not be included. The second command can start at the same key word level as the command that was just executed.

SCPI Termination and Separator Syntax

A terminator must be provided when an instrument is controlled using RS-232 (Option 1AX). There are several issues to be understood about choosing the proper SCPI terminator and separator when this is the case. There is no current SCPI standard for RS-232. Although one intent of SCPI is to be interface independent, <END> is only defined for IEEE 488 operation. At the time of this writing, the RS-232 terminator issue was in the process of being addressed in IEEE standard 1174.

A semicolon (;) is not a SCPI terminator, it is a separator. The purpose of the separator is to queue multiple commands or queries in order to obtain multiple actions and/or responses. Make sure that you do not attempt to use the semicolon as a terminator when using RS-232 control.

Basically all binary trace and response data is terminated with <NL><END>, as defined in Section 8.5 of IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

The following are some examples of good and bad commands. The examples are created from an EMC analyzer with the simple set of commands indicated below:

```
[:SENSe]
     :POWer
          [:RF]
          :ATTenuation 40dB

[:SENSe]
     :FREQuency
          :STARt
     :POWer
     [:RF]
          :MIXer
               :RANGe
              [:UPPer]
```

```
:TRIGger
     [:SEQuence]
     :EXTernal [1]
          :SLOPe
               POSitive
```

Bad Command	Good Command
`PWR:ATT 40dB`	`POW:ATT 40dB`
The short form of `POWER` is `POW`, not `PWR`.	
`FREQ:STAR 30MHz;MIX:RANG -20dBm`	`FREQ:STAR 30MHz;POW:MIX:RANG -20dBm`
The `:MIX:RANG` command is in the same `:SENSE` subsystem as `:FREQ`, but executing the `:FREQ` command puts you back at the `:SENSE` level. You must specify `:POW` to get to the `:MIX:RANG` command.	
`FREQ:STAR 30MHz;POW:MIX RANG -20dBm`	`FREQ:STAR 30MHz;POW:MIX:RANG -20dBm`
`:MIX` and `:RANG` require a colon to separate them.	
`:POW:ATT 40dB;TRIG:FREQ:STAR 2.3GHz`	`:POW:ATT 40dB;:FREQ:STAR 2.3GHz`
`:FREQ:STAR` is in the `:SENSE` subsystem, not the `:TRIGGER` subsystem.	
`:POW:ATT?:FREQ:STAR?`	`:POW:ATT?;:FREQ:STAR?`
`:POW` and `:FREQ` are within the same `:SENSE` subsystem, but they are two separate commands, so they should be separated with a semicolon, not a colon.	
`:POW:ATT -5dB;:FREQ:STAR 10MHz`	`:POW:ATT 5dB;:FREQ:STAR 10MHz`
Attenuation cannot be a negative value.	

Overview of GPIB (Option A4H)

GPIB Instrument Nomenclature

An instrument that is part of a GPIB network is categorized as a listener, talker, or controller, depending on its current function in the network.

Listener	A listener is a device capable of receiving data or commands from other instruments. Any number of instruments in the GPIB network can be listeners simultaneously.
Talker	A talker is a device capable of transmitting data or commands to other instruments. To avoid confusion, an GPIB system allows only one device at a time to be an active talker.
Controller	A controller is an instrument, typically a computer, capable of managing the various GPIB activities. Only one device at a time can be an active controller.

GPIB Command Statements

Command statements form the nucleus of GPIB programming. They are understood by all instruments in the network. When combined with the programming language codes, they provide all management and data communication instructions for the system. Refer to the your programming language manual and your computers I/O programming manual for more information.

The seven fundamental command functions are as follows:

- An abort function that stops all listener/talker activity on the interface bus, and prepares all instruments to receive a new command from the controller. Typically, this is an initialization command used to place the bus in a known starting condition (sometimes called: abort, abortio, reset, halt).
- A remote function that causes an instrument to change from local control to remote control. In remote control, the front panel keys are disabled except for the Local key and the line power switch (sometimes called: remote, resume).
- A local lockout function, that can be used with the remote function, to disable the front panel Local key. With the Local key disabled, only the controller (or a hard reset by the line power switch) can restore local control (sometimes called: local).

- A local function that is the complement to the remote command, causing an instrument to return to local control with a fully enabled front panel (sometimes called: local, resume).
- A clear function that causes all GPIB instruments, or addressed instruments, to assume a cleared condition. The definition of clear is unique for each instrument (sometimes called: clear, reset, control, send).

  In the Agilent EMC Analyzer, clear does the following:

  1. Clears the Input Buffer and the Output Queue.
  2. Resets the parser.
  3. Clears any current operations, such as *OPC, i.e., returns the device to Operation Complete Query Idle State and Operation Complete Command Idle State.
  4. Aborts /resumes the current sweep.
- An output function that is used to send function commands and data commands from the controller to the addressed instrument (sometimes called: output, control, convert, image, iobuffer, transfer).
- An enter function that is the complement of the output function and is used to transfer data from the addressed instrument to the controller (sometimes called: enter, convert, image, iobuffer, on timeout, set timeout, transfer).

Overview of RS-232 (Option 1AX)

Serial interface programming techniques are similar to most general I/O applications. Due to the asynchronous nature of serial I/O operations, special care must be exercised to ensure that data is not lost by sending to another device before the device is ready to receive. Modem line handshaking can he used to help solve this problem. These and other topics are discussed in greater detail in your programming language documentation.

Settings for the Serial Interface

Please refer to the documentation on your computer and I/O to configure the serial interface. Some common serial interface configuration settings are:

Baud Rate to	9600
Bits per character to	8
Parity to	Odd or disabled
Stop bits to	1

Handshake and Baud Rate

To determine hardware operating parameters, you need to know the answer for each of the following questions about the peripheral device:

- Which of the following signal and control lines are actively used during communication with the peripheral?

  — Data Set Ready (DSR)

  — Clear to Send (CTS)

- What baud rate is expected by the peripheral?

Character Format Parameters

To define the character format, you must know the requirements of the peripheral device for the following parameters:

- Character Length: Eight data bits are used for each character, excluding start, stop, and parity bits.
- Parity Enable: Parity is disabled (absent) for each character.
- Stop Bits: One stop bit is included with each character.

Modem Line Handshaking

To use modem line handshaking for data transfer you would consider the following tasks:

1. Set Data Terminal Ready and Request-to-Send modem lines to active state.
2. Check Data Set Ready and Clear-to-Send modem lines to be sure they are active.
3. Send information to the interface and thence to the peripheral.
4. After data transfer is complete, clear Data Terminal Ready and Request-to-Send signals.

For ENTER operations:

1. Set Data Terminal Ready line to active state. Leave Request-to-Send inactive.
2. Check Data Set Ready and Data Carrier Detect modem lines to be sure they are active.
3. Input information from the interface as it is received from the peripheral.
4. After the input operation is complete, clear the Data Terminal Ready signal.

Data Transfer Errors

The serial interface can generate several types of errors when certain conditions are encountered while receiving data from the peripheral device. Errors can be generated by any of the following conditions:

- Parity error. The parity bit on an incoming character does not match the parity expected by the receiver. This condition is most commonly caused by line noise.
- Framing error. Start and stop bits do not match the timing expectations of the receiver. This can occur when line noise causes the receiver to miss the start bit or obscures the stop bits.
- Overrun error. Incoming data buffer overrun caused a loss of one or more data characters. This is usually caused when data is received by the interface, but no ENTER statement has been activated to input the information.
- Break received. A BREAK was sent to the interface by the peripheral device. The desktop computer program must be able to properly interpret the meaning of a break and take appropriate action.

Printer Setup and Operation

Equipment

- Agilent EMC Analyzer equipped with standard I/O Option A4H (GPIB) and (Parallel Interface) or Option 1AX (RS-232 and Parallel Interface).
- IEEE 1284 compliant printer cable (such as HP C2950A).
- Supported printer equipped with a parallel interface. (A supported printer is one that accepts Printer Control Language Level 3 or 5).
  — PCL3 printers include most HP DeskJet printers.
  — PCL5 printers include most HP LaserJet printers and the 1600C DeskJet printer.

Interconnection and Setup

1. Turn off the printer and the analyzer.
2. Connect the printer to the analyzer parallel I/O interface connector using an IEEE 1284 compliant parallel printer cable.
3. If appropriate, configure your printer using configuration menus or switches. Refer to your printer's documentation for more specific information on configuring your printer.
4. Turn on the analyzer and printer.
5. Press **Print Setup** on the front panel and then press the **Printer Type** menu key. **Printer Type** accesses the following keys:

None	**None** disables the analyzer from attempting to print to a printer. This is the appropriate setting if no printer is connected to the analyzer.
Custom	**Custom** allows you to access the **Define Custom** menu keys. The **Define Custom** menu keys allow you to specify printer characteristics such as PCL Level and printer color capability.
Auto	**Auto** enables the analyzer to automatically attempt to identify the connected printer when the **Print** key is pressed or when **Printer Type** is set to **Auto**.

6. Press **Printer Type** to access the **Printer Type** menu keys. Press **Auto** to make the analyzer attempt to identify the connected printer. When you press **Auto**, the analyzer will respond in one of the three following ways:

   - The **Print Setup** menu will be displayed with the **Auto** key selected and no new message will be displayed in the display status line. This indicates that the analyzer has successfully identified the connected printer and no further setup is required. As long as **Auto** remains selected in the **Printer Type** menu, the analyzer will attempt to identify the printer when the front panel **Print** key is pressed.

   - The **Print Setup** menu will be displayed with the **Custom** key selected and one of the following diagnostic messages will be displayed in the display status line:

     `Unknown printer, Define Custom to set up printer`

     `No printer response, Define Custom to set up printer`

     `Invalid printer response, Define Custom to set up printer`

     This indicates that the analyzer was unable to automatically identify the connected printer, and **Custom** has been selected in the **Printer Type** menu. Press **Print Setup**, **Define Custom** to select specific printer characteristics such as the printer language (PCL3 or PCL5) and color printing capability. Once you have set these characteristics to match those of your connected printer, the printer setup process is complete. As long as **Custom** remains selected in the **Printer Type** menu, the analyzer will not attempt to automatically identify the connected printer when the front panel **Print** key is pressed.

   - The **Print Setup** menu will be displayed with the **None** key selected and the following message will appear in the display status line:

     `Unsupported printer, Printer Type set to None`

     This indicates that the analyzer has successfully identified the connected printer, but the printer is not supported by the analyzer. As long as **None** is selected in the **Printer Type** menu, the analyzer will respond to any print command by displaying the message `Printer Type is None` in the display status line.

Testing Printer Operation

When you have completed the printer setup for the analyzer, press **Print Setup**, **Print** (Screen), and then press **Print** on the front panel. If the printer is ready and the printer setup was successful, a printout of the analyzer display will be printed. If the printer is not ready, the message `Printer Timeout` will appear on the analyzer display. `Printer Timeout` will remain on the display until the printer is ready or until you press **ESC** to cancel the printout request.

2 Status Registers

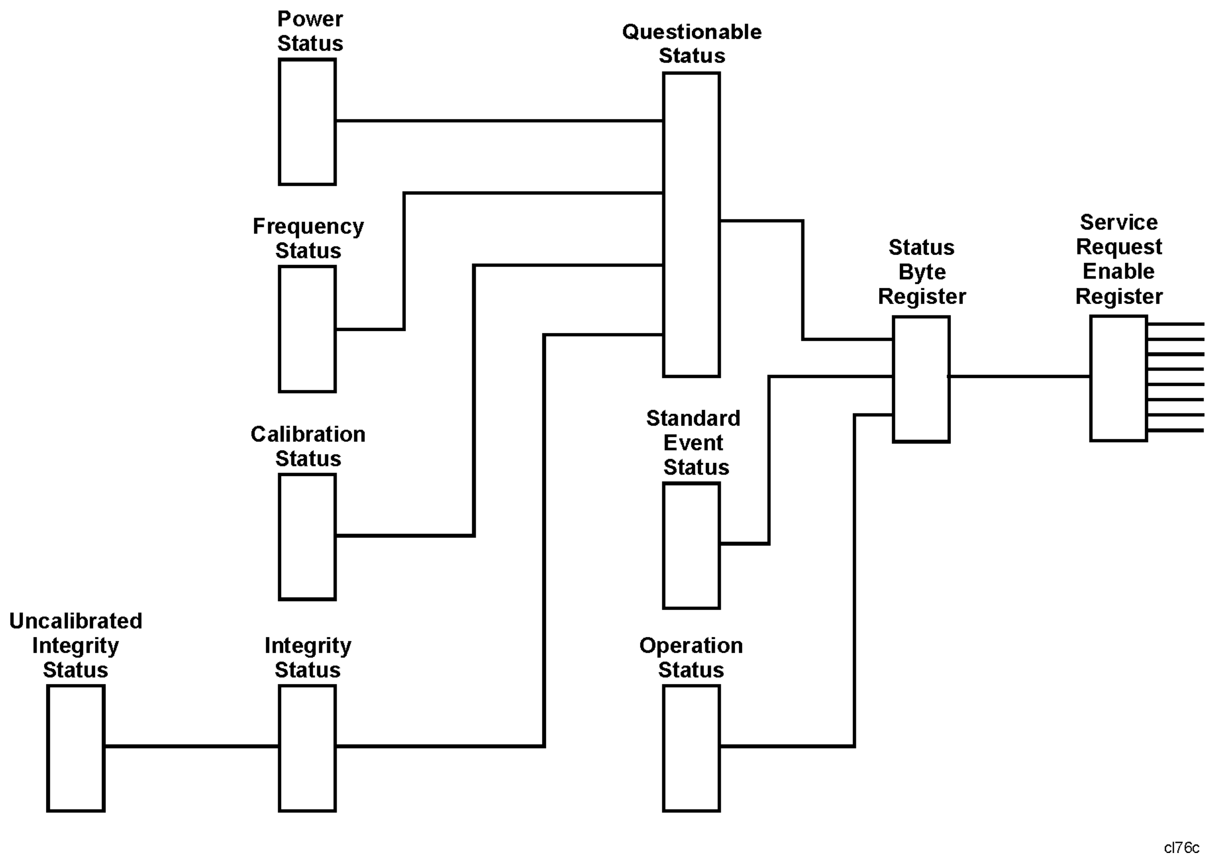
This chapter contains a comprehensive description of status registers explaining what status registers are and how to use them. Information pertaining to all bits of the registers in Agilent EMC analyzers is also provided.

Use Status Registers to Determine the State of Analyzer Events and Conditions

Programs often need to detect and manage error conditions or changes in analyzer status. Agilent EMC products allow this function to be performed using status registers. You can determine the state of certain analyzer hardware and firmware events and conditions by programming the status register system.

Refer to Figure 2-1. The status system is comprised of multiple registers arranged in a hierarchical order. The service request enable register is at the top of the hierarchy and contains the general status information for the analyzer events and conditions. The lower-priority status registers propagate their data to the higher-priority registers in the data structures by means of summary bits. These registers are used to determine the states of specific events or conditions.

Figure 2-1 **Status Register System Simplified Block Diagram**



The two methods used to programmatically access the information in status registers are the polling method and the service request method. An explanation of these methods is given in the next section "What are the Status Registers?"

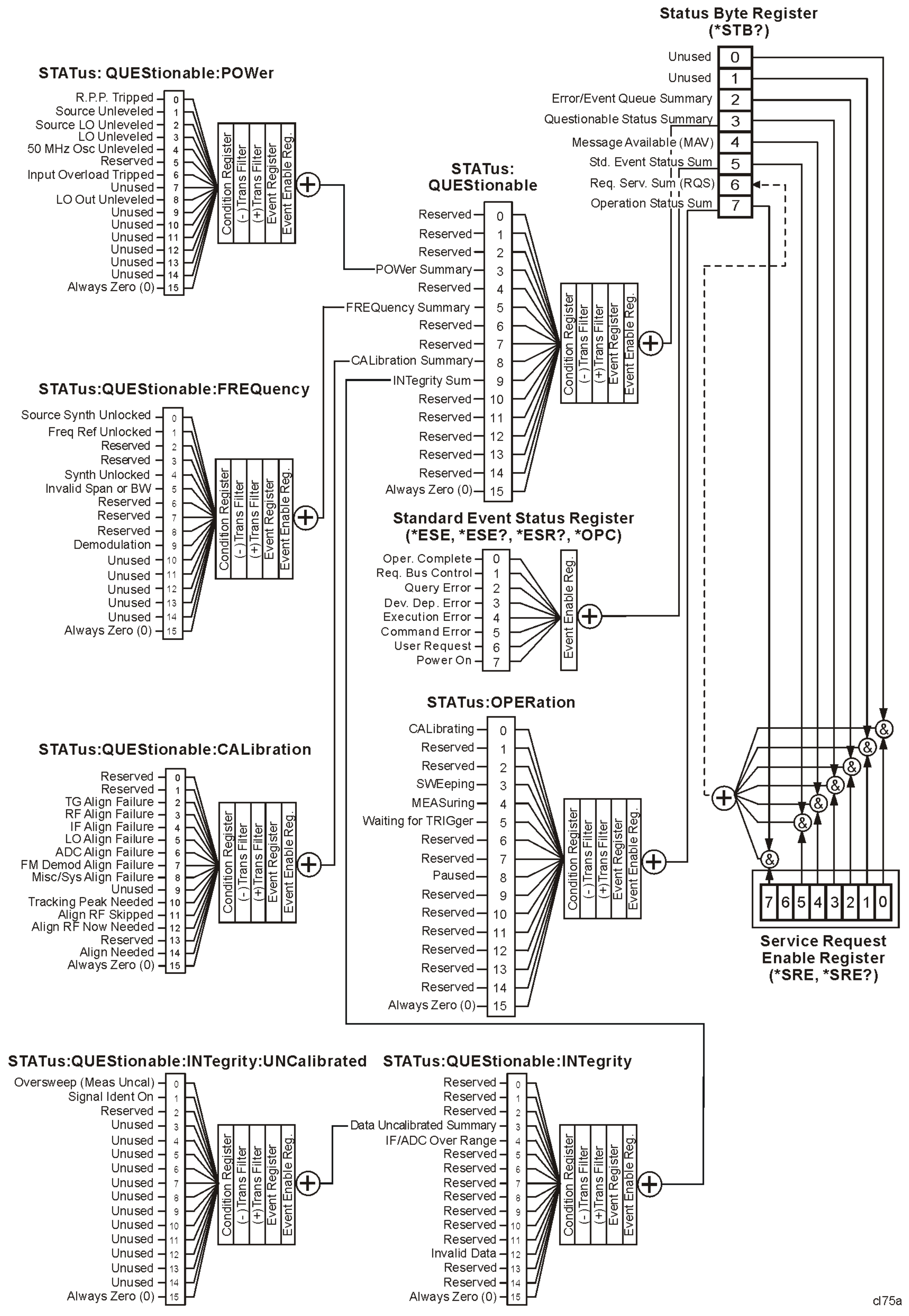
What are the Status Registers?

Refer toFigure 2-2, which shows the overall status register system in detail. Most status registers are composed of the five individual registers described below. One such status register in the figure is entitled “STATus: QUEStionable,” which is both the name of the register, and the SCPI command form used to access the register. From now on, the SCPI command form will be used when referring to the various registers. There are IEEE common SCPI commands noted under some register names in parenthesis. These commands are associated with those registers, and their effects are described under “How Do You Access the Status Registers?” in this chapter, and in the beginning of Chapter 5, “Language Reference,” in this guide.

Refer to the right-hand part of the STATus: QUEStionable register while reading the following register descriptions.

Condition Register	A condition register continuously monitors the hardware and firmware status of the analyzer. There is no latching or buffering for a condition register.
Negative Transition Filter	A negative transition filter specifies the bits in the condition register that will set corresponding bits in the event register when the condition bit changes from 1 to 0.
Positive Transition Filter	A positive transition filter specifies the bits in the condition register that will set corresponding bits in the event register when the condition bit changes from 0 to 1.
Event Register	An event register latches transition events from the condition register as specified by the positive and negative transition filters. Bits in the event register are latched, and once set, they remain set until cleared by either querying the register contents or sending the **`*CLS`** command.
Event Enable Register	An event enable register specifies the bits in the event register that can generate a summary bit. Summary bits are, in turn, used by the status byte register.

Figure 2-2 Overall Status Register System Diagram



Status registers (except for the status byte register and the standard event status register) consist of the registers whose contents can be used to produce status summary bits.

These summary bits are then manipulated as follows: The condition register passes summary bits to the negative and positive transition filters, after which they are stored in the event register. The contents of the event register are logically ANDed with the contents of the event enable register and the result is logically ORed to produce a status summary bit. The status summary bit is then passed to the status byte register either directly, or through the STATus: QUEStionable register. Next, the summary bits are logically ANDed with the contents of the service request enable register and the result is logically ORed to produce the request service (**`*RQS`**) bit in the status byte register.

How Do You Access the Status Registers?

There are two different methods to access the status registers:

- Common Commands Accesses and Controls
- Status Subsystem Commands

Common Command Access and Control

Most monitoring of the analyzer conditions is done at the highest level using the following IEEE common commands:

`*CLS` (clear status) clears the status byte by emptying the error queue and clearing all the event registers.

`*ESE,*ESE?` (event status enable) sets and queries the bits in the enable register part of the standard event status register.

`*ESR?` (event status register) queries and clears the standard event status register.

`*OPC` (operation complete) sets bit 0 in the standard event status register when all operations are complete.

`*SRE,*SRE?` (service request enable) sets and queries the value of the service request enable register.

`*STB?` (status byte) queries the value of the status byte register without erasing its contents.

Complete command descriptions are given in Chapter 5, “Language Reference,” under the subsection entitled “IEEE Common Commands” on page 193.

NOTE If you are using the status bits and the analyzer mode is changed, the status bits should be read, and any error conditions resolved, prior to switching modes. Error conditions that exist prior to switching modes cannot be detected using the condition registers after the mode change. This is true unless they recur after the mode change, although transitions of these conditions can be detected using the event registers.

Changing modes resets all SCPI status registers and mask registers to their power-on defaults. Hence any event or condition register masks must be re-established after a mode change. Also note that the power up status bit is set by any mode change, since that is the default state after power up.

Status Subsystem Commands

Individual status registers can be set and queried using the commands in the STATus subsystem in Chapter 5, "Language Reference," in this guide. There are two methods used to programmatically detect and manage error conditions or changes in analyzer status. Either method allows you to monitor one or more conditions. The two methods are:

- The Polling Method
- The Service Request (SRQ) Method

The Polling Method

In the polling method, the analyzer has a passive role. It only tells the controller that conditions have changed when the controller asks the right question. The polling method works well if you do not need to know about changes the moment they occur. This method is very efficient.

Use the polling method when either:

— your programming language/development environment does not support SRQ interrupts
— you want to write a simple, single-purpose program and don't want the added complexity of setting up an SRQ handler

The Service Request (SRQ) Method

The SRQ method allows timely communication of information without requiring continuous controller involvement. Using this method, the analyzer takes a more active role. It tells the controller when there has been a condition change without the controller asking. The SRQ method should be used if you must know immediately when a condition changes. This is in contrast to the polling method, which requires the program to repeatedly read the registers to detect a change.

Use the SRQ method when either:

— you need time-critical notification of changes
— you are monitoring more than one device which supports SRQs
— you need to have the controller do something else while the analyzer is making a measurement
— you can't afford the performance penalty inherent to polling

Using the Service Request (SRQ) Method

Your language, bus, and programming environment must be able to support SRQ interrupts (for example, using C and C++ with the GPIB). When you monitor a condition with the SRQ method, you must establish the following parameters:

1. Determine which bit monitors the condition.
2. Determine how that bit reports to the request service (RQS) bit of the status byte.
3. Send GPIB commands to enable the bits that monitor the condition and to enable the summary bits that report the condition to the RQS bit.
4. Enable the controller to respond to service requests.

When the condition changes, the analyzer sets the RQS bit and the GPIB SRQ line. The controller is informed of the change as soon as it occurs. The time the controller would otherwise have used to monitor the condition can now be used to perform other tasks. Your program also determines how the controller responds to the SRQ.

Generating a Service Request

Before using the SRQ method of generating a service request, first become familiar with how service requests are generated. Bit 6 of the status byte register is the request service summary (RQS) bit. The RQS bit is set whenever there is a change in the register bit that it has been configured to monitor. The RQS bit will remain set until the condition that caused it is cleared. It can be queried without erasing the contents using the **`*STB?`** command. Configure the RQS function using the **`*SRE`** command.

When a register set causes a summary bit in the status byte to change from 0 to 1, the analyzer can initiate the service request (SRQ) process. However, the process is only initiated if both of the following conditions are true:

- The corresponding bit of the service request enable register is also set to 1.
- The analyzer does not have a service request pending. (A service request is considered to be pending between the time the analyzer SRQ process is initiated, and the time the controller reads the status byte register.)

The SRQ process sets the GPIB SRQ line true. It also sets the status byte request service (RQS) bit to 1. Both actions are necessary to inform the controller that the analyzer requires service. Setting the SRQ line only informs the controller that some device on the bus requires service. Setting the RQS bit allows the controller to determine which device requires service.

If your program enables the controller to detect and respond to service requests, it should instruct the controller to perform a serial poll when the GPIB SRQ line is set true. Each device on the bus returns the contents of its status byte register in response to this poll. The device, whose RQS bit is set to 1, is the device that requested service.

NOTE When you read the analyzer status byte register with a serial poll, the RQS bit is reset to 0. Other bits in the register are not affected.

Restarting a measurement with the `:INITiate` command can cause the measuring bit to pulse low. A low pulse causes an SRQ if the status register is configured to SRQ upon end-of-measurement. To avoid this, perform the following steps:

1. Set `:INITiate:CONTinuous` off.
2. Set/enable the status registers.
3. Restart the measurement (send `:INITiate`).

Example of Monitoring Conditions Using the :STATus Command

Use the following steps to monitor a *specific* condition:

1. Determine which register contains the bit that reports the condition.
2. Send the unique SCPI query that reads that register.
3. Examine the bit to see if the condition has changed.
4. Act upon the cause of the condition and the SRQ to re-enable the method.

The examples below show how to use the `:STATus` command to perform the following tasks:

- Check the analyzer hardware and firmware status.
  Do this by querying the condition registers which continuously monitor status. These registers represent the current state of the analyzer. Bits in a condition register are updated in real time. When the condition monitored by a particular bit becomes true, the bit is set to 1. When the condition becomes false, the bit is reset to 0.
- Monitor a particular bit (condition), or bits.
  Once you have enabled a bit using the event enable register, the analyzer will monitor that particular bit. If the bit becomes true in the event register it will stay set until the event register is cleared. Querying the event register allows you to detect that this condition occurred even if the condition no longer exists. The event register can only be cleared by querying it or sending the `*CLS` command, which clears all event registers.
- Monitor a change in the condition of a particular bit, or bits.
  Once you have enabled a bit, the analyzer will monitor it for a change in its condition. The transition registers are preset to respond to the condition of going from 0 to 1 (positive transitions). This can be changed so that the selected bit is detected if it goes from 1 to 0 (negative transition), or if either transition occurs. Query the event register to determine whether or not a change has been made to how the transition registers respond. The event register can only be cleared by querying it or sending the `*CLS` command, which clears all event registers.

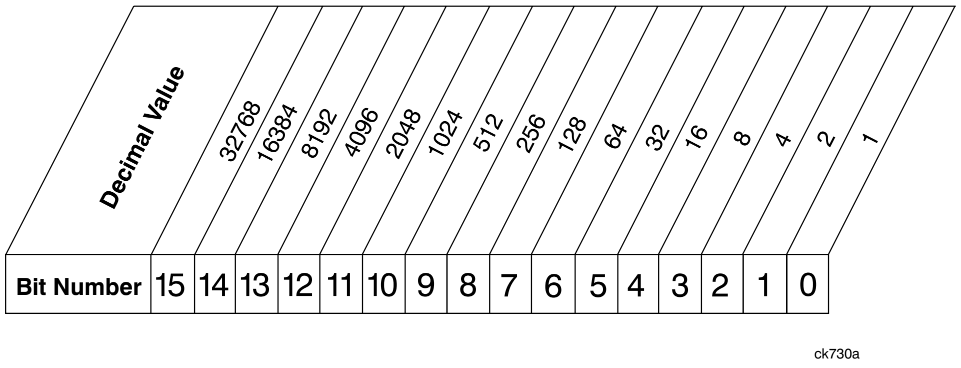
Setting and Querying the Status Register

See Figure 2-3. Each bit in a register is represented by a numerical value based on its location. This number is sent with the command to enable a particular bit. To enable more than one bit, send the sum of all of the bits involved.

For example, to enable bit 0 and bit 6 of the standard event status register, you would send the command **`*ESE 65`** (1 + 64).

The results of a query are evaluated in a similar way. If the **`*STB?`** command returns a decimal value of 140, (140 = 128 + 8 + 4) then bit 7 is true, bit 3 is true, and bit 2 is true.

Figure 2-3 Status Register Bit Values

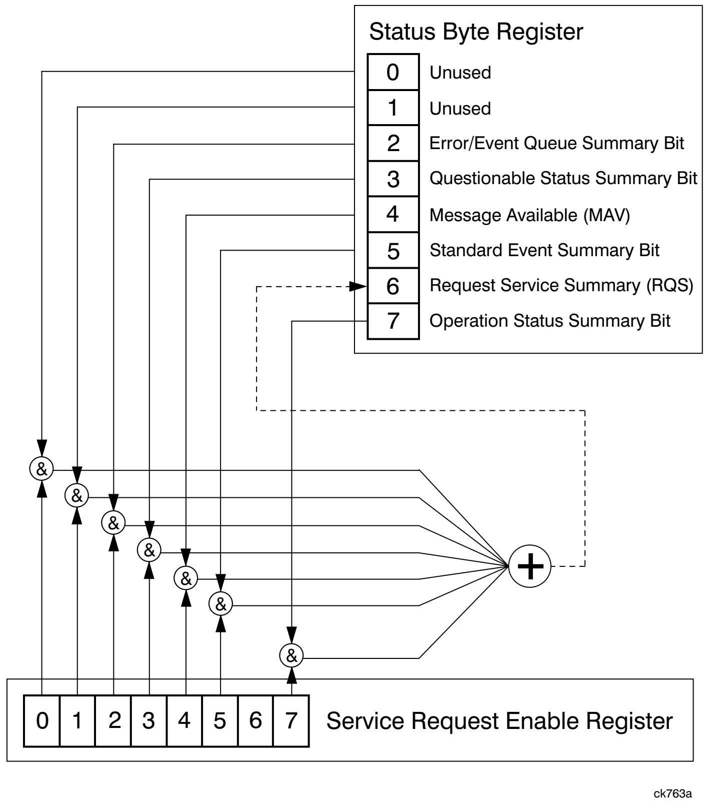


Details of Bits in All Registers

Refer to Figure 2-2. The rest of this chapter lists the bits in each register shown in the figure, along with descriptions of their purpose.

Status Byte Register

Figure 2-4 Status Byte Register Diagram


Status Byte Register
0 Unused
1 Unused
2 Error/Event Queue Summary Bit
3 Questionable Status Summary Bit
4 Message Available (MAV)
5 Standard Event Summary Bit
6 Request Service Summary (RQS)
7 Operation Status Summary Bit
&
+
0 1 2 3 4 5 6 7
Service Request Enable Register
ck763a

The status byte register contains the following bits:

Bit	Decimal Value	Description
0	1	**`Unused`**: This bit is always set to 0.
1	2	**`Unused`**: This bit is always set to 0.
2	4	**`Error/Event Queue Summery Bit`**: A 1 in this bit position indicates that the SCPI error queue is not empty. The SCPI error queue contains at least one error message.
3	8	**`Questionable Status Summary Bit`**: A 1 in this bit position indicates that the questionable status summary bit has been set. The questionable status event register can then be read to determine the specific condition that caused this bit to be set.
4	16	**`Message Available (MAV)`**: A 1 in this bit position indicates that the analyzer has data ready in the output queue. There are no lower status groups that provide input to this bit.

Bit	Decimal Value	Description
5	32	**`Standard Event Status Summary Bit`**: A 1 in this bit position indicates that the standard event status summary bit has been set. The standard event status register can then be read to determine the specific event that caused this bit to be set.
6	64	**`Request Service (RQS) Summery Bit`**: A 1 in this bit position indicates that the analyzer has at least one reason to report a status change. This bit is also called the master summary status bit (MSS).
7	128	**`Operation Status Summary Bit`**: A 1 in this bit position indicates that the operation status summary bit has been set. The operation status event register can then be read to determine the specific event that caused this bit to be set.

To query the status byte register, send the **`*STB`** command. The response will be the *decimal* sum of the bits that are set to 1. For example, if bit number 7 and bit number 3 are set to 1, the decimal sum of the 2 bits is 128 plus 8. So the decimal value 136 is returned.

Service Request Enable Register

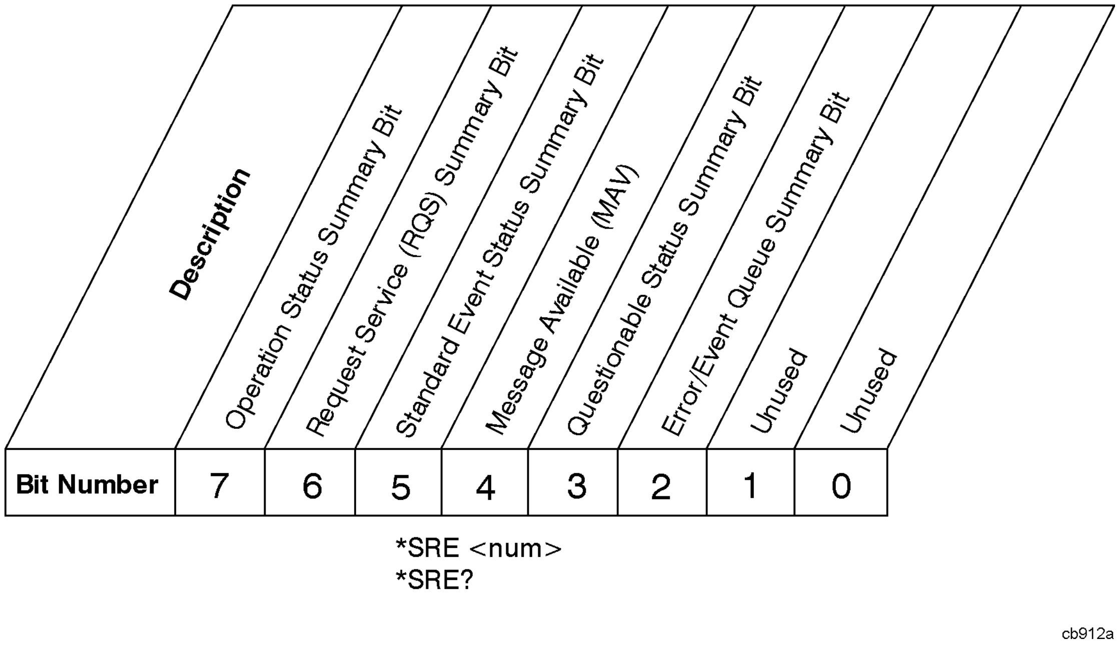
In addition to the status byte register, the status byte group also contains the service request enable register. The status byte service request enable register lets you choose which bits in the Status Byte Register will trigger a service request.

Send the **`*SRE <number>`** command (where **`<number>`** is the sum of the decimal values of the bits you want to enable plus the decimal value of bit 6). For example, assume that you want to enable bit 7 so that whenever the operation status summary bit is set to 1, it will trigger a service request. Send the **`*SRE 192`** (128 + 64) command. The **`*SRE?`** command returns the decimal value of the sum of the bits enabled previously with the **`*SRE <number>`** command.

NOTE You must always add 64 (the numeric value of RQS bit 6) to your numeric sum when you enable any bits for a service request.

The service request enable register contains the following bits:

Figure 2-5 **Service Request Enable Register**


Description
Operation Status Summary Bit
Request Service (RQS) Summary Bit
Standard Event Status Summary Bit
Message Available (MAV)
Questionable Status Summary Bit
Error/Event Queue Summary Bit
Unused
Unused
Bit Number
7
6
5
4
3
2
1
0
*SRE <num>
*SRE?
cb912a

NOTE The service request enable register presets to zeros (0).

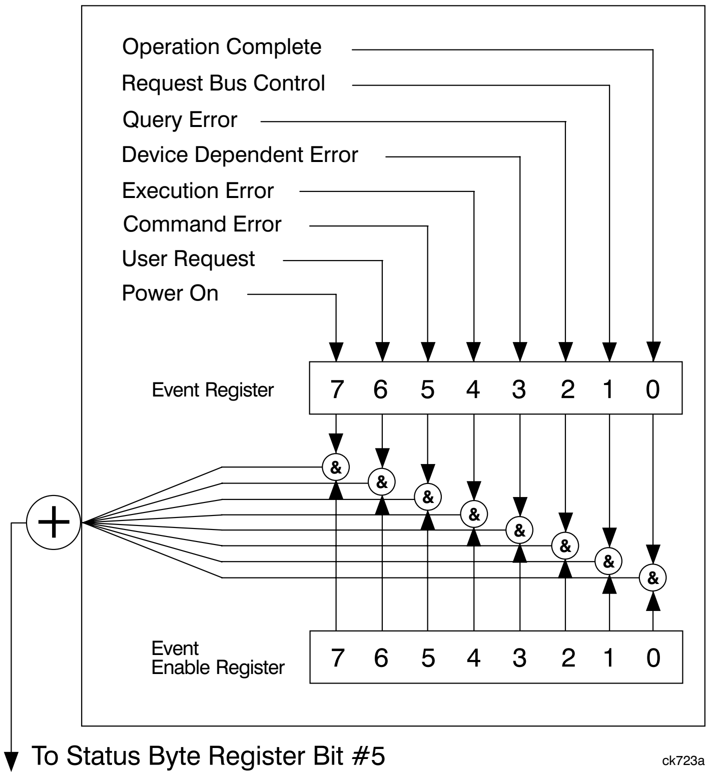
Standard Event Status Register

The standard event status register is used to determine the specific event that sets bit 5 in the status byte register. The standard event status register does *not* have negative and positive transition registers, nor a condition register. Use the IEEE common commands at the beginning of Chapter 5, “Language Reference,” in this guide to access the register.

To query the standard event status register, send the **`*ESR`** command. The response will be the *decimal* sum of the bits which are set to 1. For example, if bit number 7 and bit number 3 are set to 1, the decimal sum of the 2 bits is 128 plus 8. So the decimal value 136 is returned.

See “Setting and Querying the Status Register” on page 65 in this chapter for more information.

Figure 2-6 **Standard Event Status Register Diagram**



The standard event status register contains the following bits:

Bit	Decimal Value	Description
0	1	`Operation Complete`: A 1 in this bit position indicates that all operations were completed following execution of the `*OPC` command.
1	2	`Request Bus Control`: This bit is always set to 0. (The analyzer does not request control.)
2	4	`Query Error`: A 1 in this bit position indicates that a query error has occurred. Query errors have SCPI error numbers from –499 to –400.
3	8	`Device Dependent Error`: A 1 in this bit position indicates that a device dependent error has occurred. Device dependent errors have SCPI error numbers from –399 to –300 and 1 to 32767.
4	16	`Execution Error`: A 1 in this bit position indicates that an execution error has occurred. Execution errors have SCPI error numbers from –299 to –200.

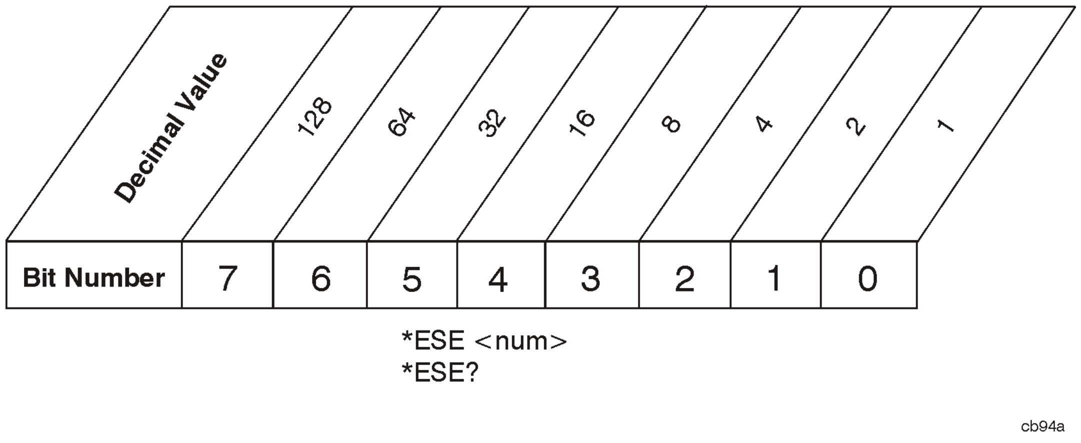
Bit	Decimal Value	Description
5	32	**`Command Error`**: A 1 in this bit position indicates that a command error has occurred. Command errors have SCPI error numbers from –199 to –100.
6	64	**`User Request Key (Local`**: A 1 in this bit position indicates that the **LOCAL** key has been pressed. This is true even if the analyzer is in local lockout mode.
7	128	**`Power On`**: A 1 in this bit position indicates that the analyzer has been turned off and then on.

Standard Event Status Event Enable Register

The event enable register (contained in the standard event status register) lets you choose which bits will set the summary bit (bit 5 of the status byte register) to 1. Send the **`*ESE <number>`** command (where **`<number>`** is the sum of the decimal values of the bits you want to enable).

For example, to enable bit 7 and bit 6 so that whenever either of those bits is set to 1, the standard event status summary bit of the status byte register will also be set to 1, send the **`*ESE 192`** (128 + 64) command. The **`*ESE?`** command returns the decimal value of the sum of the bits previously enabled with the **`*ESE <number>`** command.

Figure 2-7 **Standard Event Status Event Enable Register**



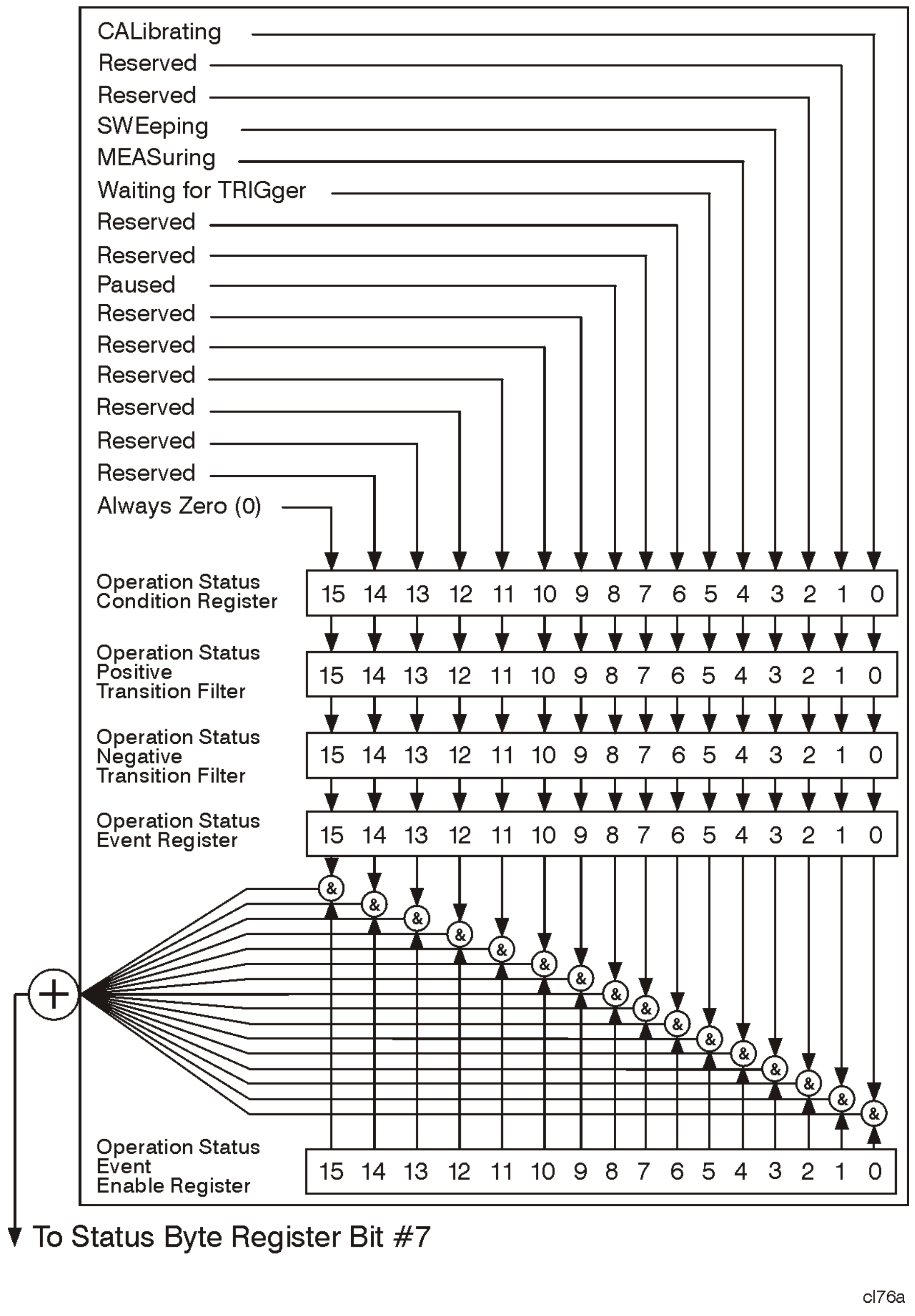
STATus:OPERation Register

The STATus:OPERation register is used to determine the specific event that sets bit 7 in the status byte register. This register also monitors the current measurement state and checks to see if the analyzer is performing any of these functions:

- measuring
- calibrating

- sweeping
- waiting for a trigger

Figure 2-8 **Status Operation Register Diagram**


CALibrating
Reserved
Reserved
SWEeping
MEASuring
Waiting for TRIGger
Reserved
Reserved
Paused
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved
Always Zero (0)
Operation Status Condition Register
Operation Status Positive Transition Filter
Operation Status Negative Transition Filter
Operation Status Event Register
Operation Status Event Enable Register
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
&
+
To Status Byte Register Bit #7
cl76a

The STATus:OPERation condition register contains the following bits:

Bit	Decimal Value	Description
0	0	`Calibrating`: A 1 in this bit position indicates that the analyzer is performing a self-calibration.
1	2	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
2	4	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
3	8	`Sweeping`: A 1 in this bit position indicates that a sweep is in progress.
4[a]	16	`Measuring`: A 1 in this bit position indicates that a measurement is in progress.
5[a]	32	`Waiting for Trigger`: A 1 in this bit position indicates that a measurement is in a “wait for trigger” state.
6	64	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
7	128	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
8	256	`Paused`: A 1 in this bit position indicates that the instrument is in the paused state of the measurement.
9	512	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
10	1024	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
11	2048	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
12	4096	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
13	8192	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
14	16384	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
15	32768	`Always Zero (0)`

a. The description of this bit refers to any measurement under the **MEASURE** key.

STATus:OPERation Condition and Event Enable Registers

The STATus:OPERation condition register continuously monitors the hardware and firmware status of the analyzer, and is read-only. To query the register, send the `:STATus:OPERation:CONDition?` command. The response will be the *decimal* sum of the bits that are set to 1. For example, if bit number 9 and bit number 3 are set to 1, the decimal sum of the 2 bits is 512 plus 8. So the decimal value 520 is returned.

The transition filter specifies which types of bit state changes in the condition register will set corresponding bits in the event register. The changes may be positive (from 0 to 1) or negative (from 1 to 0). Send the **`:STATus:OPERation:NTRansition <num>`** (negative transition) command or the **`:STATus:OPERation:PTRansition <num>`** (positive transition) command (where **`<num>`** is the sum of the decimal values of the bits you want to enable).

The STATus:OPERation event register latches transition events from the condition register as specified by the transition filters. Event registers are destructive read-only data. Reading data from an event register will clear the content of that register. To query the event register, send the **`:STATus:OPERation:[:EVENt]?`** command.

The STATus:OPERation event enable register lets you choose the bits that will set the operation status summary bit (bit 7) of the status byte register to 1. Send the **`:STATus:OPERation:ENABle <num>`** command where **`<num>`** is the sum of the decimal values of the bits you want to enable.

For example, to enable bit 9 and bit 3 (so that whenever either bit 9 or 3 is set to 1, the operation status summary bit of the status byte register will be set to 1), send the `:STATus:OPERation:ENABle 520` (512 + 8) command. The `:STATus:OPERation:ENABle?` command returns the decimal value of the sum of the bits previously enabled with the `:STATus:OPERation:ENABle <num>` command.

STATus:QUEStionable Registers

STATus:QUEStionable registers monitor the overall analyzer condition. They are accessed with the **`:STATus:OPERation`** and **`:STATus:QUEStionable`** commands in the **`:STATus`** command subsystem.

The STATus:QUEStionable registers also monitor the analyzer to see if there are any questionable events that occurred. These registers look for anything that may cause an error or that may induce a faulty measurement. Signs of a faulty measurement include the following:

- hardware problems
- out of calibration situations
- unusual signals

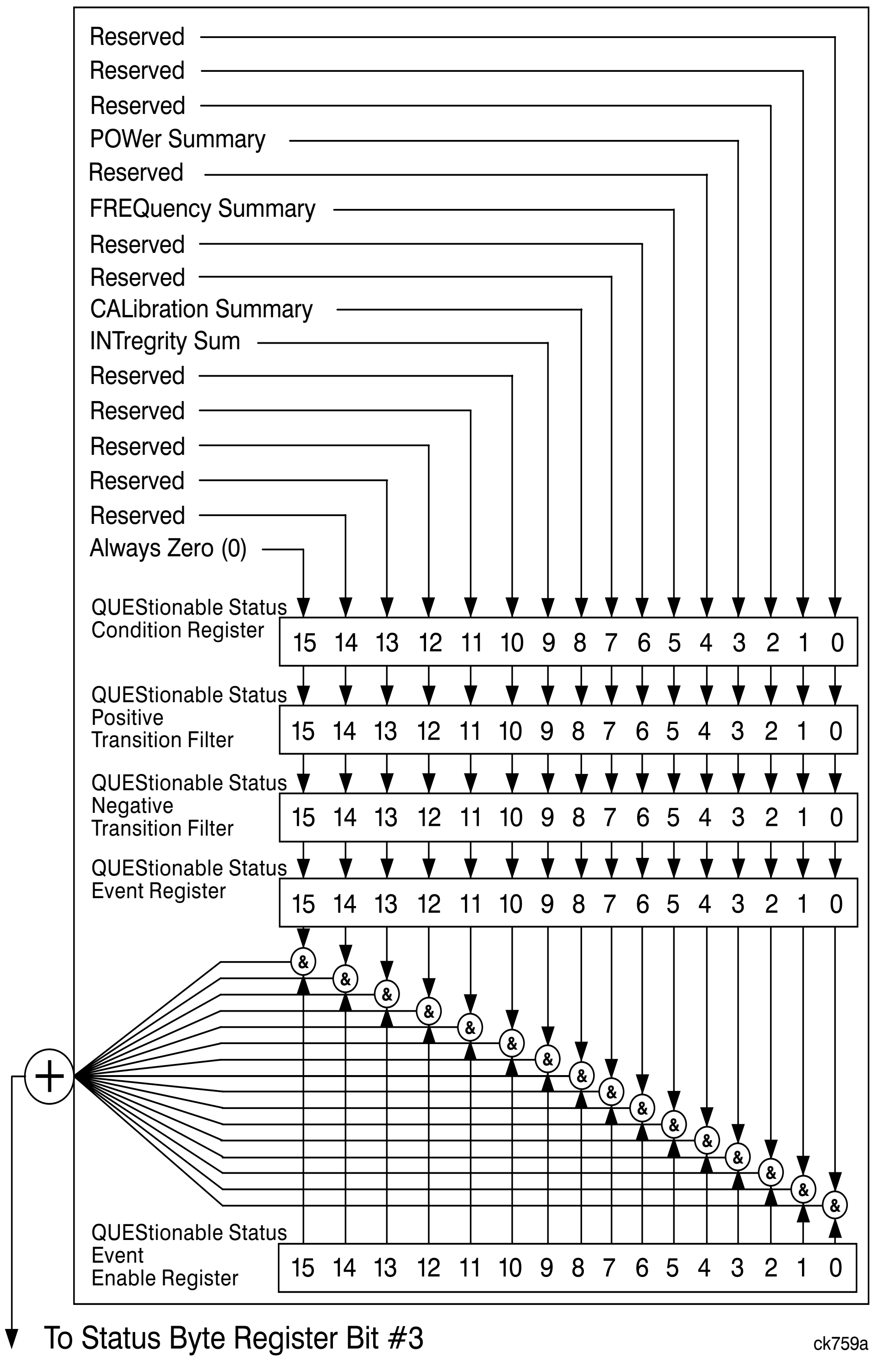
NOTE All bits are summary bits from lower-level event registers. (For a general diagram of the STATus:QUEStionable register, see Figure 2-9.)

A Questionable Status condition register query response will be the decimal sum of the bits which are set to 1. For example, if bit number 9 and bit number 3 are set to 1, the decimal sum of the 2 bits is 512 plus 8. So the decimal value 520 is returned.

The transition filter specifies which types of bit state changes in the condition register will set corresponding bits in the event register. The changes may be positive (from 0 to 1) or negative (from 1 to 0). Send the command `:STATus:QUEStionable:NTRansition <num>` (negative transition) or `:STATus:QUEStionable:PTRansition <num>` (positive transition) where `<num>` is the sum of the decimal values of the bits you want to enable.

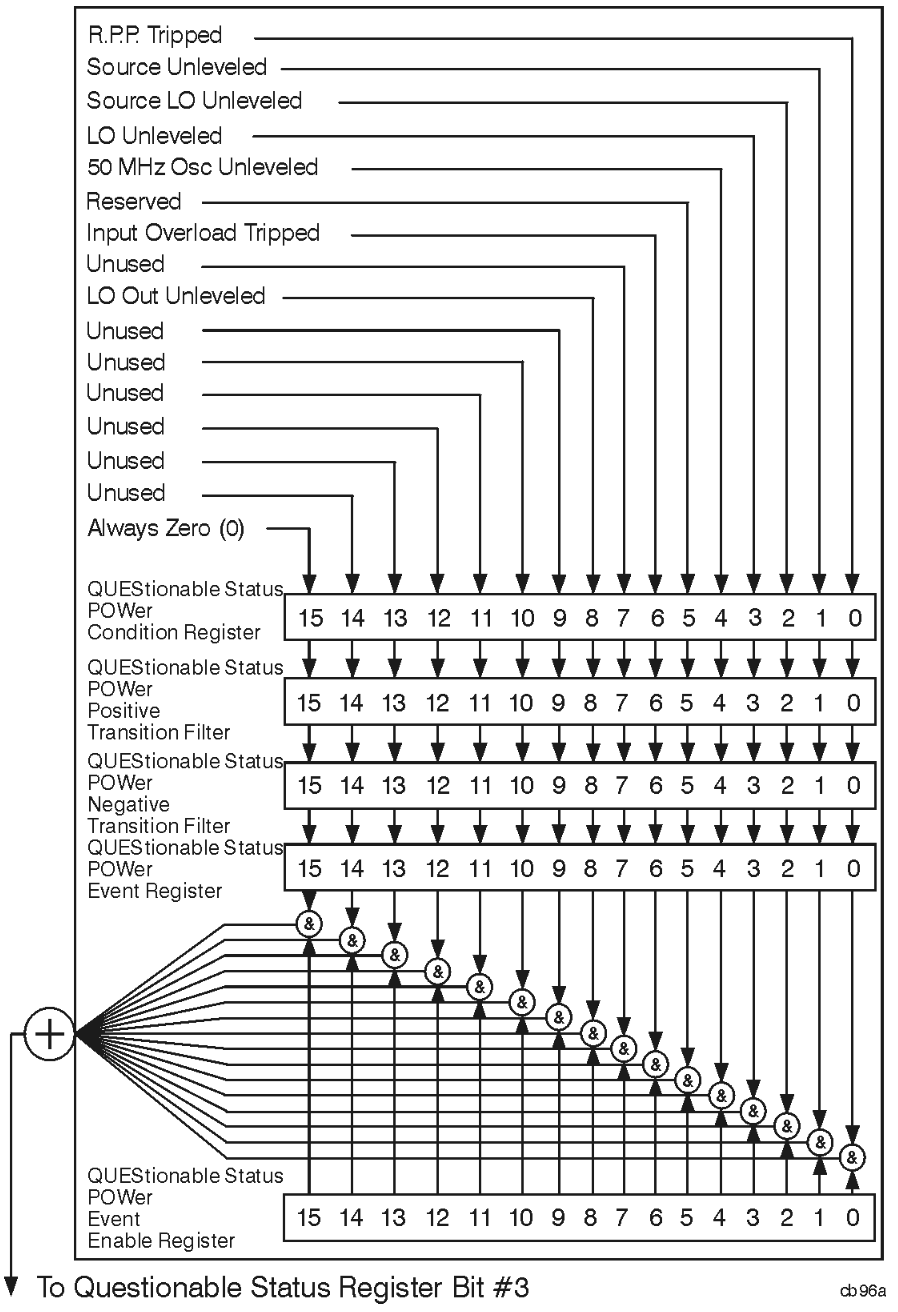
The Questionable Status event register latches transition events from the condition register as specified by the transition filters. Event registers are destructive read-only. Reading data from an event register will clear the content of that register. To query the event register, send the command `:STATus:QUEStionable[:EVENt]?`

Figure 2-9 **Status Questionable Register Diagram**



STATus:QUEStionable:POWer Register

Figure 2-10 Questionable Status Power Register Diagram


R.P.P. Tripped
Source Unleveled
Source LO Unleveled
LO Unleveled
50 MHz Osc Unleveled
Reserved
Input Overload Tripped
Unused
LO Out Unleveled
Unused
Unused
Unused
Unused
Unused
Unused
Always Zero (0)
QUEStionable Status POWer Condition Register
QUEStionable Status POWer Positive Transition Filter
QUEStionable Status POWer Negative Transition Filter
QUEStionable Status POWer Event Register
QUEStionable Status POWer Event Enable Register
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
&
To Questionable Status Register Bit #3
cb96a

Bit descriptions in the Questionable Status Power Condition Register are given in the following table.

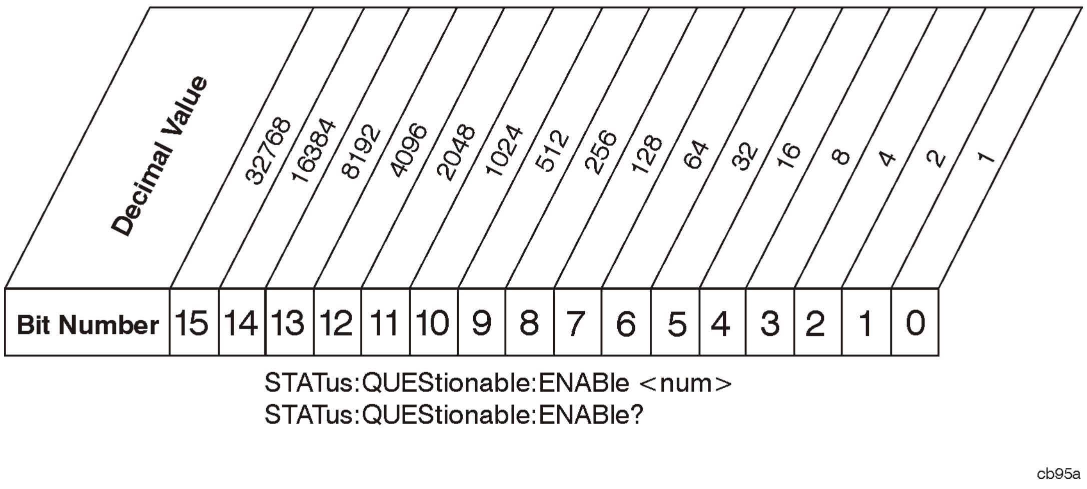
Bit	Decimal Value	Description
0	0	`R.P.P Tripped`: A 1 in this bit position indicates that the reverse power protection is tripped (Agilent model E7401A only). Reverse power protection is “overload” protection for the tracking generator.
1	2	`Source Unleveled`: A 1 in this bit position indicates that the source (tracking generator) output is unleveled.
2	4	`Source LO Unleveled`: A 1 in this bit position indicates that the local oscillator (LO) in the source (tracking generator) is unleveled.
3	8	`LO Unleveled`: A 1 in this bit position indicates that the analyzer local oscillator (LO) is unleveled.
4	16	`50 MHz Osc Unleveled`: A 1 in this bit position indicates that the 50 MHz amplitude reference signal is unleveled.
5	32	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
6	64	`Input Overload Tripped`: A 1 in this bit position indicates that the input overload protection is tripped (Agilent EMC model E7401A only).
7	128	`Unused`: This bit is always set to 0.
8	256	`LO Out Unleveled`: A 1 in this bit position indicates that the first local oscillator (LO) output is unleveled.
9	512	`Unused`: This bit is always set to 0.
10	1024	`Unused`: This bit is always set to 0.
11	2048	`Unused`: This bit is always set to 0.
12	4096	`Unused`: This bit is always set to 0.
13	8192	`Unused`: This bit is always set to 0.
14	16384	`Unused`: This bit is always set to 0.
15	32768	`Always Zero (0)`: This bit is always set to 0.

Questionable Status Event Enable Register

The Questionable Status Event Enable Register lets you choose which bits in the Questionable Status Event Register will set the summary bit (bit 3 of the Status Byte Register) to 1. Send the command `:STATus:QUEStionable:ENABle <num>` where `<num>` is the sum of the decimal values of the bits you want to enable.

For example, to enable bit 9 and bit 3 so that whenever either of those bits is set to 1, the Questionable Status Summary bit of the Status Byte Register will be set to 1, send the command `:STAT:QUES:ENAB 520` (512 + 8). The command `:STATus:QUEStionable:ENABle?` returns the decimal value of the sum of the bits previously enabled with the `:STATus:QUEStionable:ENABle <num>` command.

Figure 2-11 Questionable Status Event Enable Register



Bit descriptions in the Status Questionable Condition Register are given in the following table.

Bit	Decimal Value	Description
0	1	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
1	2	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
2	4	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
3	8	**POWer Summary**: This is the summary bit for the Questionable Power Status Register.
4	16	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
5	32	**FREQuency Summary**: This is the summary bit for the Questionable Frequency Status Register.
6	64	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.

Bit	Decimal Value	Description
7	128	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
8	256	**CALibration Summary**: This is the summary bit for the Questionable Calibration Status Register.
9	512	**INTegrity Sum**: This is the summary bit for the Questionable Integrity Status Register.
10	1024	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
11	2048	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
12	4096	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
13	8192	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
14	16384	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
15	32768	**Always Zero (0)**

Questionable Status Power Condition and Event Registers

The Questionable Status Power Condition Register continuously monitors output power status of the analyzer. Condition registers are read-only. To query the condition register, send the command **:STATus:QUEStionable:POWer:CONDition?** The response will be the decimal sum of the bits which are set to 1.

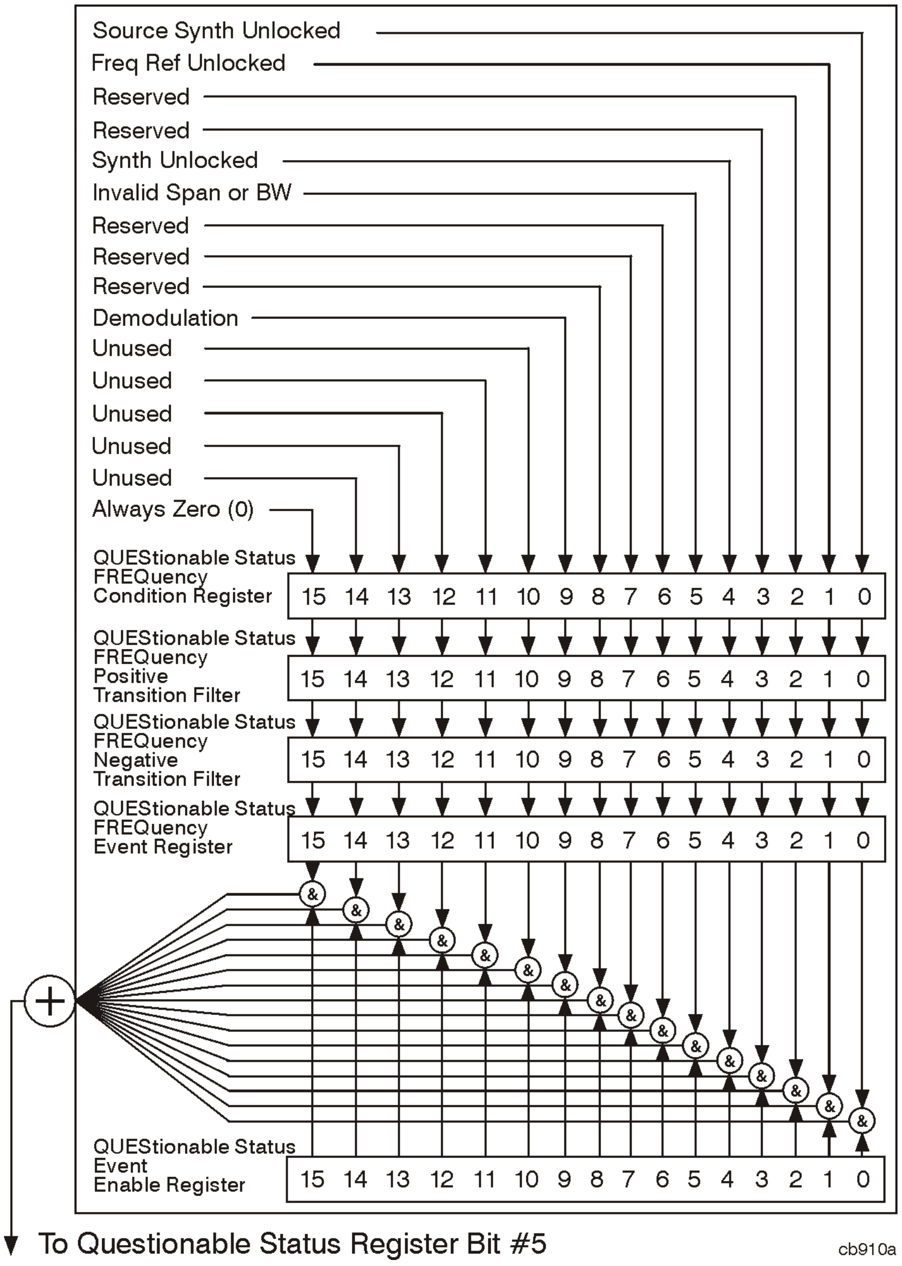
The transition filter specifies which types of bit state changes in the condition register will set corresponding bits in the event register. The changes may be positive (from 0 to 1) or negative (from 1 to 0). Send the command **:STATus:QUEStionable:POWer:NTRansition <num>** (negative transition) or **:STATus:QUEStionable:POWer:PTRansition <num>** (positive transition) where **<num>** is the sum of the decimal values of the bits you want to enable.

The Questionable Status Power Event Register latches transition events from the condition register as specified by the transition filters. Event registers are destructive read-only. Reading data from an event register will clear the content of that register. To query the event register, send the command **:STATus:QUEStionable:POWer[:EVENt]?**

See "Questionable Status Event Enable Register" on page 77 for an explanation of how to set the summary bit using the event enable register. In this case, use the command **:STATus:QUEStionable:POWer:ENABle <num>**.

STATus:QUEStionable:FREQuency Register

Figure 2-12 Questionable Status Frequency Register Diagram


Source Synth Unlocked
Freq Ref Unlocked
Reserved
Reserved
Synth Unlocked
Invalid Span or BW
Reserved
Reserved
Reserved
Demodulation
Unused
Unused
Unused
Unused
Unused
Always Zero (0)
QUEStionable Status FREQuency Condition Register
QUEStionable Status FREQuency Positive Transition Filter
QUEStionable Status FREQuency Negative Transition Filter
QUEStionable Status FREQuency Event Register
QUEStionable Status Event Enable Register
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
&
To Questionable Status Register Bit #5
cb910a

Bit descriptions in the Questionable Status Frequency Condition Register are given in the following table.

Bit	Decimal Value	Description
0	0	**Source Synth Unlocked**: A 1 in this bit position indicates that the synthesizer in the source (tracking generator) is unlocked.
1	2	**Freq Ref Unlocked**: A 1 in this bit position indicates that the analyzer frequency reference is unlocked.
2	4	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
3	8	**Reserved**: This bit is not used by the analyzer, but is for future use with other Agilent products.
4	16	**Synth Unlocked**: A 1 in this bit position indicates that the analyzer synthesizer is unlocked.
5	32	**Invalid Span or BW**: A 1 in this bit position indicates an invalid span or bandwidth during frequency count.
6	64	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
7	128	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
8	256	**Reserved**: This bit is not used by the analyzer, but are for future use with other Agilent products.
9	512	**Demodulation**: A 1 in this bit position indicates an invalid span during FM Demodulation and Listen measurements.
10	1024	**Unused**: This bit is always set to 0.
11	2048	**Unused**: This bit is always set to 0.
12	4096	**Unused**: This bit is always set to 0.
13	8192	**Unused**: This bit is always set to 0.
14	16384	**Unused**: This bit is always set to 0.
15	32768	**Always Zero (0)**: This bit is always set to 0.

Questionable Status Frequency Condition and Event Enable Registers

The Questionable Status Frequency condition register continuously monitors output frequency status of the analyzer. Condition registers are read-only. To query the condition register, send the command `:STATus:QUEStionable:FREQuency:CONDition?` The response will be the *decimal* sum of the bits which are set to 1.

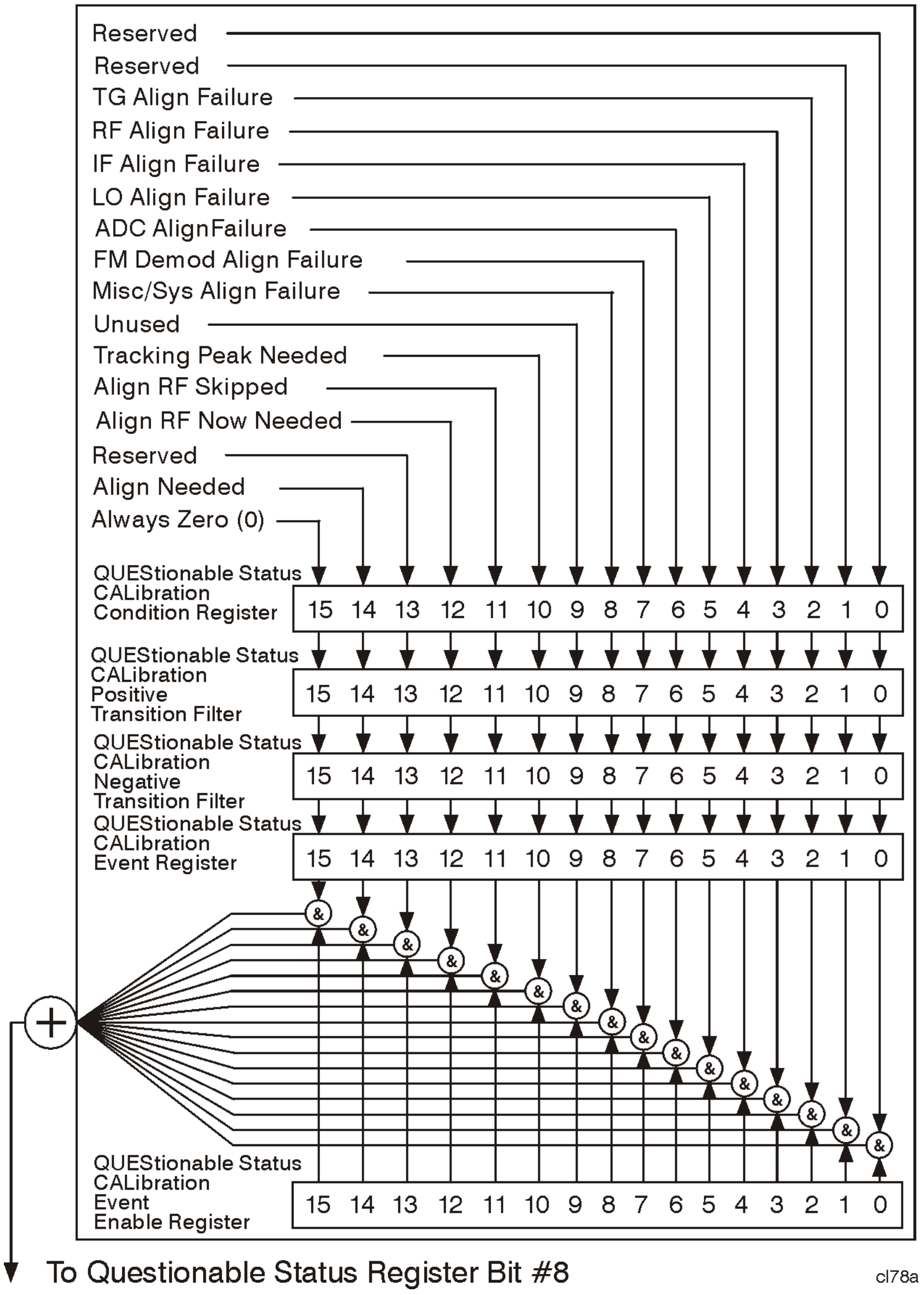
The negative and positive transition filters specify which types of bit state changes in the condition register will set corresponding bits in the event register. The changes may be positive (from 0 to 1) or negative (from 1 to 0). Send the command **`:STATus:QUEStionable:FREQuency :NTRansition <num>`** (negative transition) or **`:STATus:QUEStionable :FREQuency:PTRansition <num>`** (positive transition) where **`<num>`** is the sum of the decimal values of the bits you want to enable.

The Questionable Status Frequency Event register latches transition events from the condition register as specified by the transition filters. Event registers are destructive read-only. Reading data from an event register will clear the content of that register. To query the event register, send the command **`:STATus:QUEStionable:FREQuency [:EVENt]?`**

See "Questionable Status Event Enable Register" on page 77 for an explanation of how to set the summary bit using the event enable register. In this case, use the command **`:STATus:QUEStionable:FREQ:ENABle <num>`**.

STATus:QUEStionable:CALibration Register

Figure 2-13 Questionable Status Calibration Register Diagram



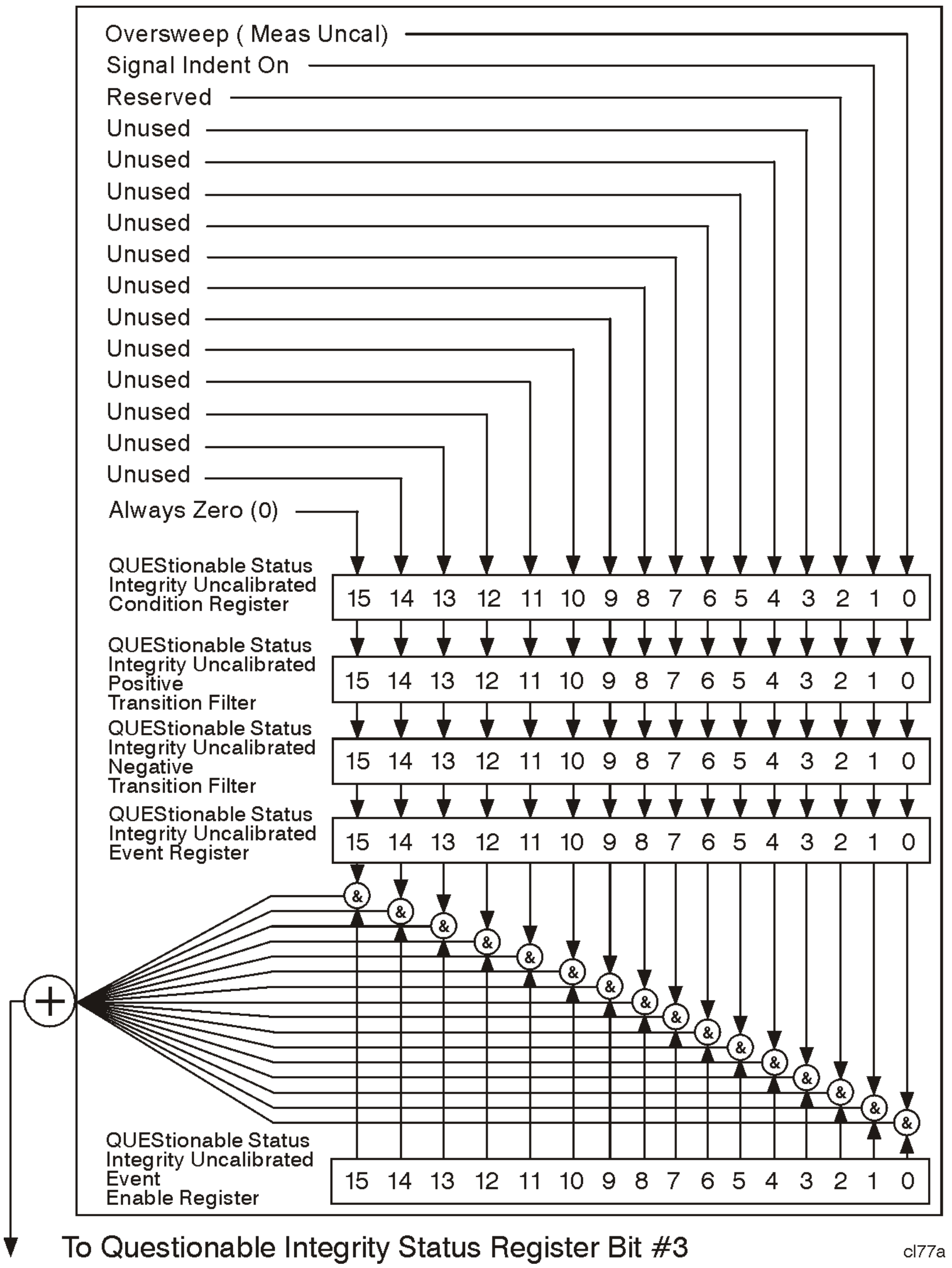
Bit descriptions in the Questionable Status Calibration Condition Register are given in the following table.

Bit	Decimal Value	Description
0	0	**`Reserved`**: This bit is not used by the analyzer, but are for future use with other Agilent products.
1	2	**`Reserved`**: This bit is not used by the analyzer, but are for future use with other Agilent products.
2	4	**`TG Align Failure`**: A 1 in this bit position indicates that a failure has occurred while trying to align the tracking generator (TG).
3	8	**`RF Align Failure`**: A 1 in this bit position indicates that a failure has occurred while trying to align the RF section.
4	16	**`IF Align Failure`**: A 1 in this bit position indicates that a failure has occurred while trying to align the IF section.
5	32	**`LO Align Failure`**: A 1 in this bit position indicates that a failure has occurred while trying to align the local oscillator (LO).
6	64	**`ADC Align Failure`**: A 1 in this bit position indicates that a failure has occurred while trying to align the analog-to-digital converter (ADC).
7	128	**`FM Demod Align Failure`**: A 1 in this bit position indicates that a failure has occurred while trying to align the FM demodulation circuitry.
8	256	**`Misc/Sys Align Failure`**: A 1 in this bit position indicates that a failure has occurred while trying to align the quasi-peak detector.
9	512	**`Unused`**: This bit is always set to 0.
10	1024	**`Tracking Peak Needed`**: A 1 in this bit position indicates that a tracking peak needs to be performed (the tracking generator is in operation). (Agilent EMC models E7402A, E7403A, E7404A, and E7405A, with Option 1DN, Tracking Generator, only).
11	2048	**`Align RF Skipped`**: A 1 in this bit position indicates that the alignment of the RF section was skipped, perhaps due to an external 50 MHz signal having been detected.
12	4096	**`Align RF Now Needed`**: A 1 in this bit position indicates that the RF section needs to be aligned.
13	8192	**`Reserved`**: This bit is not used by the analyzer, but is for future use with other Agilent products.

Bit	Decimal Value	Description
14	16384	`Align Needed`: A 1 in this bit position indicates that a full alignment is needed, perhaps due to a large temperature change having been detected with auto align off, or due to default data being used.
15	32768	`Always Zero (0)`: This bit is always set to 0.

STATus:QUEStionable:INTegrity:UNCalibrated Register

Figure 2-14 Questionable Status Integrity Uncalibrated Register Diagram


Oversweep (Meas Uncal)
Signal Indent On
Reserved
Unused
Unused
Unused
Unused
Unused
Unused
Unused
Unused
Unused
Unused
Unused
Unused
Always Zero (0)
QUEStionable Status Integrity Uncalibrated Condition Register
QUEStionable Status Integrity Uncalibrated Positive Transition Filter
QUEStionable Status Integrity Uncalibrated Negative Transition Filter
QUEStionable Status Integrity Uncalibrated Event Register
QUEStionable Status Integrity Uncalibrated Event Enable Register
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
&
+
To Questionable Integrity Status Register Bit #3
cl77a

Bit descriptions in the Questionable Status Integrity Uncalibrated Condition Register are given in the following table.

Bit	Decimal Value	Description
0	0	`Oversweep (Meas Uncal)`: A 1 in this position indicates that the analyzer is in a state that could lead to uncalibrated measurements. This is typically caused by sweeping too fast for the current combination of span, resolution bandwidth, and video bandwidth. Auto coupling may resolve this problem.
1	2	`Signal Ident ON`: A 1 in this bit position indicates that amplitude measurements may be in error due to signal identification routines being active. Amplitude accuracy is degraded when signal identification is active.
2	4	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
3	8	`Unused`: This bit is always set to 0.
4	16	`Unused`: This bit is always set to 0.
5	32	`Unused`: This bit is always set to 0.
6	64	`Unused`: This bit is always set to 0.
7	128	`Unused`: This bit is always set to 0.
8	256	`Unused`: This bit is always set to 0.
9	512	`Unused`: This bit is always set to 0.
10	1024	`Unused`: This bit is always set to 0.
11	2048	`Unused`: This bit is always set to 0.
12	4096	`Unused`: This bit is always set to 0.
13	8192	`Unused`: This bit is always set to 0.
14	16384	`Unused`: This bit is always set to 0.
15	32768	`Always Zero (0)`: This bit is always set to 0.

Questionable Status Calibration Condition and Event Enable Registers

The Questionable Status Calibration condition register continuously monitors the calibration status of the analyzer. Condition registers are read-only. To query the condition register, send the command `:STATus:QUEStionable:CALibration:CONDition?` The response will be the decimal sum of the bits which are set to 1.

The transition filter specifies which types of bit state changes in the condition register will set corresponding bits in the event register. The changes may be positive (from 0 to 1) or negative (from 1 to 0). Send the command

:STATus:QUEStionable:CALibration:NTRansition <num> (negative transition) or **:STATus:QUEStionable:CALibration:PTRansition <num>** (positive transition) where **<num>** is the sum of the decimal values of the bits you want to enable.

The Questionable Status Calibration Event register latches transition events from the condition register as specified by the transition filters. Event registers are destructive read-only. Reading data from an event register will clear the content of that register. To query the event register, send the command **:STATus:QUEStionable:CALibration [:EVENt]?**

See "Questionable Status Event Enable Register" on page 77 for an explanation of how to set the summary bit using the event enable register. In this case, use the command **:STATus:QUEStionable:CALibration:ENABle <num>**.

Questionable Status Integrity Uncalibrated Condition and Event Enable Registers

The Questionable Status Integrity Uncalibrated Condition Register continuously monitors the calibration status of the analyzer. Condition registers are read-only. To query the condition register, send the command **:STATus:QUEStionable:INTegrity:UNCalibrated:CONDition?** The response will be the decimal sum of the bits which are set to 1.

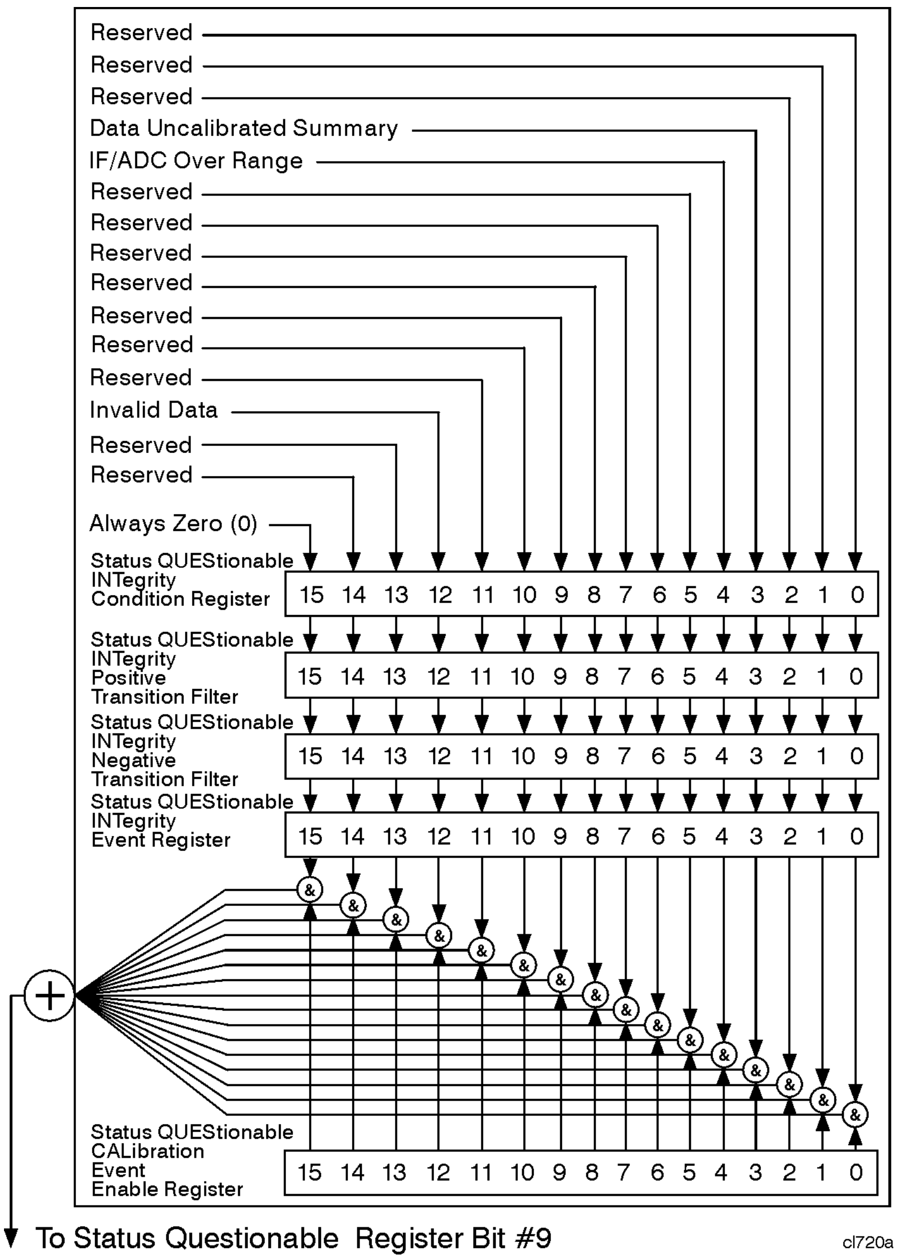
The transition filter specifies which types of bit state changes in the condition register will set corresponding bits in the event register. The changes may be positive (from 0 to 1) or negative (from 1 to 0). Send the command **:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition <num>** (negative transition) or **:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition <num>** (positive transition) where **<num>** is the sum of the decimal values of the bits you want to enable.

The Questionable Status Integrity Uncalibrated Event Register latches transition events from the condition register as specified by the transition filters. Event registers are destructive read-only. Reading data from an event register will clear the content of that register. To query the event register, send the command **:STATus:QUEStionable:INTegrity:UNCalibrated[:EVENt]?**

See "Questionable Status Event Enable Register" on page 77 for an explanation of how to set the summary bit using the event enable register. In this case, use the command **:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle <num>**.

STATus:QUEStionable:INTegrity Register

Figure 2-15 Questionable Status Integrity Register Diagram



Bit descriptions in the Questionable Status Integrity Condition Register are given in the following table.

Bit	Decimal Value	Description
0	1	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
1	2	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
2	4	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
3	8	`Data Uncalibrated Summary`: This is the summary bit for the Questionable Status Integrity Uncalibrated Register.
4	16	`IF/ADC Over Range`: The signal input level is too high, causing the analyzer analog-to-digital converter (ADC) range to be exceeded. This may occur with resolution bandwidths less than or equal to 300 Hz in zero span.
5	32	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
6	64	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
7	128	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
8	256	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
9	512	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
10	1024	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
11	2048	`Reserved`: This bit is not used by the analyzer, but are for future use with other Agilent products.
12	4096	`Invalid Data`: A 1 in this bit position indicates that the present trace data does not reflect the existing analyzer state. Trigger a new sweep and/or measurement.
13	8192	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
14	16384	`Reserved`: This bit is not used by the analyzer, but is for future use with other Agilent products.
15	32768	`Always Zero (0)`: This bit is always set to 0.

Questionable Status Integrity Event Condition and Enable Registers

The Questionable Status Integrity Condition Register continuously monitors the calibration status of the analyzer. Condition registers are read-only. To query the condition register, send the command `:STATus:QUEStionable:INTegrity:CONDition?` The response will be the decimal sum of the bits which are set to 1.

The transition filter specifies which types of bit state changes in the condition register will set corresponding bits in the event register. The changes may be positive (from 0 to 1) or negative (from 1 to 0). Send the command **`:STATus:QUEStionable:INTegrity:NTRansition <num>`** (negative transition) or **`:STATus:QUEStionable:INTegrity:PTRansition <num>`** (positive transition) where **`<num>`** is the sum of the decimal values of the bits you want to enable.

The Questionable Status Integrity Event Register latches transition events from the condition register as specified by the transition filters. Event registers are destructive read-only. Reading data from an event register will clear the content of that register. To query the event register, send the command **`:STATus:QUEStionable:INTegrity[:EVENt]?`**

See “Questionable Status Event Enable Register” on page 77 for an explanation of how to set the summary bit using the event enable register. In this case, use the command **`:STATus:QUEStionable:INTegrity:ENABle <num>`**.

3 Programming Examples

This chapter includes examples of how to program the analyzer using the analyzer SCPI programming commands. Twelve examples are written for an analyzer with an GPIB interface (Option A4H). Three examples are written for an analyzer with an RS-232 interface (Option 1AX).

List of Programming Examples

The programming examples included in this chapter are:

"Using Marker Peak Search and Peak Excursion"

"Using Marker Delta Mode and Marker Minimum Search"

"Performing Internal Self-alignment"

"Reading Trace Data using ASCII Format (GPIB)"

"Reading Trace Data Using 32-bit Real Format (GPIB)"

"Reading Trace Data Using ASCII Format (RS-232)"

"Reading Trace Data Using 32-bit Real Format (RS-232)"

"Using Limit Lines"

"Measuring Noise"

"Entering Amplitude Correction Data"

"Status Register–Determine When a Measurement is Done"

"Determine if an Error has Occurred"

"Measuring Harmonic Distortion (GPIB)"

"Measuring Harmonic Distortion (RS-232)"

"Making Faster Measurements (multiple measurements)"

Programming Examples System Requirements

These examples were written for use on an IBM compatible PC configured as follows:

- Pentium processor
- Windows 95®[1] or Windows NT® 4.0 operating system
- C programming language
- National Instruments GPIB interface card (for analyzers with Option A4H)
- National Instruments VISA Transition Libraries (VTL)
- COM1 serial port configured as follows (for analyzers with Option 1AX)
    - — 9600 baud
    - — 8 data bits
    - — 1 stop bit
    - — no parity bits
    - — hardware flow control

A HP/Agilent 82341C card may be substituted for the National Instruments GPIB, and the HP VISA libraries may be substituted for the National Instruments VISA Transition Libraries. If substitutions are made, the subdirectories for the include and library files will be different than those listed in the following paragraphs. Refer to the documentation for your interface card and the VISA libraries for details.

1. Microsoft® is a U.S. registered trademark of Microsoft Corporation.

C Programming Examples using VTL

The programming examples that are provided in this guide are written using the C programming language and the VTL (VISA transition library). This section includes some basic information about programming in the C language. Refer to your C programming language documentation for more details. (This information is taken from the manual “HP VISA Transition Library”, HP part number E2090-90026.) If you are using the National Instruments VISA library, most of this information will still apply, but the include and library files will be in different subdirectories. Also, this information assumes a computer running a Windows 95 operating system with an HP/Agilent 82341C GPIB interface card is being used. The following topics are included:

“Typical Example Program Contents” on page 96.
“Linking to VTL Libraries” on page 97.
“Compiling and Linking a VTL Program” on page 97.
“Example Program” on page 99.
“Including the VISA Declarations File” on page 99.
“Opening a Session” on page 100.
“Device Sessions” on page 100.
“Addressing a Session” on page 102.
“Closing a Session” on page 103.

Typical Example Program Contents

The following is a summary of the VTL function calls used in the example programs.

`visa.h`	This file is included at the beginning of the file to provide the function prototypes and constants defined by VTL.
`ViSession`	The **`ViSession`** is a VTL data type. Each object that will establish a communication channel must be defined as **`ViSession`**.
`viOpenDefaultRM`	You must first open a session with the default resource manager with the **`viOpenDefaultRM`** function. This function will initialize the default resource manager and return a pointer to that resource manager session.
`viOpen`	This function establishes a communication channel with the device specified. A session identifier that can be used with other VTL functions is returned. This call must be made for each device you will be using.

`viPrintf` **`viScanf`**	These are the VTL formatted I/O functions that are patterned after those used in the C programming language. The **`viPrintf`** call sends the SCPI commands to the analyzer. The **`viPrintf`** call can also be used to query the analyzer. The **`viScanf`** call is then used to read the results.
`viClose`	This function must be used to close each session. When you close a device session, all data structures that had been allocated for the session will be deallocated. When you close the default manager session, all sessions opened using the default manager session will be closed.

Linking to VTL Libraries

Your application must link to one of the VTL import libraries:

32-bit Version (assumes Windows 95® operating system):

`C:\VXIPNP\WIN95\LIB\MSC\VISA32.LIB` for Microsoft compilers

`C:\VXIPNP\WIN95\LIB\BC\VISA32.LIB` for Borland compilers

16-bit Version:

`C:\VXIPNP\WIN\LIB\MSC\VISA.LIB` for Microsoft compilers

`C:\VXIPNP\WIN\LIB\BC\VISA.LIB` for Borland compilers

See the following section for information on how to use the VTL run-time libraries.

Compiling and Linking a VTL Program

32-bit Applications (assumes Windows 95® operating system)

The following is a summary of important compiler-specific considerations for several C/C++ compiler products when developing WIN32 applications.

For Microsoft Visual C++ version 2.0 compilers:

- Select `Project | Update All Dependencies` from the menu.
- Select `Project | Settings` from the menu. Click on the `C/C++` button. Select `Code Generation` from the `Use Run-Time Libraries` list box. VTL requires these definitions for WIN32. Click on `OK` to close the dialog boxes.

- Select `Project | Settings` from the menu. Click on the `Link` button and add `visa32.lib` to the `Object / Library Modules` list box. Optionally, you may add the library directly to your project file. Click on `OK` to close the dialog boxes.
- You may wish to add the include file and library file search paths. They are set by doing the following:
  1. Select `Tools | Options` from the menu.
  2. Click on the `Directories` button to set the include file path.
  3. Select `Include Files` from the `Show Directories For` list box.
  4. Click on the `Add` button and type in the following: `C:\VXIPNP\WIN95\INCLUDE`
  5. Select `Library Files` from the `Show Directories For` list box.
  6. Click on the `Add` button and type in the following: `C:\VXIPNP\WIN95\LIB\MSC`

For Borland C++ version 4.0 compilers:

- You may wish to add the include file and library file search paths. They are set under the `Options | Project` menu selection. Double click on `Directories` from the `Topics` list box and add the following:

```
C:\VXIPNP\WIN95\INCLUDE
C:\VXIPNP\WIN95\LIB\BC
```

16-bit Applications

The following is a summary of important compiler-specific considerations for the Windows® compiler.

For Microsoft Visual C++ version 1.5:

- To set the memory model, do the following:
  1. Select `Options | Project`.
  2. Click on the `Compiler` button, then select `Memory Model` from the `Category` list.
  3. Click on the `Model` list arrow to display the model options, and select `Large`.
  4. Click on `OK` to close the `Compiler` dialog box.
- You may wish to add the include file and library file search paths. They are set under the `Options | Directories` menu selection:

```
C:\VXIPNP\WIN\INCLUDE
C:\VXIPNP\WIN\LIB\MSC
```

  Otherwise, the library and include files should be explicitly specified in the project file.

Example Program

This example program queries a GPIB device for an identification string and prints the results. Note that you must change the address if something other than the EMC default value of 18 is required.

```
/*idn.c - program filename */

#include "visa.h"
#include <stdio.h>

void main ()
{
     /*Open session to GPIB device at address 18 */
     ViOpenDefaultRM(&defaultRM);
     ViOpen(defaultRM, "GPIB0::18::INSTR", VI_NULL,
       VI_NULL, &vi);

     /*Initialize device */
     viPrintf(vi, "*RST\n");

     /*Send an *IDN? string to the device */
     printf(vi, "*IDN?\n");

     /*Read results */
     viScanf(vi, "%t", &buf);

     /*Print results */
     printf("Instrument identification string: %s\n", buf);

     /* Close the sessions */
     viClose(vi);
     viClose(defaultRM);
}
```

Including the VISA Declarations File

For C and C++ programs, you must include the **visa.h** header file at the beginning of every file that contains VTL function calls:

```
   #include "visa.h"
```

This header file contains the VISA function prototypes and the definitions for all VISA constants and error codes. The **visa.h** header file includes the **visatype.h** header file.

The `visatype.h` header file defines most of the VISA types. The VISA types are used throughout VTL to specify data types used in the functions. For example, the `viOpenDefaultRM` function requires a pointer to a parameter of type `ViSession`. If you find `ViSession` in the `visatype.h` header file, you will find that `ViSession` is eventually typed as an unsigned long.

Opening a Session

A session is a channel of communication. Sessions must first be opened on the default resource manager, and then for each device you will be using. The following is a summary of sessions that can be opened:

- A **resource manager session** is used to initialize the VISA system. It is a parent session that knows about all the opened sessions. A resource manager session must be opened before any other session can be opened.
- A **device session** is used to communicate with a device on an interface. A device session must be opened for each device you will be using. When you use a device session you can communicate without worrying about the type of interface to which it is connected. This insulation makes applications more robust and portable across interfaces. Typically a device is an instrument, but could be a computer, a plotter, or a printer.

NOTE All devices that you will be using need to be connected and in working condition prior to the first VTL function call (`viOpenDefaultRM`). The system is configured only on the *first* `viOpenDefaultRM` per process. Therefore, if `viOpenDefaultRM` is called without devices connected and then called again when devices are connected, the devices will not be recognized. You must close **ALL** resource manager sessions and re-open with all devices connected and in working condition.

Device Sessions

There are two parts to opening a communications session with a specific device. First you must open a session to the default resource manager with the `viOpenDefaultRM` function. The first call to this function initializes the default resource manager and returns a session to that resource manager session. You only need to open the default manager session once. However, subsequent calls to `viOpenDefaultRM` returns a session to a unique session to the same default resource manager resource.

Next, you open a session with a specific device with the `viOpen` function. This function uses the session returned from `viOpenDefaultRM` and returns its own session to identify the device session. The following shows the function syntax:

viOpenDefaultRM (*sesn*);

viOpen (*sesn*, *rsrcName*, *accessMode*, *timeout*, *vi*);

The session returned from **`viOpenDefaultRM`** must be used in the *sesn* parameter of the **`viOpen`** function. The **`viOpen`** function then uses that session and the device address specified in the *(resource name)* parameter to open a device session. The *vi* parameter in **`viOpen`** returns a session identifier that can be used with other VTL functions.

Your program may have several sessions open at the same time by creating multiple session identifiers by calling the **`viOpen`** function multiple times.

The following summarizes the parameters in the previous function calls:

sesn	This is a session returned from the **`viOpenDefaultRM`** function that identifies the resource manager session.
rsrcName	This is a unique symbolic name of the device (device address).
accessMode	This parameter is not used for VTL. Use VI_NULL.
timeout	This parameter is not used for VTL. Use VI_NULL.
vi	This is a pointer to the session identifier for this particular device session. This pointer will be used to identify this device session when using other VTL functions.

The following is an example of opening sessions with a GPIB multimeter and a GPIB/VXI scanner:

```
ViSession defaultRM, dmm, scanner;
.
.
viOpenDefaultRM(&defaultRM);
viOpen(defaultRM, "GPIB0::22::INSTR", VI_NULL,
    VI_NULL, &dmm);
viOpen(defaultRM, "GPIB-VXI0::24::INSTR", VI_NULL,
    VI_NULL, &scanner);
.
.
viClose(scanner);
viClose(dmm);
viClose(defaultRM);
```

The above function first opens a session with the default resource manager. The session returned from the resource manager and a device address is then used to open a session with the GPIB device at address 22. That session will now be identified as **dmm** when using other VTL functions. The session returned from the resource manager is then used again with another device address to open a session with the GPIB/VXI device at primary address 9 and VXI logical address 24. That session will now be identified as **scanner** when using other VTL functions. See the following section for information on addressing particular devices.

Addressing a Session

As seen in the previous section, the *rsrcName* parameter in the **`viOpen`** function is used to identify a specific device. This parameter is made up of the VTL interface name and the device address. The interface name is determined when you run the VTL Configuration Utility. This name is usually the interface type followed by a number. The following table illustrates the format of the *rsrcName* for the different interface types:

Interface	Syntax
VXI	VXI [*board*]::*VXI logical address*[::INSTR]
GPIB/VXI	GPIB-VXI [*board*]::*VXI logical address*[::INSTR]
GPIB	GPIB [*board*]::*primary address*[::*secondary address*][::INSTR]

The following describes the parameters used above:

board	This optional parameter is used if you have more than one interface of the same type. The default value for *board* is 0.
VXI logical address	This is the logical address of the VXI instrument.
primary address	This is the primary address of the GPIB device.
secondary address	This optional parameter is the secondary address of the GPIB device. If no secondary address is specified, none is assumed.
INSTR	This is an optional parameter that indicates that you are communicating with a resource that is of type **INSTR**, meaning instrument.

NOTE If you want to be compatible with future releases of VTL and VISA, you must include the INSTR parameter in the syntax.

The following are examples of valid symbolic names:

VXI0::24::INSTR Device at VXI logical address 24 that is of VISA type INSTR.

VXI2::128	Device at VXI logical address 128, in the third VXI system (VXI2).
GPIB-VXI0::24	A VXI device at logical address 24. This VXI device is connected via an GPIB/VXI command module.
GPIB0::7::0	An GPIB device at primary address 7 and secondary address 0 on the GPIB interface.

The following is an example of opening a device session with the GPIB device at primary address 23.

```
ViSession defaultRM, vi;

.
.

viOpenDefaultRM(&defaultRM);

viOpen(defaultRM, “GPIB0::23::INSTR”, VI_NULL,VI_NULL,&vi);

.
.

viClose(vi);

viClose(defaultRM);
```

Closing a Session

The **`viClose`** function must be used to close each session. You can close the specific device session, which will free all data structures that had been allocated for the session. If you close the default resource manager session, all sessions opened using that resource manager will be closed.

Since system resources are also used when searching for resources (**`viFindRsrc`**) or waiting for events (**`viWaitOnEvent`**), the **`viClose`** function needs to be called to free up find lists and event contexts.

Using Marker Peak Search and Peak Excursion

Example:

```
/*************************************************************/
/* Using Marker Peak Search and Peak Excursion              */
/*                                                          */
/* This example is for the E44xxB ESA Spectrum Analyzers    */
/* and E740xA EMC Analyzers.                                */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens a GPIB session at address 18                     */
/* - Clears the Analyzer                                    */
/*      *CLS                                                */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Sets the analyzer center frequency, span and units     */
/*      SENS:FREQ:CENT freq                                 */
/*      SENS:FREQ:SPAN freq                                */
/*      UNIT:POW DBM                                        */
/* - Set the input port to the 50 MHz amplitude reference   */
/*      CAL:SOUR:STAT ON                                    */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                         */
/* - Prompt the user for peak excursion and set them        */
/*      CALC:MARK:PEAK:EXC dB                               */
/* - Set the peak threshold to -90 dBm                      */
/*      TRAC:MATH:PEAK:THR:STAT ON                          */
/*      TRAC:MATH:PEAK:THR -90                              */
/* - Trigger a sweep and wait for sweep to complete         */
/*      INIT:IMM;*WAI                                       */
/* - Set the marker to the maximum peak                     */
/*      CALC:MARK:MAX                                       */
/* - Query and read the marker frequency and amplitude      */
/*      CALC:MARK:X?                                        */
/*      CALC:MARK:Y?                                        */
/* - Close the session                                      */
/*************************************************************/

#include <stdio.h>
```

```
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char    cEnter = 0;
int      iResult = 0;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B and E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf (".......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}

void main()
{
```

```
/*Program Variables*/
ViStatus viStatus  = 0;
double dMarkerFreq = 0;
double dMarkerAmpl = 0;
float fPeakExcursion =0;
long lOpc = 0L;

/*Open a GPIB session at address 18.*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
     printf("Could not open a session to GPIB device at address 18!\n");
     exit(0);
}
/*Clear the instrument*/
viClear(viESA);

/*Reset the instrument*/
viPrintf(viESA,"*RST\n");

/*Set Y-Axis units to dBm*/
viPrintf(viESA, "UNIT:POW DBM\n");

/*Set the  analyzer center frequency to 50MHZ*/
viPrintf(viESA,"SENS:FREQ:CENT 50e6\n");

/*Set the analyzer span to 50MHZ*/
viPrintf(viESA,"SENS:FREQ:SPAN 50e6\n");

/*Display the program heading */
printf("\n\t\t Marker Program \n\n" );

/* Check for the instrument model number and route the 50MHz signal accordingly*/
Route50MHzSignal();

/*Set analyzer to single sweep mode*/
viPrintf(viESA,"INIT:CONT 0 \n");

/*User enters the peak excursion value*/
printf("\t Enter PEAK EXCURSION in dB:  ");
scanf( "%f",&fPeakExcursion);

/*Set the peak excursion*/
viPrintf(viESA,"CALC:MARK:PEAK:EXC %1fDB \n",fPeakExcursion);
```

```
  /*Set the peak thresold */
  viPrintf(viESA,"CALC:MARK:PEAK:THR -90 \n");

  /*Trigger a sweep and wait for completion*/
  viPrintf(viESA,"INIT:IMM;*WAI\n");

  /*Set the marker to the maximum peak*/
  viPrintf(viESA,"CALC:MARK:MAX \n");

  /*Query and read the marker frequency*/
  viQueryf(viESA,"CALC:MARK:X? \n","%lf",&dMarkerFreq);
  printf("\n\t RESULT: Marker Frequency is: %lf MHZ \n\n",dMarkerFreq/10e5);

  /*Query and read the marker amplitude*/
  viQueryf(viESA,"CALC:MARK:Y?\n","%lf",&dMarkerAmpl);
  printf("\t RESULT: Marker Amplitude is: %lf dBm \n\n",dMarkerAmpl);

  /*Close the session*/
  viClose(viESA);
  viClose(defaultRM);
}
```

Using Marker Delta Mode and Marker Minimum Search

```
/***********************************************************/
/* Using Marker Delta Mode and Marker Minimum Search       */
/*                                                         */
/* This example is for the E44xxB ESA Spectrum Analyzers   */
/* and E740xA EMC Analyzers.                               */
/*                                                         */
/* This C programming example does the following.          */
/* The SCPI instrument commands used are given as          */
/* reference.                                              */
/*                                                         */
/* - Opens a GPIB session at address 18                    */
/* - Clears the Analyzer                                   */
/* - Resets the Analyzer                                   */
/*      *RST                                               */
/* - Set the input port to the 50 MHz amplitude reference  */
/*      CAL:SOUR:STAT ON                                   */
/* - Set the analyzer to single sweep mode                 */
/*      INIT:CONT 0                                        */
/* - Prompts the user for the start and stop frequencies   */
/* - Sets the start and stop frequencies                   */
/*      SENS:FREQ:START freq                               */
/*      SENS:FREQ:STOP freq                                */
/* - Trigger a sweep and wait for sweep completion         */
/*      INIT:IMM;*WAI                                      */
/* - Set the marker to the maximum peak                    */
/*      CALC:MARK:MAX                                      */
/* - Set the analyzer to activate the delta marker         */
/*      CALC:MARK:MODE DELT                                */
/* - Trigger a sweep and wait for sweep completion         */
/*      INIT:IMM;*WAI                                      */
/* - Set the marker to the minimum amplitude mode          */
/*      CALC:MARK:MIN                                      */
/* - Query and read the marker amplitude                   */
/*      CALC:MARK:Y?                                       */
/* - Close the session                                     */
/***********************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
```

```
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256] ={0};
char   cEnter = 0;
int      iResult =0;

/*Set the input port to the 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /* E4401B, E4411B and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

       /*Externally route the 50MHz Signal*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}

void main()
{
  /*Program Variable*/
  ViStatus viStatus  = 0;
  double dStartFreq =0.0;
  double dStopFreq  =0.0;
  double dMarkerAmplitude = 0.0;
  long   lOpc =0L;

  /* Open an GPIB session at address 18*/
  viStatus=viOpenDefaultRM(&defaultRM);
  viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
  if(viStatus)
  {
       printf("Could not open a session to GPIB device at address 18!\n");
       exit(0);
  }
  /*Clear the instrument*/
  viClear(viESA);

  /*Reset the instrument*/
  viPrintf(viESA,"*RST\n");

  /*Display the program heading */
  printf("\n\t\t Marker Delta Program \n\n" );

  /*Check for the instrument model number and route the 50MHz signal accordingly*/
  Route50MHzSignal();

  /*Set the analyzer to single sweep mode*/
  viPrintf(viESA,"INIT:CONT 0\n");

  /*Prompt the user for the start frequency*/
  printf("\t Enter the Start frequency in MHz ");

  /*The user enters the start frequency*/
  scanf("%lf",&dStartFreq);

  /*Prompt the user for the stop frequency*/
  printf("\t Enter the Stop frequency in MHz ");

  /*The user enters the stop frequency*/
  scanf("%lf",&dStopFreq);

  /*Set the analyzer to the values given by the user*/
  viPrintf(viESA,"SENS:FREQ:STAR %lf MHZ \n;:SENS:FREQ:STOP %lf
MHZ\n",dStartFreq,dStopFreq);

  /*Trigger a sweep, wait for completion*/
  viPrintf(viESA,"INIT:IMM;*WAI\n");

  /*Set the marker to the maximum peak*/
  viPrintf(viESA,"CALC:MARK:MAX\n");

  /*Set the analyzer to  activate  delta marker mode*/
    viPrintf(viESA,"CALC:MARK:MODE DELT\n");

  /*Trigger a sweep, wait for completion*/
  viPrintf(viESA,"INIT:IMM;*WAI\n");

  /*Set the marker to minimum amplitude*/
  viPrintf(viESA,"CALC:MARK:MIN\n");

  /*Query and read the marker amplitude*/
  viQueryf(viESA,"CALC:MARK:Y?\n","%lf",&dMarkerAmplitude);

  /*print the marker amplitude*/
  printf("\n\n\tRESULT: Marker Amplitude Delta = %lf dB\n\n",dMarkerAmplitude);

  /*Close the session*/
  viClose(viESA);
  viClose(defaultRM);
}
```

Performing Internal Self-alignment

```
/*************************************************************/
/* Performing Internal Self-alignment                        */
/*                                                           */
/* This example is for the E44xxB ESA Spectrum Analyzers     */
/* and E740xA EMC Analyzers.                                 */
/*                                                           */
/* This example shows two ways of executing an internal      */
/* self-alignment. The first demonstrates using the *OPC?    */
/* query to determine when the alignment has completed. The */
/* second demonstrates using the query form of the CAL:ALL  */
/* command to not only determine when the alignment has      */
/* been completed, but the pass/fail status of the align-    */
/* ment process.                                             */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used  are given as           */
/* reference.                                                */
/*                                                           */
/* - Opens a GPIB session at address 18                      */
/* - Clears the Analyzer                                     */
/*      *CLS                                                 */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - VISA function sets the time out to infinite             */
/* - Initiate self-alignment                                 */
/*      CAL:ALL                                              */
/* - Query for operation complete                            */
/*      *OPC?                                                */
/* - Query for results of self-alignment                     */
/*      CAL:ALL?                                             */
/* - Report the results of the self-alignment                */
/* - Close the session                                       */
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
```

```
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char     cEnter = 0;
int      iResult = 0;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B, and E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf (".......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}

void main()
{
  /*Program Variables*/
  ViStatus viStatus  = 0;
  long lOpc =0L;
  long lResult =0L;

  /* Open a GPIB session at address 18*/
  viStatus=viOpenDefaultRM(&defaultRM);
  viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
```

```
if(viStatus)
{
     printf("Could not open a session to GPIB device at address 18!\n");
     exit(0);
}
/*Clear the instrument*/
viClear(viESA);

/*Reset the instrument*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Internal Self-Alignment Program \n\n" );

/*Check for the instrument model number and route the 50MHz-signal accordingly*/
Route50MHzSignal();

/*VISA function sets the time out to infinite for this specified session*/
viSetAttribute(viESA, VI_ATTR_TMO_VALUE, VI_TMO_INFINITE);
printf("\t Performing first self alignment ..... " );

/*Initiate a self-alignment */
viPrintf(viESA,"CAL:ALL\n");

/*Query for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
printf ("\n\n\t First Self Alignment is Done \n\n");
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
printf ("\n\n\t Press Return to continue with next alignment \n\n");
scanf( "%c",&cEnter);
printf("\t Performing next self alignment ..... " );

/* Query for self-alignment results*/
viQueryf(viESA,"CAL:ALL?\n","%d",&lResult);
if (lResult)
     printf ("\n\n\t Self-alignment Failed \n");
else
     printf ("\n\n\t Self-alignment Passed \n");

/* Query for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
```

```
  {
       printf("Program Abort! error ocurred: last command was not completed!\n");
       exit(0);
  }
  /*Close the session*/
  viClose(viESA);
  viClose(defaultRM);
}
```

Reading Trace Data using ASCII Format (GPIB)

```
/*********************************************************************/
/* Reading Trace Data using ASCII Format (GPIB)                      */
/*                                                                   */
/* This example is for the E44xxB ESA Spectrum Analyzers             */
/* and E740xA EMC Analyzers.                                         */
/*                                                                   */
/* This C programming example does the following.                    */
/* The required SCPI instrument commands are given as                */
/* reference.                                                        */
/*                                                                   */
/* - Opens a GPIB session at address 18                              */
/* - Clears the Analyzer                                             */
/* - Resets the Analyzer                                             */
/*      *RST                                                         */
/* - Set the input port to the 50 MHz amplitude reference            */
/*   E4411B or E4401B                                                */
/*      CAL:SOUR:STAT ON                                             */
/*   E4402, E4403B, E4404BE, 4405B, E4407B or E4408B                 */
/*      Prompt to connect AMPTD REF OUT to INPUT                     */
/*      CAL:SOUR STAT ON                                             */
/* - Query for the number of sweep points (only applies to           */
/*   firmware revisions A.04.00 and later); default is 401           */
/*      SENS:SWE:POIN?                                               */
/* - Sets the analyzer center frequency to 50 MHz                    */
/*      SENS:FREQ:CENT 50 MHZ                                        */
/* - Sets the analyzer span to 50 MHz                                */
/*      SENS:FREQ:SPAN 50 MHZ                                        */
/* - Set the analyzer to single sweep mode                           */
/*      INIT:CONT 0                                                  */
/* - Trigger a sweep and wait for sweep to complete                  */
/*      INIT:IMM;*WAI                                                */
/* - Specify units in dBm                                            */
/*      UNIT:POW DBM                                                 */
/* - Set the analyzer trace data to ASCII                            */
/*      FORM:DATA: ASC                                               */
/* - Trigger a sweep and wait for sweep to complete                  */
/*      INIT:IMM;*WAI                                                */
/* - Query the trace data                                            */
/*      TRAC:DATA? TRACE1                                            */
/* - Remove the "," from the ACSII data                              */
/* - Save the trace data to an ASCII file                            */
/* - Close the session                                               */
```

```
/***********************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256] ={0};
char  cEnter =0;
int      iResult =0;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4401B, E4411B and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf (".......Press Return to continue \n");
       scanf( "%c",&cEnter);

       /*Externally route the 50MHz Signal*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}
```

```
void main()
{
  /*Program Variable*/
  ViStatus viStatus = 0;
  /*Dimension cResult to 13 bytes per sweep point, 8192 sweep points maximum*/
     ViChar _VI_FAR cResult[106496] = {0};
  FILE *fTraceFile;
     static ViChar *cToken ;
  int iNum  =0;
  int iSwpPnts = 401;
  long lCount=0L;
  long lOpc=0;

  /*iNum set to 13 times number of sweep points, 8192 sweep points maximum*/
  iNum =106496;
     lCount =0;

  /* Open a GPIB session at address 18*/
  viStatus=viOpenDefaultRM(&defaultRM);
  viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
  if(viStatus)
  {
       printf("Could not open a session to GPIB device at address 18!\n");
       exit(0);
  }
  /* Clear the instrument */
  viClear(viESA);

  /*Reset the instrument. This will set number of sweep points to default of 401*/
  viPrintf(viESA,"*RST\n");

  /*Display the program heading */
  printf("\n\t\t Read in Trace Data using ASCII Format (GPIB) Program \n\n" );

  /* Check for the instrument model number and route the 50MHz signal accordingly*/
  Route50MHzSignal();

  /*Query number of sweep points per trace (firmware revision A.04.00 and later)*/
  /*For firmware revisions prior to A.04.00, the number of sweep points is 401*/
  iSwpPnts = 401;
     viQueryf(viESA,"SENSE:SWEEP:POINTS?\n","%d",&iSwpPnts);

  /*Set the analyzer center frequency to 50MHz*/
  viPrintf(viESA,"SENS:FREQ:CENT 50 MHz\n");

  /*Set the analyzer to 50MHz Span*/
  viPrintf(viESA,"SENS:FREQ:SPAN 50 MHz\n");

  /*Set the analyzer to single sweep mode */
  viPrintf(viESA,"INIT:CONT 0 \n");

  /*Trigger a sweep and wait for sweep to complete */
  viPrintf(viESA,"INIT:IMM;*WAI\n");

  /* Specify units in dBm*/
  viPrintf(viESA,"UNIT:POW DBM \n");

  /*Set analyzer trace data format to ASCII Format*/
  viPrintf(viESA,"FORM:DATA ASC \n");

  /*Trigger a sweep and wait for sweep to complete */
  viPrintf(viESA,"INIT:IMM;*WAI\n");

  /*Query the Trace Data using ASCII Format */
  viQueryf(viESA,"%s\n", "%#t","TRAC:DATA? TRACE1" , &iNum , cResult);

  /*Remove the "," from the ASCII trace data for analyzing data*/
    cToken  = strtok(cResult,",");

  /*Save trace data to an ASCII to a file, by removing the "," token*/
  fTraceFile=fopen("C:\\temp\\ReadAscGpib.txt","w");
  fprintf(fTraceFile,"ReadAscGpib.exe Output\nAgilent Technologies 2000\n\n");
  fprintf(fTraceFile,"\tAmplitude of point[%d] = %s dBm\n",lCount+1,cToken);
    while (cToken != NULL)
      {
      lCount++;
      cToken =strtok(NULL,",");
            if (lCount != iSwpPnts)
            fprintf(fTraceFile,"\tAmplitude of point[%d] = %s 
dBm\n",lCount+1,cToken);
      }
  fprintf(fTraceFile,"\nThe Total trace data points of the spectrum are:[%d]
\n\n",lCount);
  fclose(fTraceFile);

  /*Close the session*/
  viClose(viESA);
  viClose(defaultRM);
}
```

Reading Trace Data Using 32-bit Real Format (GPIB)

```
/***********************************************************/
/* Reading Trace Data using 32-bit Real Format (GPIB)      */
/*                                                         */
/* This example is for the E44xxB ESA Spectrum Analyzers   */
/* and E740xA EMC Analyzers.                               */
/*                                                         */
/* This C programming example does the following.          */
/* The SCPI instrument commands used are given as          */
/* reference.                                              */
/*                                                         */
/* - Opens a GPIB session at address 18                    */
/* - Clears the Analyzer                                   */
/* - Resets the Analyzer                                   */
/*      *RST                                               */
/* - Set the input port to the 50 MHz amplitude reference  */
/*      CAL:SOUR:STAT ON                                   */
/* - Query for the number of sweep points (for firmware    */
/*   revisions A.04.00 and later). Default is 401.         */
/*       SENS:SWE:POIN?                                    */
/* - Calculate the number of bytes in the header           */
/* - Set the analyzer to single sweep mode                 */
/*      INIT:CONT 0                                        */
/* - Sets the analyzer center frequency and span to 50 MHz */
/*      SENS:FREQ:CENT 50 MHZ                              */
/*      SENS:FREQ:SPAN 50 MHZ                              */
/* - Specify 10 dB per division for the amplitude scale in */
/*   and dBm Units                                         */
/*      DISP:WIND:TRAC:Y:SCAL:PDIV 10 dB                   */
/*      UNIT:POW DBM                                       */
/* - Set the analyzer trace data to 32-bit Real            */
/*      FORM:DATA: REAL,32                                 */
/* - Set the binary order to swap                          */
/*      FORM:BORD SWAP                                     */
/* - Trigger a sweep and wait for sweep to complete        */
/*      INIT:IMM;*WAI                                      */
/* - Calculate the number of bytes in the trace record     */
/* - Query the trace data                                  */
/*      TRAC:DATA? TRACE1                                  */
/* - Remove the "," from the ACSII data                    */
/* - Save the trace data to an ASCII file                   */
/* - Close the session                                     */
```

```
/************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViChar    cIdBuff[256];
char  cEnter =0;
int      iResult =0;

void Route50MHzSignal()
{
viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz internal reference source for the models*/
     /*E4401B, E4411B and E7401A*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user to*/
     /* connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf ("......Press Return to continue \n");
     scanf( "%c",&cEnter);

     /*Externally route the 50MHz Signal*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
```

```
/*Program Variables*/
ViStatus viStatus= 0;
    ViChar _VI_FAR cResult[5000] = {0};
ViReal32 dTraceArray[401] = {0};
char cBufferInfo[6]= {0};
long lNumberBytes =0L;
long lOpc =0L;
unsigned long lRetCount = 0L;
int iSize = 0;
/*BytesPerPoint is 4 for Real32 or Int32 formats, 8 for Real64, and 2 for Uint16*/
int iBytesPerPnt = 4;
int iSwpPnts = 401;
int iDataBytes=1604;
int iHeaderBytes=6;
FILE *fTraceFile;

/* Open a GPIB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
     printf("Could not open a session to GPIB device at address 18!\n");
     exit(0);
}
/*Clear the instrument */
viClear(viESA);

/*Reset the instrument. This will set number of sweep points to default of 401*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Read in Trace Data using 32-bit Real Format (using GPIB) \n\n" );

/* Set the input port to the 50MHz amplitude reference*/
Route50MHzSignal();

/*Query number of sweep points per trace (firmware revision A.04.00 and later)*/
/*For firmware revisions prior to A.04.00, the number of sweep points is 401*/
iSwpPnts=401;
viQueryf(viESA,"SENSE:SWEEP:POINTS?\n","%d",&iSwpPnts);

/*Calculate number of bytes in the header. The header consists of the "#" sign*/
/*followed by a digit representing the number of digits to follow. The digits */
/*which follow represent the number of sweep points multiplied by the number  */
/*of bytes per point. */
iHeaderBytes = 3;              /*iDataBytes >3, plus increment for "#" and "n"*/
```

```
iDataBytes = (iSwpPnts*iBytesPerPnt);
lNumberBytes = iDataBytes;
while ((iDataBytes = (iDataBytes / 10 )) > 0 )
{
     iHeaderBytes++;
}

/*Set analyzer to single sweep mode */
viPrintf(viESA,"INIT:CONT 0 \n");

/*Set the analyzer to 50MHz-center frequency */
viPrintf(viESA,"SENS:FREQ:CENT 50 MHZ\n");

/*Set the analyzer to 50MHz Span */
viPrintf(viESA,"SENS:FREQ:SPAN 50 MHZ\n");

/* Specify dB per division of each vertical division and  Units */
viPrintf(viESA,"DISP:WIND:TRAC:Y:SCAL:PDIV 10dB\n");
viPrintf(viESA,"UNIT:POW DBM\n");

/*Set analyzer trace data format to 32-bit Real */
viPrintf(viESA,"FORM:DATA REAL,32 \n");

/*Set the binary byte order to SWAP */
viPrintf(viESA, "FORM:BORD SWAP\n");

/*Trigger a sweep and wait for sweep to complete*/
viPrintf(viESA,"INIT:IMM;*WAI\n");

/*Calculate size of trace record. This will be sum of HeaderBytes, NumberBytes*/
/*(the actual data bytes) and the "/n" terminator*/
iSize = lNumberBytes +iHeaderBytes+1;

/*Get trace header data and trace data */
viPrintf(viESA,"TRAC:DATA? TRACE1\n");
viRead (viESA,(ViBuf)cResult,iSize,&lRetCount);

/*Extract the trace data*/
 memcpy(dTraceArray,cResult+iHeaderBytes,(size_t)lNumberBytes);

/*Save trace data to an ASCII file*/
fTraceFile=fopen("C:\\temp\\ReadTrace32Gpib.txt","w");
fprintf(fTraceFile,"ReadTrace32Gpib.exe Output\nAgilent Technologies 2000\n\n");
fprintf(fTraceFile,"The %d trace data points of the
spectrum:\n\n",(lNumberBytes/4));
for ( long i=0;i<lNumberBytes/4;i++)
```

```
     fprintf(fTraceFile,"\tAmplitude of point[%d] = %.2lf
dBm\n",i+1,dTraceArray[i]);
fclose(fTraceFile);

/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Reading Trace Data Using ASCII Format (RS-232)

```
/*********************************************************/
/* Reading Trace Data using ASCII Format (RS-232)        */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens an RS-232 session at COM1/COM2                */
/* - Clears the Analyzer                                 */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Set the input port to the 50 MHz amplitude reference */
/*      CAL:SOUR:STAT ON                                 */
/* - Query for the number of sweep points (for firmware  */
/*   revisions A.04.00 and later). Default is 401.       */
/*      SENS:SWE:POIN?                                   */
/* - Set the analyzer to single sweep mode               */
/*      INIT:CONT 0                                      */
/* - Sets the analyzer center frequency and span to 50 MHz */
/*      SENS:FREQ:CENT 50 MHZ                            */
/*      SENS:FREQ:SPAN 50 MHZ                            */
/* - Trigger a sweep                                     */
/*      INIT:IMM                                         */
/* - Check for operation complete                        */
/*      *OPC?                                            */
/* - Specify dBm Unit                                    */
/*      UNIT:POW DBM                                     */
/* - Set the analyzer trace data ASCII                   */
/*      FORM:DATA: ASC                                   */
/* - Trigger a sweep                                     */
/*      INIT:IMM                                         */
/* - Check for operation complete                        */
/*      *OPC?                                            */
/* - Query the trace data                                */
/*      TRAC:DATA? TRACE1                                */
/* - Remove the "," from the ACSII data                  */
/* - Save the trace data to an ASCII file                */
/* - Close the session                                   */
/*********************************************************/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define  hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256] ={0};
char  cEnter ={0};
int      iResult =0;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B and E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user to/*
      /* connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
```

```
{
/*Program Variable*/
ViStatus viStatus = 0;
/*Dimension cResult to 13 bytes per sweep point, 8192 sweep points maximum*/
ViChar _VI_FAR cResult[106496] = {0};
FILE *fTraceFile;
    static ViChar *cToken;
int  iNum  =0;
int  iSwpPnts = 401;
long lCount=0L;
long lOpc=0L;

/*iNum set to 13 times number of sweep points, 8192 sweep points maximum*/
iNum =106496;
    lCount =0;

/* Open a serial session at COM1 */
viStatus=viOpenDefaultRM(&defaultRM);
if (viStatus =viOpen(defaultRM,"ASRL1::INSTR",VI_NULL,VI_NULL,&viESA) !=
VI_SUCCESS)
{
      printf("Could not open a session to ASRL device at COM1!\n");
      exit(0);
}
/* Clear the instrument */
viClear(viESA);

/*Reset the instrument. This will set number of sweep points to default of 401*/
  viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\tRead in Trace Data using ASCII Format (RS232) Program \n\n" );

/* Check for the instrument model number and route the 50MHz signal accordingly*/
Route50MHzSignal();

/*Query number of sweep points per trace (firmware revision A.04.00 and later)*/
/*For firmware revisions prior to A.04.00, the number of sweep points is 401  */
iSwpPnts = 401;
viQueryf(viESA, "SENSE:SWEEP:POINTS?\n","%d",&iSwpPnts);

/*Set the analyzer center frequency to 50MHz */
viPrintf(viESA,"SENS:FREQ:CENT 50 MHz\n");

/*Set the analyzer to 50MHz Span*/
viPrintf(viESA,"SENS:FREQ:SPAN 50 MHz\n");
```

```
/*set the analyzer to single sweep mode*/
viPrintf(viESA,"INIT:CONT 0 \n");

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Read the operation complete query*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}
/*Specify units in dBm*/
viPrintf(viESA,"UNIT:POW DBM \n");

/*Set analyzer trace data format to ASCII Format*/
viPrintf(viESA,"FORM:DATA ASC \n");

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Read the operation complete query */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}
/*Query the Trace Data using ASCII Format */
viQueryf(viESA,"%s\n", "%#t","TRAC:DATA? TRACE1" , &iNum , cResult);

/*Remove the "," from the ASCII trace data for analyzing data*/
    cToken  = strtok(cResult,",");

/*Save trace data to an ASCII to a file, by removing the "," token*/
fTraceFile=fopen("C:\\temp\\ReadAscRS232.txt","w");
fprintf(fTraceFile,"ReadAscRS232.exe Output\nHewlett-Packard 1999\n\n");
fprintf(fTraceFile,"\tAmplitude of point[%d] = %s dBm\n",lCount+1,cToken);
    while (cToken != NULL)
      {
      lCount++;
      cToken =strtok(NULL,",");
            if (lCount != iSwpPnts)
            fprintf(fTraceFile,"\tAmplitude of point[%d] = %s
dBm\n",lCount+1,cToken);
       }
fprintf(fTraceFile,"\nThe Total trace data points of the spectrum are:[%d]
\n\n",lCount);
fclose(fTraceFile);

/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Reading Trace Data Using 32-bit Real Format (RS-232)

```
/*************************************************************/
/* Reading Trace Data using 32-bit Real Format (RS-232)     */
/*                                                           */
/* This example is for the E44xxB ESA Spectrum Analyzers    */
/* and E740xA EMC Analyzers.                                 */
/*                                                           */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                                */
/*                                                           */
/* - Opens an RS-232 session at COM1/COM2                   */
/* - Clears the Analyzer                                     */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Set the input port to the 50 MHz amplitude reference   */
/*      CAL:SOUR:STAT ON                                     */
/* - Query for the number of sweep points (for firmware     */
/*   revision A.04.00 and later). Default is 401.           */
/*      SENS:SWE:POIN?                                       */
/* - Calculate the number of bytes in the header            */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                          */
/* - Sets the analyzer center frequency and span to 50 MHz  */
/*      SENS:FREQ:CENT 50 MHZ                                */
/*      SENS:FREQ:SPAN 50 MHZ                                */
/* - Specify 10 dB per division for the amplitude scale in  */
/*   and dBm Units                                           */
/*      DISP:WIND:TRAC:Y:SCAL:PDIV 10 dB                     */
/*      UNIT:POW DBM                                         */
/* - Set the analyzer trace data to 32-bit Real             */
/*      FORM:DATA: REAL,32                                   */
/* - Set the binary order to swap                           */
/*      FORM:BORD SWAP                                       */
/* - Trigger a sweep                                         */
/*      INIT:IMM                                             */
/* - Check for operation complete                           */
/*      *OPC?                                                */
/* - Calculate the number of bytes in the trace record      */
/* - Set VISA timeout to 60 seconds, to allow for slower    */
/*   transfer times caused by higher number of sweep points */
/*   at low baud rates.                                      */
/* - Set VISA to terminate read after buffer is empty       */
/* - Query the trace data                                   */
/*      TRAC:DATA? TRACE1                                   */
/* - Reset VISA timeout to 3 seconds                        */
/* - Remove the "," from the ACSII data                     */
/* - Save the trace data to an ASCII file                   */
/* - Close the session                                      */
/***********************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char    cEnter = 0;
int    iResult = 0;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz amplitude reference for the models*/
     /*E4411B and E4401B*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user to*/
     /* connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf (".......Press Return to continue \n");
     scanf( "%c",&cEnter);

     /*Externally route the 50MHz Signal*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
/*Program Variables*/
ViStatus viStatus= 0;
    ViChar _VI_FAR cResult[1024000] = {0};
ViReal32 dTraceArray[1024] = {0};
char cBufferInfo[7]= {0};
long lNumberBytes =0L;
long lOpc =0L;
unsigned long lRetCount = 0L;
int iSize = 0;
/*BytesPerPnt is 4 for Real32 or Int32 formats, 8 for Real64, and 2 for Uint16*/
int iBytesPerPnt = 4;
int iSwpPnts = 401;   /*Number of points per sweep*/
int iDataBytes = 1604;/*Number of data points, assuming 4 bytes per point*/
int iHeaderBytes = 6; /*Number of bytes in the header, assuming 1604 data bytes*/
FILE *fTraceFile;

/* Open a serial session at COM1 */
viStatus=viOpenDefaultRM(&defaultRM);
if (viStatus =viOpen(defaultRM,"ASRL1::INSTR",VI_NULL,VI_NULL,&viESA) !=
VI_SUCCESS)
{
     printf("Could not open a session to ASRL device at COM1!!\n");
     exit(0);
}
/*Clear the instrument */
viClear(viESA);

/*Reset the instrument. This will set number of sweep points to default of 401*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Read in Trace Data using ASCII Format (using RS-232) Program \n\n"
);

/* Set the input port to the internal 50MHz reference source */
Route50MHzSignal();
```

```
/*Query number of sweep points per trace (firmware revision A.04.00 or later)*/
/*For firmware revisions prior to A.04.00, the number of sweep points is 401 */
iSwpPnts = 401;
viQueryf(viESA,"SENSE:SWEEP:POINTS?\n","%d",&iSwpPnts);

/*Calculate number of bytes in the header. The header consists of the "#" sign*/
/*followed by a digit representing the number of digits to follow. The digits */
/*which follow represent the number of sweep points multiplied by the number  */
/*of bytes per point. */
iHeaderBytes = 3;            /*iDataBytes >0, plus increment for "#" and "n" */
iDataBytes = (iSwpPnts*iBytesPerPnt);
    lNumberBytes = iDataBytes;
while ((iDataBytes = (iDataBytes / 10 )) > 0 )
{
     iHeaderBytes++;
}

/*Set analyzer to single sweep mode */
viPrintf(viESA,"INIT:CONT 0 \n");

/* Set the analyzer to 50MHz-center frequency */
viPrintf(viESA,"SENS:FREQ:CENT 50 MHZ\n");

/*Set the analyzer to 50MHz Span */
viPrintf(viESA,"SENS:FREQ:SPAN 50 MHZ\n");

/* Specify dB per division of each vertical division & Units */
viPrintf(viESA,"DISP:WIND:TRAC:Y:SCAL:PDIV 10dB\n");
viPrintf(viESA,"UNIT:POW DBM\n");

/*Set analyzer trace data format to 32-bit Real */
viPrintf(viESA,"FORM:DATA REAL,32\n");

/*Set the binary byte order to SWAP */
viPrintf(viESA, "FORM:BORD SWAP\n");

/*Trigger a sweep */
viPrintf(viESA,"INIT:IMM\n");

/*Read the operation complete query */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
```

```
/*Calculate size of trace record. This will be the sum of HeaderBytes, Number*/
/*Bytes (the actual data bytes) and the "\n" terminator*/
iSize = lNumberBytes + iHeaderBytes + 1;

/*Increase timeout to 60 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

/*Set RS-232 interface to terminate when the buffer is empty*/
viSetAttribute(viESA,VI_ATTR_ASRL_END_IN,VI_ASRL_END_NONE);

/*Get trace header data and trace data*/
viPrintf(viESA,"TRAC:DATA? TRACE1\n");
viRead (viESA,(ViBuf)cResult,iSize,&lRetCount);

/*Reset timeout to 3 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

/*Extract the trace data*/
memcpy(dTraceArray,cResult+iHeaderBytes,(size_t)lNumberBytes);

/*Save trace data to an ASCII file*/
fTraceFile=fopen("C:\\temp\\ReadTrace32Rs232.txt","w");
fprintf(fTraceFile,"ReadTrace32Rs232.exe Output\nHewlett-Packard 1999\n\n");
fprintf(fTraceFile,"The %d trace data points of the
spectrum:\n\n",(lNumberBytes/4));
for ( long i=0;i<lNumberBytes/iBytesPerPnt;i++)
     fprintf(fTraceFile,"\tAmplitude of point[%d] =%.2lf
dBm\n",i+1,dTraceArray[i]);
fclose(fTraceFile);

/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Using Limit Lines

```
/************************************************************/
/* Using Limit Lines                                        */
/*                                                          */
/* This example is for the E44xxB ESA Spectrum Analyzers    */
/* and E740xA EMC Analyzers.                                */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/*                                                          */
/* - Open a GPIB session at address 18.                     */
/* - Clear the analyzer.                                    */
/*       *CLR                                               */
/* - Reset the analyzer.                                    */
/*       *RST                                               */
/* - Set Y-Axis Units to dBm                                */
/*       UNIT:POW DBM                                       */
/* - Define the upper limit line to have frequency/         */
/*   amplitude pairs.                                       */
/*       CALC:LLINE1:CONT:DOM FREQ                          */
/*       CALC:LLINE1:TYPE UPP                               */
/*       CALC:LLINE1:DISP ON                                */
/*       CALC:LLINE1:DATA freq1,amp1,1,freq2,amp2,1...      */
/* - Define the lower limit line to have frequency/amplitude*/
/*   pairs.                                                 */
/*       CALC:LLINE2:CONT:DOM FREQ                          */
/*       CALC:LLINE2:TYPE LOW                               */
/*       CALC:LLINE2:DISP ON                                */
/*       CALC:LLINE2:DATA freq1,amp1,1,freq2,amp2,1...      */
/* - Turn the limit line test function on.                  */
/*       CALC:LLINE2:STAT ON                                */
/* - Set the analyzer to a center frequency of 50 MHz, span */
/*   to 20 MHz, and resolution bandwidth to 1 MHz.          */
/*       SENS:FREQ:SPAN 20 MHZ                              */
/*       SENS:FREQ:CENT 50 MHZ                              */
/*       SENS:BWID:RES 1 MHZ                                */
/* - Turn the limit line test function on.                  */
/* - Set the analyzer reference level to 0 dBm.             */
/*       DISP:WIND:TRAC:Y:SCAL:RLEV 0                       */
/* - Set the input port to the 50 MHz amplitude reference.  */
/*       CAL:SOUR:STAT ON                                   */
/* - Check to see if limit line passes or fails.  It should */
/*   pass.                                                  */
/*       CALC:LLINE:FAIL?                                   */
/* - Pause for 5 seconds.                                   */
/* - Deactivate the 50 MHz alignment signal.                */
/*       CAL:SOUR:STAT OFF                                  */
/* - The limit line test should fail.                       */
/* - Close the session.                                     */
/************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <windows.h>
#include "visa.h"
#define  YIELD Sleep(5000)

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char    cEnter = 0;
int      iResult = 0;
long   lLimitTest =0L;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4401B, E4411B, and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
        /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
        /* to connect the amplitude reference output to the input*/
        printf ("Connect AMPTD REF OUT to the INPUT \n");
        printf (".......Press Return to continue \n");
        scanf( "%c",&cEnter);

        /*Externally route the 50MHz Signal*/
        viPrintf(viESA,"CAL:SOUR:STAT ON \n");
    }
}

void printResult()
{
  viQueryf(viESA, "CALC:CLIM:FAIL?\n", "%ld", &lLimitTest);
     if (lLimitTest!=0)
  {
        printf ("\n\t..Limit Line Failed.....\n");
        viQueryf(viESA, "CALC:LLINE1:FAIL?\n", "%ld", &lLimitTest);
        if (lLimitTest==0)
              printf ("\n\t......Limit Line1 Passed \n");
        else  printf ("\n\t......Limit Line1 Failed \n");

        viQueryf(viESA, "CALC:LLINE2:FAIL?\n", "%ld", &lLimitTest);
        if (lLimitTest==0)
              printf ("\n\t......Limit Line2 Passed \n");

        else printf ("\n\t......Limit Line2 Failed \n");
  }
  else
  printf ("\n\t..Limit Test Pass\n");
}

void main()
{
  /*Program Variable*/
  ViStatus viStatus  = 0;
  long   lOpc =0L;

  /* Open a GPIB session at address 18*/
  viStatus=viOpenDefaultRM(&defaultRM);
  viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
  if(viStatus)
  {
        printf("Could not open a session to GPIB device at address 18!\n");
        exit(0);
  }
  /*Clear the instrument*/
  viClear(viESA);

  /*Reset the instrument*/
  viPrintf(viESA,"*RST\n");

  /* Check for the instrument model number and route the 50MHz signal accordingly*/
/*Route50MHzSignal();

  /*Display the program heading */
  printf("\n\t\t Limit Lines Program \n\n" );

  /*Set the Y-Axis Units to dBm */
  viPrintf(viESA, "UNIT:POW DBM\n");

  /*Set to Frequency Domain Mode*/
    viPrintf(viESA,"CALC:LLINE1:CONT:DOM FREQ\n");

  /*Delete any current limit line and define the upper limit line
    to have the following frequency/amplitude pairs*/
    viPrintf(viESA,"CALC:LLINE1:TYPE UPP\n");

  /* Turn on display*/
    viPrintf(viESA,"CALC:LLINE1:DISP ON\n");

  /*Send the upper limit line data*/
    viPrintf(viESA,"CALC:LLINE1:DATA 40E06,-50,1, 45E06,-20,1, 50E06,-15,1,
55E06,-20,1, 60E06,-50,1\n");

  /* Turn on display*/
    viPrintf(viESA,"CALC:LLINE1:DISP ON\n");

  /*Delete any current limit line and define the lower limit line
    to have the following frequency/amplitude pairs*/
    viPrintf(viESA,"CALC:LLINE2:TYPE LOW\n");

  /*Send the lower limit line data*/
    viPrintf(viESA,"CALC:LLINE2:DATA
40E06,-100,1,49.99E06,-100,1,50E06,-30,1,50.01E06,-100,1,60E06,-100,1\n");

  /* Turn on display*/
    viPrintf(viESA,"CALC:LLINE2:DISP ON\n");

  /*Turn the limit line test function on.*/
  viPrintf(viESA,"CALC:LLINE2:STAT ON\n");
```

```
/*Set the analyzer to a center frequency of 50 MHz, span to 20 MHz,
  and resolution bandwidth to 1 MHz.*/
   viPrintf(viESA,"SENS:FREQ:CENT 50e6\n");
   viPrintf(viESA,"SENS:FREQ:SPAN 20e6\n");
   viPrintf(viESA,"SENS:BWID:RES 1e6\n");

/*Set the analyzer reference level to 0 dBm*/
   viPrintf(viESA,"DISP:WIND:TRAC:Y:SCAL:RLEV 0 \n");

/* Check for the instrument model number and route the 50MHz-signal accordingly*/
Route50MHzSignal();

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");
/*Check for operation complete */

viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
/*Check to see if limit line passes or fails. It should pass.*/
printf ("\n\t Limit Line status after activating the 50MHz signal \n");

/*Print the limits line result*/
printResult();

/*Pause for 5 seconds*/
YIELD;

/*Deactivate the 50 MHz alignment signal.*/
viPrintf(viESA,"CAL:SOUR:STAT OFF\n");

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Check for operation complete */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
/* The limit line test should fail.*/
printf ("\n\t Limit Line status after de-activating the 50MHz signal \n");
```

```
  /*Print the limits line result*/
  printResult();

  /*Close the session*/
  viClose(viESA);
  viClose(defaultRM);
}
```

Measuring Noise

```
/*************************************************************/
/* Measuring Noise                                          */
/*                                                          */
/* This example is for the E44xxB ESA Spectrum Analyzers    */
/* and E740xA EMC Analyzers.                                */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens a GPIB session at address 18                     */
/* - Clears the Analyzer                                    */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Sets the center frequency and span                     */
/*      SENS:FREQ:CENT 50 MHZ                               */
/*      SENS:FREQ:SPAN 10 MHZ                               */
/* - Set the input port to the 50 MHz amplitude reference   */
/*      CAL:SOUR:STAT ON                                    */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                         */
/* - Trigger a sweep and wait for sweep completion          */
/*      INIT:IMM;*WAI                                       */
/* - Set the marker to the maximum peak                     */
/*      CALC:MARK:MAX                                       */
/* - Set the analyzer to active delta marker                */
/*      CALC:MARK:MODE DELT                                 */
/* - Set the delta marker to 2 MHZ                          */
/*      CALC:MARK:X 2E+6                                    */
/* - Activate the noise marker function                     */
/*      CALC:MARK:FUNC NOIS                                 */
/* - Trigger a sweep and wait for sweep completion          */
/*      INIT:IMM;*WAI                                       */
/* - Query the marker delta amplitude from the analyzer     */
/*      CALC:MARK:Y?                                        */
/* - Report the marker delta amplitude as the carrier to    */
/*   noise ratio in dBc/Hz                                  */
/* - Close the session                                      */
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>
```

```
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char    cEnter = 0;
int      iResult = 0;
long      lOpc =0L;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B amd E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
```

```
/*Program Variables*/
ViStatus viStatus  = 0;
double dMarkAmp =0.0;
long lOpc=0L;

/*Open a GPIB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
      printf("Could not open a session to GPIB device at address 18!\n");
      exit(0);
}
/*Clear the Instrument*/
viClear(viESA);

/*Reset the Instrument*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Noise Program \n\n" );

/* Check for the instrument model number and route the 50MHz signal accordingly*/
Route50MHzSignal();

/*Set the analyzer center frequency to 50MHz*/
viPrintf(viESA,"SENS:FREQ:CENT 50e6\n");

/*Set the analyzer span to 10MHz*/
viPrintf(viESA,"SENS:FREQ:SPAN 10e6\n");

/*Set the analyzer in a single sweep mode*/
viPrintf(viESA,"INIT:CONT 0 \n");

/*Trigger a sweep and wait for sweep completion*/
viPrintf(viESA,"INIT:IMM;*WAI \n");

/*Set the marker to the maximum peak*/
viPrintf(viESA,"CALC:MARK:MAX \n");

/*Check for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}

/*Set the analyzer in a single sweep mode*/
viPrintf(viESA,"INIT:CONT 0 \n");

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Set the analyzer in active delta marker mode*/
    viPrintf(viESA,"CALC:MARK:MODE DELT \n");

/*Set the marker delta frequency to 2 MHz. This places the
  active marker two divisions to the right of the input signal.*/
    viPrintf(viESA,"CALC:MARK:X 2E+6 \n");

/*Activate the noise marker function.*/
    viPrintf(viESA,"CALC:MARK:FUNC NOIS \n");

/*Trigger a sweep and wait for sweep completion*/
viPrintf(viESA,"INIT:IMM;*WAI \n");

/*Query and read the marker delta amplitude from the analyzer*/
viQueryf(viESA,"CALC:MARK:Y? \n","%lf",&dMarkAmp);

/*Report the marker delta amplitude as the carrier-to-noise ratio in dBc/Hz*/
printf("\t Marker Amplitude = %lf dBc/Hz\n",dMarkAmp);

/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Entering Amplitude Correction Data

```
/*************************************************************/
/* Entering Amplitude Correction Data                        */
/*                                                           */
/* This example is for the E44xxB ESA Spectrum Analyzers     */
/* and E740xA EMC Analyzers.                                 */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used are given as            */
/* reference.                                                */
/*                                                           */
/* - Opens a GPIB session at address 18                      */
/* - Clears the Analyzer                                     */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Sets the stop frequency to 1.5 GHz                      */
/*      SENS:FREQ:STOP 1.5 GHZ                               */
/* - Set the input port to the 50 MHz amplitude reference    */
/*      CAL:SOUR:STAT ON                                     */
/* - Enter amplitude correction frequency/amplitude pairs:   */
/*   0 Hz/ 0 dB, 100 MHz/5 dB, 1 GHz/-5 dB, 1.5 GHz/ 10 dB   */
/*      SENS:CORR:CSET1:DATA 0,0,100E6,5.0,1.0E9,-5.0,...    */
/* - Activate amplitude correction                           */
/*      SENS:CORR:CSET1:DATA                                 */
/*      SENS:CORR:CSET1:ALL:STAT ON                          */
/* - Query the analyzer for the amplitude corection factors */
/*      SENS:CORR:CSET1:DATA?                                */
/* - Store them in an array                                  */
/* - Display the array                                       */
/* - Close the session                                       */
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"
```

```
ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char     cEnter = 0;
int       iResult = 0;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B, and E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
/*Program Variables*/
    ViChar _VI_FAR cResult[1024] ={0};
ViReal64 _VI_FAR aRealArray[2][100] ={0};
ViStatus viStatus  = 0;
int  iNum =0;
int  iNoOfPoints =0;
long lCount = 0;
long lFreq=0L;
long lAmpltd=1;
    static ViChar *cToken;
```

```
/*No of amplitude corrections points */
iNoOfPoints = 4;

/* Open a GPIB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
      printf("Could not open a session to GPIB device at address 18!\n");
      exit(0);
}
/*Clear the instrument*/
viClear(viESA);

/*Reset the instrument*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Amplitude Correction Program \n\n" );

/*Set the stop frequency to 1.5 GHz */
viPrintf(viESA,"SENS:FREQ:STOP 1.5 GHz\n");

/* Check for the instrument model number and route the 50MHz signal accordingly*/
Route50MHzSignal();

/*  Purge any currently-loaded amplitude correction factors*/
    viPrintf(viESA,"SENS:CORR:CSET1:DEL \n");

/* Enter amp cor frequency/amplitude pairs:
   0 Hz, 0 dB, 100 MHz, 5 dB, 1 GHz, -5 dB, 1.5GHz,10*/
viPrintf(viESA,"SENS:CORR:CSET1:DATA ");
viPrintf(viESA,"0, 0.0,");
viPrintf(viESA,"100.E6, 5.0,");
viPrintf(viESA,"1.E9, -5.0,");
viPrintf(viESA,"1.5E9, 10  \n");

/* Activate amplitude correction. Notice that the noise floor slopes
up from 0 Hz to 100 MHz, then downward by 10 dB to 1 GHz, then upwards
again by 15 dB to 1.5 GHz.*/
    viPrintf(viESA,"SENS:CORR:CSET1:STATE ON \n");
    viPrintf(viESA,"SENS:CORR:CSET:ALL:STAT ON \n");

/*Query the analyzer for its amplitude correction factors */
   viQueryf(viESA,"SENS:CORR:CSET1:DATA?" , "%s" , &cResult);
```

```
/*Remove the "," from the amplitude correction for analyzing data*/
    cToken  = strtok(cResult,",");

/*Store the array (frequency) value into a two-dimensional real array*/
aRealArray[lFreq=0][lCount=0] = atof( cToken);

/*Remove the "," from the amplitude correction for analyzing data*/
cToken =strtok(NULL,",");

/*Store the array(amplitude) value into a two-dimensional real array*/
aRealArray[lAmpltd=1][lCount] = atof(cToken);
while (cToken != NULL)
{
      lCount++;
      if (lCount == iNoOfPoints)
      {
            lCount --;
            break;
      }
      /*Remove the "," from the amplitude correction for analyzing data*/
      cToken =strtok(NULL,",");

      /*Store the array (frequency) value into a two-dimensional real array*/
      aRealArray[lFreq][lCount] = atof(cToken);
      cToken =strtok(NULL,",");

      /*Store the array (amplitude) value into a two-dimensional real array*/
      aRealArray[lAmpltd][lCount] = atof(cToken);
}
/*Display the contents of the array.*/
for (long i=0;i<=lCount;i++)
{
      printf("\tFrequency of point[%d] = %f MHz\n",i,aRealArray[lFreq][i]/1e6);
      printf("\tAmplitude of point[%d] = %f dB\n",i,aRealArray[lAmpltd][i]);
}
/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Status Register–Determine When a Measurement is Done

```
/************************************************************/
/* Status Register - Determine when a measurement is done   */
/*                                                          */
/* This example is for the E44xxB ESA Spectrum Analyzers    */
/* and E740xA EMC Analyzers.                                */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens a GPIB session at address 18                     */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Clears the analyzer status byte                        */
/*      *CLS                                                */
/* - Sets the analyzer to single sweep mode                 */
/*      INIT:CONT 0                                         */
/* - Route the amplitude reference to the analyzer input    */
/*      CAL:SOUR:STAT ON                                    */
/* - Set the analyzer center frequency, span and Res BW     */
/*      SENS:FREQ:CENT 50 MHz                               */
/*      SENS:FREQ:SPAN 10 MHz                               */
/*      SENS:BAND:RES 300 kHz                               */
/* - Trigger a sweep and wait for completion of sweep       */
/*      INIT:IMM                                            */
/*      *OPC?                                               */
/* - Sets the service request mask to assert SRQ when       */
/*   either a measurement is uncalibrated or an error       */
/*   message has occurred.                                  */
/*      *SRE 96                                             */
/*      *ESE 35                                             */
/* - Set the computer to response to an interrupt           */
/* - Send an undefined command to the ESA                   */
/*       IDN (illegal command)                              */
/* - Wait for the SRQ                                       */
/* - When an interrupt occurs, poll all instruments         */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Pause 5 seconds to observe the analyzer                */
/* - Set the ESA to perform 80 video averages               */
/*       SENS:AVER:TYPE LPOW                                */
/*       SENS:AVER:COUN 80                                  */
/*       SENS:AVER:STAT ON                                     */
/* - Trigger a measurement, and set OPC bit when done          */
/*       INIT:IMM                                              */
/*       *OPC                                                  */
/* - Wait for the SRQ                                          */
/* - When an interrupt occurs, poll all instruments            */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Clear the status register enable                          */
/*      *SRE 0                                                 */
/* - Clear the status byte of the ESA                          */
/*      *CLS                                                   */
/* - Close the session                                         */
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <windows.h>
#include "visa.h"

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"
#define  YIELD Sleep(10)

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256] = {0};
ViAddr  iAddress;
char  cEnter =0;
int      iResult =0;
int       iSrqOccurred=0;
char  cBuf[3]={0};


/*Wait until SRQ is generated and for the handler to be called. Print
  something while waiting. When interrupt occurs it will be handled by
  interrupt handler*/
void WaitForSRQ()
{
long   lCount = 0L;
iSrqOccurred    =0;
```

```
printf ("\t\nWaiting for an SRQ to be generated ...");
for (lCount =0;(lCount<10) && (iSrqOccurred    ==0); lCount++)
{
      long lCount2 =0;
      printf(".");
      while ((lCount2++ < 100) && (iSrqOccurred   ==0))
      {
            YIELD;
      }
}
printf("\n");

}

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B and E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

/*Interrupt handler,trigger event handler */
ViStatus _VI_FUNCH  mySrqHdlr(ViSession viESA, ViEventType eventType, ViEvent
ctx,ViAddr userHdlr)
{
ViUInt16 iStatusByte;
```

```
/* Make sure it is an SRQ event, ignore if stray event*/
if (eventType!=VI_EVENT_SERVICE_REQ)
{
      printf ("\n Stray event type0x%1x\n",eventType);

      /*Return successfully*/
      return VI_SUCCESS;
}
/* When an interrupt occurs,determine which device generated the interrupt
(if an instrument other than the ESA generates the interrupt, simply report
"Instrument at GPIB Address xxx Has Generated an Interrupt").*/
printf ("\n\n SRQ event occurred!\n");
printf ("\n ... Original Device Session = %t\n",viESA);

/*Get the GPIB address of the insrument, which has interrupted*/
viQueryf(viESA,"SYST:COMM:GPIB:SELF:ADDR?\n","%t", cBuf);
printf ("\n Instrument at GPIB address %s has generated an interrupt!\n",cBuf);

/*Get the status byte*/
/* If the ESA generated the interrupt, determine the nature of the interrupt;
  did the measurement complete or an error message occur?*/
viQueryf(viESA, "*ESR?\n", "%d", &iStatusByte);
if ( (0x01 & iStatusByte))
      printf("\n SRQ message:\t Measurement complete\n");
else if ( (0x02 | 0x10 | 0x20 & iStatusByte ))
      printf ("\n SRQ message:\t Error Message Occurred\n");

/*Return successfully*/
iSrqOccurred   =1;
viReadSTB(viESA,&iStatusByte);
return VI_SUCCESS;
}
/* Main Program*/
void main()
{
/*Program Variables*/
ViStatus viStatus  = 0;
long   lOpc=0;

/* Open a GPIB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
int address =18;
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
        printf("Could not open a session to GPIB device at address 18!\n");
        exit(0);
}
/*Clear the instrument*/
viClear(viESA);

/*Reset the instrument*/
viPrintf(viESA,"*RST\n");

/*Clear the status byte of the instrument*/
viPrintf(viESA,"*CLS\n");

/*Display the program heading */
printf("\n\t Status Register - Determine When a Measurement is Done \n\n" );

/*Put the analyzer in a single sweep*/
viPrintf(viESA,"INIT:CONT 0 \n");

/* Check for the instrument model number and route the 50MHz-signal accordingly*/
Route50MHzSignal();

/*Set the analyzer to 50MHz center frequency*/
viPrintf(viESA,"SENS:FREQ:CENT 50 MHz\n");

/*Set the analyzer resolution bandwidth to 300 Khz*/
viPrintf(viESA,"SENS:BAND:RES 300 KHz\n");

/*Set the analyzer to 10MHz span*/
viPrintf(viESA,"SENS:FREQ:SPAN 10MHz\n");

/*Trigger a sweep*/
viPrintf(viESA,"INIT:IMM\n");

/*Make sure the previous command has been completed*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
        printf("Program Abort! error ocurred: last command was not completed!\n");
        exit(0);
}
/* Set the service request mask to assert SRQ when either a measurement
   is completed or an error message has occurred.*/
viPrintf(viESA,"*SRE 96\n");
viPrintf(viESA,"*ESE 35\n");

/* Configure the computer to respond to an interrupt*/
```

```
/*install the handler and enable it */
viInstallHandler(viESA, VI_EVENT_SERVICE_REQ, mySrqHdlr,iAddress);
viEnableEvent(viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR,VI_NULL);

/*Send an undefined command to the device*/
viPrintf(viESA,"IDN\n");

/*Wait for SRQ */
WaitForSRQ();

/* Pause 5 seconds to observe error message displayed on ESA*/
Sleep(5000);

/*Averaging the successive measurements, Set video averaging to 80 sweeps,
/*Turn the avarage On*/
viPrintf(viESA,":SENS:AVER:TYPE LPOW;:SENS:AVER:COUN 80;:SENS:AVER:STAT ON\n");

/* Set the service request mask to assert SRQ when either a measurement
   is completed or an error message has occurred.*/
viPrintf(viESA,"*SRE 96\n");
viPrintf(viESA,"*ESE 35\n");

/*Trigger the sweeps and set the *OPC bit after the sweeps are completed*/
viPrintf(viESA,":INIT:IMM;*OPC\n");

/*Wait for SRQ */
WaitForSRQ();

/*Disable and uninstall the interrupt handler*/
viDisableEvent  (viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR);
viUninstallHandler(viESA, VI_EVENT_SERVICE_REQ, mySrqHdlr,iAddress);


/*Clear the instrument status register*/
viPrintf(viESA,"*SRE 0 \n");

/*Clear the status byte of the instrument*/
viPrintf(viESA,"*CLS\n");

/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Determine if an Error has Occurred

```
/*********************************************************/
/* Determine if an error has occurred                    */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB session at address 18                  */
/* - Clears the Analyzer                                 */
/*      *CLS                                             */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Sets the service request mask to assert SRQ when    */
/*   either a measurement is uncalibrated or an error    */
/*   message has occurred.                               */
/*      STAT:QUES:ENAB 512                               */
/*      STAT:QUES:INT:ENAB 8                             */
/*      *ESE 35                                          */
/*      *SRE 104                                         */
/* - Set the center frequency to 500MHz and span to 100MHz  */
/*      SENS:FREQ:CENT 500 MHZ                           */
/*      SENS:FREQ:SPAN 100 MHZ                           */
/* - Set the analyzer to an uncalibrated state           */
/* - When an interrupt occurs, poll all instruments      */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Pause 5 seconds to observe the analyzer             */
/* - Sets the service request mask to assert SRQ when    */
/*   either a measurement is uncalibrated or an error    */
/*   message has occurred.                               */
/*      *ESE 35                                          */
/*      *SRE 96                                          */
/* - Send an illegal command to the ESA                  */
/*       IDN  (illegal command)                          */
/* - When an interrupt occurs, poll all instruments      */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Clear the analyzer status registers                 */
/*      *SRE 0                                           */
/*      *ESE 0                                           */
/*      STAT:QUES:ENAB 0                                 */
/*      STAT:QUES:INT:ENAB 0                                 */
/*      *CLS                                                 */
/* - Continue monitoring for an interrupt                    */
/* - Close the session                                       */
/************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <windows.h>
#include "visa.h"

#define   hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define   hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define  hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"
#define   YIELD Sleep(10)

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256] = {0};
char  cEnter =0;
int      iResult =0;
int       iSrqOccurred = 0;
char  cBuf[3]={0};


/*Wait until SRQ is generated and for the handler to be called. Print
 something while waiting. When interrupt occurs it will be handled by
 interrupt handler*/
void WaitForSRQ()
{
long   lCount = 0L;
iSrqOccurred    =0;

printf ("\t\nWaiting for an SRQ to be generated ...");
for (lCount =0;(lCount<10) && (iSrqOccurred    ==0); lCount++)
{
      long lCount2 =0;
      printf(".");
      while ((lCount2++ < 100) && (iSrqOccurred   ==0))
      {
            YIELD;
      }
}
printf("\n");

}

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B and E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

/*Interrupt handler,trigger event handler */
ViStatus _VI_FUNCH  sSrqHdlr(ViSession viESA, ViEventType eventType, ViEvent
ctx,ViAddr userHdlr)
{
ViUInt16 iStatusByte=0;
long lCond = 0L;

/* Make sure it is an SRQ event, ignore if stray event*/
if (eventType!=VI_EVENT_SERVICE_REQ)
{
      printf ("\n Stray event type0x%1x\n",eventType);
      /*Return successfully*/
      return VI_SUCCESS;
}
```

```
/* When an interrupt occurs, determine which device generated the interrupt
   (if an instrument other than the ESA generates the interrupt, simply report
   "Instrument at GPIB Address xxx Has Generated an Interrupt").*/
printf ("\n SRQ Event Occurred!\n");
printf ("\n ... Original Device Session = %1d\n",viESA);

/*Get the GPIB address of the insrument, which has interrupted*/
viQueryf(viESA,"SYST:COMM:GPIB:SELF:ADDR?\n","%t", cBuf);
printf ("\n Instrument at GPIB Address %s Has Generated an Interrupt!\n",cBuf);

/*Get the status byte*/
/* If the ESA generated the interrupt, determine the nature of the interrupt
   either a measurement is uncalibrated or an error message has occurred?*/
viQueryf(viESA, "STAT:QUES:INT:EVEN?\n", "%d", &iStatusByte);
if ( (0x08 & iStatusByte))
      printf("\n SRQ message:\t Measurement uncalibrated\n");

/* If the ESA generated the interrupt, determine the nature of the interrupt;
  did is the measurement complete or an error message occur?*/
viQueryf(viESA, "*ESR?\n", "%d", &iStatusByte);
if ( (iStatusByte !=0) && (0x01 & iStatusByte))
      printf("\n SRQ message:\t Measurement complete\n");
else if ( (iStatusByte !=0) && (0x02 | 0x10 | 0x20 & iStatusByte ))
      printf ("\n SRQ message:\t Error Message Occurred\n");

/*Return successfully*/
iSrqOccurred   =1;
viReadSTB(viESA, &iStatusByte);
return VI_SUCCESS;
}

void main()
{
/*Program Variables*/
ViStatus viStatus  = 0;
long     lOpc= 0L;
int      iIntNum= 0;
long    lCount= 0L;

/* Open a GPIB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
      printf("Could not open a session to GPIB device at address 18!\n");
      exit(0);
}
/*Clear the instrument*/
viClear(viESA);

/*Reset the instrument*/
viPrintf(viESA,"*RST\n");

/*Clear the status byte of the instrument*/
viPrintf(viESA,"*CLS\n");

/*Display the program heading */
printf("\n\t\t Status register - Determine if an Error has Occurred\n\n" );

/* Check for the instrument model number and route the 50MHz-signal accordingly*/
Route50MHzSignal();

/*Put the analyzer in single sweep*/
viPrintf(viESA,"INIT:CONT 0 \n");

/*Set the service request mask to assert SRQ when either a measurement
is uncalibrated (i.e. "Meas Uncal" displayed on screen) or an error
message has occurred.*/
viPrintf(viESA,"STAT:QUES:ENAB 512\n");
 viPrintf(viESA,"STAT:QUES:INT:ENAB 8\n");
viPrintf(viESA,"*ESE 35\n");
viPrintf(viESA,"*SRE 104\n");

/*Configure the computer to respond to an interrupt,install the handler
  and enable it */
viInstallHandler(viESA, VI_EVENT_SERVICE_REQ, sSrqHdlr,ViAddr(10));
viEnableEvent(viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR,VI_NULL);
iSrqOccurred    =0;

/*Set the analyzer to a 500 MHz center frequency*/
viPrintf(viESA,"SENS:FREQ:CENT 500 MHZ \n");

/*Set the analyzer to a 100 MHz span*/
viPrintf(viESA,"SENS:FREQ:SPAN 100 MHZ\n");

/*Set the analyzer to a  auto resolution BW*/
viPrintf(viESA,"SENS:BAND:RES:AUTO 1\n");

/*Set the analyzer to a  Auto Sweep Time*/
viPrintf(viESA,"SENS:SWE:TIME:AUTO 1\n");

/*Allow analyzer to sweep several times.*/
```

```
viPrintf(viESA,"INIT:CONT 1 \n");

/*Manually couple sweeptime to 5ms. reduce resolution BW to 30 KHz.
"Meas Uncal" should be displayed on the screen, and an interrupt should
 be generated.*/
viPrintf(viESA,"SENS:SWE:TIME 5 ms \n");
viPrintf(viESA,"SENS:BAND:RES 30 KHZ  \n");

/*Wait for SRQ*/
WaitForSRQ();

/*Pause for 5 seconds to observe "Meas Uncal" message on ESA display*/
Sleep(5000);

/* Set the service request mask to assert SRQ when either a measurement
   is completed or an error message has occurred.*/
viPrintf(viESA,"*SRE 96\n");
viPrintf(viESA,"*ESE 35\n");

/*Send an undefined command to the device*/
viPrintf(viESA,"IDN\n");

/*Wait for SRQ*/
WaitForSRQ();

/*Disable and uninstall the interrupt handler*/
viDisableEvent  (viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR);
viUninstallHandler(viESA, VI_EVENT_SERVICE_REQ, sSrqHdlr,ViAddr(10));

/*Clear the instrument status register*/
viPrintf(viESA,"*SRE 0 \n");
viPrintf(viESA,"*ESE 0 \n");
viPrintf(viESA,"STAT:QUES:ENAB 0\n");
 viPrintf(viESA,"STAT:QUES:INT:ENAB 0\n");

/*Clear the status byte of the instrument*/
viPrintf(viESA,"*CLS\n");

/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Measuring Harmonic Distortion (GPIB)

```
/*************************************************************/
/* Measuring Harmonic Distortion (GPIB)                      */
/*                                                           */
/* This example is for the E44xxB ESA Spectrum Analyzers     */
/* and E740xA EMC Analyzers.                                 */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used are given as            */
/* reference.                                                */
/*                                                           */
/* - Opens a GPIB session at address 18                      */
/* - Clears the Analyzer                                     */
/*      *CLS                                                 */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Set the input port to the 50 MHz reference              */
/*      CAL:SOUR:STAT ON                                     */
/* - Set the analyzer center frequency to the fundamental    */
/*      SENS:FREQ:CENT freq                                  */
/* - Set the analyzer to 10 MHz span                         */
/*      SENS:FREQ:SPAN 10 MHZ                                */
/* - Set the analyzer to single sweep mode                   */
/*      INIT:CONT 0                                          */
/* - Take a sweep and wait for sweep completion              */
/*      INIT:IMM;*WAI                                        */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Set the marker to reference level                       */
/*      CALC:MARK:SET:RLEV                                   */
/* - Take a sweep and wait for sweep completion              */
/*      INIT:IMM;*WAI                                        */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Change VISA timeout to 60 seconds                       */
/* - Activate signal track                                   */
/*      CALC:MARK:TRCK:STAT ON                               */
/* - Perform narrow span and wait                            */
/*      SENS:FREQ:SPAN 10e4                                  */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - De-activate signal track                                */
/*      CALC:MARK:TRCK:STAT OFF                              */
/* - Reset VISA timeout to 3 seconds                         */
/* - Set units to dBm                                        */
/*      UNIT:POW DBM                                         */
/* - Take a sweep and wait for sweep completion              */
/*      INIT:IMM;                                            */
/*      *OPC?                                                */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Read the marker amplitude,this is the fundamental Level*/
/*      CALC:MARK:Y?                                         */
/* - Change the amplitude units to volts                     */
/*      UNIT:POW V                                           */
/* - Take a sweep                                            */
/*      INIT:IMM                                             */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - Read the marker amplitude in volts, this is the         */
/*   fundamental amplitude in volts.                         */
/*      CALC:MARK:Y?                                         */
/* - Read the marker frequency                               */
/*      CALC:MARK:X?                                         */
/* - Measure each harmonic amplitude as follows:             */
/*     Set the span to 20 MHz                                */
/*        SENS:FREQ:SPAN 20 MHZ                              */
/*     Set the center frequency to the desired harmonic      */
/*        SENS:FREQ:CENT <freq>                              */
/*     Take a sweep and wait for operation complete          */
/*        INIT:IMM                                           */
/*        *OPC?                                              */
/*     Perform peak search                                   */
/*        CALC:MARK:MAX                                      */
/*     Set VISA timeout to 60 seconds                        */
/*     Activate signal track                                 */
/*        CALC:MARK:TRCK:STAT ON                             */
/*     Zoom down to a 100 kHz span                           */
/*        SENS:FREQ:SPAN 10E4                                */
/*     Take a sweep and wait for operation complete          */
/*        INIT:IMM                                           */
/*        *OPC?                                              */
/*     Signal track off                                      */
/*        CALC:MARK:TRCK:STAT OFF                            */
/*     Reset VISA timeout to 3 seconds                       */
/*     Perform Peak Search                                   */
/*        CALC:MARK:MAX                                      */
/*     Set marker amplitude in volts                         */
/*        UNIT:POW V                                         */
/*      Query, read the marker amplitude in volts              */
/*         CALC:MARK:Y?                                        */
/*      Change the amplitude units to dBm and read the         */
/*      marker amplitude.                                      */
/*         UNIT:POW DBM                                        */
/* - Calculate the relative amplitude of each harmonic         */
/*   reletive to the fundamental                               */
/* - Calculate the total harmonic distortion                   */
/* - Display the fundamental amplitude in dBm, fundamental     */
/*   frequency in MHz, relative amplitude of each harmonic     */
/*   in dBc and total harmonic distortion in percent           */
/* - Close the session                                         */
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char    cEnter = 0;
int    iResult = 0;
long      lOpc =0L;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz amplitude reference for the models*/
     /*E4401B, E4411B, and E7401A*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
     /* to connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf ("......Press Return to continue \n");
     scanf( "%c",&cEnter);
     /*Externally route the 50MHz Signal*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void TakeSweep()
{
/*Take a sweep and wait for the sweep completion*/
viPrintf(viESA,"INIT:IMM\n");
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! Error occurred: last command was not completed! \n");
     exit(0);
}
}

void main()
{
/*Program Variables*/
ViStatus viStatus  = 0;
double dFundamental = 0.0;
double dHarmFreq =0.0;
float fHarmV[10] ={0.0};
float fHarmDbm[10]={0.0};
float fRelAmptd[10]={0.0};
float fFundaAmptdDbm=0.0;
double dFundaAmptdV=0.0;
double dMarkerFreq = 0.0;
double dPrcntDistort =0.0;
double dSumSquare =0.0;
long   lMaxHarmonic =0L;
long   lNum=0L;

/*Setting default values*/
lMaxHarmonic =5;
dFundamental =50.0;
```

```
/* Open a GPIB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
     printf("Could not open a session to GPIB device at address 18!\n");
     exit(0);
}
/*Clear the instrument*/
viClear(viESA);

/*Reset the instrument*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Harmonic Distortion Program \n\n" );

/* Check for the instrument model number and route the 50MHz-signal accordingly*/
Route50MHzSignal();

/*Prompt user for fundamental frequency*/
printf("\t Enter the input signal fundamental frequency in MHz ");

/*The user enters fundamental frequency*/
scanf("%lf",&dFundamental);

/*Set the analyzer center frequency to the fundamental frequency. */
viPrintf(viESA,"SENS:FREQ:CENT %lf MHZ \n;",dFundamental);

/*Set the analyzer to 10MHz Span */
viPrintf(viESA,"SENS:FREQ:SPAN 10 MHZ\n");

/*Put the analyzer in a single sweep  */
viPrintf(viESA,"INIT:CONT 0 \n");

/*Trigger a sweep, wait for sweep completion*/
viPrintf(viESA,"INIT:IMM;*WAI\n");

/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX \n");

/* Place the signal at the reference level using the
   marker-to-reference level command and take sweep */
viPrintf(viESA,"CALC:MARK:SET:RLEV \n");

/*Trigger a sweep, wait for sweep completion*/
```

```
viPrintf(viESA,"INIT:IMM;*WAI\n");

/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX \n");

/*increase timeout to 60 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

/*Perform activate signal track */
viPrintf(viESA,"CALC:MARK:TRCK:STAT ON \n");

/*Take a sweep and wait for the sweep completion*/
TakeSweep();

/*Perform  narrow span and wait */
viPrintf(viESA,"SENS:FREQ:SPAN 10e4 \n");

/*Take a sweep and wait for the sweep completion*/
TakeSweep();

/*De activate the signal track */
viPrintf(viESA,"CALC:MARK:TRCK:STAT OFF \n");

/*Reset timeout to 3 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

/*Set units to DBM*/
viPrintf(viESA,"UNIT:POW DBM \n");

/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX \n");

/*Read the marker amplitude, this is the fundamental amplitude
  in dBm */
viQueryf(viESA,"CALC:MARK:Y?\n","%lf", &fFundaAmptdDbm);

/*Change the amplitude units to Volts */
viPrintf(viESA,"UNIT:POW V \n");

/*Read the marker amplitude in volts, This is the fundamental amplitude
  in Volts (necessary for the THD calculation).*/
viQueryf(viESA,"CALC:MARK:Y?\n","%lf",&dFundaAmptdV);

/*Read the marker frequency. */
viQueryf(viESA,"CALC:MARK:X? \n","%lf",&dMarkerFreq);
dFundamental = dMarkerFreq;
```

```
/*Measure each harmonic amplitude as follows: */
for ( lNum=2;lNum<=lMaxHarmonic;lNum++)
{
     /*Measuring the Harmonic No#[%d] message */
     printf("\n\t Measuring the Harmonic No [%d] \n",lNum );

     /*Set the span to 20 MHz*/
     viPrintf(viESA,"SENS:FREQ:SPAN 20 MHZ \n");

     /*Set the center frequency to the nominal harmonic frequency*/
     dHarmFreq = lNum*dFundamental;
     viPrintf(viESA,"SENS:FREQ:CENT %lf HZ \n;",dHarmFreq);

     /*Take a sweep and wait for the sweep completion*/
     TakeSweep();

     /*Perform a peak search and wait for completion */
     viPrintf(viESA,"CALC:MARK:MAX\n");

     /*increase timeout to 60 sec*/
     viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

     /*Activate signal track */
     viPrintf(viESA,"CALC:MARK:TRCK:STAT ON\n");

     /*Zoom down to a 100 kHz span */
     viPrintf(viESA,"SENS:FREQ:SPAN 10e4\n");

     /*Take a sweep and  wait for the sweep completion*/
     TakeSweep();

     /* Signal track off */
     viPrintf(viESA,"CALC:MARK:TRCK:STAT OFF\n");

     /*Reset timeout to 3 sec*/
     viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

     /*Set Marker Amplitude in Volts*/
     viPrintf(viESA,"UNIT:POW V\n");

     /*Perform a peak search and  wait for completion*/
     viPrintf(viESA,"CALC:MARK:MAX\n");

     /*Query and read the Marker Amplitude in Volts*/
     /*Store the result in the array.*/
     viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmV[lNum]);

     /*Change the amplitude units to DBM  */
     viPrintf(viESA,"UNIT:POW DBM\n");

     /* Read the marker amplitude */
     viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmDbm[lNum]);
     }

/*Sum the square of each element in the fHarmV array. Then
  calculate the relative amplitude of each harmonic relative
  to the fundamental */
for (lNum=2;lNum<=lMaxHarmonic;lNum++)
{
     dSumSquare= dSumSquare + (pow (double(fHarmV[lNum]) ,2.0));
     /* Relative Amplitude  */
     fRelAmptd[lNum] = fHarmDbm[lNum] - fFundaAmptdDbm ;
}
/*Calculate the total harmonic distortion by dividing the square root of
the sum of the squares (dSumSquare) by the fundamental amplitude in Volts
(dFundaAmptdV).Multiply this value by 100 to obtain a result in percent*/
dPrcntDistort = ((sqrt(double (dSumSquare)))  /dFundaAmptdV) *100 ;

/*Fundamental amplitude in dBm */
printf("\n\t Fundamental Amplitude: %lf dB \n\n",fFundaAmptdDbm);

/*Fundamental Frequency in MHz*/
printf("\t Fundamental Frequency is: %lf MHz \n\n",dFundamental/10e5);

/*Relative amplitude of each harmonic in dBc*/
for (lNum=2;lNum<=lMaxHarmonic;lNum++)
     printf("\t Relative amplitude of Harmonic[%d]: %lf dBc
\n\n",lNum,fRelAmptd[lNum]);

/*Total harmonic distortion in percent*/
printf("\t Total Harmonic Distortion: %lf percent \n\n",dPrcntDistort);

/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Measuring Harmonic Distortion (RS-232)

```
/*************************************************************/
/* Measuring Harmonic Distortion (RS-232)                   */
/*                                                           */
/* This example is for the E44xxB ESA Spectrum Analyzers    */
/* and E740xA EMC Analyzers.                                 */
/*                                                           */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                                */
/*                                                           */
/* - Opens an RS-232 session to the COM1 serial port        */
/* - Clears the Analyzer                                     */
/*      *CLS                                                 */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Set the input port to the 50 MHz reference             */
/*      CAL:SOUR:STAT ON                                     */
/* - Set the analyzer center frequency to the fundamental   */
/*      SENS:FREQ:CENT freq                                  */
/* - Set the analyzer to 10 MHz span                        */
/*      SENS:FREQ:SPAN 10 MHZ                                */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                          */
/* - Trigger a sweep and wait for sweep completion          */
/*      INIT:IMM;*WAI                                        */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Set the marker to reference level                      */
/*      CALC:MARK:SET:RLEV                                   */
/* - Trigger a sweep and wait for sweep completion          */
/*      INIT:IMM;*WAI                                        */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Change VISA timeout to 60 seconds                      */
/* - Activate signal track                                   */
/*      CALC:MARK:TRCK:STAT ON                               */
/* - Perform narrow span and wait                           */
/*      SENS:FREQ:SPAN 10e4                                  */
/* - Check for operation complete                           */
/*      *OPC?                                                */
/* - De-activate signal track                               */
/*      CALC:MARK:TRCK:STAT OFF                              */
/* - Reset VISA timeout to 3 seconds                          */
/* - Set units to dBm                                         */
/*      UNIT:POW DBM                                          */
/* - Take a sweep and wait for sweep completion               */
/*      INIT:IMM;                                             */
/*      *OPC?                                                 */
/* - Perform the peak search                                  */
/*      CALC:MARK:MAX                                         */
/* - Read the marker amplitude,this is the fundamental Level*/
/*      CALC:MARK:Y?                                          */
/* - Change the amplitude units to volts                      */
/*      UNIT:POW V                                            */
/* - Take a sweep                                             */
/*      INIT:IMM                                              */
/* - Check for operation complete                             */
/*      *OPC?                                                 */
/* - Read the marker amplitude in volts, this is the          */
/*   fundamental amplitude in volts.                          */
/*      CALC:MARK:Y?                                          */
/* - Read the marker frequency                                */
/*      CALC:MARK:X?                                          */
/* - Measure each harmonic amplitude as follows:              */
/*     Set the span to 20 MHz                                 */
/*        SENS:FREQ:SPAN 20 MHZ                               */
/*     Set the center frequency to the desired harmonic       */
/*        SENS:FREQ:CENT <freq>                               */
/*     Take a sweep and wait for operation complete           */
/*        INIT:IMM                                            */
/*        *OPC?                                               */
/*     Perform peak search                                    */
/*        CALC:MARK:MAX                                       */
/*     Set VISA timeout to 60 seconds                         */
/*     Activate signal track                                  */
/*        CALC:MARK:TRCK:STAT ON                              */
/*     Zoom down to a 100 kHz span                            */
/*        SENS:FREQ:SPAN 10E4                                 */
/*     Take a sweep and wait for operation complete           */
/*        INIT:IMM                                            */
/*        *OPC?                                               */
/*     Signal track off                                       */
/*        CALC:MARK:TRCK:STAT OFF                             */
/*     Reset VISA timeout to 3 seconds                        */
/*     Perform Peak Search                                    */
/*        CALC:MARK:MAX                                       */
/*     Set marker amplitude in volts                          */
/*        UNIT:POW V                                          */
/*     Query, read the marker amplitude in volts            */
/*        CALC:MARK:Y?                                      */
/*     Change the amplitude units to dBm and read the       */
/*     marker amplitude.                                    */
/*        UNIT:POW DBM                                      */
/* - Calculate the relative amplitude of each harmonic      */
/*   reletive to the fundamental                            */
/* - Calculate the total harmonic distortion                */
/* - Display the fundamental amplitude in dBm, fundamental  */
/*   frequency in MHz, relative amplitude of each harmonic  */
/*   in dBc and total harmonic distortion in percent        */
/* - Close the session                                      */
/***********************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <visa.h>

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char    cEnter = 0;
int      iResult = 0;
long      lOpc =0L ;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B, E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON\n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON\n");
}
}

void TakeSweep()
{
/*Take a sweep and wait for the sweep completion*/
viPrintf(viESA,"INIT:IMM\n");
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! Error occurred: last command was not completed! \n");
      exit(0);
}
}

void main()
{
/*Program Variables*/
ViStatus viStatus  = 0;
double dFundamental = 0.0;
double dHarmFreq = 0.0;
float fHarmV[10] ={0.0};
float fHarmDbm[10]={0.0};
float fRelAmptd[10]={0.0};
float fFundaAmptdDbm=0.0;
double dFundaAmptdV=0.0;
double dMarkerFreq = 0.0;
double dPrcntDistort =0.0;
double dSumSquare =0.0;
long   lMaxHarmonic =0L;
long   lNum=0L;


/*Setting default values*/
lMaxHarmonic =5;
```

```
dFundamental =50.0;


/* Open a serial session at COM1 */
viStatus=viOpenDefaultRM(&defaultRM);
if (viStatus =viOpen(defaultRM,"ASRL1::INSTR",VI_NULL,VI_NULL,&viESA) !=
VI_SUCCESS)
{
      printf("Could not open a session to ASRL device at COM1!\n");
      exit(0);
}
/*Clear the instrument*/
viClear(viESA);

/*Reset the instrument*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Harmonic Distortion Program \n\n" );

/* Check for the instrument model number and route the 50MHz-signal accordingly*/
Route50MHzSignal();

/*Prompt user for fundamental frequency*/
printf("\t Enter the input signal fundamental frequency in MHz ");

/*The user enters fundamental frequency*/
scanf("%lf",&dFundamental);

/*Set the analyzer center frequency to the fundamental frequency. */
viPrintf(viESA,"SENS:FREQ:CENT %lf MHZ\n",dFundamental);

/*Set the analyzer to 10MHz Span */
viPrintf(viESA,"SENS:FREQ:SPAN 10 MHZ\n");

/*Put the analyzer in a single sweep mode */
viPrintf(viESA,"INIT:CONT 0\n");

/*Trigger a sweep, wait for sweep completion*/
viPrintf(viESA,"INIT:IMM;*WAI\n");

/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX\n");

/* Place the signal at the reference level using the
   marker-to-reference level command and take sweep */
```

```
viPrintf(viESA,"CALC:MARK:SET:RLEV\n");

/*Trigger a sweep, wait for sweep completion*/
viPrintf(viESA,"INIT:IMM;*WAI\n");

/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX\n");

/*Increase timeout to 60 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

/*Perform activate signal track */
viPrintf(viESA,"CALC:MARK:TRCK:STAT ON\n");

/*Take a sweep and wait for the sweep completion*/
 TakeSweep();

/*Perform  narrow span and  wait */
viPrintf(viESA,"SENS:FREQ:SPAN 10e4\n");

/*Take a sweep and wait for the sweep completion*/
TakeSweep();

/*De activate the signal track */
viPrintf(viESA,"CALC:MARK:TRCK:STAT OFF\n");

/*Reset timeout to 3 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

/*Set units to dBm*/
viPrintf(viESA,"UNIT:POW DBM\n");

/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX\n");

/*Read the marker amplitude, this is the fundamental amplitude
  in dBm */
viQueryf(viESA,"CALC:MARK:Y?\n","%lf", &fFundaAmptdDbm);

/*Change the amplitude units to Volts */
viPrintf(viESA,"UNIT:POW V\n");

/*Read the marker amplitude in volts, This is the fundamental amplitude
  in Volts (necessary for the THD calculation).*/
viQueryf(viESA,"CALC:MARK:Y?\n","%lf",&dFundaAmptdV);
```

```
/*Read the marker frequency. */
viQueryf(viESA,"CALC:MARK:X? \n","%lf",&dMarkerFreq);
dFundamental = dMarkerFreq;

/*Measure each harmonic amplitude as follows: */
for ( lNum=2;lNum<=lMaxHarmonic;lNum++)
{
      /*Measuring the Harmonic No#[%d] message */
      printf("\n\t Measuring the Harmonic No [%d] \n",lNum );

      /*Set the span to 20 MHz*/
      viPrintf(viESA,"SENS:FREQ:SPAN 20 MHZ\n");

      /*Set the center frequency to the nominal harmonic frquency*/
      dHarmFreq = lNum * dFundamental;
      viPrintf(viESA,"SENS:FREQ:CENT %lf HZ\n",dHarmFreq);

      /*Take a sweep and wait for the sweep completion*/
      TakeSweep();

      /*Perform a peak search and  wait for completion */
      viPrintf(viESA,"CALC:MARK:MAX\n");

      /*Increase timeout to 60 sec*/
      viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

      /*Activate signal track */
      viPrintf(viESA,"CALC:MARK:TRCK:STAT ON\n");

      /*Zoom down to a 100 KHz span */
      viPrintf(viESA,"SENS:FREQ:SPAN 10e4\n");

      /*Take a sweep and wait for the sweep completion*/
      TakeSweep();

      /* Signal track off */
      viPrintf(viESA,"CALC:MARK:TRCK:STAT OFF\n");

      /*Reset timeout to 3 sec*/
      viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

      /*Set marker amplitude in Volts*/
      viPrintf(viESA,"UNIT:POW V\n");

      /*Perform a peak search and wait for completion*/
      viPrintf(viESA,"CALC:MARK:MAX\n");
```

```
        /*Query and read the marker amplitude in Volts*/
        /*Store the result in the fHarmV array.*/
        viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmV[lNum]);

        /*Change the amplitude units to dBm */
        viPrintf(viESA,"UNIT:POW DBM\n");

        /* Read the marker amplitude */
        viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmDbm[lNum]);
        }

/*Sum the square of each element in the fHarmV array and calculate
  the relative amplitude of each harmonic relative to the fundamental*/
for (lNum=2;lNum<=lMaxHarmonic;lNum++)
{
        dSumSquare= dSumSquare + (pow (double(fHarmV[lNum]) ,2.0));

        /* Relative Amplitude  */
        fRelAmptd[lNum] = fHarmDbm[lNum] - fFundaAmptdDbm ;
}
/*Calculate the total harmonic distortion by dividing the square root of
the sum of the squares (dSumSquare) by the fundamental amplitude in Volts
(dFundaAmptdV).Multiply this value by 100 to obtain a result in percent*/
dPrcntDistort = ((sqrt(double (dSumSquare)))  /dFundaAmptdV) *100 ;

/*Fundamental amplitude in dBm */
printf("\nFundamental Amplitude: %lf dB \n",fFundaAmptdDbm);

/*Fundamental frequency in MHz*/
printf("Fundamental Frequency is: %lf MHz \n",dFundamental/10e5);

/*Relative amplitude of each harmonic in dBc*/
for (lNum=2;lNum<=lMaxHarmonic;lNum++)
        printf("Relative amplitude of Harmonic[%d]: %lf dBc
\n",lNum,fRelAmptd[lNum]);

/*Total harmonic distortion in percent*/
printf("Total Harmonic Distortion: %lf percent\n",dPrcntDistort);

/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Making Faster Measurements (multiple measurements)

```
/*************************************************************/
/*  Average.c   Agilent Technologies 1999                    */
/*                                                           */
/* This C programming example does the following:            */
/* Performs Power Averaging of Multiple ESA Measurements     */
/* and Writes the Result back to a Trace for display         */
/*                                                           */
/* The required SCPI instrument commands are given as        */
/* reference.                                                */
/*                                                           */
/* - Opens a GPIB device at address 18                       */
/* - Clears and Resets the Analyzer to a known state         */
/*      SYST:PRES:TYPE FACT                                  */
/*      *RST                                                 */
/* - Identify the Instrument model                           */
/*      *IDN?                                                */
/* - Sets the analyzer center frequency and span             */
/*      SENS:FREQ:CENT freq                                  */
/*      SENS:FREQ:SPAN freq                                  */
/* - Sets the analyzer resolution bandwidth                  */
/*      SENS:BAND rbw                                        */
/* - Selects sampled as the detector mode                    */
/*      SENS:DET SAMP                                        */
/* - Disable optional Input/Output functions                 */
/*     :SYST:PORT:IFVS:ENAB OFF                             */
/* - Turn off auto-alignment                                 */
/*      CAL:AUTO OFF                                         */
/* - Select the desired number of sweep points               */
/*      SWE:POINTS points                                    */
/* - Select the appropriate display reference level and     */
/*     amplitude reference routing                           */
/*   E4402B/03B/04B/05B/07B/08B or E7402A/03A/04A/05A        */
/*      DISP:WIND:TRAC:Y:RLEV -20 DBM                        */
/*      CAL:SOUR:STAT ON                                     */
/*   E4401B, E4411B, or E7401A                               */
/*      DISP:WIND:TRAC:Y:RLEV -25 DBM    */
/*      CAL:SOUR:STAT ON;                                    */
/* - Select single sweep mode                                */
/*      INIT:CONT OFF                                        */
/* - Disable local display                                   */
/*      DISP:ENAB OFF                                        */
/* - Select internal machine binary data format (milli-dBm) */
/*      FORM:DAT INT,32                                         */
/* - Select appropriate byte order (Intel)                      */
/*      FORM:BORD SWAP                                          */
/* - Repeat the following the requested number of times:        */
/*    - Trigger a measurement and wait for completion           */
/*      INIT:*OPC?                                              */
/*    - Read the resulting measurement trace                    */
/*      TRAC:DATA? TRACE1                                       */
/* - Compute running averaged power at all trace points         */
/* - Display measurement statistics                             */
/* - Write averaged data to second trace display                */
/*      TRAC:DATA TRACE2 <definite length block of data>        */
/* - Enable viewing of second trace                             */
/*      TRACE2:MODE VIEW                                        */
/* - Enable local display for viewing                           */
/*      DISP:ENAB ON                                            */
/* - Select continuous sweep mode                               */
/*      INIT:CONT ON                                            */
/* - Close session and Return instrument to local control       */
/**************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <sys\timeb.h>
#include <visa.h>

#define   hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define   hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define  hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

#define NUM_TRACES 100     /* number of traces to average
*/
#define NUM_POINTS 401     /* requested number of points/trace
*/
#define CENTER 50          /* center frequency in MHz, an integer
*/
#define SPAN 20           /* span frequency in MHz, an integer                 */
#define RBW 300           /* resolution BW in kHz, an integer                  */

#define DISPLAY  0         /* ESA display enable, disable for speed
*/

#define DATA_LENGTH     4   /* number of data bytes in one trace point
*/
#define MAX_POINTS 8192     /* maximum number of points/trace in ESA
```

```
*/

int iNumTraces = NUM_TRACES,/* number of traces to average
*/
    iRbw = RBW,              /* resolution bandwidth                              */
    iNumPoints = NUM_POINTS,/* actual number of trace points per sweep
*/
    iSpan = SPAN,            /* Analyzer Frequency Span in MHz                    */
    iCenter = CENTER;        /* Analyzer Center frequency in MHz
*/

int iResult =0;

unsigned long lRetCount;    /* the number of bytes transferred in one trace record
*/

double dDelta, dTimePer, dPower;

struct timeb start_time, stop_time, elapsed_time;

char cCommand[100];
char cBuffer[100];
char cEnter;
double dPwrAvgArray[MAX_POINTS];

ViUInt32 iHeaderLength,     /* header is "#nyyy..." n is number of chars in yyy,
*/
                           /* yyy is the total data length in bytes              */
         iArrayLength,      /* iArrayLength is number of bytes of data
*/
         iTermLength = 1,   /* the response message includes a LF character
*/
         iBlockSize,        /* number of bytes expected in one trace definite block
*/
         iTotalRetCount;    /* total number of bytes actually transferred
*/

ViSession defaultRM, viESA;

                           /* reserve space for the header, data and terminator    */
ViChar cInBuffer[sizeof("#nyyyyl") + (MAX_POINTS * DATA_LENGTH) ];
ViChar cOutBuffer[sizeof("TRAC:DATA TRACE2,#nyyyyl") + (MAX_POINTS * DATA_LENGTH
) ];

/******************   Calculate length byte in block header
**********************/
int HeaderLength(int iArrayLength)      {
        int iHeaderLength;
        iHeaderLength = 3;   /* iArrayLength >0 plus increment for "#" and  "n"
*/
        while ( (iArrayLength = (iArrayLength / 10)) > 0 )      {
                iHeaderLength++;
        }

        return(iHeaderLength);
}

/*******************      prepare ESA for measurement
************************/
void setup() {

        viPrintf(viESA, ":SENS:FREQ:CENT %i MHz\n", iCenter);
        viPrintf(viESA,":SENS:FREQ:SPAN %i MHZ\n", iSpan);
        viPrintf(viESA, ":SENS:BAND %i KHZ\n", iRbw);

        /* use the sampling detector for power-average calculations              */
        viPrintf(viESA, ":DET SAMP\n");

      /* Turn off analog output of option board to maximize measurement rate   */
        viPrintf(viESA, ":SYST:PORT:IFVS:ENAB OFF\n");

        /* Turn auto align off to maximize measurement rate                      */
        viPrintf(viESA, ":CAL:AUTO OFF\n");

        /* set requested number of points                                        */
        viPrintf(viESA, ":SWE:POINTS %i\n", NUM_POINTS);

        printf("This program will measure and calculate\n");
        printf ("the power average of %i %i-point
traces.\n",iNumTraces,iNumPoints);

        /* Turn on 50 MHz amplitude reference signal                             */
        viPrintf(viESA, ":CAL:SOUR:STAT ON\n");

/* Identify the instrument and get the model number                        */
        viQueryf(viESA, "*IDN?\n", "%t", &cBuffer);

       iResult = (strncmp( cBuffer, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cBuffer, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cBuffer,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
/*Set the input port to the 50MHz amplitude reference for the models*/
/*E4401B, E4411B and E7401A*/
viPrintf(viESA,":DISP:WIND:TRAC:Y:RLEV -25 DBM\n");
viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
/* For the analyzers having frequency limits >= 3GHz, prompt the user*/
/* to connect the amplitude reference output to the input*/
printf ("Connect AMPTD REF OUT to the INPUT \n");
printf (".......Press Return to continue \n");
scanf( "%c",&cEnter);

/*Externally route the 50MHz Signal*/
viPrintf(viESA, ":DISP:WIND:TRAC:Y:RLEV -20 DBM\n");
viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}


        /* Single sweep mode                                               */
        viPrintf(viESA, ":INIT:CONT OFF\n");

        /* Turn off the local display to maximize measurement rate         */
        if(!DISPLAY)    {
            viPrintf(viESA, ":DISP:ENAB OFF\n");
        }

        /* transfer data in definite length,32 bit integer blocks. Select  */
        /* machine units (milli-dBm) to maximize measurement rate          */
        viPrintf(viESA, ":FORM:DATA INT,32\n" );

        /* select the byte order; low-byte first for Intel platforms       */
        /* To further increase measurement rate,:FORM:BORD NORM could      */
        /* be used instead. The byte ordering would then need to be        */
        /* done within this program.                                       */
        viPrintf(viESA, ":FORM:BORD SWAP\n");

        /* pre-calculate amount of data to be transferred per measurement  */
        iTermLength = 1;
        iArrayLength = iNumPoints * DATA_LENGTH;
        iHeaderLength = HeaderLength(iArrayLength);
        iBlockSize = iHeaderLength + iArrayLength + iTermLength;

}

/****************        Write binary trace data to ESA        *******************/
void write_binary_trace(char *cScpiCommand, int *ipTraceData)   {
    /*  trace data must point to an integer array of size NUM_POINTS            */
        memcpy(&cOutBuffer[strlen(cScpiCommand)], ipTraceData, iArrayLength);
        memcpy(&cOutBuffer, cScpiCommand, strlen(cScpiCommand));

    /* Add a <newline> to the end of the data, This isn't necessary             */
    /* if the GPIB card has been configured to assert EOI when the last         */
    /* character is sent, but it ensures a valid iTermLength is provided.       */
    cOutBuffer[iArrayLength + strlen(cScpiCommand)] = 0x0A;
    iBlockSize = (strlen(cScpiCommand) + iArrayLength + 1);
    viWrite(viESA,(ViBuf) cOutBuffer, iBlockSize, &lRetCount );
}

/*******   Measure and calculate power-average of multiple measurements
*********/
void average()  {
    int i=0, iLoop=0;
    int iArray[NUM_POINTS];

    long lOpc =0L;
    double dLogTen = log(10.0);

    setup();

    iTotalRetCount = lRetCount = 0;

    /* start the timer                                                          */
    ftime( &start_time );

    /* Now run through the event loop iNumTraces times                          */
    for(i=0; i<iNumTraces; i++)     {

/* trigger a new measurement and wait for complete                  */
        viPrintf(viESA, ":INIT:IMM;*WAI\n");

        /* Read the trace data into a buffer                                    */
        viPrintf(viESA, ":TRAC:DATA? TRACE1\n");
        viRead(viESA,(ViBuf) cInBuffer, (ViUInt32) iBlockSize, &lRetCount );
        iTotalRetCount += lRetCount;

        /* copy trace data to an array,                                         */
        /* byte order swapping  could be done here rather than in ESA           */
        memcpy(iArray, &cInBuffer[iHeaderLength], iArrayLength);

        /* calculate a running dPower-average                                   */
        for(iLoop = 0; iLoop < NUM_POINTS; iLoop++) {
            /* running average of dPower, in milliwatts                         */
            dPower = exp( dLogTen * (iArray[iLoop]/10000.0));
            if(i > 0)   {
                dPwrAvgArray[iLoop] += ((dPower - dPwrAvgArray[iLoop])/(i+1));
            }
            else    {
                dPwrAvgArray[iLoop] = dPower;
            }
        }
    }   /* end of event loop                                                   */

        /* stop the timer                                                      */
        ftime( &stop_time );

        /* Calculate elapsed time                                              */
        if (start_time.millitm > stop_time.millitm) {
            stop_time.millitm += 1000;
            stop_time.time--;
        }
        elapsed_time.millitm = stop_time.millitm - start_time.millitm;
        elapsed_time.time = stop_time.time - start_time.time;

        /* This is measurement time in milliseconds                            */
        dDelta = (1000.0 * elapsed_time.time) + (elapsed_time.millitm);

        /* show measurement statistics                                         */
        dTimePer=dDelta/((float)iNumTraces);
        printf("\tPower average of %i %i-point traces performed in %3.1f
seconds\n",iNumTraces,iNumPoints,dDelta/1000);
        printf("\t%6.1f milliseconds per averaged measurement\n",dTimePer);
        printf("\t%6.1f averaged measurements per second\n",1000.0/dTimePer);
        printf("\t%i bytes transferred per trace, %i bytes total\n\n",lRetCount,
iTotalRetCount);
        return;
}

/********************************       Main      ******************************/
void  main(void)  {
    int iLoop;
    int iAvgArray[NUM_POINTS];
    ViStatus viStatus;

    /* Open a GPIB session at address 18                                       */
    viStatus = viOpenDefaultRM(&defaultRM);
    viStatus = viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
    if(viStatus)
    {
        printf("Could not open a session to GPIB device at address 18!\n");
                exit(0);
    }

/*Clear the Instrument                                                  */
    viClear(viESA);

    /* go to known instrument state with cleared status byte                  */
    viPrintf(viESA, ":SYST:PRES:TYPE FACT;*RST\n");

    /* measure, transfer and calculate power average of multiple traces        */
    average();

    /* convert average power array back to integer array                      */
    for (iLoop = 0; iLoop < iNumPoints; iLoop++)    {
        dPower = 10.0 * log10( dPwrAvgArray[iLoop]);
        iAvgArray[iLoop] = (int) (1000.0 * dPower);
    }

    /* build 'TRAC:DATA TRACE2,#nyyy' header and write the result to Trace 2    */
    sprintf(cCommand,":TRAC:DATA TRACE2,#%i%i", HeaderLength(iArrayLength)-2,
iArrayLength);
    write_binary_trace(cCommand, iAvgArray);

    /* enable the trace, local display and return to continuous sweep          */
    viPrintf(viESA,":TRACE2:MODE VIEW;:DISP:ENAB ON;:INIT:CONT ON\n");

    /* Close session                                                          */
    viClose(viESA);
    viClose(defaultRM);

}  /*****************************   End of Main   *******************************/
```

4 Programming Command Cross-References

Functional Index to SCPI Subsection

The following table lists the SCPI subsections or subsystems associated with the instrument function category you wish to perform. The commands listed that begin with an asterisk (*) are IEEE common commands. These commands, and the SCPI commands in the subsection or subsystem, are documented in Chapter 5, "Language Reference," on page 189.

Function Category	SCPI Subsection or Subsystem

ALIGNMENT	`*CAL?`
	`*TST?`
	`:CALibration`
	`:STATus:QUEStionable`

ATTENUATOR	see function category: Internal Attenuation and Source

BANDWIDTH	`:CALCulate`
	`:INITiate`
	`:MEASure`
	`[:SENSe]:BANDwidth`

CONFIGURATION and STATUS	`*RCL <register>`
	`*SRE <integer>`
	`*STB?`
	`:SYSTem`

CONTROL	`:ABORt`
CORRECTED MEASUREMENTS	`[:SENSe]:CORRection`

COUPLING	`:COUPle`
	`[:SENSe]:BANDwidth`

DELETE, LOAD, OR SAVE	`*SAV <register>`
	`:MMEMory`

DEMODULATION	`[:SENSe]:DEMod`
DISPLAY	`:UNIT`
EMI DIAGNOSTICS	`:MEASure`

EMI MEASUREMENTS	`:MEASure`
	`[:SENSe]:EMI`

FREQUENCY	`[:SENSe]:FREQuency`
	`:STATus:QUEStionable`

FREQUENCY SPAN	`[:SENSe]:FREQuency`

Function Category	SCPI Subsection or Subsystem

INPUT and OUTPUT	`:INPut`
	`:OUTPut`
	`[:SENSe]:AVERage`
	`[:SENSe]:BANDwidth`
	`[:SENSe]:CORRection`
	`[:SENSe]:DEMod`
	`[:SENSe]:DETector`
	`[:SENSe]:POWer`
	`[:SENSe]:SWEep`
	`:STATus:QUEStionable`
	`:UNIT`

INTERNAL ATTENUATION and SOURCE	`:OUTPut`
	`[:SENSe]:POWer`
	`:SOURce`

LIMIT LINES	`:CALCulate:LLINe`
	`:MMEMory`
	`:TRACe`

MARKER	`:CALCulate:MARKer`

MEASURE	`:EMI`
	`:INITiate`
	`:MEASure`
	`[:SENSe]:AVERage`
	`[:SENSe]:POWer`
	`[:SENSe]:SWEep`

PRESET	`*RST`
	`:STATus`
	`:SYSTem`

PRINTING	`:HCOPy`
SIGNAL LIST	`:CALCulate` `:MEASure`
SOURCE	see function category: Internal Attenuation and Source
SPAN	see also functional category: FREQUENCY SPAN
SPEAKER	`:SYSTem`

SWEEP	`[:SENSe]:SWEep`
	`:SOURce`

SYNCHRONIZATION	`*OPC?`
	`*WAI`
	`:SYSTem`

Function Category	SCPI Subsection or Subsystem

SYSTEM INFORMATION	`*CLS`
	`*ESE <number>`
	`*IDN?`
	`*ESR?`
	`*LRN?`
	`:STATus`
	`:STATus:QUEStionable`
	`:SYSTem`

TRACE	`:DISPlay`
	`:FORMat`
	`[:SENSe]:EBWidth`
	`:TRACe`

TRACE MATH	`:CALCulate:NTData`
	`:DISPlay`
	`:TRACe`

TRIGGER	`*TRG`
	`:ABORt`
	`:INITiate`
	`:TRIGger`

5 Language Reference

This chapter contains SCPI (Standard Commands for Programmable Instruments) programming commands for the Agilent EMC analyzers.

The first few pages of this chapter contain common commands specified in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992. Following these commands, the Agilent EMC analyzers SCPI commands are listed.

NOTE Refer to Chapter 2 , “Status Registers” which supplements the information presented in this chapter. In addition, refer to Chapter 6, “Front-Panel Key Reference,” in the “Agilent EMC Analyzers User’s Guide” for additional information about the operation of each analyzer function. Use the analyzer **HELP** key to obtain similar information about analyzer key functions.

Refer to Chapter 6, “Agilent 8590/EMC Analyzers Programming Conversion Guide” for specific backwards compatibility information between commands for HP/Agilent 8590-Series spectrum analyzers and Agilent EMC analyzers.

SCPI Sections and Subsections

SCPI commands related to major functional areas (such as calculate or) are grouped into blocks, or subsystems. Some of these subsystems are further divided into subsections (such as calculate/marker, or sense/harmonics). An instrument model is then created to represent the way in which instrument functionality is viewed and categorized by SCPI. Refer to *IEEE SCPI-1997 Volume 2: Command Reference, Standard Commands for Programmable Instruments,* Version 1997.0, May, 1997 for a more complete description of the SCPI instrument model.

The SCPI subsystems in this chapter are listed in alphabetical order. Likewise, the SCPI commands are in alphabetical order within the subsystem in which they belong. Refer to the following table to locate SCPI command subsystems and subsections by page number.

SCPI Subsystem/Subsection	Page
IEEE Common Commands	page 193
:ABORt	page 198
:CALCulate	page 199
:CALCulate:EMI:SLISt	page 201
:CALCulate:LLINe	page 209
:CALCulate:MARKer	page 216
:CALCulate:NTData	page 228
:CALibration	page 229
:CONFigure (see :MEAS)	page 254
:COUPle	page 233
:DISPlay	page 235
:FETCh (see :MEASure)	page 254
:FORMat	page 243
:HCOPy	page 245
:INITiate	page 249
:INPut	page 252
:MEASure	page 254
:MMEMory	page 260
:OUTPut	page 267

SCPI Subsystem/Subsection	Page
:READ (see :MEASure)	page 254
[:SENSe]:	page 268
[:SENSe]:AVERage	page 269
[:SENSe]:BANDwidth	page 272
[:SENSe]:CORRection	page 276
[:SENSe]:DEMod	page 280
[:SENSe]:DETector	page 282
[:SENSe]:EMI	page 287
[:SENSe]:FREQuency	page 290
[:SENSe]:POWer	page 295
[:SENSe]:SWEep	page 298
:SOURce	page 303
:STATus	page 308
:STATus:QUEStionable	page 310
:SYSTem	page 319
:TRACe	page 328
:TRIGger	page 334
:UNIT	page 338

IEEE Common Commands

These commands are specified in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

Calibration Query

`*CAL?`

Performs a full alignment and returns a number indicating the success of the alignment. A zero is returned if the alignment is successful. The SCPI equivalent for this command is the same as **`:CALibrate[:ALL]?`**

NOTE Before executing this command, connect a cable between front panel connector AMPTD REF OUT and the INPUT connector for all Agilent EMC analyzers except Agilent model E7401A.

If the cable is not connected, **`CAL:ALL`** will perform a subset of the RF alignment and a subsequent **`CAL:RF`** will be required for the analyzer to meet its specified performance.

The query performs a full alignment and returns a number indicating the success of the alignment. A zero is returned if the alignment is successful, even if only a subset of the RF alignment is performed.

Front Panel Access:	**System**, **Alignments**, **Align All Now**

Clear Status

`*CLS`

Clears the status byte. It does this by emptying the error queue and clearing all bits in all of the event registers. The status byte registers summarize the states of the other registers. It is also responsible for generating service requests.

Remarks:	See **`*STB?`**

Standard Event Status Enable

`*ESE <number>`

`*ESE?`

Sets the bits in the standard event status enable register. This register monitors I/O errors and synchronization conditions such as operation complete, request control, query error, device dependent error, execution error, command error and power on. A summary bit is generated on execution of the command.

Query returns the state of the standard event status enable register.

Range: Integer, 0 to 255

Standard Event Status Register Query

`*ESR?`

Queries and clears the standard event status event register. (This is a destructive read.)

Range: Integer, 0 to 255

Identification Query

`*IDN?`

Returns an instrument identification information string. The string will contain the model number, serial number and firmware revision. The response is organized into four fields separated by commas. The field definitions are as follows:

Manufacturer

Model

Serial number

Firmware version

Example: `Hewlett-Packard, E7402A, US39120213, A.06.00`

NOTE As shown in the example, the analyzer returns “Hewlett-Packard” as the manufacturer even though it is now manufactured by Agilent Technologies. This is intentional. Agilent Technologies was created out of the Hewlett-Packard company, and the Hewlett-Packard name is retained to support those customers who have purchased EMC analyzers in the past.

Front Panel Access: **System, Show System**

Instrument State Query

`*LRN?`

Returns current instrument state data in a block of defined length. The information is in a machine readable format only. Sending the query returns the following format:

`#PQQQSYST:SET #NMMM<state_data>`

The following example is a response to **`*LRN?`** The actual sizes will vary depending on the instrument state data size.

Example: `#42031SYST:SET #42016<state data>`

The number 4 (P in the preceding query response format) means there are 4 numbers that make up the size of the data that follows. In this example, 2031 bytes will follow the number 4 (42031).

2031 and 2016 (QQQ and MMM in the preceding query response format) represent data size in bytes.

The state can be changed by sending this block of data to the instrument after removing the size information:

Example:	`SYST:SET #NMMM<state_data>`

Operation Complete

`*OPC`

`*OPC?`

Sets bit 0 in the standard event status register to “1” when all pending operations have finished.

The query stops any new commands from being processed until the current processing is complete. Then it returns a “1”, and the program continues. This query can be used to synchronize events of other instruments on the external bus.

`*OPC` and `*OPC?` are currently effective only when immediately preceded by either the `:INITiate:IMMediate` or a `:CALibration` command.

Query Instrument Options

This function is provided in the analyzer SCPI language reference in the SYSTem subsystem under `:SYSTem:OPTions?`.

Recall

`*RCL <register>`

This command recalls the instrument state from the specified instrument memory register.

Range:	Registers are an integer, 0 to 127
Remarks:	See also commands `:MMEMory:LOAD:STATe` and `:MMEMory:STORe:STATe`

If the state being loaded has a newer firmware revision than the revision of the instrument, no state is recalled and an error is reported.

If the state being loaded has an equal firmware revision than the revision of the instrument, the state will be loaded.

If the state being loaded has an older firmware revision than the revision of the instrument, the instrument will only load the parts of the state that apply to the older revision.

Front Panel Access:	**File, Recall State**

Reset

`*RST`

This command presets the instrument to a factory defined condition that is appropriate for remote programming operation. **`*RST`** is equivalent to performing the two commands :**`SYSTem:PRESet`** and **`*CLS`**. This command always performs a factory preset.

NOTE The preset performed by **`*RST`** is always a factory preset. That is, the same preset performed by :**`SYSTem:PRESet`** when :**`SYSTem:PRESet:TYPE`** is set to **`FACTory`**.

Front Panel Access:	**Preset**

Save

`*SAV <register>`

This command saves the instrument state to the specified instrument memory register.

Range:	Registers are an integer, 0 to 127
Remarks:	See also commands :**`MMEMory:LOAD:STATe`** and :**`MMEMory:STORe:STATe`**
Front Panel Access:	**File, Save State**

Service Request Enable

`*SRE <integer>`

`*SRE?`

This command sets the value of the service request enable register.

The query returns the value of the register.

Range:	Integer, 0 to 255

Read Status Byte Query

`*STB?`

Returns the value of the status byte register without erasing its contents.

Remarks:	See **`*CLS`**

Trigger

`*TRG`

This command triggers the instrument. Use the :**`TRIGger:SEQuence:SOURce`** command to select the trigger source.

Remarks: See also the :**`INITiate:IMMediate`** command

Self Test Query

`*TST?`

This query is used by some instruments for a self test.

For Agilent ESA analyzers, **`*TST?`** always returns 0; no tests are performed.

Front Panel
Access: **System, Alignments, Align All Now**

Wait-to-Continue

`*WAI`

This command causes the instrument to wait until all pending commands are completed before executing any additional commands. There is no query form to the command.

ABORt Subsystem

Abort

`:ABORt`

Restarts any sweep or measurement in progress and resets the sweep or trigger system. A measurement refers to any of the measurements found in the **MEASURE** menu.

> If :**`INITiate:CONTinuous`** is off (single measure), then :**`INITiate:IMMediate`** will start a new single measurement.
>
> If :**`INITiate:CONTinuous`** is on (continuous measure), a new continuous measurement begins immediately.

The INITiate and TRIGger subsystems contain additional related commands.

Front Panel Access:	**Restart** for continuous measurement mode

CALCulate Subsystem

This subsystem is used to perform post-acquisition data processing. In effect, the collection of new data triggers the CALCulate subsystem. In this instrument, the primary functions in this subsystem are markers and limits.

NdBpoints

`:CALCulate:BWIDth|BANDwidth:NDB <rel_ampl>`

`:CALCulate:BWIDth|BANDwidth:NDB?`

Selects the power level, below the peak of the signal, at which the signal bandwidth will be measured by the markers.
:**`CALCulate:BWIDth|BANDwidth[:STATe]`** must be ON.

NOTE To query the result of NdBpoints, use the command **`:CALCulate:BWIDth|BANDwidth:RESult?`**

Factory Preset and *RST:	–3 dB
Range:	–80 dB to –1 dB
Default Unit:	dB
Remarks:	Refer to :**`CALCulate:BWIDth|BANDwidth[:STATe]`** for an explanation of this marker function.
Front Panel Access:	**Peak Search (or Search), N dB Points**

NdBresults

`:CALCulate:BWIDth|BANDwidth:RESult?`

Returns the measured bandwidth at the power level defined by **`:CALCulate:BWIDth:NDB`**. –100 is returned if **`:CALCulate:BWIDth|BANDwidth[:STATe]`** is off, or when a result is not available. Refer to **`CALCulate:BWIDth|BANDwidth[:STATe]`** for an explanation of this marker function.

Range:	Real value less than the current frequency span
Default Unit:	Hz
Remarks:	When segmented sweep is on, a result will not be available when the NDB marker crosses a segment boundary.
Front Panel Access:	**Peak Search (or Search), N dB Points**

NdBstate

:CALCulate:BWIDth|BANDwidth[:STATe] OFF|ON|0|1

:CALCulate:BWIDth|BANDwidth[:STATe]?

Controls the bandwidth measurement function. The function measures the bandwidth, at the number of dB down specified in :**CALCulate:BWIDth:NDB**, of the maximum signal on the display.

Factory Preset and *RST:	Off
Remarks:	When this command is turned on, the bandwidth measurement function (N dB Points) is associated with the active marker. If no marker is active at the time this command is turned on, marker 1 becomes the active marker, and a peak search is performed. No restrictions exist for moving the bandwidth measurement function markers to any other signal on the display. However, when this function is turned on, all other concurrent marker functions are suspended.
Front Panel Access:	**Peak Search (or Search), N dB Points On Off**

Calculate Correction at Frequency

:CALCulate:CORRection:ATFREquency? <freq>

Calculates the total correction factor for a specified frequency point.

Example:	**:CALC:CORR:ATFRE? 1.5E8**
Remarks:	Refer to **[:SENSe]:CORRection:CSET[1]|2|3|4:DATA <freq>, <rel_ampl> {,<freq>, <rel_ampl}** for an explanation of correction factors.

Test Current Trace Data Against all Limit Lines

:CALCulate:CLIMits:FAIL?

Queries the status of the limit line testing. Returns a 0 if the trace data passes when compared with all the current limit lines. Returns a 1 if the trace data fails any limit line test.

CALCulate:EMI:SLISt Subsection

Add Measure to List

`:CALCulate:EMI:SLISt:ADD:MEASured`

Adds results of most recent measure to marker list.

Factory Preset and *RST:	Not affected by preset
Front Panel Access:	**MEASURE, Meas to List**

Add Marker to List

`:CALCulate:EMI:SLISt:ADD:MARKer [1]|2|3|4`

Adds marker frequency and amplitude to signal list.

Factory Preset and *RST:	Not affected by preset
Front Panel Access:	**MEASURE, Marker to List**

Append Signal Data to List

`CALCulate:EMI:SLISt:ADD <string>`

and

`CALCulate:EMI:SLISt:FETCH CURRent | <integer>`

Adds data to signal list using the defined format. The string is a data string formatted as a quote delimited string with comma separated fields.

Data Format

The string is a quoted delimited string with comma separated fields.

The meaning of a value is determined by its position in the data string. The data format string is designed to allow complementary use between the ADD and FETCH features. That is, the output of the FETCH command is directly usable as input to the ADD command.

The string positions are defined in Table 5-1. Note that while the ADD format requires only a frequency and allows defaults, the data returned by the FETCH will always return a fully formatted data string. See the field descriptions for allowed values and formats.

The ADD command always appends signals to the end of the signal list. If the

Table 5-1 Field Description Table

Position	ADD Required	ADD Optional	Definition	Default
1	Yes		Frequency	Value required
2		Yes	Peak Ampl	0 (Peak detector flag off)
3		Yes	QPK Ampl	0 (Quasi-Peak detector flag off)
4		Yes	Avg Ampl	0 (Average amplitude flag off)
5		Yes	Uncertainty	0 (Applied when measured)
6		Yes	Total Correction	0 (Applied when measured)
7		Yes	Marked Status	0 (off)
8		Yes	Peak Det. Flag	0 (off) * Implicitly On if peak amplitude provided
9		Yes	QPK Det. Flag	0 (off) * Implicitly On if QPk amplitude provided
10		Yes	Avg Flag	0 (off) * Implicitly On if Avg amplitude provided
11		Yes	Comment	Empty

ADD command fails (e.g. signal list at capacity), a message is placed on the status line and in the error queue. An invalid data format places a message on the status line and into the error queue.

Because all of the values may not be meaningful during the ADD process, the ADD command allows defaults to be used. Because of this the data string submitted may be abbreviated. Each ADD data format string must contain a valid frequency string. The rule applied to ADD strings is that values need to be provided only as far to the right as a non-default value is used.

For example, the data format to add a signal at 1 MHz can be indicated as a frequency only. The two commands below are equivalent. The remaining fields are set to their default values.

`CALCulate:EMI:SLISt:ADD "1e6"` or

`CALCulate:EMI:SLISt:ADD "1000000"`

If the frequency and a comment are desired, the missing fields between the frequency and comment position must be shown by marking the positions using the comma separator. Either type of quote delimiting is available. The preceding example would be expressed as:

`CALCulate:EMI:SLISt:ADD "1e6,,,,,,,,,,This is my comment"` or

`CALCulate:EMI:SLISt:ADD '1e6,,,,,,,,,,This is my comment'`

If a mark were desired in the above example, the command would be:

```
CALCulate:EMI:SLISt:ADD "1e6,,,,,,1,,,,This is my comment"
```

The Mark and detector fields allow human readable input forms:

```
CALCulate:EMI:SLISt:ADD "1e6,,,,,,On,,,,This is my comment"
```

NOTE If an amplitude value is provided in the format string, the associated detector flag is automatically turned on unless explicitly turned off in the same data format string.

For example:

```
CALCulate:EMI:SLISt:ADD "1e6,-10000,,,,,On"
```

This provides a peak detector amplitude and turns the mark flag on. The peak detector flag is turned on implicitly which allows the amplitude value to be displayed in the signal list.

The FETCH process allows the selection of the current signal or the signal at an explicit index. If the index is out of range, an empty string is returned.

The FETCH process also returns the current state of the signal. Continuing the example above, if a signal were added, the list cursor were set to the last signal and a fetch immediately performed, the command sequence would be:

```
CALCulate:EMI:SLISt:ADD "1e6,3012,,,,,On,,,,This is my
comment"
CALCulate:EMI:SLISt:SEL LAST
CALCulate:EMI:SLISt:FETCH? CURRent
```

The returned value would appear in this form:

```
1000000,3012,0,0,0,0,1,1,0,0,'This is my comment'
```

Additional detail is provided when the signal has been measured.

A measurement provides new detector values, amplitudes from the detectors used, a valid uncertainty and a potential correction factor.

Detailed Field Descriptions

Frequency

ADD - Frequency is the only required field for the ADD command. The frequency may be expressed in Hz or engineering notation. Examples: `1000000, 1e6, 1.545e6`.

FETCH - The frequency is returned in Hz using engineering notation.

Peak, Quasi-Peak, and Average Amplitudes

ADD - The amplitudes may be entered in integer or engineering notation. The units are mdBm

FETCH - The amplitudes are returned in mdBm.

Uncertainty	ADD – The uncertainty for a signal may be entered for a signal. If not provided the default value for the instrument is applied to the signal. The uncertainty may be entered in Hz or engineering notation.
	FETCH – The uncertainly is returned in Hz.
Total Correction	ADD – The total correction factor may be entered in integer or engineering format. The units are mdB.
	FETCH – The total correction factor is returned in mdB.
Mark	ADD – The signal may be marked during the ADD process by setting this flag to 1 or On. It may also be explicitly turned Off or 0.
	FETCH – The signal mark is returned as a 1 (On) or 0 (Off).
Peak, Quasi-Peak, and Average Detector Amplitude Flags	ADD – The presence of an amplitude number in the add string automatically turns on the amplitude flag for that detector. That value may be overridden by explicitly setting the detector to 0 or Off.
	FETCH – The detector flag is returned as a 1 (On) or 0 (Off).
Comment	ADD – The comment is a single line of text with a maximum length of 31 characters. Characters in excess of 31 are truncated.
	FETCH – The comment is returned in the 11th position and delimited by commas.
Factory Preset and *RST:	Not affected by preset

Clear Marks

```
:CALCulate:EMI:SLISt:CLEar ALL|CURRent|<integer>
```

Clears marks on all, current, or specific signals

Factory Preset and *RST:	Not affected by preset
Remarks:	This command takes an <integer> value which corresponds to the position of a signal in the signal list. Valid range is 1 to 2000.
Front Panel Access:	**MEASURE, More, Signal Marking, Clear**

Set Comment for Signals

`:CALCulate:EMI:SLISt:COMMent:ALL <string>`

Set comment for all signals.

Factory Preset and *RST:	Not affected by preset
Remarks:	The comment is a string consisting of a single line of text with a maximum length of text characters of 31. Characters in excess of 31 are truncated.

Set Comment for Current Signal

`:CALCulate:EMI:SLISt:COMMent:CURRent <string>`

Set comment for current signal.

Factory Preset and *RST:	Not affected by preset
Remarks:	The comment is a string consisting of a single line of text with a maximum length of text characters of 31. Characters in excess of 31 are truncated.

Set Comment for Marked Signals

`:CALCulate:EMI:SLISt:COMMent:MARKed <string>`

Set comment for all marked signals.

Factory Preset and *RST:	Not affected by preset
Remarks:	The comment is a string consisting of a single line of text with a maximum length of text characters of 31. Characters in excess of 31 are truncated.

Delete Signal

`:CALCulate:EMI:SLISt:DELete ALL|CURRent|MARKed| <integer>`

Deletes all, current, marked, or specific signals from signal list.

Factory Preset and *RST:	Not affected by preset
Remarks:	This command takes an <integer> value which corresponds to the position of a signal in the signal list. Valid range is 1 to 2000.
Front Panel Access:	**MEASURE, More, Signal List, Delete Signal**

Turn Signal List On or Off

:CALCulate:EMI:SLISt:DISPlay[:STATe] OFF|ON|0|1

:CALCulate:EMI:SLISt:DISPlay:STATe?

Sets the state of the signal list, On or Off.

Factory Preset and *RST:	Not affected by preset
Remarks:	Once the state function is selected, the selected state is persistent. Persistent means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**MEASURE, More, Signal List** **MEASURE, More, Signal List, List Edit** **MEASURE, More, Signal List, Delete Signals** **MEASURE, More, Signal List, Remeasure** **MEASURE, More, Signal List, Signal Marking** **MEASURE, More, Signal List, Sort Signals**

Retrieve Signal as a String

:CALCulate:EMI:SLISt:FETCh? CURRent|<integer>

Retrieve the current or specific signal as a string (comma separated fields).

Factory Preset and *RST:	Not affected by preset
Remarks:	This command takes an <integer> value which corresponds to the position of a signal in the signal list. Valid range is 1 to 2000.

Retrieve Signals in List

:CALCulate:EMI:SLISt:LENGth?

Retrieve the number of signals in the signal list.

Factory Preset and *RST:	Not affected by preset

Mark Selected Signals

:CALCulate:EMI:SLISt:MARK ALL|CURRent|TOEnd|DUPLicate|LOWer|COMPlement <integer>

Marks signals based on the current parameter. Default is current.

ALL - all signals

CURRent - current signal.
TOEnd - from the current signal to the end of the list.
DUPLicates - all duplicate signals.
LOWer- all lower amplitude duplicates.
COMPlement - complement signals.

Factory Preset and *RST:	Not affected by preset
Front Panel Access:	**MEASURE, More, Signal List, Signal Marking, Mark Signal**

Position Cursor to Signal List

`:CALCulate:EMI:SLISt:SELect FIRSt|LAST|NEXT|PREVious|<integer>`

Position cursor to signal in list.

Factory Preset and *RST:	Not affected by preset
Remarks:	Once this function is defined (**Signal List On**), the cursor is positioned and the selected state is persistent. Persistent means that it retains the setting previously selected, even through a power cycle. This command takes a signal-index value which corresponds to the position of a signal in the signal list. Valid range is 1 to 2000.
Front Panel Access:	**MEASURE, More, Signal List, Signal List**

Specify Sort Key

```
:CALCulate:EMI:SLISt:SORT
FREQuency|PEAK|QPEak|AVERage|LLINE1|LLINE2
ASCending|DESending
```

```
:CALCulate:EMI:SLISt:SORT FREQ, ASC?
```

Specifies the sort key and order for signal list sorting. Both the sort and order are required. The format is comma separated.

Factory Preset and *RST:	Not affected by preset
Front Panel Access:	**MEASURE, More, Signal List, Sort Signals, By Freq** **MEASURE, More, Signal List, Sort Signals, By Pk Ampl** **MEASURE, More, Signal List, Sort Signals, By QP Ampl** **MEASURE, More, Signal List, Sort Signals, By Δ LL1** **MEASURE, More, Signal List, Sort Signals, By Δ LL2**

Specify Signal List Display Parameters

:CALCulate:EMI:SLISt:View
COMMent|PEAK|QPEak|AVERage

:CALCulate:EMI:SLISt:View?

Specifies the signal display parameters given by the following:

COMM - Displays the comment for each of the signals in the list.
PEAK - Displays the difference values, in dB, from the measured peak value and the limit lines.
QPEak - Displays the difference values, in dB, from the measured quasi-peak value and the limit lines.
AVERage - Displays the difference values, in dB, from the measured average value and the limit lines.

Factory Preset and *RST:	Not affected by preset
Front Panel Access:	**MEASURE, More, Signal List, Edit List, Comment**
	MEASURE, More, Signal List, Edit List, Pk Ampl
	MEASURE, More, Signal List, Edit List, QP Ampl
	MEASURE, More, Signal List, Edit List, AV Ampl

CALCulate:LLINe Subsection

Limit lines can be defined for your measurement. You can then have the instrument compare the data to your defined limits and indicate a pass/fail condition.

NOTE Refer also to **`:MMEMory`** and **`:TRACe`** subsystems for more trace and limit line commands.

Delete All Correction Sets in Memory

`:CALCulate:LLINe:ALL:DELete`

Deletes all correction sets in volatile memory.

History:	Added with firmware revision A.08.00.
Front Panel Access:	**Display, Limits, Delete All Limits**

Control Limit Line Amplitude Interpolation

`:CALCulate:LLINe[1]|2:AMPLitude:INTerpolate:TYPE LOGarithmic|LINear`

`:CALCulate:LLINe[1]|2:AMPLitude:INTerpolate:TYPE?`

Selects the type of interpolation done for the amplitude values of the designated limit line when comparing to measured data.

Factory Preset and *RST:	Not affected by preset
Remarks:	Once this function is defined, the selected type is persistent. Persistent means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**Display, Limits, Limit 1|2, Amptd Interp Log Lin**

Set Fixed or Relative Limit Lines

`:CALCulate:LLINe:CMODe FIXed|RELative`

`:CALCulate:LLINe:CMODe?`

Specifies whether the current limit lines are fixed or relative.

NOTE If you need to change the domain with :**`CALCulate:LLINe:CONTrol:DOMain`**, do it before this command. Changing the domain deletes all the existing limit line values.

Factory Preset and *RST:	Not affected by preset
Remarks:	For Amplitude Parameters:

Regardless of whether the limit line is based on frequency or sweep time, amplitude parameters in a limit line table represent absolute values or relative values. In fixed, the limit line amplitude values are specified in absolute amplitude and do not depend on the reference level. In relative, the limit line amplitude values are relative to the current reference level.

For Fixed Frequency Parameters:

The frequency values in a limit line table are fixed values, and the limit line is positioned accordingly. Fixed limit lines are specified in absolute frequency and do not depend upon the center frequency values.

For Relative Frequency Parameters:

The frequency values in a limit line table are relative values and positions the limit line relative to the center frequency settings. Relative limit lines are specified in relative frequency and are positioned with respect to the current center frequency. When the current center frequency value is changed, the segment frequencies are converted according to the current center frequency value.

For Time Parameters:

Limit lines that are based on sweep time are always relative to the start time. The horizontal position of the limit line is not affected by this command.

Front Panel Access: **Display, Limits, Limits Fixed Rel**

Set Limit Line X-axis Units

```
:CALCulate:LLINe:CONTrol:DOMain FREQuency|TIME
```

```
:CALCulate:LLINe:CONTrol:DOMain?
```

Selects how the limit line segments are defined: according to frequency, or according to the sweep time setting of the spectrum analyzer.

NOTE Changing this setting deletes *all* existing limit data from the analyzer. In other words, if a limit line has already been defined, changing the type clears the existing limit line.

Factory Preset and *RST: Not affected by Preset

Remarks: For TIME, the limit line segments are placed on the spectrum analyzer display with respect to the sweep time setting of the analyzer, with 0 at the left edge of the display.

For FREQuency, segments are placed according to the frequency that is specified for each segment.

Front Panel Access: **Display, Limits, X Axis Units Freq Time**

Control Limit Line Frequency Interpolation

:CALCulate:LLINe[1]|2:CONTrol:INTerpolate:TYPE LOGarithmic|LINear

:CALCulate:LLINe[1]|2:CONTrol:INTerpolate:TYPE?

Selects the type of interpolation done for the frequency values of the designated limit line when comparing to measured data. This only applies in the frequency domain. This function does not work in zero span (when the analyzer is in the time domain).

Remarks: Once this function is defined, the selected type is persistent. Persistent means that it retains the setting previously selected, even through a power cycle.

Front Panel Access: **Display, Limits, Limit 1|2, Freq Interp Log Lin**

Define Limit Line Values

:CALCulate:LLINe[1]|2:DATA <x-axis>,<ampl>,<connected>{,<x-axis>,<ampl>,<connected>}

:CALCulate:LLINe[1]|2:DATA?

Defines limit line values, and destroys all existing data. Up to 200 points may be defined for each limit. No units are allowed.

- <x-axis> – can be frequency or time values as specified by :**CALCulate:LLINe:CONTrol:DOMain**. Frequencies are always in Hz. Time is always in seconds. No unit is allowed in this parameter.
- <ampl> – amplitude values are in the current Y-axis units. Up to two amplitude values can be provided for each x-axis value, by repeating <x-axis> in the data list. No unit is allowed in this parameter.
- <connected> – connected values are either 0 or 1. A 1 means this point should be connected to the previously defined point to define the limit line. A 0 means that it is a point of discontinuity and is not connected to the preceding point. The "connected" value is ignored for the first point.

Example: **CALC:LLIN1:DATA 1000000000,-20,0,200000000,-30,1**

Range: **<x-axis>** –30 Gs to +30 Gs for time limits

`<x-axis>` –30 GHz to +350 GHz for frequency limits

`<ampl>` –120 dBm to +100 dBm

`<connected>` 0 or 1

Remarks: If two amplitude values are entered for the same frequency, a single vertical line is the result. In this case, if an upper line is chosen, the amplitude of lesser frequency (amplitude 1) is tested. If a lower line is chosen, the amplitude of greater frequency (amplitude 2) is tested.

For linear amplitude interpolation and linear frequency interpolation, the interpolation is computed as:

$$y = \frac{y_{i+1} - y_i}{f_{i+1} - f_i}(f - f_i) + y_i$$

For linear amplitude interpolation and log frequency interpolation, the interpolation is computed as:

$$y = \frac{y_{i+1} - y_i}{\log f_{i+1} - \log f_i}(\log f - \log f_i) + y_i$$

For log amplitude interpolation and linear frequency interpolation, the interpolation is computed as:

$$\log y = \frac{\log y_{i+1} - \log y_i}{f_{i+1} - f_i}(f - f_i) + \log y_i$$

For log amplitude interpolation and log frequency interpolation, the interpolation is computed as:

$$\log y = \frac{\log y_{i+1} - \log y_i}{\log f_{i+1} - \log f_i}(\log f - \log f_i) + \log y_i$$

Front Panel Access:

Display, Limits, X Axis Units Freq Time

Display, Limits, Limit 1|2, Edit

Display, Limits, Limit 1|2, Edit, Point

Display, Limits, Limit 1|2, Edit, Frequency

Display, Limits, Limit 1|2, Edit, Amplitude

Display, Limits, Limit 1|2, Edit, Connected

Display, Limits, Limit 1|2, Edit, Delete Point

Merge Additional Values into the Existing Limit Line

`:CALCulate:LLINe[1]|2:DATA:MERGe <x-axis>,<ampl>,<connected>{,<x-axis>,<ampl>,<connected>}`

Adds the points with the specified values to the current limit line, allowing you to merge limit line data. Up to two amplitude values are allowed for each x value. If too much data is merged, as many points as possible are merged into the existing limit and then an error is reported. Up to 200 points total may be defined for each limit.

- <x-axis> can be frequency or time values as specified by :**`CALCulate:LLINe:CONTrol:DOMain`**. Frequencies are always in Hz. Time is always in seconds. No unit is allowed in this parameter.
- <ampl> – amplitude values are in the current Y-axis units. No unit is allowed in this parameter.
- <connected> connected values are either 0 or 1. A 1 means this point should be connected to the previously defined point to define the limit line. A 0 means that it is a point of discontinuity and is not connected to the preceding point. The “connected” value is ignored for the first point.

Range:	<x-axis> –30 Gs to +30 Gs for time limits
	<x-axis> –30 GHz to +350 GHz for frequency limits
	<ampl> –120 dBm to +100 dBm
	<connected> 0 or 1
Front Panel Access:	**Display, Limits, X Axis Units Freq Time**

Delete Limit Line

`:CALCulate:LLINe[1]|2:DELete`

Deletes the selected limit line.

Display the Limit Line

`:CALCulate:LLINe[1]|2:DISPlay OFF|ON|0|1`

`:CALCulate:LLINe[1]|2:DISPlay?`

Controls the display of the current limit line.

Factory Preset and *RST:	Off
Front Panel Access:	**Display, Limits, Limit 1|2, Limit On Off**

Test the Data Against the Limit Line

`:CALCulate:LLINe[1]|2:FAIL?`

Queries the status of the limit line testing. Returns a 0 if the data passes, and returns a 1 if there is a failure. This query value is valid only if margin or limit test is On. Use the command **`:CALCulate:LLINe[1]|2:STATe OFF|ON|0|1`** to activate limit line testing.

Set the Margin Size

`:CALCulate:LLINe[1]|2:MARGin <rel_ampl>`

`:CALCulate:LLINe[1]|2:MARGin?`

Allows you to define the amount of measurement margin that is added to the designated limit line.

Factory Preset and *RST:	not affected
Default Units:	dB
Remarks:	The margin must be negative for upper limit lines, and positive for lower limits.
Front Panel Access:	**Display, Limits, Limit 1|2, Margin On Off**

Display the Limit Margin

`:CALCulate:LLINe[1]|2:MARGin:STATe OFF|ON|0|1`

`:CALCulate:LLINe[1]|2:MARGin:STATe?`

Allows you to display a measurement margin that is added to the designated limit line to do secondary testing of the data.

Factory Preset and *RST:	Off
Front Panel Access:	**Display, Limits, Limit 1|2, Margin On Off**

Control Limit Line Testing

`:CALCulate:LLINe[1]|2:STATe OFF|ON|0|1`

`:CALCulate:LLINe[1]|2:STATe?`

Turns limit line testing on/off. The limit and margin will only be tested if they are displayed. Use `:CALCulate:LLINe[1]|2:DISPlay` to turn on the display of limit lines, and `:CALCulate:LLINe[1]|2:MARGin:STATe` to turn on the display of margins. If margin and limit display are both turned off, limit test is automatically turned off. Use `:CALCulate:LLINe[1]|2:FAIL?` to return the state of pass or fail after limit line state has been turned on.

Factory Preset and *RST:	Off
Front Panel Access:	**Display, Limits, Limit 1|2, Limit On Off**

Select the Type of Limit Line

`:CALCulate:LLINe[1]|2:TYPE UPPer|LOWer`

`:CALCulate:LLINe[1]|2:TYPE?`

Sets a limit line to be either an upper or lower type limit line. An upper line will be used as the maximum allowable value when comparing with the data. A lower limit line defines the minimum value.

Factory Preset and *RST:	Upper; not affected by preset
Remarks:	If a margin has already been set for this limit line, and this command is used to change the limit type, then the margin value is reset to 0 dB.
Front Panel Access:	**Display, Limits, Limit 1|2, Type Upper Lower**

CALCulate:MARKer Subsection

Markers All Off on All Traces

```
:CALCulate:MARKer:AOFF
```

Turns off all markers on all the traces.

Front Panel Access:	**Marker, Marker All Off**

Continuous Peaking Marker Function

```
:CALCulate:MARKer[1]|2|3|4:CPEak[:STATe] OFF|ON|0|1
```

```
:CALCulate:MARKer[1]|2|3|4:CPEak[:STATe]?
```

Turns on or off continuous peaking. It continuously puts the selected marker on the highest displayed signal peak.

Factory Preset and *RST:	Off
Remarks:	This command may not be used to activate a given marker.
Front Panel Access:	**Peak Search (or Search), Continuous Pk On Off**

Frequency Counter Marker Resolution

```
:CALCulate:MARKer:FCOunt:RESolution <real>
```

```
:CALCulate:MARKer:FCOunt:RESolution?
```

Sets the resolution of the marker frequency counter. Setting the resolution to AUTO will couple the marker counter resolution to the frequency span.

Factory Preset and *RST:	1 kHz
Range:	1 Hz to 100 kHz
Default Unit:	Hz
Front Panel Access:	**Freq Count, Resolution Auto Man**

Frequency Counter Marker Automatic Resolution

`:CALCulate:MARKer:FCOunt:RESolution:AUTO OFF|ON|0|1`

`:CALCulate:MARKer:FCOunt:RESolution:AUTO?`

Sets the resolution of the marker frequency counter so it is automatically coupled to the frequency span, generating the fastest accurate count.

Factory Preset and *RST:	On
Front Panel Access:	**Freq Count, Resolution Auto Man**

Frequency Counter Marker

`:CALCulate:MARKer[1]|2|3|4:FCOunt[:STATe] OFF|ON|0|1`

`:CALCulate:MARKer[1]|2|3|4:FCOunt[:STATe]?`

Turns on or off the marker frequency counter. To query the frequency counter, use **`:CALCulate:MARKer[1]:FCOunt:X?`** If the specified marker number is not the active marker, it becomes the active marker. If the specified marker number is not on, it is turned on and becomes the active marker. Once the marker count is on, it is on for any active marker, not just for the one used in the command. A 1 is returned only if marker count is on and the selected number is the active marker.

Factory Preset and *RST:	Off
Remarks:	If a frequency count x value is generated when the frequency count state is off, then 9e15 is returned.
Front Panel Access:	**Freq Count, Marker Count On Off**

Frequency Counter Marker Query

`:CALCulate:MARKer[1]|2|3|4:FCOunt:X?`

Queries the marker frequency counter.

Remarks:	If a frequency count x value is generated when the frequency count state is off, then 9e15 is returned.

Marker Function

`:CALCulate:MARKer[1]|2|3|4:FUNCtion BPOWer|NOISe|OFF`

`:CALCulate:MARKer[1]|2|3|4:FUNCtion?`

Selects the marker function for the specified marker. To query the value returned by the function, use **`:CALCulate:MARKer[1]|2|3|4:Y?`**

BPOWer is the power integrated within the bandwidth

NOISe is a noise measurement

OFF turns off all functions

Remarks: When a measurement under the front panel **`MEASURE`** key is started, this command is turned off. If this command is turned on when any of the **`MEASURE`** key measurements are in progress, that measurement will be stopped.

Front Panel Access: **Marker, Function**

Marker Peak (Maximum) Search

`:CALCulate:MARKer[1]|2|3|4:MAXimum`

Performs a peak search based on the search mode settings of **`:CALCulate:MARKer:PEAK:SEARch:MODE`**.

NOTE See command **`:CALCulate:MARKer:PEAK:SEARch:MODE`**

Front Panel Access: **Peak Search (or Search), Meas Tools, Peak Search**

Marker Peak (Maximum) Left Search

`:CALCulate:MARKer[1]|2|3|4:MAXimum:LEFT`

Places the selected marker on the next highest signal peak to the left of the current marked peak.

Remarks: The marker will be placed at the next highest peak that rises and falls by at least the peak excursion above the peak threshold. If no peak meets the excursion and threshold criteria, a No Peak Found error (202) is given.

Front Panel Access: **Peak Search (or Search), Next Pk Left**

Marker Next Peak (Maximum) Search

`:CALCulate:MARKer[1]|2|3|4:MAXimum:NEXT`

Places the selected marker on the next highest signal peak from the current marked peak.

Remarks: The marker will be placed at the highest peak that rises and falls by at least the peak excursion above the peak threshold. If no peak meets the excursion and threshold criteria, a No Peak Found error (202) is given.

Front Panel
Access: **Peak Search (or Search), Next Peak**

Marker Peak (Maximum) Right Search

:CALCulate:MARKer[1]|2|3|4:MAXimum:RIGHt

Places the selected marker on the next highest signal peak to the right of the current marked peak.

Remarks: The marker will be placed at the highest peak that rises and falls by at least the peak excursion above the peak threshold. If no peak meets the excursion and threshold criteria, a No Peak Found error (202) is given.

Front Panel
Access: **Peak Search (or Search), Next Pk Right**

Marker Peak (Minimum) Search

:CALCulate:MARKer[1]|2|3|4:MINimum

Places the selected marker on the lowest point on the trace that is assigned to that particular marker number.

Front Panel
Access: **Peak Search (or Search), Min Search**

Marker Mode

:CALCulate:MARKer[1]|2|3|4:MODE POSition|DELTa|BAND|SPAN

:CALCulate:MARKer[1]|2|3|4:MODE?

Selects the type of markers that you want to activate. Refer to the "*Agilent EMC Analyzers User's Guide*" for a more complete explanation of this function.

Position selects a normal marker that can be positioned on a trace and from which trace information will be generated.

Delta activates a pair of markers, one of which is fixed at the current marker location. The other marker can then be moved around on the trace. The marker readout shows the difference between the two markers.

Band activates a pair of band markers, where each marker can be independently positioned on the trace. The marker readout shows the difference between the two markers.

Span activates a pair of span markers, where the marker positioning is controlled by changing the span and/or center frequency between the two markers. The marker readout shows the difference between the two markers.

Remarks: If a marker is not active when the mode is queried, "Off" will be returned.

Front Panel Access:	**Marker, Normal**
	Marker, Delta
	Marker, Delta Pair Ref Delta
	Marker, Span Pair Span Center

Define Peak Excursion

`:CALCulate:MARKer:PEAK:EXCursion <rel_ampl>`

`:CALCulate:MARKer:PEAK:EXCursion?`

Specifies the minimum signal excursion above the threshold for the internal peak identification routine to recognize a signal as a peak. This applies to all traces and all windows. (The excursion is the delta power from the noise level to the signal peak.)

NOTE See command **`:CALCulate:MARKer:PEAK:SEARch:MODE`**

Factory Preset and *RST:	6 dB
Range:	0 to 100 dB
Default Unit:	dB
Front Panel Access:	**Peak Search (or Search), Search Criteria, Peak Excursion**

Define Peak Search

`:CALCulate:MARKer:PEAK:SEARch:MODE PARameter|MAXimum`

`:CALCulate:MARKer:PEAK:SEARch:MODE?`

Sets the peak search mode.

Factory Preset and *RST:	MAXimum
Remarks:	If mode is set to MAXimum, peak search will place the marker at the maximum amplitude in the trace. If mode is set to PARameter, peak search will place the marker at the highest peak that rises and falls by at least the peak excursion above the peak threshold. If no peak meets the excursion and threshold criteria, a No Peak Found error (error 202) is issued.
	Next peak, next peak right, next peak left, and peak table are not affected by this command. They will always use peak excursion and peak threshold for search criteria.

Front Panel

Access:	**Peak Search (or Search), Search Criteria, Peak Search Type, Max Value|Excursion & Threshold**

Define Peak Threshold

`:CALCulate:MARKer:PEAK:THReshold <ampl>`

`:CALCulate:MARKer:PEAK:THReshold?`

Specifies the minimum signal level for the analyzers internal peak identification routine to recognize a signal as a peak. This applies to all traces and all windows.

NOTE See command **`:CALCulate:MARKer:PEAK:SEARch:MODE`**

Range:	Reference level to the bottom of the display
Default Unit:	Amplitude units
Front Panel Access:	**Peak Search (or Search), Search Criteria, Peak Threshold**

Peak to Peak Delta Markers

`:CALCulate:MARKer[1]|2|3|4:PTPeak`

Positions delta markers on the highest and lowest points on the trace.

Factory Preset and *RST:	Off
Front Panel Access:	**Peak Search (or Search), Pk-Pk Search**

Set Center Frequency to the Marker Value

`:CALCulate:MARKer[1]|2|3|4[:SET]:CENTer`

Sets the center frequency equal to the specified marker frequency, which moves the marker to the center of the screen. In delta marker mode, the center frequency is set to the marker delta value. This command is not available in zero span.

Front Panel Access:	**Marker –>, Mkr –> CF**

Set Reference Level to the Marker Value

`:CALCulate:MARKer[1]|2|3|4[:SET]:RLEVel`

Sets the reference level to the specified marker amplitude. In delta marker mode, the reference level is set to the amplitude difference between the markers.

Front Panel Access:.	**Marker –>, Mkr –> Ref Lvl**
.	**Peak Search (or Search), Meas Tools, Mkr –> Ref Lvl**

Set Span to the Marker Value

`:CALCulate:MARKer[1]|2|3|4[:SET]:SPAN`

Sets the span to the value of the specified marker frequency. The specified marker must be in delta mode. Select the delta marker mode with **`CALCulate:MARKer[1]|2|3|4:MODE DELTa`**. This command is not available in zero span.

Front Panel
Access: **Marker, Delta, Marker –>, Mkr Δ –> Span**

Set Start Frequency to the Marker Value

`:CALCulate:MARKer[1]|2|3|4[:SET]:STARt`

Sets the start frequency to the value of the specified marker frequency. In delta marker mode, the start frequency is set to the marker delta value. This command is not available in zero span.

Front Panel
Access: **Marker –>, Mkr –> Start**

Set Center Frequency Step Size to the Marker Value

`:CALCulate:MARKer[1]|2|3|4[:SET]:STEP`

Sets the center frequency step size to match the marker frequency. In delta marker mode, the center frequency step size will be set to the frequency difference between the markers. Select the delta marker mode with **`:CALCulate:MARKer[1]|2|3|4:MODE DELTa`**. This command is not available if the delta marker is off, or in zero span.

Front Panel
Access: **Marker –>, Mkr –> CF Step**

Peak Search (or Search), Meas Tools, Mkr –> CF

Set Stop Frequency to the Marker Value

`:CALCulate:MARKer[1]|2|3|4[:SET]:STOP`

Sets the stop frequency to the value of the active marker frequency. In delta marker mode, the stop frequency is set to the marker delta value. This command is not available in zero span.

Front Panel
Access: **Marker –>, Mkr –> Stop**

Marker On/Off

`:CALCulate:MARKer[1]|2|3|4:STATe OFF|ON|0|1`

`:CALCulate:MARKer[1]|2|3|4:STATe?`

Turns the selected marker on or off.

Front Panel Access:	**Marker, Off**

Marker Table On/Off

`:CALCulate:MARKer:TABLe:STATe OFF|ON|0|1`

`:CALCulate:MARKer:TABLe:STATe?`

Turns the marker table on or off

Front Panel Access:	**Marker, Marker Table On Off**

Marker to Trace

`:CALCulate:MARKer[1]|2|3|4:TRACe <integer>`

`:CALCulate:MARKer[1]|2|3|4:TRACe?`

Assigns the specified marker to the designated trace 1, 2, or 3.

Factory Preset and *RST:	1
Range:	1 to 3
Front Panel Access:	**Marker, Marker Trace Auto 1 2 3**

Marker to Trace Auto

`:CALCulate:MARKer[1]|2|3|4:TRACe:AUTO OFF|ON|0|1`

`:CALCulate:MARKer[1]|2|3|4:TRACe:AUTO?`

Turns on or off the automatic marker to trace function.

Factory Preset and *RST:	`AUTO ON`
Front Panel Access:	**Marker, Marker Trace Auto 1 2 3**

Continuous Signal Tracking Function

`:CALCulate:MARKer[1]|2|3|4:TRCKing[:STATe] OFF|ON|0|1`

`:CALCulate:MARKer[1]|2|3|4:TRCKing[:STATe]?`

Turns on or off marker signal tracking. It continuously puts the selected marker on the highest displayed signal peak and moves it to the center frequency. This allows you to keep a signal that is drifting in frequency, on the display.

Factory Preset and *RST:	Off
Remarks:	When a measurement under the front panel **MEASURE** key is started, this command is turned off. If this command is turned on when any of the **MEASURE** key measurements are in progress, that measurement will be stopped.
Front Panel Access:	**FREQUENCY/Channel, Signal Track On Off**

Marker X Value

`:CALCulate:MARKer[1]|2|3|4:X <param>`

`:CALCulate:MARKer[1]|2|3|4:X?`

Position the designated marker on its assigned trace at the specified trace X value. The value is in the X-axis units (which is often frequency or time).

The query returns the current X value of the designated marker.

Default Unit:	Matches the units of the trace on which the marker is positioned
Front Panel Access:	**Marker**

Span Markers Center Frequency X Value

`:CALCulate:MARKer[1]|2|3|4:X:CENTer <param>`

`:CALCulate:MARKer[1]|2|3|4:X:CENTer?`

Position the center frequency, of the designated span-type marker pair, at the specified trace X value. The value is in the X-axis units (which is often frequency or time) Use `:CALCulate:MARKer:MODE SPAN` to select span markers.

The query returns the current X value center frequency of the designated markers.

Range:	Matches the units of the trace on which the markers are positioned
Front Panel Access:	**Marker, <active marker>, Span Pair**

Marker X Position

`:CALCulate:MARKer[1]|2|3|4:X:POSition <integer>`

`:CALCulate:MARKer[1]|2|3|4:X:POSition?`

Position the designated marker on its assigned trace at the specified X position.

The query returns the current X position for the designated marker.

Range:	Refer to the `[:SENSe]:SWEep:POINts` command.

Front Panel
Access: **Marker**

Span Markers Center Frequency X Position

`:CALCulate:MARKer[1]|2|3|4:X:POSition:CENTer <param>`

`:CALCulate:MARKer[1]|2|3|4:X:POSition:CENTer?`

Position the center frequency, of the designated span-type marker pair, at the specified trace X position. Use `:CALCulate:MARKer:MODE SPAN` to select span markers.

The query returns the current X position center frequency of the designated markers.

Range: Refer to the `[:SENSe]:SWEep:POINts` command.

Front Panel
Access: **Marker, <active marker>, Span Pair**

Span Markers Span X Position

`:CALCulate:MARKer[1]|2|3|4:X:POSition:SPAN <param>`

`:CALCulate:MARKer[1]|2|3|4:X:POSition:SPAN?`

Change the frequency span, of the designated span-type marker pair, to position the markers at the desired trace X positions. Use `:CALCulate:MARKer:MODE SPAN` to select span markers.

The query returns the current X position frequency span of the designated markers.

Range: Refer to the `[:SENSe]:SWEep:POINts` command.

Front Panel
Access: **Marker, <active marker>, Span Pair**

Delta Pair Markers Start Frequency X Position

`:CALCulate:MARKer[1]|2|3|4:X:POSition:STARt <param>`

`:CALCulate:MARKer[1]|2|3|4:X:POSition:STARt?`

Position the left-most marker, the start (reference) frequency of the designated band-type marker pair, at the specified trace X position. Use `:CALCulate:MARKer:MODE BAND` to select band markers.

The query returns the current X position start/reference frequency of the designated marker.

Range: Refer to the `[:SENSe]:SWEep:POINts` command.

Front Panel
Access: **Marker, <active marker>, Delta Pair**

Delta Pair Markers Stop Frequency X Position

:CALCulate:MARKer[1]|2|3|4:X:POSition:STOP <param>

:CALCulate:MARKer[1]|2|3|4:X:POSition:STOP?

Position the right-most marker, the stop frequency of the designated band-type marker pair, at the specified trace X position. Use **:CALCulate:MARKer:MODE BAND** to select band markers.

The query returns the current X position stop frequency of the designated marker.

Range:	Refer to the [:**SENSe]:SWEep:POINts** command.
Front Panel Access:	**Marker, <active marker>, Delta**

Marker X-Axis Readout

:CALCulate:MARKer[1]|2|3|4:X:READout FREQuency|TIME|ITIMe|PERiod

:CALCulate:MARKer[1]|2|3|4:X:READout?

Selects the units for the x-axis readout of the marker. Available units are:

Frequency
Time
Inverse of time
Period

Factory Preset and *RST:	Frequency
Front Panel Access:	**Marker, Readout, Frequency**
	Marker, Readout, Time
	Marker, Readout, Inverse Time
	Marker, Readout, Period

Span Markers Span X Value

:CALCulate:MARKer[1]|2|3|4:X:SPAN <param>

:CALCulate:MARKer[1]|2|3|4:X:SPAN?

Change the frequency span of the designated span-type marker pair to position the markers at the desired trace X values. The value is in the X-axis units (which is usually frequency or time). Use **:CALCulate:MARKer:MODE SPAN** to select span markers.

The query returns the current X value frequency span of the designated markers. If span markers are not selected, the query returns the latest marker reading as a span (always positive).

Default Unit:	Matches the units of the trace on which the markers are positioned.
Front Panel Access:	**Marker, <active marker>, Span Pair**

Delta Pair Markers Start Frequency X Value

`:CALCulate:MARKer[1]|2|3|4:X:STARt <param>`

`:CALCulate:MARKer[1]|2|3|4:X:STARt?`

Position the start (reference) frequency of the designated band-type marker pair, at the specified trace X value. The value is in the X-axis units (which is often frequency or time). Use :**`CALCulate:MARKer:MODE BAND`** to select band markers.

The query returns the current X value start/reference frequency of the designated marker.

Default Unit:	Matches the units of the trace on which the markers are positioned
Front Panel Access:	**Marker, <active marker>, Delta Pair**

Delta Pair Markers Stop Frequency X Value

`:CALCulate:MARKer[1]|2|3|4:X:STOP <param>`

`:CALCulate:MARKer[1]|2|3|4:X:STOP?`

Position the stop frequency of the designated band-type marker pair, at the specified trace X value. The value is in the X-axis units (which is often frequency or time). Use :**`CALCulate:MARKer:MODE BAND`** to select band markers.

The query returns the current X value stop frequency of the designated marker.

Default Unit:	Matches the units of the trace on which the markers are positioned
Front Panel Access:	**Marker, <active marker>, Delta Pair**

Marker Read Y Value

`:CALCulate:MARKer[1]|2|3|4:Y?`

Read the current Y value for the designated marker or delta on its assigned trace. The value is in the Y-axis units for the current trace (which is often dBm).

Default Unit:	Matches the units of the trace on which the marker is positioned
Remarks:	This command can be used to read the results of marker functions such as band power and noise that are displayed in the marker value field on the analyzer.

CALCulate:NTData Subsection

Normalize the Trace Data

`:CALCulate:NTData[:STATe] OFF|ON|0|1`

`:CALCulate:NTData[:STATe]?`

One sweep of trace data is copied to trace 3 (firmware version greater then A.03.03, **`NRML`** in firmware version less than or equal to A.03.03), which is used as the reference trace. Then for all subsequent trace sweeps, display trace 1 = data collected into trace 1 – data in trace 3 (firmware version greater than A.03.03, NRML in firmware version less than or equal to A.03.03).

Front Panel Access:	**View/Trace, Normalize, Normalize On Off**

CALibration Subsystem

These commands control the self-alignment and self-diagnostic processes.

Align All Instrument Assemblies

`:CALibration[:ALL]`

`:CALibration[:ALL]?`

Performs an alignment of all the assemblies within the instrument, except for the tracking generator (Option 1DN), if installed (except Agilent model E7401A).

Before executing this command, connect a cable between front panel connector **AMPTD REF OUT** and the **INPUT** connector for all Agilent EMC analyzers except Agilent model E7401A.

If the cable is not connected, **`CAL:ALL`** will perform a subset of the RF alignment and a subsequent **`CAL:RF`** will be required for the analyzer to meet its specified performance.

The query performs a full alignment and returns a number indicating the success of the alignment. A zero is returned if the alignment is successful, even if only a subset of the RF alignment is performed.

Front Panel Access:	**System, Alignments, Align Now, All**

Set Auto Align Mode All or Not RF

`:CALibration:AUTO:MODE ALL|NRF`

`:CALibration:AUTO:MODE?`

This command determines whether or not to include RF alignment as part of the automatic alignment routines. Eliminating automatic alignment of the RF prevents changes in the input impedance between sweeps, which could cause input device instability.

Factory Preset and *RST:	All at power-up
Front Panel Access:	**System, Alignments, Auto Align, All**
	System, Alignments, Auto Align, All but RF

Automatic Alignment

`:CALibration:AUTO OFF|ON|0|1`

`:CALibration:AUTO?`

Turns the automatic alignment on and off. This is run continuously, at the completion of each sweep.

Factory Preset and *RST:	On at power-up
Front Panel Access:	**System, Alignments, Auto Align, All**
	System, Alignments, Auto Align, All but RF
	System, Alignments, Auto Align, Off

Return to the Default Alignment Data

`:CALibration:DATA:DEFault`

Initializes the alignment data to the factory defaults.

Front Panel Access:	**System, Alignments, Load Defaults**

Align FM Demodulation

`:CALibration:FMDemod`

`:CALibration:FMDemod?`

Performs an alignment of the FM Demodulation board. The query form of this command performs the alignment and returns zero if the alignment is successful.

Front Panel Access:	**System, Alignments, Align Now, FM Demod**

Query the Internal or External Frequency Reference

`:CALibration:FREQuency:REFerence?`

This is a query only. It reports the location of where the instrument frequency reference is generated.

Range:	INT or EXT

Coarse Adjust the Frequency Reference

`:CALibration:FREQuency:REFerence:COARse <setting>`

`:CALibration:FREQuency:REFerence:COARse?`

Allows coarse adjustment of the internal 10 MHz reference oscillator timebase of the analyzer.

NOTE **`:CALibration:ALL`** is required after **`COARse`** is set.

Range:	Integer, 0 to 255
Front Panel Access:	**System, Alignments, Time Base, Coarse**

Fine Adjust the Frequency Reference

`:CALibration:FREQuency:REFerence:FINE <setting>`

`:CALibration:FREQuency:REFerence:FINE?`

Allows fine adjustment of the analyzer internal 10 MHz reference oscillator timebase.

NOTE **`:CALibration:ALL`** is required after **`FINE`** is set.

Range:	Integer, 0 to 255
Front Panel Access:	**System, Alignments, Time Base, Fine**

Select the Frequency Corrections

`:CALibration:FREQuency[:STATe] OFF|ON|0|1`

`:CALibration:FREQuency[:STATe]?`

Turns on or off the frequency corrections.

Factory Preset and *RST:	On
Front Panel Access:	**System, Alignments, Freq Correct On Off**

Align the RF Circuitry

`:CALibration:RF`

`:CALibration:RF?`

Performs an alignment of the RF assembly.

The query performs the alignment and returns a zero if the alignment is successful.

Before executing this command, connect a cable between front panel connector **AMPTD REF OUT** and the **INPUT** connector for all Agilent EMC analyzers except Agilent model E7401A. If the cable is not connected, the alignment will fail.

Front Panel Access: **System, Alignments, Align Now, RF**

Select the Source State for Calibration

`:CALibration:SOURce:STATe OFF|ON|0|1`

`:CALibration:SOURce:STATe?`

Controls the state of the 50 MHz alignment signal.

NOTE The alignment signal is internally switched to the **INPUT** for Agilent model E7401A. For all other models, connect a cable between front panel connector **AMPTD REF OUT** and the **INPUT** connector before performing a calibration.

Factory Preset and *RST: Off

Front Panel Access: For Agilent EMC model E7401A:

Input/Output (or Input), Amptd Ref (f=50 MHz) On Off

For all other Agilent EMC models:

Input/Output (or Input), Amptd Ref Out (f=50 MHz) On Off

Calibrate the Tracking Generator

`:CALibration:TG`

`:CALibration:TG?`

Performs an alignment of the tracking generator assembly.

The query performs the alignment and returns a zero if the alignment is successful.

NOTE This command is applicable on all Agilent EMC models except E7401A with Option 1DN installed. Before executing this command, connect a cable between front panel connector **RF OUT** and the **INPUT** connector. The alignment will fail using command **`CAL:TG`** if the cable is not connected.

Front Panel Access: **System, Alignments, Align Now, TG**

COUPle Subsystem

Some measurement settings are automatically coupled together to optimize speed and accuracy. These commands control that coupling.

COUPle the Function to Other Settings

`:COUPle ALL|NONE`

`:COUPle?`

The instrument can automatically couple instrument settings together for accurate measurements and optimum dynamic range. This command is used to override the coupling for special measurement needs.

`COUPle NONE` puts these functions into the manually set (not coupled) mode. **`COUPle ALL`** puts the functions into the auto coupled mode, and also puts the sweep coupling mode into SA (couple all).

The following list of analyzer functions can be automatically coupled:

Resolution bandwidth
Center Frequency

Average type (Firmware revision A.08.00 or greater)
Marker functions

Detector (Firmware revision A.08.00 or greater)
Marker functions
Average On Off
Average type

Attenuation
Reference level
External amplifier gain
Preamp

Center frequency step
Span (in swept spans)
Resolution bandwidth (in zero spans)

Video bandwidth
Resolution bandwidth

Tracking Generator
Sweep coupling mode (SR/SA)

VBW/RBW ratio (Firmware revision A.08.00 or greater)

Sweep time
Span
Video bandwidth
Resolution bandwidth

Sweep points
Phase noise optimization

Phase Noise Optimization (Firmware revision A.08.00 or greater)
Span

NOTE Although marker count, gate time, and marker trace have auto settings, they are not affected by Couple.

Factory Preset and *RST:	All
Front Panel Access:	**Auto Couple, Auto All**

DISPlay Subsystem

The DISPlay subsystem controls the selection and presentation of textual, graphical, and trace information. Within a display, information may be separated into individual windows.

Display Viewing Angle

`:DISPlay:ANGLe <integer>`

`:DISPlay:ANGLe?`

Changes the viewing angle for better viewing in different environments.

Factory Preset and *RST:	The factory default is 4. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Range:	Integer, 1 to 7
Front Panel Access:	**Viewing angle keys**

Date and Time Display Format

`:DISPlay:ANNotation:CLOCk:DATE:FORMat MDY|DMY`

`:DISPlay:ANNotation:CLOCk:DATE:FORMat?`

Allows you to set the format for displaying the real-time clock. To set the date time use: SYSTem:DATE <year>, <month>, <day>.

Factory Preset and *RST:	The factory default is MDY. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**System, Time/Date, Date Format MDY DMY**

Date and Time Display

`:DISPlay:ANNotation:CLOCk[:STATe] OFF|ON|0|1`

`:DISPlay:ANNotation:CLOCk[:STATe]?`

Turns on and off the display of the date and time on the spectrum analyzer screen.

Factory Preset
and *RST: The factory default is On. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Front Panel
Access: **System, Time/Date, Time/Date On Off**

Display Annotation Title Data

`:DISPlay:ANNotation:TITLe:DATA <string>`

`:DISPlay:ANNotation:TITLe:DATA?`

Enters the text that will be displayed in the user title area of the display.

Front Panel
Access: **Display, Title**

Display, Title, Change Title

Display, Title, Clear Title

Turn the Entire Display On/Off

`:DISPlay:ENABle OFF|ON|0|1`

Turns the display on or off. Having the display turned off may increase repetitive measurement rate.

The following key presses will turn display enable back on:

1. If in local, press any key
2. If in remote, press the local (system) key
3. If in local lockout, no key (the computer must either cancel local lockout, or re-enable the display)

Factory Preset
and *RST: On

Turn the Full Screen Display On/Off

`:DISPlay:MENU:STATe OFF|ON|0|1`

`:DISPlay:MENU:STATe?`

Turns the full screen display mode on or off. Press **System, System** to turn off full screen mode.

History: Added with firmware revision A.08.00.

Window Annotation

`:DISPlay:WINDow:ANNotation[:ALL] OFF|ON|0|1`

`:DISPlay:WINDow:ANNotation[:ALL]?`

Turns the screen annotation on or off for all windows.

Factory Preset and *RST:	On
Front Panel Access:	**Display, Preferences, Annotation On Off**

Trace Graticule Display

`:DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe] OFF|ON|0|1`

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

Turns the graticule on or off.

Factory Preset and *RST:	On
Front Panel Access:	**Display, Preferences, Graticule On Off**

Trace X-Axis Scale Offset

`:DISPlay:WINDow:TRACe:X[:SCALe]:OFFSet <freq>`

`:DISPlay:WINDow:TRACe:X[:SCALe]:OFFSet?`

Specifies the frequency offset for all frequency readouts such as center frequency, except that it does not affect marker count.

Factory Preset and *RST:	0 Hz
Range:	–500 THz to 500 THz
Default Unit:	Hz
History:	Prior to firmware revision A.06.00, the lower range is –3 GHz.
Remarks:	Frequency offset is not available when frequency scale type is Log (`[:SENSe]:SWEep:SPACing LINear|LOGarithmic`).
Front Panel Access:	**FREQUENCY/Channel, Freq Offset**

Display Line Amplitude

`:DISPlay:WINDow:TRACe:Y:DLINe <ampl>`

`:DISPlay:WINDow:TRACe:Y:DLINe?`

Defines the level of the display line, in the active amplitude units if no units are specified.

Factory Preset and *RST:	2.5 divisions below the reference level
Range:	10 display divisions below the reference level to the reference level
Default Unit:	Current active units
Front Panel Access:	**Display, Display Line On Off**

Display Line On/Off

`:DISPlay:WINDow:TRACe:Y:DLINe:STATe OFF|ON|0|1`

`:DISPlay:WINDow:TRACe:Y:DLINe:STATe?`

Turns the display line on or off.

Factory Preset and *RST:	Off
Front Panel Access:	**Display, Display Line On Off**

IF Gain Auto/Reference Level Auto Ranging

`:DISPlay:WINDow:TRACe:Y[:SCALe]:LOG:RANGe:AUTO OFF|ON|0|1`

`:DISPlay:WINDow:TRACe:Y[:SCALe]:LOG:RANGe:AUTO?`

This command enables and disables auto ranging. The speed benefits gained with this command are realized only when in narrow resolution (digital) bandwidths. The setting of auto range has no effect when in analog resolution bandwidths.

Factory Preset and *RST:	On
History:	This command is available with firmware revision A.04.00 and later.
Remarks:	When using digital resolution bandwidths (RBW < 1 kHz) the analyzer uses IF Gain auto ranging to set the optimum signal gain for digital processing. This technique produces the greatest measurement range without overloading the digital system. To increase the measurement speed this IF Gain auto ranging may be set to fixed mode. When in fixed mode, make sure the signal

is not set above the reference level and the reference is set so that the signal is within the display range. When in fixed mode the measurement has approximately 70 dB of display range.

Front Panel Access:	**AMPLITUDE/Y Scale, IF Gain Auto Fixed** (front panel access is available with firmware revision A.06.00 and later).

Normalized Reference Level

`:DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel <rel_ampl>`

`:DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel?`

Sets the normalized reference level.

NOTE See command **`:CALCulate:NTData[STATe] OFF|ON|0|1`**

Factory Preset and *RST:	0 dB
Range:	–327.6 to 327.6 dB
Default Unit:	Current active units
Front Panel Access:	**View/Trace, Normalize, Norm Ref Lvl**

Normalized Reference Level Position

`:DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition <integer>`

`:DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition?`

Selects the position of the normalized reference level. The top and bottom graticule lines correspond to 10 and 0, respectively.

NOTE See command **`:CALCulate:NTData[STATe] OFF|ON|0|1`**

Factory Preset and *RST:	10
Range:	integer
Front Panel Access:	**View/Trace, Normalize, Norm Ref Posn**

Trace Y-Axis Amplitude Scaling

`:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision <rel_ampl>`

`:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision?`

Sets the per-division display scaling for the y-axis when y-axis units are set to amplitude units.

Factory Preset and *RST:	10 dB
Range:	0.1 to 20.0 dB
Default Unit:	dB
Front Panel Access:	**AMPLITUDE/Y Scale, Scale/Div**

Trace Y-Axis Frequency Scaling

`:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision:FREQuency <freq>`

`:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision:FREQuency?`

This command sets the per-division display scaling for the y-axis, when the y-axis units are set to frequency units, such as when looking at FM deviation with the command **`[:SENSe]:DEMod:VIEW[:STATe] OFF|ON|0|1`**.

Factory Preset and *RST:	20 kHz
Range:	1 kHz to 240 kHz
Default Unit:	Hz
Front Panel Access:	**AMPLITUDE/Y Scale, Scale/Div**

Trace Y-Axis Reference Level

`:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel <ampl>`

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

Sets the amplitude value of the reference level for the y-axis.

Factory Preset and *RST:	107 dBμV
Range:	EMC E7401A: –149.9 to 50 dBm EMC E7402A, E7403A, E7404A, E7405A: –149.9 to 55 dBm

	–42.9 to 162 dBµV with zero reference level offset and max mixer level = 97 dBµV.
Default Unit:	Current active units
Remarks:	The input attenuator setting may be affected. The minimum displayed value of reference level is –327.6 dBm, and the maximum displayed value is 327.6 dBm. See the remarks given for the command **`:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet <rel_ampl>`**
Front Panel Access:	**Amplitude Y Scale, Ref Level**

Trace Y-Axis Reference Level Offset

`:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet <rel_ampl>`

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

Sets the amplitude level offset for the y-axis.

Factory Preset and *RST:	0 dB
Range:	–327.6 to 327.6 dB
Default Unit:	dB
Remarks:	The sum of (reference level offset + reference level) is clipped to the range –327.6 to 327.6 dB. The maximum limits are determined by the setting of the first of these two parameters, within the boundaries of their individual limits when initially set. For example, if the reference level is (first) set to –20 dBm, then the reference level offset can be set to values of –307.6 dB to 327.6 dB. In the case of a 327.6 dB reference level offset, the resultant reference level value changes to 307.6 dBm. The reference level value range can be initially set to values from –149.9 to 55 dBm. Setting the reference level offset value first yields the following: If the reference level offset is (first) set to –30 dB, then the reference level can be set to values of –327.6 to 25 dBm. The reference level is “clamped” at 25 dBm because its positive value of 55 dBm is reached at 25 dBm with an offset of –30 dB. Its own positive amplitude limit applies. If the reference level offset is (first) set to 30 dB, then the reference level can be set to values of –327.6 to 85 dBm. Again, the positive amplitude limit of reference level (alone) is factored in to the resultant combined limit.

Front Panel Access:	**Amplitude Y Scale, Ref Level Offst**

Vertical Axis Scaling

```
:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing LINear|LOGarithmic
```

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

Specifies the vertical graticule divisions as log or linear units.

Factory Preset and *RST:	Logarithmic
Front Panel Access:	**AMPLITUDE/Y Scale, Scale Type Log Lin**

FORMat Subsystem

The FORMat subsystem sets a data format for transferring numeric and array information. `TRACe[:DATA]` and `TRACe[:DATA]?` are affected by FORMat subsystem commands.

Byte Order

`:FORMat:BORDer NORMal|SWAPped`

`:FORMat:BORDer?`

This command selects the binary data byte order for data transfer. It controls whether binary data is transferred in normal or swapped mode. This command affects only the byte order for setting and querying trace data for the **`:TRACe[:DATA]`** and query **`:TRACe[:DATA]?`** commands.

NOTE Normal mode is when the byte sequence begins with the most significant byte (MSB) first, and ends with the least significant byte (LSB) last in the sequence: 1|2|3|4. Swapped mode is when the byte sequence begins with the LSB first, and ends with the MSB last in the sequence: 4|3|2|1.

Factory Preset and *RST:	Normal

Numeric Data format

`:FORMat[:TRACe][:DATA]ASCii|INTeger,32|REAL,32|REAL,64|UINTeger,16`

`:FORMat[:TRACe][:DATA]?`

This command changes the format of the trace data input and output. It affects only the data format for setting and querying trace data for the **`:TRACe[:DATA]`** and query **`:TRACe[:DATA]?`** commands.

NOTE This command specifies the formats used for trace data during data transfer across any remote port.

For corrected trace data (**`:TRACe[:DATA]`** with parameter **`<trace_name>`**), **`REAL`**, and **`ASCii`** formats will provide trace data in the current amplitude units. **`INTeger`** format will provide trace data in mdBm. The fastest mode is **`INTeger,32`**.

For uncorrected trace data (**`:TRACe[:DATA]`** with parameter **`RAWTRACE`**), **`UINTeger`**, and **`INTeger`** formats apply to **`RAWTRACE`** queries, and return uncorrected ADC values. The fastest mode is **`UINTeger,16`**.

For state data, the format cannot be changed. It is always in a machine readable format only (machine units).

Table 5-2

Corrected Trace Data Types `:TRACe:DATA? <trace_name>`	
Data Type	**Result**
ASCII	Amplitude Units
INT,32 (fastest)	Internal Units
REAL,32	Amplitude Units
REAL,64	Amplitude Units

Table 5-3

Uncorrected Trace Data Types `:TRACe:DATA? RAWTRACE`	
Data Type	**Result**
INT,32	Uncorrected ADC Values
UINT,16 (fastest)	Uncorrected ADC Values

`ASCii` - Amplitude values are in ASCII, in amplitude units, separated by commas.

`INTeger, 32` - Binary 32-bit integer values in internal units (mdBm), in a definite length block.

`REAL, 32` (or 64) - Binary 32-bit, or 64-bit, real values in amplitude units), in a definite length block.

`UINTeger, 16` - Binary 16-bit unsigned integer uncorrected ADC values, in a definite length block.

Factory Preset and *RST:	ASCII

HCOPy Subsystem

The HCOPy subsystem controls the setup of plotting and printing to an external device.

Abort the Print

`:HCOPy:ABORt`

Aborts hard copy printout of results.

Front Panel Access:	**ESC** (with print in progress)

Printer Type

`:HCOPy:DEVice:TYPE AUTO|CUSTom|NONE`

`:HCOPy:DEVice:TYPE?`

Sets up the printer by selecting printer type.

AUTO - the instrument queries the printer to determine the printer type and automatically sets itself for that printer

CUSTom - allows you to select a custom printer if your printer cannot be auto-configured

NONE - tells the instrument that the hard copy output device is not a printer

Factory Preset and *RST:	The factory default is AUTO. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**Print Setup, Printer Type**

Select Report Content

`:HCOPy:EMI:ITEM:REPort:SCReen|SLISt|ISETups[:State] |ON|OFF|1|0`

`:HCOPy:EMI:ITEM:REPort:SCReem|SLISt|ISETups[:State]?`

Selection of report content: screen dump, signal list, and instrument setups.

Print Report

`:HCOPy:EMI:REPort:IMMediate`

Prints a report.

Remarks:	This parameter retains the setting previously selected, even through a power cycle.
Key Access:	**Print**

Select a Signal List to Include in a Report

:HCOPy:EMI:ITEM:SLISt:PPEak|QPEak|AVERage|LLINe1|LLINe2|CORRection|UNCertainty|COMMent:STATe ON|OFF|1|0

:HCOPy:EMI:ITEM:SLISt:PPEak|QPEak|AVERage|LLINe1|LLINe2|CORRection|UNCertainty|COMMent:STATe?

Selection of signal list columns to include in the report. (PK, QP, AVG, limit1, limit2, correction, uncertainty, comment)

Factory Preset and *RST:	Off

Select a Signal List to Include in a Report (Delta)

:HCOPy:EMI:ITEM:SLISt:DELTa1|DELTa2:PPEak|QPEak|AVERage :STATe ON|OFF|1|0

Selects delta from limit line values to include in report. For each combination of limit lines (1 and 2) and detectors (PPEak. QPEak, and AVERage), this command can be used to include or exclude the delta from the limit for that detector.

Factory Preset and *RST:	Off

Color Hard Copy

:HCOPy:IMAGe:COLor[:STATe] OFF|ON|0|1

:HCOPy:IMAGe:COLor[:STATe]?

Selects between color and monochrome mode for hard copy output.

Factory Preset and *RST:	The factory default is On. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Front Panel
Access: **Print Setup, Color On Off**

Print a Hard Copy

`:HCOPy[:IMMediate]`

The entire screen is output to the parallel port.

Front Panel
Access: **Print**

Form Feed the Print Item

`:HCOPy:ITEM:FFEed[:IMMediate]`

Sends the printer a command to form feed.

Front Panel
Access: **Print Setup, Eject Page**

Page Orientation

`:HCOPy:PAGE:ORIentation LANDscape|PORTrait`

`:HCOPy:PAGE:ORIentation?`

Specifies the orientation of the print.

NOTE Landscape mode is not presently supported for PCL-3 printers.

Factory Preset
and *RST: The factory default is Landscape. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Front Panel
Access: **Print Setup, Orientation, Landscape**

Print Setup, Orientation, Portrait

Number of Items Printed on a Page

`:HCOPy:PAGE:PRINts <integer>`

`:HCOPy:PAGE:PRINts?`

Sets the number of display print outputs sent to print on one piece of paper, before a form feed is sent.

Factory Preset and *RST:	The factory default is 1. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Range:	Integer, 1 or 2
Front Panel Access:	**Print Setup, Prints/Page 1 2**

Printed Page Size

`:HCOPy:PAGE:SIZE A|B|A3|A4|LETTer|LEGal|EXECutive|LEDGer`

`:HCOPy:PAGE:SIZE?`

Formats the print image for the selected page size. Page size "A" is letter, and page size "B" is ledger. There is no size standardization for "legal" or "executive."

Factory Preset and *RST:	The factory default is letter. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**Print Setup, /Page Size**

Select Report Type

`:HCOPy:REPOrt:TYPE SCREen|REPort`

Toggles between screen dump or report.

Remarks:	This parameter retains the setting previously selected, even through a power cycle.
Key Access:	**Print Setup, Print Screen Report**

INITiate Subsystem

The INITiate subsystem is used to control the initiation of the trigger. Refer to the TRIGger and ABORt subsystems for related commands.

Continuous or Single Measurements

`:INITiate:CONTinuous OFF|ON|0|1`

`:INITiate:CONTinuous?`

Selects whether the trigger system is continuously initiated or not.

This command affects sweep if not in a measurement, and affects trigger when in a measurement. A “measurement” refers to any of the functions under the **MEASURE** key. This corresponds to continuous sweep or single sweep operation when not in a measurement, and continuous measurement or single measurement operation when in a measurement.

When not in a measurement, this command does the following:

- When ON at the completion of each sweep cycle, the sweep system immediately initiates another sweep cycle.
- When OFF, the sweep system remains in an “idle” state until CONTinuous is set to ON or an **`:INITiate[:IMMediate]`** command is received. On receiving the **`:INITiate[:IMMediate]`** command, it will go through a single sweep cycle, and then return to the “idle” state.
- The query returns 1 or 0 into the output buffer. 1 is returned when there is continuous sweeping. 0 is returned when there is only a single sweep.

When in a measurement, this command does the following:

- When ON at the completion of each trigger cycle, the trigger system immediately initiates another trigger cycle.
- When OFF, the trigger system remains in an “idle” state until CONTinuous is set to ON or an **`:INITiate[:IMMediate]`** command is received. On receiving the **`:INITiate[:IMMediate]`** command, it will go through a single trigger cycle, and then return to the “idle” state.
- The query returns 1 or 0 into the output buffer. 1 is returned when in a continuous measurement state. 0 is returned when there is only a single measurement.

Factory Preset:	Continuous
*RST:	Continuous, or On
Front Panel Access:	**Sweep**, **Sweep Cont Single**
	Single

Meas Control, **Measure Cont Single**

Take New Data Acquisitions

`:INITiate[:IMMediate]`

This command initiates a sweep if not in a measurement. If in a measurement, it triggers the measurement. A "measurement" refers to any function under the **MEASURE** key.

Remarks: See also the `*TRG` command

Use the `:TRIGer[:SEQuence]:SOURce EXTernal` command to select the external trigger.

The instrument must be in the single measurement mode. If :INITiate:CONTinuous is ON then the command is ignored.

Use `:FETCh?` to transfer a measurement result from memory to the output buffer. Refer to individual commands in the MEASure subsystem for more information.

Front Panel Access: **Sweep, Sweep Cont Single**

Single

Meas Control, Measure Cont Single

Abort Measurement

`:INITiate:ABort`

This command applies to measurements found in the MEASURE menu. Use this command to abort the current measurement.

Remarks: This command is equivalent of sending an :ABORt command followed by an :INITiate[:IMMediate] command.

Front Panel Access: **Meas Control, Abort**

Pause the Measurement

`:INITiate:PAUSe`

This command applies to measurements found in the **MEASURE** menu. Use this command to pause the current measurement by changing the current measurement state from the "wait for trigger" state to the "paused" state. If the measurement is not in the "wait for trigger" state when the command is issued, the transition will be made the next time that state is entered as part of the trigger cycle. When in the pause state, the analyzer auto-align process stops. If the analyzer is paused for a long period of time, measurement accuracy may degrade.

Remarks: Only applies to auto measure.

Front Panel
Access: **Meas Control, Pause**

Restart the Measurement

`:INITiate:RESTart`

This command applies to measurements found in the **MEASURE** menu. Use this command to restart the present measurement from the “idle” state, regardless of its operating state. It is equivalent to **`:INITiate[:IMMediate]`** for single measurement mode, or **`:ABORt`** for continuous measurement mode.

Front Panel
Access: **Restart**

Meas Control, Restart

Resume the Measurement

`:INITiate:RESume`

This command applies to measurements found in the **MEASURE** menu. Use this command to resume the current measurement by changing the current measurement state from the “paused” state back to the “wait for trigger” state.

Front Panel
Access: **Meas Control, Resume**

INPut Subsystem

The INPut subsystem controls the characteristics of analyzer input ports.

Input Port Coupling

```
:INPut:COUPling AC|DC
```

```
:INPut:COUPling?
```

Selects ac or dc coupling for the front panel INPUT port. A blocking capacitor is switched in for the ac mode.

CAUTION Instrument damage can occur if there is a dc voltage present at the INPUT and dc coupling is selected.

Factory Preset and *RST:	ac
Remarks:	This command is available only on Agilent EMC analyzer models E7402A Option UKB, E7405A Option UKB, E7403A or E7404A.

Table 5-4 Selecting Input Coupling

Model Number	AC Frequency Range	DC Frequency Range
E7402A with *Option UKB*	100 kHz to 3 GHz	100 Hz to 3 GHz
E7403A	100 kHz to 6.7 GHz	9 kHz to 6.7 GHz
E7403A with *Option UKB*	100 kHz to 6.7 GHz	100 Hz to 6.7 GHz
E7404A	100 kHz to 13.2 GHz	9 kHz to 13.2 GHz
E7404A with *Option UKB*	100 kHz to 13.2 GHz	100 Hz to 13.2 GHz
E7405A with *Option UKB*	10 MHz to 26.5 GHz	100 Hz to 26.5 GHz

Front Panel Access:	**Input/Output (or Input), Coupling AC DC**

Clear the Input Overload

```
:INPut:PROTection:CLEar
```

Resets the overload protection circuitry for the input connector. There is no query form of this command.

NOTE This command is valid only for Agilent EMC model E7401A.

The excessive input signal may have caused 15 dB of attenuation to be switched in, or it may have completely switched the input connector out so that it is connected to the internal reference signal.

MEASure Group of Commands

This group includes commands used to make measurements and return results. The different commands can be used to provide fine control of the overall measurement process. Most measurements should be done in single measurement mode, rather than doing the measurement continuously.

If you need to change some of the measurement parameters from the factory default settings, set up the measurement with the CONFigure command. Use the commands in the **`[:SENSe]`** subsystem to change the settings. Then use the **`:READ?`** command, or the **`:INITiate`** and **`:FETCh?`** commands to initiate the measurement and query the results.

Each measurement sets the instrument state that is appropriate for that measurement. Other commands are available for each **Mode** to allow changing settings such as view and limits, etc. Refer to the following command subsystems:

SENSe:<measurement>
SENSe:CHANnel
SENSe:CORRection
SENSe:FREQuency
SENSe:POWer

CALCulate:<measurement>
CALCulate:CLIMits/DATA

DISPlay:<measurement>

TRIGger

Configure Commands

`:CONFigure:<measurement>`

This command stops the current measurement and sets up the instrument for the specified measurement using the factory default instrument settings. It does not initiate the taking of measurement data. This command also turns the averaging function on and sets the number of averages to 10 for all measurements.

The query `:CONFigure?` returns the current measurement name in quotes.

The CONFigure? query returns the current measurement name.

Fetch Commands

`:FETCh:<measurement>[n]?`

This command puts valid data into the output buffer, but does not initiate data acquisition. Use the INITiate[:IMMediate] command to acquire data before you use the FETCh command. You can only fetch results from the measurement that is currently selected.

`:FETCh <meas>?` will return valid data only when the measurement is in one of the following states:

idle
initiated
paused

If the optional [n] value is not included, or is set to 1, the scalar measurement results will be returned. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available.

Measure Commands

`:MEASure:<measurement>[n]?`

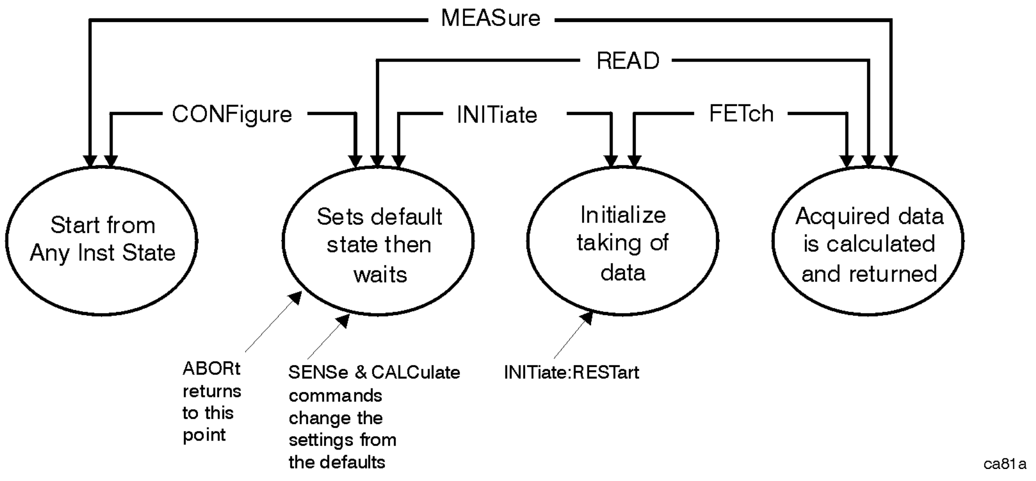
`MEASure` commands stop the present measurement and set up the instrument for the specified measurement using the factory default values. Other SCPI communication is blocked until the measurement is complete. After the data is valid, the scalar result is returned for the specified measurement. These are the settings and units that conform to the measurement specific standard.

- Stops the current measurement and sets up the instrument for the specified measurement using the factory defaults
- Initiates the data acquisition for the measurement
- Blocks other SCPI communication, waiting until the measurement is complete before returning results.
- Turns the averaging function on and sets the number of averages to the default number of averages for the measurement; typically, this is 10.
- After the data is valid it returns the scalar results, or the trace data, for the specified measurement.

  If the optional [n] value is not included, or is set to 1, the scalar measurement results will be returned. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available.

If you need to change some of the measurement parameters from the factory default settings you can set up the measurement with the CONFigure command. Use the commands in the SENSe:<measurement> and CALCulate:<measurement> subsystems to change the settings. Then you can use the READ? command, or the INITiate and FETCh? commands, to initiate the measurement and query the results. See Figure 5-1.

Figure 5-1 **Measurement Group of Commands**


MEASure
READ
CONFigure
INITiate
FETch
Start from Any Inst State
Sets default state then waits
Initialize taking of data
Acquired data is calculated and returned
ABORt returns to this point
SENSe & CALCulate commands change the settings from the defaults
INITiate:RESTart
ca81a

If you need to repeatedly make a given measurement with settings other than the factory defaults, you can use the commands in the SENSe:<measurement> and CALCulate:<measurement> subsystems to set up the measurement. Then use the READ? command or INITiate and FETCh? commands, to initiate the measurement and query results.

Measurement settings persist if you initiate a different measurement and then return to a previous one. Use READ:<measurement>? if you want to use those persistent settings. If you want to go back to the default settings, use MEASure:<measurement>?.

Read Commands

`:READ:<measurement>[n]?`

- Does not preset the measurement to the factory defaults. (The MEASure? and CONFigure? commands reset the parameters to the default values.) It uses the settings from the last measurement.
- Initiates the measurement and puts valid data into the output buffer. If a measurement other than the current one is specified, the instrument will switch to that measurement before it initiates the measurement and returns results.
- Blocks other SCPI communication, waiting until the measurement is complete before returning the results

  If the optional [n] value is not included, or is set to 1, the scalar measurement results will be returned. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available. The binary data formats should be used when handling large blocks of data since they are smaller and faster then the ASCII format.

Measurement settings persist if you initiate a different measurement and then return to a previous one. Use READ:<measurement>? if you want to use those persistent settings. If you want to go back to the default settings, use MEASure:<measurement>?.

Get Measurement Results

`:FETCH?`

Returns the results of the last measurement.

Factory Preset and *RST:	Not affected by *RST or Preset
Remarks:	Only returns data if a measurement has been made. Returns the results of the measurement as a string.

Read Command

`READ?`

Performs configured measurement and returns the results in a string format.

Factory Preset and *RST:	N/A
Remarks:	Output format for measurement results of a single signal from the Measure at Marker or Measure at Frequency keys is: "`<PEAK>, <QUASI-PEAK>, <AVG>, <FREQ' UNCERTAINTY>, <TOTAL AMPLITUDE CORRECTION>, <COMMENT>`" Peak and signal list measurement output is the number of signals measured.

Configure for Measuring Frequency

`:CONFigure:EMI:AFRequency <freq><unit>`

`:MEASure:EMI:AFRequency? <freq><unit>`

Configures for/measures a given frequency.

Factory Preset and *RST:	Not affected by *RST or Preset.

Measure at Marker

:CONFigure:EMI:MARKer[1|2|3|4]?

:MEASure:EMI:MARKer[1|2|3|4]?

:MEASure:MARKer?

Measures at selected marker. Loads the internal buffer.

Factory Preset and *RST:	Not affected by *RST or Preset.
Key Access:	**MEASURE, Meas at Marker**

Measure at Marker and Add to List

:MEASure:EMI:MARKer[1|2|3|4]ADD?

Measures at selected marker and adds to signal list.

Factory Preset and *RST:	1
Key Access:	**MEASURE, Marker to List**

Setting Max/Min On or Off

:MEASure:EMI:MMIN:STATe OFF|ON|0|1

:MEASure:EMI:MMIN:STATe?

Turns Max/Min View On or Off

Factory Preset and *RST:	Off
Remarks:	Max/Min View is active when Max/Min On is selected.
Key Access:	**View/Trace, More, Max/Min**

Max/Min View ->Max

:MEASure:EMI:MMIN:VIEW MAX|MIN|BOTH

Selects Max trace view.

Key Access:	**View/Trace, More, Max/Min View, Max**

Measure Peaks

`:CONFigure:EMI:PEAKs`

`:MEASure:EMI:PEAKs?`

Measures all peaks on screen and adds them to the signal list.

Factory Preset and *RST:	Start
Key Access:	**MEASURE, More, Auto-measure, Start/Abort**

Remeasure Current Signal

`:CONFigure:EMI:SLISt[CURRent]|MARKed|ALL`

`:MEASure:EMI:SLISt?[CURRent]MARKed|ALL`

Remeasures current, marked, or all signals in the signal list.

Factory Preset and *RST:	Current
Key Access:	**MEASURE, More, Signal List, Remeasure**

MMEMory Subsystem

The purpose of the MMEMory subsystem is to provide access to mass storage devices such as internal or external disk drives.

NOTE Refer also to **`:CALCulate`** and **`:TRACe`** subsystems for more trace and limit line commands.

Agilent EMC analyzers use two types of mass storage devices:

- 3.5 inch disk drive (high density, 1.44 MBytes formatted) designated “A:”
- Part of flash memory and treated as a device designated “C:”

The MMEMory command syntax term **`<file_name>`** is a specifier having the form: drive:\directory\name.ext, where the following rules apply:

- “drive” is “A:” or “C:”
- “\directory\” is the path name
- “name” is a DOS file name of up to eight characters, letters (A-Z, a-z) and numbers (0-9) only (lower case letters are read as uppercase)
- “ext” is an optional file extension using the same rules as “name,” but consists of up to three characters total

Catalog the Selected Memory Location

`:MMEMory:CATalog? <drive>`

where “drive” is “A:” or “C:”

Lists all files in the specified drive. The return data will be of the format: `<mem_used>,<mem_free>,<file_listing>`

Each `<file listing>` indicates the name, and size of one file in the directory list: `<file_name>,<file_size>`

Example: `Catalog drive C:`, which is in instrument memory:
`:MMEMory:CATalog? "C:"`

Front Panel Access: **File**

Copy a File

`:MMEMory:COPY <file_name1>,<file_name2>`

To copy a file, the source file name is <file_name1> and the destination file name is <file_name2>.

Example: **`:MMEM:COPY "C:oldname.sta","A:\newname.sta"`**

Front Panel
Access: **File, Copy**

Move Data to File

`:MMEMory:DATA <file_name>,<definite_length_block>`

`:MMEMory:DATA? <file_name>`

Loads **`<definite_length_block>`** into the memory location **`<file_name>`**.

The query returns the contents of the <file_name> in the format of a definite length block. This command can be used for copying files out of the analyzer over the remote bus. Refer to chapter 3, Programming Examples, for more information.

Example: Load “abcd” into C:source.txt:
`:MMEM:DATA “C:source.txt”,“#14abcd”`

Delete a File

`:MMEMory:DELete <file_name>`

Delete a file.

Example: **`:MMEM:DEL “C:source.txt”`**

Remarks: If <file_name> does not exist, a “File Name Error” will occur.

Front Panel
Access: **File, Delete**

Load a Corrections Table from a File

`:MMEMory:LOAD:CORRection`
`ANTenna|CABLe|OTHer|USER,<file_name>`

Loads the data in the file <file_name> to the specified correction set.

Example: **`:MMEM:LOAD:CORR ANT, “A:TEST5.CBL”`**

Front Panel
Access: **File, Load, Type, Corrections**

Load a Limit Line from Memory to the Instrument

`:MMEMory:LOAD:LIMit LLINE1|LLINE2,<file_name>`

Loads a limit line, from the specified file in mass storage to the instrument. Loading a time limit line deletes any frequency limit lines. Similarly, loading a frequency limit line deletes any time limit lines.

Example:. **`:MMEM:LOAD:LIM LLINE2,“C:mylimit.lim”`**

Remarks:. There is no SCPI short form for parameters **`LLINE1|LLINE2`**.

Front Panel
Access:. **File, Load, Type, Limits**

Load an Instrument State from a File

`:MMEMory:LOAD:STATe 1,<file_name>`

The contents of the state file are loaded into the current instrument state.

Example: **`:MMEM:LOAD:STAT 1,“C:mystate.sta”`**

Remarks: See also commands **`:MMEMory:LOAD:STATe`** and **`:MMEMory:STORe:STATe`**

If the revision of the state being loaded is newer than the revision of the instrument, no state is recalled and an error is reported.

If the revision of the state being loaded is equal to the revision of the instrument, all regions of the state will be loaded.

If the revision of the state being loaded is older than the revision of the instrument, the instrument will only load the older regions of the state.

Front Panel Access: **File, Load, Type, State**

Load a Trace From a File to the Instrument

`:MMEMory:LOAD:TRACe <file_name>`

The contents of the file are loaded into TRACE1. The file name must have a file extension of :trc or :csv. The file extension determines whether a trace is loaded, or a trace with its state, are loaded. The :csv extension is for trace files using the CSV (comma-separated values) format. The :trc extension is for files that include both trace and state data.

Example: **`:MMEM:LOAD:TRAC “C:mytrace.trc”`**

Remarks: See also commands **`:MMEMory:LOAD:STATe`** and **`:MMEMory:STORe:STATe`**

If the revision of the state being loaded is newer than the revision of the instrument, no state is recalled and an error is reported.

If the revision of the state being loaded is equal to the revision of the instrument, all regions of the state will be loaded.

If the revision of the state being loaded is older than the revision of the instrument, the instrument will only load the older regions of the state.

Make a Directory

`:MMEMory:MDIRectory <dir_path>`

where “path” is “A:\” or “C:\”

Makes a directory or subdirectory in the specified path.

Example: `Make a directory in C:\`, which is in instrument memory:
`:MMEMory:MDIRectory "C:\"`

Front Panel
Access: **File, Create Dir**

Store (Load/Save) a Signal List

`:MMEMory:LOAD:SIGNallist <file_name>`

`:MMEMory:STORe:SIGNallist <file_name>`

These commands are used to recall the signal list data from a disk or to save the signal list data to a disk.

Example: **`:MMEM:LOAD:SIGN, 'A:myfiles.lis'`**
`:MMEM:STOR:SIGN, 'A:xfiles.lis'`

Remarks: Signal list files should be saved with a `.lis` extension since this is the extension assumed by the file manager. You can save signal list files under a different extension but you will not be able to load these files via the front panel keys.

The load command for a signal list behaves differently than the **`LOAD`** command for **`AMPCOR`** or limit lines. For most kinds of instrument data the **`LOAD`** command performs a “destructive read”; it replaces any existing data with the data from the disk. The **`:MEM:LOAD:SIGN`** command does an “additive read”; it merges the data loaded with any data currently in the signal list. Signal list files are stored in ASCII text files with section markers to delineate content. The data in a signal list is stored in the [DATA] section of the file. The contents of this section are described in the table below.

Column[a]	**Contents**
1	Row number (starts at 1)
2	Frequency (Hz)
3	Peak detector amplitude (dBmV)
4	Quasi-peak detector amplitude (dBmV)
5	Average detector amplitude (dBmV)
6	Total amplitude correction (dBm)

Column[a]	Contents
7	Comment (enclosed in double quotes)
8	Marked/unmarked state (0=unmarked, 1=marked)
9	Uncertainty (MHz)
10	Status word (used internally)
11	Peak delta from limit line 1
12	Peak delta from limit line 2
13	Quasi-peak delta from limit line 1
14	Quasi-peak delta from limit line 2
15	Average delta from limit line 1
16	Average delta from limit line 2

a. An asterisk (*) in the column indicates that the data was missing or not applicable at the time the signal list data was saved to disk.

Front Panel
Access: **File, Save, Signal List**

File, Load, Signal List

Delete a Directory

`:MMEMory:RDIRectory <dir_name>`

Deletes the specified directory and all files and subdirectories within that directory.

Front Panel
Access: **File, Delete**

Store a Corrections Table to a File

`:MMEMory:STORe:CORRection`
`ANTenna|CABLe|OTHer|USER,<file_name>`

Stores the specified correction set to the file named <file_name>.

Example: **`:MMEM:STOR:CORR ANT, "A:TEST1.ANT"`**

Remarks: This command will fail if the **`<file_name>`** already exists.

Front Panel
Access: **File, Save, Type, Corrections**

Store a Limit Line in a File

`:MMEMory:STORe:LIMit LLINE1|LLINE2,<file_name>`

Stores the specified limit line to the specified file in memory.

Example: **`:MMEM:STOR:LIM LLINE2,"C:mylimit.lim"`**

Remarks: This command will fail if the **`<file_name>`** already exists. There is no SCPI short form for parameters **`LLINE1|LLINE2`**.

Front Panel Access: **File, Save, Type, Limits**

Store Measurement Results in a File

`:MMEMory:STORe:RESults <file_name>`

Saves the results of the current measurement into a comma-separated file. Only works when a measurement has been chosen from the **MEASURE** menu. The filename extension is .CSV. This command will fail if the file **`<file_name>`** already exists.

Example: **`:MMEM:STOR:RES "A:ACP.CSV"`**

Front Panel Access: **File, Save, Type, Measurement Results**

Store a Screen Image in a Graphic File

`:MMEMory:STORe:SCReen <file_name>`

Saves the current instrument screen image, as a graphic file, to the specified file in memory. The file must have a :gif or :wmf file extension. The specified file extension determines which file format the instrument will use to save the image.

Example: **`:MMEM:STOR:SCR "C:myscreen.gif"`**

Remarks: This command will fail if the **`<file_name>`** already exists.

Front Panel Access: **File, Save, Type, Screen**

Store an Instrument State in a File

`:MMEMory:STORe:STATe 1,<file_name>`

Saves the instrument state to the file in memory.

Example: **`:MMEM:STOR:STAT 1,"C:mystate.sta"`**

Remarks: This command will fail if the **`<file_name>`** already exists.

Store a Trace in a File

`:MMEMory:STORe:TRACe <label>,<file_name>`

Saves the specified trace to a file in memory. The file name must have a file extension of :trc or :csv. The file extension determines whether a trace is stored, or a trace with its state, are stored. The :csv extension is for trace files using the CSV (comma-separated values) format. The :trc extension is for files that include both trace and state data.

Example:	**`:MMEM:STOR:TRAC TRACE3,“C:mytrace.trc”`**
Range:	Trace labels are: TRACE1|TRACE2|TRACE3|ALL
Remarks:	This command will fail if the **`<file_name>`** already exists.
Front Panel Access:	**File, Save, Type, Trace**

OUTPut Subsystem

The OUTPut subsystem controls the characteristics of the tracking generator output port. Refer to the “SOURce Subsystem”, which also contains commands that control the characteristics of the tracking generator.

Turn Output On/Off

```
:OUTPut[:STATe] OFF|ON|0|1
```

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

Controls the tracking generator output.

Factory Preset and *RST:	Off
Front Panel Access:	**Source, Amplitude On Off**

SENSe Subsystem

Sets the instrument state parameters so that you can measure the input signal.

SENSe subsystem commands used for measurements in the **MEASURE** and **Meas Setup** menus may only be used to set parameters of a specific measurement when the measurement is active. Otherwise, an error will occur. You must first select the appropriate measurement using the `:CONFigure:<measurement>` command. If a `:SENSe` command is used to change a parameter during a measurement (while not in its idle state), the measurement will be restarted.

[:SENSe]:AVERage Subsection

Clear the Current Average

`[:SENSe]:AVERage:CLEar`

Re-start the trace averaging function.

NOTE Re-start the trace at the beginning of a sweep to obtain valid average data. To do this, remotely abort the sweep and initiate a single sweep.

Set the Average Count

`[:SENSe]:AVERage:COUNt <integer>`

`[:SENSe]:AVERage:COUNt?`

Specifies the number of measurements that are combined.

Factory Preset and *RST:	100
Range:	1 to 8192
Front Panel Access:	**BW/Avg, Average On Off**

Turn Averaging On/Off

`[:SENSe]:AVERage[:STATe] OFF|ON|0|1`

`[:SENSe]:AVERage[:STATe]?`

This command toggles averaging off and on. Averaging combines the value of successive measurements to average out measurement variations.

Factory Preset and *RST:	Off
Remarks:	When a measurement under the front panel **MEASURE** key is started, this command is turned off for video averaging (**`[:SENSe]:AVERage:TYPE VIDeo`**). If this command is turned on for video averaging when any of the **MEASURE** key measurements are in progress, that measurement will be stopped.
Front Panel Access:	**BW/Avg, Average On Off**

Turn Automatic Averaging On/Off

[:SENSe]:AVERage:TYPE:AUTO OFF|ON|0|1

[:SENSe]:AVERage:TYPE:AUTO?

Sets the averaging to be automatically set to the appropriate type for the current measurement setup. Or allows you to manually choose the type of averaging with **[:SENSe]:AVERage:TYPE**.

When AUTO is On:

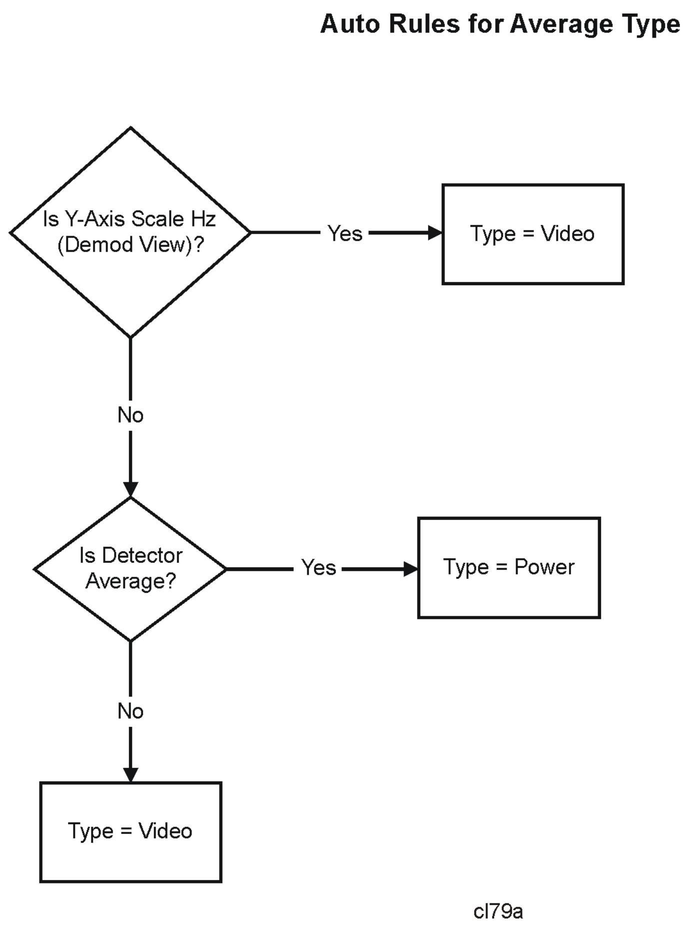
If the Y Axis Scale is not Linear or Log, then average type is Video (Y Axis Scale) Averaging.

If the Y Axis Scale is Linear or Log, then average type is Power Averaging.

If the Detector is Peak, Sample, or Negative Peak (not Average), then average type is Video Average.

See Figure 5-2, which shows these auto rules for average type in flowchart format.

Figure 5-2 Auto Rules for Average Type



Factory Preset and *RST:	On
History:	Added with firmware revision A.08.00.
Front Panel Access:	**BW/Avg, Avg Type, Auto Man**

Type of Averaging for Measurements

`[:SENSe]:AVERage:TYPE VIDeo|RMS`

`[:SENSe]:AVERage:TYPE?`

Successive measurements of data can be combined to average out measurement variations. Detector is set to average and Avg type is set to power (RMS) to measure RMS voltage (avg power).

NOTE As a best practice, set amplitude scale (**`:DISP:WIND:TRAC:Y:SPAC`**) prior to average type.

`VIDeo` logarithmically averages the power of the video data (typical units are dBm). This command is equivalent to pressing front panel keys **BW/Avg, Avg Type, Video.**

`RMS` averages the linear power of successive measurements (typical units are watts).

The following parameters of this command are supported, but not recommended for new designs. They are provided for limited compatibility to other analyzers. When used, the parameters are converted as follows:

TYPE LINear maps to RMS.

TYPE LPOWer maps to VIDeo.

TYPE POWer maps to RMS.

TYPE SCALar and VOLTage will map to VIDeo. If the amplitude scale is LOG, the setting is allowed, but an error is generated.

TYPE LOG maps to VIDeo. If the amplitude scale is not LOG (linear or Y Axis Units = Hz), the setting is allowed, but an error is generated.

For compatibility with firmware revisions prior to A.08.00, query **`[:SENSe]:AVERage:TYPE?`** will return LPOW or POW if LPOW or POW is used during the setting and no further changes have occurred to set the average type (such as from the front panel).

Factory Preset and *RST:	VID
History:	Changed with firmware revision A.08.00.
Front Panel Access:	**BW/Avg, Avg Type**

[:SENSe]:BANDwidth Subsection

Resolution Bandwidth

`[:SENSe]:BANDwidth|BWIDth[:RESolution] <freq>`

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

Specifies the resolution bandwidth.

Example:	`BAND 1 kHz`
Range:	10 Hz to 5 MHz (with Option 1DR)
	1 Hz to 5 MHz (E7401A, E7402A, E7403A, E7404A, E7405A with Option 1D5)
Default Unit:	Hz
Front Panel Access:	**BW/Avg, Resolution BW Auto Man**

Resolution Bandwidth Automatic

`[:SENSe]:BANDwidth|BWIDth[:RESolution]:AUTO OFF|ON|0|1`

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

Couples the resolution bandwidth.

Factory Preset and *RST:	On
Example:	`BWID:AUTO On`
History:	This command function changed with firmware revision A.08.00. With **`:AUTO ON`** in zero span, an error will be generated.
Remarks:	Auto-couple resolution bandwidth is not available in zero span.

Resolution Bandwidth Mode

`[:SENSe]:BANDwidth|BWIDth[:RESolution]:MODE EMI|SAN|OFF`

`[:SENSe]:BANDwidth|BWIDth[:RESolution]:MODE?`

Couples the resolution bandwidth to the frequency span in SAN mode, to Center Frequency in EMI mode, or uncoupled when OFF.

Factory Preset and *RST:	EMI

Video Bandwidth

`[:SENSe]:BANDwidth|BWIDth:VIDeo <freq>`

`[:SENSe]:BANDwidth|BWIDth:VIDeo?`

Specifies the video bandwidth.

Factory Preset and *RST:	300 kHz
Range:	1 Hz to 3 MHz. This range is dependent upon the setting of `[:SENSe]:BANDwidth|BWIDth[:RESolution]` and installed options.
Default Unit:	Hz
Front Panel Access:	**BW/Avg, Video BW Auto Man**

Video Bandwidth Automatic

`[:SENSe]:BANDwidth|BWIDth:VIDeo:AUTO OFF|ON|0|1`

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

Couples the video bandwidth to the resolution bandwidth.

Factory Preset and *RST:	On
Front Panel Access:	**BW/Avg, Video BW Auto Man**

Video to Resolution Bandwidth Ratio

`[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio <number>`

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

Specifies the ratio of the video bandwidth to the resolution bandwidth.

Factory Preset and *RST:	3.0
Range:	0.00001 to 3.0e6
Front Panel Access:	**BW/Avg, VBW/RBW Ratio**

Video to Resolution Bandwidth Ratio Mode Select

`[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio:AUTO OFF|ON|0|1`

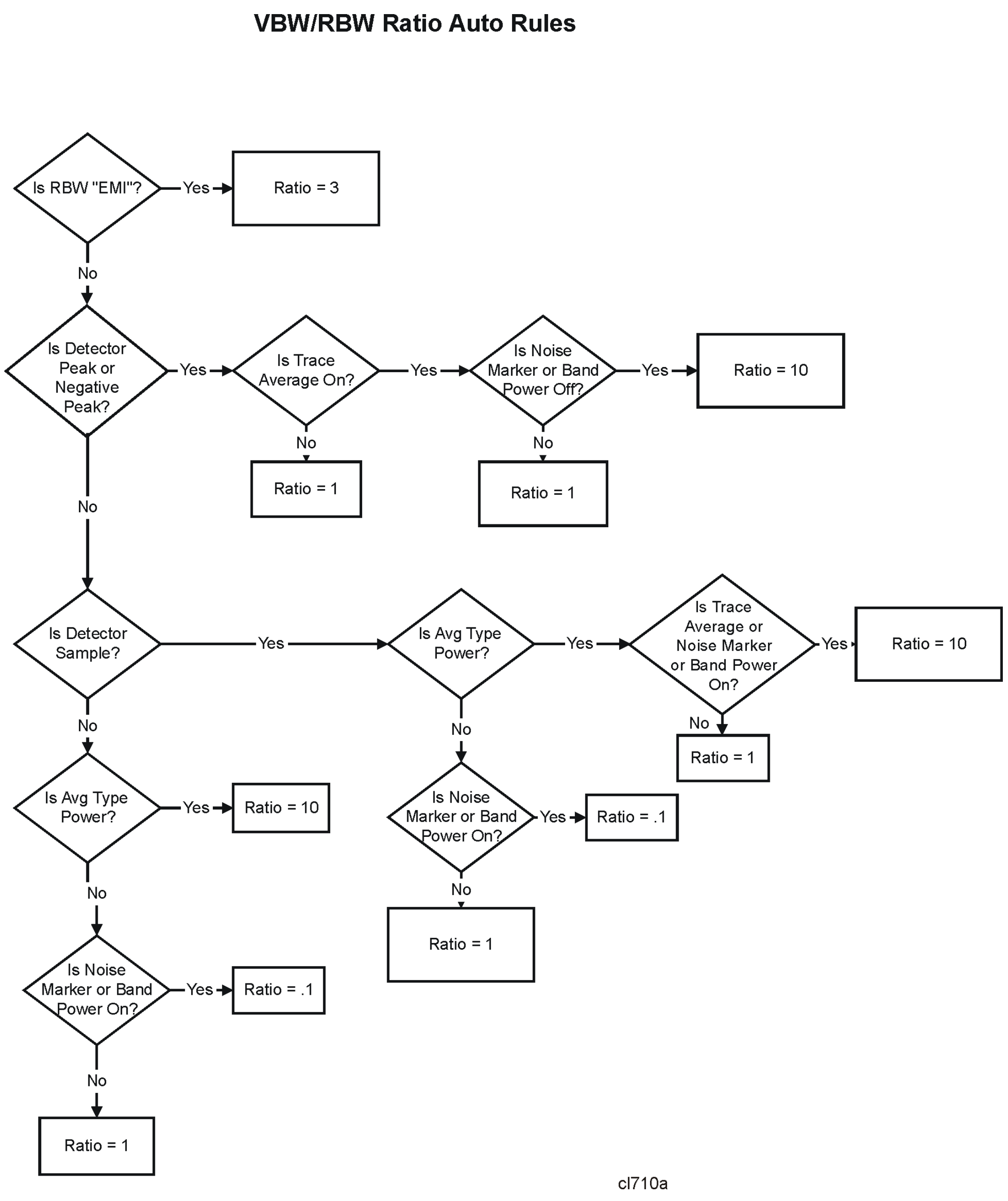
`[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio:AUTO?`

Selects auto or manual mode for video bandwidth to resolution bandwidth ratio.

Refer to Figure 5-3, which is a flowchart that illustrates VBW and RBW Ratio auto rules.

Factory Preset and *RST:	On
History:	Added with firmware revision A.08.00.
Front Panel Access:	**BW/Avg, VBW/RBW, Auto Man**

Figure 5-3 VBW and RBW Ratio Auto Rules



Resolution Bandwidth Type

`[:SENSe]:BANDwidth:TYPE IMPulse|DB6|DB3`

`[:SENSe]:BANDwidth:TYPE?`

Selects the type of 1 MHz resolution bandwidth (RBW) used. FCC regulations specify a 6 dB, 1 MHz resolution bandwidth for measurements greater than 1 GHz. CISPR regulations (1999) specify a 1 MHz impulse resolution bandwidth. Spectrum analyzers use a 3 dB resolution bandwidth.

Factory Preset and *RST:	IMPulse
Front Panel Access:	**BW/Avg, 1 MHz BW Type**

[:SENSe]:CORRection Subsection

Delete All Corrections

`[:SENSe]:CORRection:CSET:ALL:DELete`

This command deletes all existing corrections.

History: Added with firmware revision A.08.00.

Front Panel Access: **Amplitude/Y Scale, Corrections, Delete All Corrections**

Perform Amplitude Correction

`[:SENSe]:CORRection:CSET:ALL[:STATe] OFF|ON|0|1`

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

Turns On or Off the amplitude corrections. When turned On, only the correction sets that were turned on are enabled. When turned Off, all of the correction sets are disabled.

Factory Preset and *RST: Off

Remarks: To turn On or Off an individual correction set, use: `[:SENSe]:CORRection:CSET[1]|2|3|4[:STATe]`.

Front Panel Access: **Amplitude/Y Scale, Corrections, Antenna, Correction On Off**

Amplitude/Y Scale, Corrections, Cable, Correction On Off

Amplitude/Y Scale, Corrections, Other, Correction On Off

Amplitude/Y Scale, Corrections, User, Correction On Off

Set Amplitude Correction Data

```
[:SENSe]:CORRection:CSET[1]|2|3|4:DATA
<freq>,<rel_ampl>{,<freq>,<rel_ampl>}
```

`[:SENSe]:CORRection:CSET[1]|2|3|4:DATA?`

Sets the amplitude correction data. These frequency/amplitude corrections will be applied to the displayed data to correct for system losses/gains outside the analyzer. Four different sets of correction data can be stored.

Example:
```
:CORR:CSET1:DATA
900E6,0.3,1.0E9,0.35,1.3E9,0.2
```

Range: 200 points per set

Default Unit: There are no units on the frequency and amplitude pairs. They must be entered in hertz (Hz) and decibels (dB).

Remarks: CSET number equivalents to front panel access definitions are as follows:

CSET or CSET1 is Antenna
CSET2 is Cable
CSET3 is Other
CSET4 is User

Front Panel Access:

Amplitude/Y Scale, Corrections, Antenna, Edit Point|Frequency|Amplitude|Delete Point

Amplitude/Y Scale, Corrections, Cable, Edit Point|Frequency|Amplitude|Delete Point

Amplitude/Y Scale, Corrections, Other, Edit Point|Frequency|Amplitude|Delete Point

Amplitude/Y Scale, Corrections, User, Edit Point|Frequency|Amplitude|Delete Point

Merge Additional Values into the Existing Amplitude Correction Data

```
[:SENSe]:CORRection:CSET[1]|2|3|4:DATA:MERGe
<freq>,<rel_ampl>{,<freq>,<rel_ampl>}
```

Adds the points with the specified values to the current amplitude correction data, allowing you to merge correction data. If too much data is merged, as many points as possible are merged into the existing data and then an error is reported.

- `<freq>` is the frequency (in Hz) where the correction should be applied; no unit is allowed in this parameter
- `<rel_ampl>` is the amount of relative amplitude correction (in dB) needed; no unit is allowed in this parameter

Remarks: CSET number equivalents to front panel access definitions are as follows:

CSET or CSET1 is Antenna
CSET2 is Cable
CSET3 is Other
CSET4 is User

Delete Amplitude Correction

`[:SENSe]:CORRection:CSET[1]|2|3|4:DELete`

Deletes the specified correction set. If the set was On, it is turned Off.

Front Panel Access:	**AMPLITUDE/Y Scale, Corrections, Antenna|Cable|Other|User, Delete Correction**

Set Amplitude Correction Frequency Interpolation

`[:SENSe]:CORRection:CSET[1]|2|3|4:X:SPACing LINear|LOGarithmic`

Sets the frequency interpolation to linear or logarithmic for the specified correction set.

Remarks:	Logarithmic frequency scale corrections are linearly interpolated between correction points with respect to the logarithm of the frequency. Linear frequency scale corrections are interpolated along straight lines, connecting adjacent points on a linear scale.
Front Panel Access:	**AMPLITUDE/Y Scale, Corrections, Freq Interp Log Lin**

Perform Amplitude Correction

`[:SENSe]:CORRection:CSET[1]|2|3|4[:STATe] OFF|ON|0|1`

`[:SENSe]:CORRection:CSET[1]|2|3|4[:STATe]?`

Turns the amplitude correction function on or off for the given set.

NOTE `[:SENSe]:CORRection:CSET:ALL[:STATe]` must be on for this command to function.

Factory Preset and *RST:	Off
Remarks:	CSET number equivalents to front panel access definitions are as follows: CSET or CSET1 is Antenna CSET2 is Cable CSET3 is Other CSET4 is User
Front Panel Access:	**AMPLITUDE/Y Scale, Corrections, Antenna|Cable|Other|User, Correction On Off**

Input Impedance Correction

`[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude] <number>`

`[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude]?`

Amplitude correction is applied to the display data to adjust for measurement situations where the unit under test has a different impedance than the 50Ω input impedance of the analyzer.

Factory Preset and *RST:	The factory default is the input impedance of the analyzer.
Range:	50 or 75 ohms
Default Unit:	ohms
Front Panel Access:	**Input, Input Z Corr 50 Ω 75 Ω**

External Amplifier Correction

`[:SENSe]:CORRection:OFFSet[:MAGNitude] <rel_ampl>`

`[:SENSe]:CORRection:OFFSet[:MAGNitude]?`

A single value of amplitude correction can be applied to the displayed trace data to compensate for signal losses or gains that are due to other devices in the measurement setup, rather than the unit under test.

Factory Preset and *RST:	0 dB
Range:	–81.9 to 81.9
Default Unit:	dB
Front Panel Access:	**AMPLITUDE/Y Scale, Ext Amp Gain**

[:SENSe]:DEMod Subsection

Type of Demodulation

`[:SENSe]:DEMod AM|FM`

`[:SENSe]:DEMod?`

Sets the type of demodulation.

Factory Preset and *RST:	AM
Front Panel Access:	**Det/Demod, Demod, AM**
	Det/Demod, Demod, FM

FM Deviation

`[:SENSe]:DEMod:FMDeviation <freq>`

`[:SENSe]:DEMod:FMDeviation?`

Sets the total FM frequency deviation for full screen demodulation.

Factory Preset and *RST:	100 kHz
Range:	5 kHz to 1.2 MHz
Default Unit:	Hz
Front Panel Access:	**AMPLITUDE, Scale/Div**

Squelch

[:SENSe]:DEMod:SQUelch <integer>

Sets the squelch level on FM demod.

Factory Preset and *RST:	integer, 0 to 100
Key Access:	**Det/Demod, Demod/Audio, Squelch**

Demodulation Control

`[:SENSe]:DEMod:STATe OFF|ON|0|1`

`[:SENSe]:DEMod:STATe?`

Turns demodulation on or off.

Factory Preset and *RST:	Off
Front Panel Access:	**Det/Demod, Demod, Off**

Demod Time

`[:SENSe]:DEMod:TIME <time>`

`[:SENSe]:DEMod:TIME?`

Sets the time used for frequency domain demodulation.

Factory Preset and *RST:	500 ms
Range:	2 ms to 100 s
Default Unit:	seconds
Front Panel Access:	**Det/Demod, Demod, Demod Time**

Demod View

`[:SENSe]:DEMod:VIEW[:STATe] OFF|ON|0|1`

`[:SENSe]:DEMod:VIEW[:STATe]?`

This command causes the demodulated signal to be displayed. If FM Demod is on, then the display scales the y-axis in units of kHz. The scale/div is set with the command **`:DISPlay:WINDow:TRACe:Y [:SCALe]:PDIVision:FREQuency <freq>`** if FM Demod is on. If FM Demod is on, then several functions are not available; these include: Log/Lin (display is always in linear), Y-Axis Units, Marker Search functions, Normalize, Display Line, Peak Excursion, and Peak Threshold. There is no effect when AM demodulation is used (only applicable for FM demodulation).

Factory Preset and *RST:	Off
Remarks:	This command is not available when Demod is set to Off.
Front Panel Access:	**Det/Demod, Demod, FM, Demod View**

[:SENSe]:DETector Subsection

Automatic Detection Type Selected

`[:SENSe]:DETector:AUTO OFF|ON|0|1`

`[:SENSe]:DETector:AUTO?`

Switches automatically to the optimum detection type for typical measurements using the current instrument settings.

The detector type is average if any of these are on:

Noise marker
Band power markers
Trace averaging when the Average Type is Power (RMS).

The detector type is sample if any of the following conditions are true:

Trace averaging is on with average type of video
Both max and min hold trace modes are on
Resolution bandwidth is less than 1 kHz, and noise marker, band power markers, or trace averaging is on

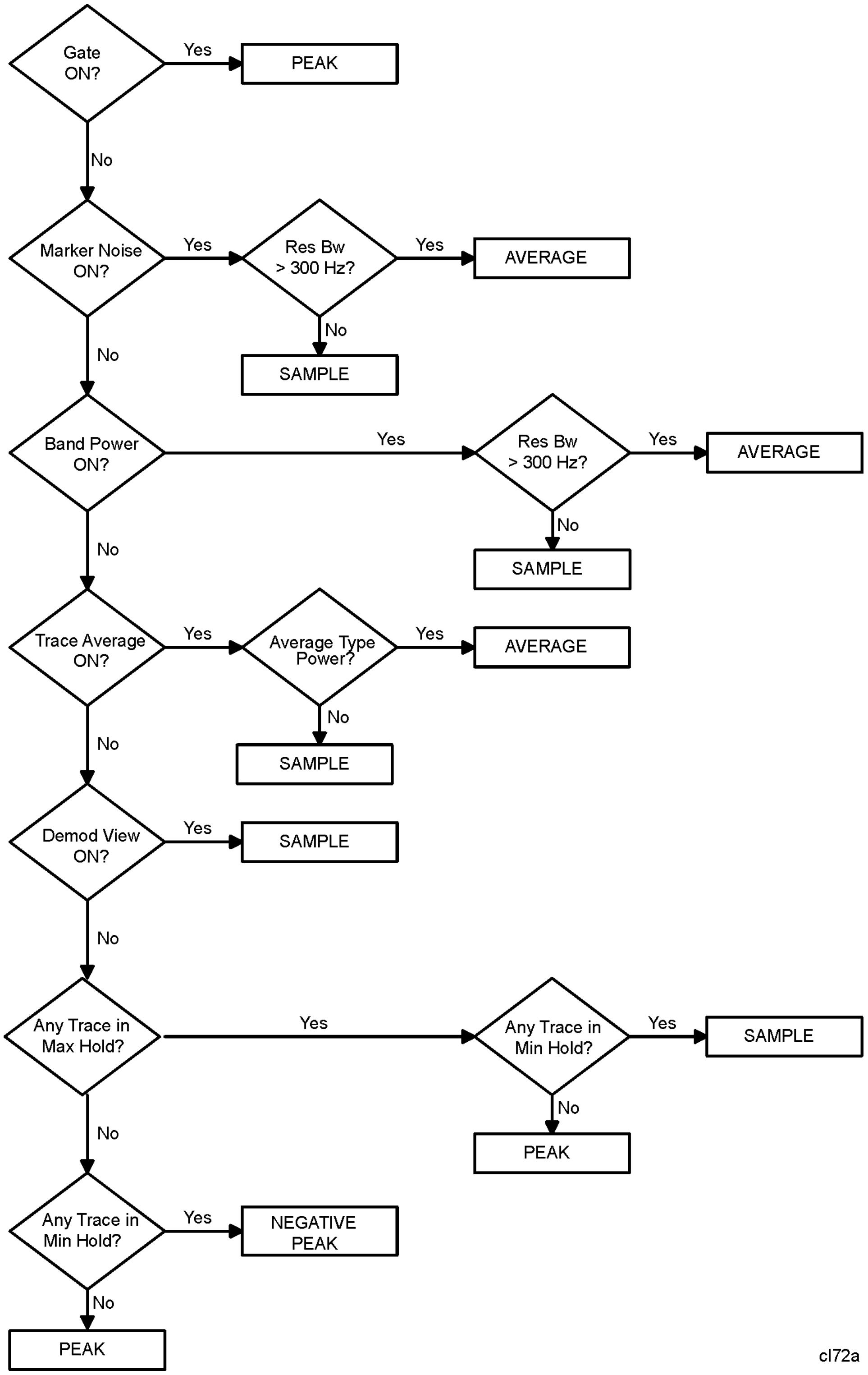
The detector type is negative peak if any trace is in min hold and no traces are in max hold.

The detector type is peak if the above conditions are off.

Manually changing the detector function turns Auto off.

Refer to Figure 5-4, which shows a decision tree of how detection type is determined.

Figure 5-4 Auto Rules of Detector Selection



Factory Preset and *RST:	On
History:	Added with firmware revision A.08.00.
Front Panel Access:	**Det/Demod**, **Detector**

Type of Detection

[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle|AVERage|RMS

[:SENSe]:DETector[:FUNCtion]?

Specifies the detection mode.

For each trace interval (bucket), average detection displays the average of all the samples within the interval. The averaging can be done using two methods:

the power method (RMS)
the video method (Y Axis Units)

The method is controlled by the BW/Avg, Avg Type key.

NOTE The combination of the average detector and the power average type is equivalent to what is sometimes referred to as "RMS detection."

Negative peak detection displays the lowest sample taken during the interval being displayed.

Positive peak detection displays the highest sample taken during the interval being displayed.

Sample detection displays the sample taken during the interval being displayed, and is used primarily to display noise or noise-like signals. In sample mode, the instantaneous signal value at the present display point is placed into memory. This detection should not be used to make the most accurate amplitude measurement of non noise-like signals.

Average detection is used when measuring the average value of the amplitude across each trace interval (bucket). The averaging method used by the average detector is set to either video or power as appropriate when the average type is auto coupled.

Factory Preset and *RST:	Positive
History:	Added Average and RMS elements to the command with firmware revision A.08.00.
Front Panel Access:	**Det/Demod**, **Detector** **Det/Demod**, **Detector**, **Peak** **Det/Demod**, **Detector**, **Sample**

Det/Demod, **Detector**, **Negative Peak**
Det/Demod, **Detector**, **Average**

Type of EMI Detector

`[:SENSe]:DETector[:FUNCtion]: EMI QPEak|AVERage|OFF`

`[:SENSe]:DETector[:FUNCtion]:EMI?`

Specifies the type of EMI detection mode. Quasi-peak detection displays a weighted, sample-detected amplitude using specific, charge, discharge, and meter time constants as described in CISPR Publication 16. Average detection displays the average value of a sample-detected amplitude.

Factory Preset and *RST:	Off
Remarks:	When either of the EMI detectors are selected for the first time, a ranging operation is performed which adjusts the reference level to a reasonable level for performing measurements. The ranging operation will first adjust the reference level in LOGarithmic scale units and then in LINear scale units. While doing so, “EMIPk” will be displayed in the upper-left corner of the display. Once the reference level has been properly adjusted, the selected EMI detector will be activated. Depending on the detector chosen, “EMI QP” or “EMIAv” will be displayed in the upper-left display corner. When setting the EMI detector to Off, the analyzer performs an UNRange and will display the various instrument settings adjusted by the range operation. The previous detector is also restored. When the QPeak or AVERage EMI detectors are selected, the detector setting is locked in SAMPle mode and the user is not allowed to adjust this value until the EMI detector is set to Off.
Key Access:	**Det/Demod, EMI Detector, Quasi Peak** **Det/Demod, EMI Detector, EMI Average** **Det/Demod, EMI Detector, Off**

EMI View

[:SENSe]:DETector[:FUNCtion]EMI:VIEW POSitive|EMI

[:SENSe]:DETector[:FUNCtion]EMI:VIEW?

Selects between Quasi-Peak/Average EMI detector mode or Peak detector mode without reranging. When POSitive is selected only the peak detector is used. When EMI is selected, the previously selected EMI detector is used.

Factory Preset and *RST:	EMI
Remarks:	This command is not available when the EMI detector is Off.
Key Access:	**Det/Demod, EMI Detector, View**

Range Immediate

[:SENSe]:DETector:RANGe:IMMediate

[:SENSe]:DETector:RANGe:IMMediate

Performs detector ranging (if enabled) when an EMI detector is selected.

Factory Preset and *RST:	Positive

Unrange

[:SENSe]:DETector[:UNRange]

Restores settings prior to last range operation.

[:SENSe]:EMI Subsection

Auto Measure Average On or Off

`[:SENSe]:EMI:MEASure:DETector:AVERage[STATe] OFF|ON|0|1`

`[:SENSe]:EMI:MEASure:DETector:AVERage?`

Sets auto measure On or Off.

Factory Preset and *RST:	Off
Remarks:	Determines if the average detector is measured (by auto measure, measure at marker, measure frequency, or remeasure).
Key Access:	**MEASURE, More, Auto Measure**

Auto Measure Peak On or Off

`[:SENSe]:EMI:MEASure:DETector:PPEak[STATe]OFF|ON|0|1|`

`[:SENSe]:EMI:MEASure:DETector:PPEak?`

Sets automeasure peak On or Off.

Factory Preset and *RST:	Off
Remarks:	Determines if the peak detector is measured (by auto measure, measure at marker, measure frequency, or remeasure).
Key Access:	**MEASURE, More, Auto Measure**

Auto Measure Quasi Peak On or Off

`[:SENSe]:EMI:MEASure:DETector:QPEak[STATe]OFF|ON|0|1|`

`[:SENSe]:EMI:MEASure:DETector:QPEak?`

Sets automeasure quasi peak On or Off.

Factory Preset and *RST:	Off
Remarks:	Determines if the quasi-peak detector is measured (by auto measure, measure at marker, measure frequency, or remeasure).
Key Access:	**MEASURE, More, Auto Measure**

Setting the Dwell Time for Peak

`[:SENSe]:EMI:MEASure:DETector:PPEak:DWELl<time>`

`[:SENSe]:EMI:MEASure:DETector:PPEak:DWELl?`

Sets the dwell time for the peak detector for Measure at Marker, Automeasure, and Remeasure.

Factory Preset and *RST:	Off
Remarks:	Sets the dwell time

Setting the Dwell Time for Quasi Peak

`[:SENSe]:EMI:MEASure:DETector:QPEak:DWELl<time>`

`[:SENSe]:EMI:MEASure:DETector:QPEak:DWELl?`

Sets the dwell time for the quasi-peak detector for Measure at Marker, Automeasure, and Remeasure.

Factory Preset and *RST:	Off
Remarks:	Sets the dwell time

Setting the Dwell Time for Average Peak

`[:SENSe]:EMI:MEASure:DETector:AVERage:DWELl<time>`

`[:SENSe]:EMI:MEASure:DETector:AVERage:DWELl?`

Sets the dwell time for the average detector for Measure at Marker, Automeasure, and Remeasure.

Factory Preset and *RST:	Off
Remarks:	Sets the dwell time

Preselector Centering On or Off (E7403A, E7404A, E7405A only)

`[:SENSe]:EMI:MEASure:PCENter[:STATe] OFF|ON|0|1`

`[:SENSe]:EMI:MEASure:PCENter[:STATe]?`

Determines if preselector centering should be performed prior to Measure at Marker, Automeasure, and Remeasure.

Factory Preset and *RST:	Off
Remarks:	*Available on Agilent E7403A, E7404A, and E7405A only.*

Setting the Dwell Time for Range

`[:SENSe]:EMI:MEASure:RANGe:DWELl<time>`

`[:SENSe]:EMI:MEASure:RANGe:DWELl?`

Sets the dwell time for ranging for Measure at Marker, Automeasure, and Remeasure.

Factory Preset and *RST:	200 ms

Auto Measure Margin On or Off

`[:SENSe]:EMI:MEASure:PEAKs:SGTMargin[:STATe]ON|OFF|1|0`

`[:SENSe]:EMI:MEASure:PEAKs:SGTMargin?`

Sets automeasure margin On or Off.

Factory Preset and *RST:	Off
Remarks:	If on when automeasuring, only the signals above the margin are measured and added to the signal list.
Key Access:	**MEASURE, More, Auto Measure**

[:SENSe]:FREQuency Subsection

Center Frequency

`[:SENSe]:FREQuency:CENTer <freq>`

`[:SENSe]:FREQuency:CENTer UP|DOWN`

`[:SENSe]:FREQuency:CENTer?`

Set the center frequency.

NOTE In log sweep mode, the minimum start frequency is 10 Hz.

Factory Preset and *RST:	600 MHz
Range:	EMC E7401A: –80 MHz[1] to 1.58 GHz
	EMC E7402A: –80 MHz[1] to 3.10 GHz
	EMC E7403A: –80 MHz[1] to 6.78 GHz
	EMC E7404A: –80 MHz[1] to 13.3 GHz
	EMC E7405A: –80 MHz[1] to 27.0 GHz
Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, Center Freq**

Center Frequency Step Size Automatic

`[:SENSe]:FREQuency:CENTer:STEP:AUTO OFF|ON|0|1`

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

Specifies whether the step size is set automatically based on the span.

Factory Preset and *RST:	On
Front Panel Access:	**FREQUENCY/Channel, CF Step Auto Man**

1. 10 Hz minimum in log sweep mode.

Center Frequency Step Size

`[:SENSe]:FREQuency:CENTer:STEP[:INCRement] <freq>`

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

Specifies the center frequency step size.

Factory Preset and *RST:	Span/10
Range:	Maximum negative frequency to the maximum positive frequency listed below:
	EMC E7401A: –1.58 to 1.58 GHz
	EMC E7402A: –3.10 to 3.10 GHz
	EMC E7403A: –6.78 to 6.78 GHz
	EMC E7404A: –13.3 to 13.3 GHz
	EMC E7405A: –27.0 to 27.0 GHz
Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, CF Step Man**

Frequency Span

`[:SENSe]:FREQuency:SPAN <freq>`

`[:SENSe]:FREQuency:SPAN?`

Set the frequency span. Setting the span to 0 Hz puts the analyzer into zero span.

Factory Preset and *RST:	800 MHz
Range:	EMC E7401A: 0 Hz, 100 Hz to 1.58 GHz
	EMC E7402A: 0 Hz, 100 Hz to 3.10 GHz
	EMC E7403A: 0 Hz, 100 Hz to 6.78 GHz
	EMC E7404A: 0 Hz, 100 Hz to 13.3 GHz
	EMC E7405A: 0 Hz, 100 Hz to 27.0 GHz
Default Unit:	Hz
Front Panel Access:	**SPAN/X Scale, Span**
	SPAN/X Scale, Zero Span

Full Frequency Span

`[:SENSe]:FREQuency:SPAN:FULL`

Set the frequency span to full scale.

Factory Preset and *RST:	800 MHz
Front Panel Access:	**SPAN/X Scale, Full Span**

Last Frequency Span

`[:SENSe]:FREQuency:SPAN:PREVious`

Set the frequency span to the previous span setting.

Front Panel Access:	**SPAN/X Scale, Last Span**

Start Frequency

`[:SENSe]:FREQuency:STARt <freq>`

`[:SENSe]:FREQuency:STARt?`

Set the start frequency.

NOTE In log sweep mode, the minimum start frequency is 10 Hz.

Factory Preset and *RST:	200 MHz
Range:	EMC E7401A: –80 MHz[1] to 1.58 GHz
	EMC E7402A: –80 MHz[1] to 3.10 GHz
	EMC E7403A: –80 MHz[1] to 6.78 GHz
	EMC E7404A: –80 MHz[1] to 13.3 GHz
	EMC E7405A: –80 MHz[1] to 27.0 GHz
Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, Start Freq**

1. 10 Hz minimum in log sweep mode.

Stop Frequency

`[:SENSe]:FREQuency:STOP <freq>`

`[:SENSe]:FREQuency:STOP?`

Set the stop frequency.

Factory Preset and *RST:	1.06 GHz
Range:	EMC E7401A: –80 MHz[1] to 1.58 GHz
	EMC E7402A: –80 MHz[1] to 3.10 GHz
	EMC E7403A: –80 MHz[1] to 6.78 GHz
	EMC E7404A: –80 MHz[1] to 13.3 GHz
	EMC E7405A: –80 MHz[1] to 27.0 GHz
Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, Stop Freq**

Frequency Synthesis Mode

`[:SENSe]:FREQuency:SYNThesis 1|2|3`

`[:SENSe]:FREQuency:SYNThesis?`

This command switches between two phase noise optimization modes. Mode 2 optimizes the analyzer for close-in phase noise. Mode 3 optimizes the analyzer for tuning speed. Mode 1 is not recommended for new designs.

This command is available for the following models only:

E7402A, E7403A, E7404A, E7405A

Factory Preset and *RST:	3
History:	Added with firmware revision A.08.00.
Front Panel Access:	**AUTO COUPLE, PhNoise Opt**

1. 10 Hz minimum in log sweep mode.

Frequency Synthesis State

`[:SENSe]:FREQuency:SYNThesis:AUTO OFF|ON|0|1`

`[:SENSe]:FREQuency:SYNThesis:AUTO?`

This command switches between auto and manual phase noise selection.

When in auto mode, the phase noise optimization is set as follows:

- For spans ≤ 10 MHz, the analyzer is optimized for phase noise.
- For spans > 10 MHz, the analyzer is optimized for fast tuning.

This command is available for the following models only:

E7402A, E7403A, E7404A, E7405A

Factory Preset and *RST:	On
History:	Added with firmware revision A.08.00.
Front Panel Access:	**AUTO COUPLE, PhNoise Opt**

[:SENSe]:POWer Subsection

Enable/Disable QPD X10 Gain

`[:SENSe]:POWer:QPGain[:STATe]ON|OFF|1|0`

`[:SENSe]:POWer:QPGain[:STATe]?`

Sets the quasi peak (QP) gain state On or Off in the quasi peak detector (QPD) board.

Factory Preset and *RST:	Off
Key Access:	Det/Demod, EMI Detector, AV/QP Gain X1 X10

Input Attenuation

`[:SENSe]:POWer[:RF]:ATTenuation <rel_ampl>`

`[:SENSe]:POWer[:RF]:ATTenuation?`

Set the input attenuator. This value is set at its auto value if input attenuation is set to auto.

Factory Preset and *RST:	10 dB
Range:	EMC E7401A: 0 to 60 dB
	EMC E7402A: 0 to 75 dB
	EMC E7403A: 0 to 75 dB
	EMC E7404A: 0 to 75 dB
	EMC E7405A: 0 to 65 dB
Default Unit:	dB
Front Panel Access:	**AMPLITUDE/Y Scale, Attenuation Auto Man**

Input Port Attenuator Auto

`[:SENSe]:POWer[:RF]:ATTenuation:AUTO OFF|ON|0|1`

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

Select the input port attenuator range to be set either automatically or manually.

On – Input attenuation is automatically set as determined by the Reference Level Setting.

Off – Input attenuation is manually set

Factory Preset and *RST:	On
Front Panel Access:	**AMPLITUDE Y Scale, Attenuation**

Input Port Power Gain

`[:SENSe]:POWer[:RF]:GAIN[:STATe] OFF|ON|0|1`

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

Turns the internal preamp on or off.

Factory Preset and *RST:	Off
Front Panel Access:	AMPLITUDE/Y Scale, Int Preamp On Off

Input Port Maximum Mixer Power

`[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer] <ampl>`

`[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer]?`

Specifies the maximum power at the input mixer.

Factory Preset and *RST:	–10 dBm
Range:	–100 dBm to 10 dBm
Default Unit:	dBm
Front Panel Access:	**AMPLITUDE/Y Scale, Max Mixer Lvl**

Optimize Preselector Frequency

`[:SENSe]:POWer[:RF]:PADJust <freq>`

`[:SENSe]:POWer[:RF]:PADJust?`

This command allows user-defined adjustment of the preselector frequency to optimize its response on the signal of interest.

Factory Preset and *RST:	0 Hz
Range:	–250 MHz to 250 MHz
Default Unit:	None. Use the MHz terminator in order for this command to work.

Remarks:	This command is available only on Agilent EMC models E7403A, E7404A, and E7405A. Use this command for signals close to the noise level, multiple signals close together, or for other conditions when the preselector is not tuned to the frequency of interest.
Front Panel Access:	**AMPLITUDE/Y Scale, Presel Adjust**

Preselector Center

```
[:SENSe]:POWer[:RF]:PCENter
```

This command centers the preselector filter at the signal of interest. This command has no effect if it is activated in non-preselected bands. This command is usable from 3 GHz to the maximum frequency of the analyzer.

NOTE This command is available only on Agilent EMC models E7403A, E7404A, and E7405A. This command has no effect with markers set to less than 3 GHz.

Remarks:	A peak search will be done if no marker is on.
Front Panel Access:	**AMPLITUDE/Y Scale, Presel Center**

[:SENSe]:SWEep Subsection

Sweep Points

`[:SENSe]:SWEep:POINts <number of points>`

`[:SENSe]:SWEep:POINts?`

This command sets the number of sweep points.

Factory Preset and *RST:	401
Example:	**`:SWEep:POIN 401`**
History:	This command is available only on analyzers with firmware revision A.04.00 and later. Analyzers with firmware revisions prior to A.04.00 have the number of sweep points fixed at 401.
Range:	101 to 8192, (2 to 8192 in zero span for analyzers with firmware revision A.05.00 and later)
Remarks:	For analyzers with firmware revisions prior to A.08.00, any change to sweep points sets the following commands as shown: :CALCulate:LLINe1:DISPlay to off, and :CALCulate:LLINe2:DISPlay to off. Whenever the number of sweep points change, the following functions are affected: • All trace data is erased • Any traces in view mode will go to blank mode • Sweep time is re-calculated • Any limit lines that are on will be turned off (For analyzers with firmware revisions prior to A.08.00)
Front Panel Access:	**Sweep, Points**

Set Frequency Domain Scale Type

`[:SENSe]:SWEep:SPACing LINear|LOGarithmic`

`[:SENSe]:SWEep:SPACing?`

Selects either linear or logarithmic for the frequency domain (X-axis) scale. The trace query of comma-separated values maps frequency/amplitude pairs for the mathematical interpolation of the log frequency axis. The value of **`[:SENSe]:SWEep:POINts`** is adjusted to reflect the acquisition of data for the given sweep span when log sweep spacing is enabled.

Factory Preset and *RST:	Linear
Remarks:	Refer to the User’s Guide for detailed information on the interactions of this command with other functions.
History:	Added with firmware revision A.08.00.
Front Panel Access:	**FREQUENCY, Scale Type**

Sweep Time

`[:SENSe]:SWEep:TIME <time>`

`[:SENSe]:SWEep:TIME?`

Specifies the time in which the instrument sweeps the display.

Factory Preset and *RST:	120.8 ms
Range:	The range depends upon the installed options, number of sweep points, and firmware revision of your instrument. See “Sweep Time Range” in the Specifications Guide for details.
Default Unit:	seconds
Remarks:	A span value of 0 Hz causes the analyzer to enter zero span mode. In zero span the X-axis represents time rather than frequency. In this mode, the sweep time may be set to faster values when Option AYX is installed.
Front Panel Access:	**Sweep, Sweep Time Auto Man**

Automatic Sweep Time

`[:SENSe]:SWEep:TIME:AUTO OFF|ON|0|1`

`[:SENSe]:SWEep:TIME:AUTO?`

Automatically selects the fastest sweep time for the current settings.

Factory Preset and *RST:	On
History:	This command function changed with firmware revision A.08.00. With :**`AUTO ON`** in zero span, an error will be generated.
Front Panel Access:	**Sweep, Sweep Time Auto Man**

Sweep Time Mode

`[:SENSe]:SWEep:TIME:AUTO:MODE SRESponse|SANalyzer`

`[:SENSe]:SWEep:TIME:AUTO:MODE?`

Specifies the type of automatic coupling for the fastest sweep time at the current settings.

Stimulus response

Spectrum analyzer

Factory Preset and *RST:	SANalyzer
Front Panel Access:	**Sweep, Sweep Coupling SR SA**

Time Gating Delay (Option 1D6 Only)

`[:SENSe]:SWEep:TIME:GATE:DELay <time>`

`[:SENSe]:SWEep:TIME:GATE:DELay?`

Sets the delay time from when the gate trigger occurs to when the gate opens. This is for **`EDGE`** triggering only.

Factory Preset and *RST:	1 μs
Range:	0.3 μs to 429 seconds
Default Unit:	seconds
Front Panel Access:	**Sweep, Gate Setup, Edge Setup, Gate Delay**

Time Gate Length (Option 1D6 Only)

`[:SENSe]:SWEep:TIME:GATE:LENGth <time>`

`[:SENSe]:SWEep:TIME:GATE:LENGth?`

Specifies the gate time length in seconds; for **`EDGE`** triggering only.

Factory Preset and *RST:	1 μs
Range:	0.3 μs to 429 seconds
Default Unit:	seconds
Front Panel Access:	**Sweep, Gate Setup, Edge Setup, Gate Length**

Time Gate Level (Option 1D6 Only)

`[:SENSe]:SWEep:TIME:GATE:LEVel HIGH|LOW`

`[:SENSe]:SWEep:TIME:GATE:LEVel?`

Selects the level of the gate signal; this command is for LEVel triggering only.

Factory Preset and *RST:	High
Front Panel Access:	**Sweep, Gate Setup, Level Setup**

Time Gate Polarity (Option 1D6 Only)

`[:SENSe]:SWEep:TIME:GATE:POLarity NEGative|POSitive`

`[:SENSe]:SWEep:TIME:GATE:POLarity?`

Selects the polarity of the gate signal; this command is for EDGE triggering only.

Factory Preset and *RST:	Positive
Front Panel Access:	**Sweep, Gate, Edge Gate, Slope Pos Neg**

Preset Time Gate (Option 1D6 Only)

`[:SENSe]:SWEep:TIME:GATE:PRESet`

Presets the time-gated spectrum analysis capability.

Remarks: This command resets gate parameters to default values, as follows:

Gate trigger type = edge

Gate polarity = positive

Gate delay = 1 µs

Gate length = 1 µs

Gate level = high

Control Time Gate (Option 1D6 Only)

`[:SENSe]:SWEep:TIME:GATE[:STATe] OFF|ON|0|1`

`[:SENSe]:SWEep:TIME:GATE[:STATe]?`

Turns time gating on or off.

NOTE Time gate cannot be turned on if external trigger delay is on.

Factory Preset and *RST:	Off
Front Panel Access:	**Sweep, Gate, Gate On Off**

Time Gate Trigger Type (Option 1D6 Only)

`[:SENSe]:SWEep:TIME:GATE:TYPE LEVel|EDGE`

`[:SENSe]:SWEep:TIME:GATE:TYPE?`

Selects between edge and level mode for time-gated spectrum analysis.

Level triggers the gate when the signal surpasses a specific level, set to either low or high.

Edge triggers the gate when the edge of a signal is encountered, set to either a negative-going edge or a positive-going edge.

Factory Preset and *RST:	Edge
Front Panel Access:	**Sweep, Gate, Gate Control Edge Level**

SOURce Subsystem

¶The SOURce subsystem controls the signal characteristics of the tracking generator. Refer also to the "OUTPut Subsystem" on page 267 which contains a command that controls the tracking generator output.

Sets the Output Power Offset Correction

`:SOURce:CORRection:OFFSet <rel_ampl>`

`:SOURce:CORRection:OFFSet?`

Specifies an offset for the displayed output power level. An offset power level can be added to the displayed level to compensate for system losses (for example, cable loss) or gains (for example, preamplifier gain.) This offset does not change the power out of the source, it only changes the display so that it reads out the actual power delivered to the device under test.

Factory Preset and *RST:	0 dB
Range:	–327.6 dB to 327.6 dB
Default Unit:	Currently selected source power units
Front Panel Access:	**`Source, Amptd Offset`**

Source Attenuation

`:SOURce:POWer:ATTenuation <ampl>`

`:SOURce:POWer:ATTenuation?`

Attenuates the source output level. Specifically setting **`:SOURce:POWer:ATTenuation <ampl>`** sets the mode to manual (**`:SOURce:POWer:ATTenuation:AUTO OFF`**).

CAUTION When source attenuation is set to manual (**`:SOURce:POWer:ATTenuation:AUTO OFF`**), source amplitude may be set to values beyond actual output levels to accommodate the full range of analyzer capabilities. Therefore, exercise caution when connecting a power-level sensitive device to the tracking generator output.

Factory Preset and *RST:	EMC E7401A: 0 dB
	EMC E7402A: 8 dB
	EMC E7403A: 8 dB
	EMC E7404A: 8 dB

	EMC E7405A: 8 dB
Range:	EMC E7401A: 0 dB to 60 dB in 10 dB steps
	EMC E7402A: 0 dB to 56 dB in 8 dB steps
	EMC E7403A: 0 dB to 56 dB in 8 dB steps
	EMC E7404A: 0 dB to 56 dB in 8 dB steps
	EMC E7405A: 0 dB to 56 dB in 8 dB steps
Default Unit:	dB
Front Panel Access:	**Source, Attenuation Auto Man**

Automatic Source Attenuation

`:SOURce:POWer:ATTenuation:AUTO OFF|ON|0|1`

`:SOURce:POWer:ATTenuation:AUTO?`

Selects if the source output level attenuator will be set automatically, or manually.

CAUTION Power-level sensitive devices connected to the tracking generator output may be accidentally damaged. This is because the actual source amplitude may be greater than the amplitude indicated on the analyzer when the source attenuation is set manually.

When source attenuation is set to manual (**`:SOURce:POWer:ATTenuation:AUTO OFF`**), source amplitude may be set to values beyond actual output levels to accommodate the full range of analyzer capabilities. Therefore, exercise caution when connecting a power-level sensitive device to the tracking generator output.

Factory Preset and *RST:	On
Front Panel Access:	**Source, Attenuation Auto Man**

Sets the Output Power

`:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude] <ampl>`

`:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude] UP|DOWN`

`:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]?`

Specifies the source output power level. Use **`:SOURce:POWer:SWEep`** to set the change in power level across the sweep. Also see **`:SOURce:POWer:STARt`** and **`OUTPut[:STATe]`**.

CAUTION Power-level sensitive devices connected to the tracking generator output may be accidentally damaged. This is because the actual source amplitude may be greater than the amplitude indicated on the analyzer when the source attenuation is set manually.

When source attenuation is set to manual (**`:SOURce:POWer:ATTenuation:AUTO OFF`**), source amplitude may be set to values beyond actual output levels to accommodate the full range of analyzer capabilities. Therefore, exercise caution when connecting a power-level sensitive device to the tracking generator output.

Factory Preset and *RST:	97 dBμV	
Range:	EMC E7401A:	37 dBμV to 110 dBμV
	EMC E7402A:	41 dBμV to 110 dBμV
	EMC E7403A:	41 dBμV to 110 dBμV
	EMC E7404A:	41 dBμV to 110 dBμV
	EMC E7405A:	41 dBμV to 110 dBμV
Default Unit:	dBm	
Front Panel Access:	**Source, Amplitude On Off**	

Sets the Source Output Power Mode

`:SOURce:POWer:MODE FIXed|SWEep`

`:SOURce:POWer:MODE?`

Sets the source output to be at a single amplitude (fixed) or to sweep through a range of power levels.

Factory Preset and *RST:	Fixed
Front Panel Access:	**Source, Power Sweep On Off**

Set the Source Sweep Power Range

`:SOURce:POWer:SPAN <rel_ampl>`

`:SOURce:POWer:SPAN?`

Specifies the range of power levels through which the source output will sweep. Use :**`SOURce:POWer:STARt`** to set the power level at the start of the power sweep. This command is equivalent to :**`SOURce:POWer:SWEep.`**

Factory Preset and *RST:	0 dB
Range:	0 dB to 20 dB
Default Unit:	dB

Set the Output Power at the Start of the Sweep

:SOURce:POWer:STARt <ampl>

:SOURce:POWer:STARt?

Specifies the source output power level at the start of the power sweep. Use **:SOURce:POWer:SPAN** to set the change in power level across the sweep. This command is equivalent to **:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]**.

CAUTION Power-level sensitive devices connected to the tracking generator output may be accidentally damaged. This is because the actual source amplitude may be greater than the amplitude indicated on the analyzer when the source attenuation is set manually.

When source attenuation is set to manual (**:SOURce:POWer:ATTenuation:AUTO OFF**), source amplitude may be set to values beyond actual output levels to accommodate the full range of analyzer capabilities. Therefore, exercise caution when connecting a power-level sensitive device to the tracking generator output.

Set the Output Power to Step Automatically

:SOURce:POWer:STEP:AUTO OFF|ON|0|1

:SOURce:POWer:STEP:AUTO?

Specifies the source power step size to be one vertical scale division when in logarithmic scale, or 10 dB when in linear scale.

Factory Preset and *RST:	On
Front Panel Access:	**Source, Amptd Step Auto Man**

Set the Output Power Step Size

:SOURce:POWer:STEP[:INCRement] <ampl>

:SOURce:POWer:STEP[:INCRement]?

Specifies the source power step size.

Default Unit:	dB

Front Panel Access:	**Source, Amptd Step Auto Man**

Set the Source Sweep Power Range

`:SOURce:POWer:SWEep <rel_ampl>`

`:SOURce:POWer:SWEep?`

Specifies the range of power levels through which the source output will sweep. Use :**`SOURce:POWer:STARt`** to set the power level at the start of the power sweep. See also :**`SOURce:POWer:SPAN`**.

Factory Preset and *RST:	0 dB
Range:	0 dB to 20 dB
Default Unit:	dB
Front Panel Access:	**Source, Power Sweep On Off**

Output Power Tracking

`:SOURce:POWer:TRCKing <integer>`

`:SOURce:POWer:TRCKing?`

Adjusts the tracking of the source output with the spectrum analyzer sweep in the present resolution bandwidth.

Factory Preset and *RST:	This command is persistent. The term persistent means that the command retains the setting previously selected, even through a power cycle.
Range:	Integer, 0 to 4095
Remarks:	This command is not needed with the 1.5 GHz tracking generator.
Front Panel Access:	**Source, Man Track Adj**

Output Power Tracking Peak

`:SOURce:POWer:TRCKing:PEAK`

Automatically adjusts the tracking of the source output with the spectrum analyzer sweep so that the power is maximized for the present resolution bandwidth.

Remarks:.	This command is not applicable for the 1.5 GHz tracking generator.
Front Panel Access:.	**Source, Tracking Peak**

STATus Subsystem

The STATus subsystem controls the SCPI-defined status-reporting structures.

Operation Condition Query

`:STATus:OPERation:CONDition?`

This query returns the decimal value of the sum of the bits in the Status Operation Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Operation Enable

`:STATus:OPERation:ENABle<integer>`

`:STATus:OPERation:ENABle?`

This command determines which bits in the Operation Condition Register will set bits in the Operation Event register, which also sets the Operation Status Summary bit (bit 7) in the Status Byte Register. The variable <integer> is the sum of the decimal values of the bits you want to enable.

NOTE Preset sets all bits in this enable register to 0. To have any Operation Events reported to the Status Byte Register, 1 or more bits must be set to 1.

Factory Preset and *RST:	0
Range:	Integer, 0 to 32767

Operation Event Query

`:STATus:OPERation[:EVENt]?`

This query returns the decimal value of the sum of the bits in the Operation Event register.

NOTE The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the data is cleared.

Operation Negative Transition

`:STATus:OPERation:NTRansition <integer>`

`:STATus:OPERation:NTRansition?`

This command determines which bits in the Operation Condition register will set the corresponding bit in the Operation Event register when that bit has a negative transition (1 to 0). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	0
Range:	Integer, 0 to 32767

Operation Positive Transition

`:STATus:OPERation:PTRansition <integer>`

`:STATus:OPERation:PTRansition?`

This command determines which bits in the Operation Condition register will set the corresponding bit in the Operation Event register when that bit has a positive transition (0 to 1). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	Integer, 0 to 32767

Preset the Status Byte

`:STATus:PRESet`

Sets bits in most of the enable and transition registers to their default state. It presets all the Transition Filters, Enable Registers, and the Error/Event Queue Enable. It has no effect on Event Registers, Error/Event Queue ESE, and SRE Registers as described in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

STATus:QUEStionable Subsection

This subsection controls the SCPI-defined status-reporting structures.

Questionable Calibration Condition

`:STATus:QUEStionable:CALibration:CONDition?`

This query returns the decimal value of the sum of the bits in the Questionable Calibration Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Questionable Calibration Enable

`:STATus:QUEStionable:CALibration:ENABle <integer>`

`:STATus:QUEStionable:CALibration:ENABle?`

This command determines which bits in the Questionable Calibration Condition Register will set bits in the Questionable Calibration Event register, which also sets the Calibration Summary bit (bit 8) in the Questionable Register. The variable <integer> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST:	32767 (all 1’s)
Range:	Integer, 0 to 32767

Questionable Calibration Event Query

`:STATus:QUEStionable:CALibration[:EVENt]?`

This query returns the decimal value of the sum of the bits in the Questionable Calibration Event register.

NOTE The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the data is cleared.

Questionable Calibration Negative Transition

`:STATus:QUEStionable:CALibration:NTRansition <integer>`

`:STATus:QUEStionable:CALibration:NTRansition?`

This command determines which bits in the Questionable Calibration Condition register will set the corresponding bit in the Questionable Calibration Event register when that bit has a negative transition (1 to 0). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	0
Range:	Integer, 0 to 32767

Questionable Calibration Positive Transition

`:STATus:QUEStionable:CALibration:PTRansition <integer>`

`:STATus:QUEStionable:CALibration:PTRansition?`

This command determines which bits in the Questionable Calibration Condition register will set the corresponding bit in the Questionable Calibration Event register when that bit has a positive transition (0 to 1). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	32767 (all 1’s)
Range:	Integer, 0 to 32767

Questionable Condition

`:STATus:QUEStionable:CONDition?`

This query returns the decimal value of the sum of the bits in the Questionable Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Questionable Enable

`:STATus:QUEStionable:ENABle <integer>`

`:STATus:QUEStionable:ENABle?`

This command determines which bits in the Questionable Condition Register will set bits in the Questionable Event register, which also sets the Questionable Status Summary bit (bit3) in the Status Byte Register. The variable <integer> is the sum of the decimal values of the bits you want to enable.

NOTE The preset condition is to have all bits in this enable register set to 0. To have any Questionable Events reported to the Status Byte Register, 1 or more bits need to be set to 1. The Status Byte Event Register should be queried after each measurement to check the Questionable Status Summary (bit 3). If it is equal to 1, a condition during the test made the test results invalid. If it is equal to 0, this indicates that no hardware problem or measurement problem was detected by the analyzer.

Factory Preset and *RST:	0
Range:	Integer, 0 to 32767

Questionable Event Query

`:STATus:QUEStionable[:EVENt]?`

This query returns the decimal value of the sum of the bits in the Questionable Event register.

NOTE The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the data is cleared.

Questionable Frequency Condition

`:STATus:QUEStionable:FREQuency:CONDition?`

This query returns the decimal value of the sum of the bits in the Questionable Frequency Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Questionable Frequency Enable

`:STATus:QUEStionable:FREQuency:ENABle <integer>`

`:STATus:QUEStionable:FREQuency:ENABle?`

This command determines which bits in the Questionable Frequency Condition Register will set bits in the Questionable Frequency Event register, which also sets the Frequency Summary bit (bit 5) in the Questionable Register. The variable <integer> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	Integer, 0 to 32767

Questionable Frequency Event Query

`:STATus:QUEStionable:FREQuency[:EVENt]?`

This query returns the decimal value of the sum of the bits in the Questionable Frequency Event register.

NOTE The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the data is cleared.

Questionable Frequency Negative Transition

`:STATus:QUEStionable:FREQuency:NTRansition <integer>`

`:STATus:QUEStionable:FREQuency:NTRansition?`

This command determines which bits in the Questionable Frequency Condition register will set the corresponding bit in the Questionable Frequency Event register when that bit has a negative transition (1 to 0). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	0
Range:	Integer, 0 to 32767

Questionable Frequency Positive Transition

:STATus:QUEStionable:FREQuency:PTRansition <integer>

:STATus:QUEStionable:FREQuency:PTRansition?

This command determines which bits in the Questionable Frequency Condition register will set the corresponding bit in the Questionable Frequency Event register when that bit has a positive transition (0 to 1). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	Integer, 0 to 32767

Questionable Integrity Condition

:STATus:QUEStionable:INTegrity:CONDition?

This query returns the decimal value of the sum of the bits in the Questionable Integrity Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Questionable Integrity Enable

:STATus:QUEStionable:INTegrity:ENABle <integer>

:STATus:QUEStionable:INTegrity:ENABle?

This command determines which bits in the Questionable Integrity Condition Register will set bits in the Questionable Integrity Event register, which also sets the Integrity Summary bit (bit 9) in the Questionable Register. The variable <integer> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	Integer, 0 to 32767

Questionable Integrity Event Query

:STATus:QUEStionable:INTegrity[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Integrity Event register.

NOTE The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the data is cleared.

Questionable Integrity Negative Transition

```
:STATus:QUEStionable:INTegrity:NTRansition <integer>
```

```
:STATus:QUEStionable:INTegrity:NTRansition?
```

This command determines which bits in the Questionable Integrity Condition register will set the corresponding bit in the Questionable Integrity Event register when that bit has a negative transition (1 to 0). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	0
Range:	Integer, 0 to 32767

Questionable Integrity Positive Transition

```
:STATus:QUEStionable:INTegrity:PTRansition <integer>
```

```
:STATus:QUEStionable:INTegrity:PTRansition?
```

This command determines which bits in the Questionable Integrity Condition register will set the corresponding bit in the Questionable Integrity Event register when that bit has a positive transition (0 to 1). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	Integer, 0 to 32767

Questionable Integrity Uncalibrated Enable

```
:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle
<integer>
```

```
:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle?
```

This command determines which bits in the Questionable Integrity Uncalibrated Condition Register will set bits in the Questionable Integrity Uncalibrated Event register, which also sets the Data Uncalibrated Summary bit (bit 3) in the Questionable Integrity Register. The variable <integer> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	Integer, 0 to 32767

Questionable Integrity Uncalibrated Event Query

`:STATus:QUEStionable:INTegrity:UNCalibrated[:EVENt]?`

This query returns the decimal value of the sum of the bits in the Questionable Integrity Uncalibrated Event register.

NOTE The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the data is cleared.

Questionable Integrity Uncalibrated Negative Transition

`:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition <integer>`

`:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition?`

This command determines which bits in the Questionable Integrity Uncalibrated Condition register will set the corresponding bit in the Questionable Integrity Uncalibrated Event register when that bit has a negative transition (1 to 0). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	0
Range:	integer, 0 to 32767

Questionable Integrity Uncalibrated Positive Transition

`:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition <integer>`

`:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition?`

This command determines which bits in the Questionable Integrity Uncalibrated Condition register will set the corresponding bit in the Questionable Integrity Uncalibrated Event register when that bit has a positive transition (0 to 1). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	32767 (all 1’s)
Range:	integer, 0 to 32767

Questionable Negative Transition

```
:STATus:QUEStionable:NTRansition <integer>
```

```
:STATus:QUEStionable:NTRansition?
```

This command determines which bits in the Questionable Condition register will set the corresponding bit in the Questionable Event register when that bit has a negative transition (1 to 0). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	0
Range:	integer, 0 to 32767

Questionable Power Condition

```
:STATus:QUEStionable:POWer:CONDition?
```

This query returns the decimal value of the sum of the bits in the Questionable Power Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Questionable Power Enable

```
:STATus:QUEStionable:POWer:ENABle <integer>
```

```
:STATus:QUEStionable:POWer:ENABle?>
```

This command determines which bits in the Questionable Power Condition Register will set bits in the Questionable Power Event register, which also sets the Power Summary bit (bit 3) in the Questionable Register. The variable <integer> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST:	32767 (all 1’s)
Range:	integer, 0 to 32767

Questionable Power Event Query

```
:STATus:QUEStionable:POWer[:EVENt]?
```

This query returns the decimal value of the sum of the bits in the Questionable Power Event register.

NOTE The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the data is cleared.

Questionable Power Negative Transition

`:STATus:QUEStionable:POWer:NTRansition <integer>`

`:STATus:QUEStionable:POWer:NTRansition?`

This command determines which bits in the Questionable Power Condition register will set the corresponding bit in the Questionable Power Event register when that bit has a negative transition (1 to 0). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	0
Range:	integer, 0 to 32767

Questionable Power Positive Transition

`:STATus:QUEStionable:POWer:PTRansition <integer>`

`:STATus:QUEStionable:POWer:PTRansition?`

This command determines which bits in the Questionable Power Condition register will set the corresponding bit in the Questionable Power Event register when that bit has a positive transition (0 to 1). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	integer, 0 to 32767

Questionable Positive Transition

`:STATus:QUEStionable:PTRansition <integer>`

`:STATus:QUEStionable:PTRansition?`

This command determines which bits in the Questionable Condition register will set the corresponding bit in the Questionable Event register when that bit has a positive transition (0 to 1). The variable <integer> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	integer, 0 to 32767

SYSTem Subsystem

This subsystem is used to set the controls and parameters associated with the overall system communication. These functions are not related to instrument performance.

GPIB Address

`:SYSTem:COMMunicate:GPIB[1][:SELF]:ADDRess <integer>`

`:SYSTem:COMMunicate:GPIB[1][:SELF]:ADDRess?`

Sets and queries the GPIB address.

NOTE This command applies to analyzers having the standard GPIB I/O interface Option A4H. Only one GPIB I/O interface Option A4H can be installed in an instrument.

Factory Preset and *RST:	It is set to 18 by **`:SYSTem:PRESet:PERSistent`**, which sets the persistent state values to their factory defaults. This command is persistent. The term persistent means that the command retains the setting previously selected, even through a power cycle.
Range:	Integer, 0 to 30
Front Panel Access:	**System, Remote Port**

Serial Port DTR Setup

`:SYSTem:COMMunicate:SERial[1]:CONTrol:DTR OFF|ON|IBFull`

`:SYSTem:COMMunicate:SERial[1]:CONTrol:DTR?`

Sets the hardware pacing scheme. Only one Option 1AX can be installed in an instrument.

Off - holds the DTR line in the unasserted (off) condition

On - holds the DTR line in the asserted (on) condition

IBFull - selects the input buffer full mode for the DTR line. The IBFull parameter sets the DTR line to indicate when the device is ready to receive. When the number of received bytes in the input buffer of the device reaches the stop threshold, the device will unassert the DTR line. When the number of bytes has been reduced to the start threshold, the device will assert DTR indicating that it can receive input again. The device will also monitor the state of CTS and will stop transmission if the line becomes unasserted.

Factory Preset

(no *RST):	The factory default is On. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Serial Port RTS Setup

:SYSTem:COMMunicate:SERial[1]:CONTrol:RTS OFF|ON|IBFull

:SYSTem:COMMunicate:SERial[1]:CONTrol:RTS?

Sets the hardware pacing (hand-shaking) scheme. Many high speed asynchronous modems use this line (paired with CTS) as receive/transmit pacing. Only one Option 1AX can be installed in an instrument.

Off - indicates that the RTS line should always be asserted

On - indicates that the RTS line should always be unasserted

IBFull - selects the input buffer full mode for the RTS line. IBFull sets the RTS line to indicate when the device is ready to receive. When the number of received bytes in the input buffer of the device reaches the stop threshold, the device will unassert the RTS line. When the number of bytes has been reduced to the start threshold, the device will assert RTS indicating that it can receive input again. RTS is sometimes called RFR (ready for receiving). The device will also monitor the state of CTS and will stop transmission if that line becomes unasserted.

Factory Preset (no *RST):	The factory default is IBFull. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Serial Port Baud Rate Setup

:SYSTem:COMMunicate:SERial[1][:RECeive]:BAUD <baud_rate>

:SYSTem:COMMunicate:SERial[1][:RECeive]:BAUD?

Only one Option 1AX can be installed in an instrument.

Factory Preset (no *RST):	The factory default is 9600. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Range:	Supported baud rates are 110|300|600|1200|2400|4800|9600|19200|38400|57600|115200
Front Panel Access:	**System, Remote Port**

Serial Port Receive Pace Setup

`:SYSTem:COMMunicate:SERial[1][:RECeive]:PACE XON|NONE`

`:SYSTem:COMMunicate:SERial[1][:RECeive]:PACE?`

Set the receive pace to on or none for an instrument, with the RS-232 interface installed. Only one Option 1AX can be installed in an instrument. If no optional serial port number is specified, port 1 is assumed.

Factory Preset (no *RST):	The factory default is none. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Serial Port Transmit Pace Setup

`:SYSTem:COMMunicate:SERial[1]:TRANsmit:PACE XON|NONE`

`:SYSTem:COMMunicate:SERial[1]:TRANsmit:PACE?`

Set the transmit pace to on or none for an instrument, with the RS-232 interface installed. Only one Option 1AX can be installed in an instrument. If no optional serial port number is specified, port 1 is assumed.

Factory Preset (no *RST):	The factory default is none. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Hardware Configuration Query

`:SYSTem:CONFigure:HARDware?`

Returns string of information about the current hardware in the instrument.

Front Panel Access:	**System, Show Hardware**

Display the Hardware Configuration

`:SYSTem:CONFigure:HARDware:STATe OFF|ON|0|1`

`:SYSTem:CONFigure:HARDware:STATe?`

Shows the current hardware configuration of the instrument on the display.

Factory Preset and *RST:	Off
Front Panel Access:	**System, Show Hdwr**

System Configuration Query

:SYSTem:CONFigure[:SYSTem]?

Returns string of information about the configurations of the instrument.

Front Panel
Access: **System, Show System**

Display System Configuration

:SYSTem:CONFigure[:SYSTem]:STATe OFF|ON|0|1

:SYSTem:CONFigure[:SYSTem]:STATe?

Shows the current system configuration of the instrument on the display.

Factory Preset
and *RST: Off

Front Panel
Access: **System, Show System**

Set Date

:SYSTem:DATE <year>,<month>,<day>

:SYSTem:DATE?

Sets the date of the real-time clock of the instrument.

Year is a 4-digit integer

Month is an integer 1 to 12

Day is an integer 1 to 31 (depending on the month)

Front Panel
Access: **System, Time/Date, Set Date**

Error Information Query

:SYSTem:ERRor[:NEXT]?

This command queries the earliest entry to the error queue and then deletes that entry. *CLS clears the entire error queue.

Front Panel
Access: **System, Show Errors**

Locate SCPI Command Errors

:SYSTem:ERRor:VERBose OFF|ON|0|1

:SYSTem:ERRor:VERBose?

Adds additional information to the error messages returned by the **:SYSTem:ERRor?** command. It indicates which SCPI command was executing when the error occurred and what about that command was unacceptable.

<error number>,"<error message>;<annotated SCPI command>"

The maximum length of the <annotated SCPI command> is 80 characters. If the error occurs in a SCPI command longer than 80 characters, the <Err> sentinel is placed at the end of the <annotated SCPI command>.

Example: First set **SYST:ERR:VERBose ON**

If the command **SENSe:FREQuently:CENTer 942.6MHz** is sent, then sending **SYST:ERR?** returns:

```
−113,"Undefined header;SENSe:FREQuently:<Err>CENTer 942.6MHz $<NL>"
```

The <Err> shown after FREQuently shows you the spelling error. (The $<NL> is the typical representation for the command terminator.

If the command **SENSe:FREQ:CENTer 942.6Sec** is sent, then sending **SYST:ERR?** returns:

```
−131,"Invalid suffix;SENSe:FREQuency:CENTer 942.6Sec<Err> $<NL>"
```

The <Err> shown after Sec shows you the invalid suffix.

Factory Preset and *RST:	Not affected by *RST
Remarks:	The verbose SCPI error debugging state is global to all the SCPI interfaces.
History:	Added with firmware revision A.08.00.
Front Panel Access:	System, Show Errors, Verbose SCPI ON OFF

Host Identification Query

:SYSTem:HID?

This command returns a string that contains the host identification. This ID is required in order to obtain the license key that enables a new application or option.

Front Panel Access:	**System, Show System**

License Key – Install Application/Option

`:SYSTem:LKEY <"option">, <"license key">`

`:SYSTem:LKEY? <"option">`

This command enters the license key required for installing the specified new application or option. The query returns a string that contains the license key for a specified application or option that is already installed in the instrument. The license key will also be returned if the application is not currently in memory, but had been installed at some previous time.

Example: **`:SYST:LKEY "BAC", "123A456B789C"`**

An option is a three character string that specifies the option or application that is to be installed, as found in the Ordering Guide (for example, BAH for GSM Measurement Personality). The option name must be enclosed in quotes.

A license key is a 12-character hexadecimal string given with the option. The license key is unique to a specific option installed in the instrument with a specific host ID, as returned by **`:SYST:HID?`**. The license key must be enclosed in quotes.

Front Panel
Access: **System, Licensing**

Delete a License Key

`:SYSTem:LKEY:DELete <"option">`

This command allows you to delete the license key from instrument memory for the selected option.

NOTE In general, deleting the license key number is not recommended. If the license key is deleted, you will be unable to reload or update the application in instrument memory without re-entering the license key. The license key works with one particular instrument host ID only.

Query Instrument Options

`:SYSTem:OPTions?`

Returns a list of the options that are installed.

It is a comma separated list such as: “1DS,1D6,A4H,A4J,1DN”

Front Panel
Access: **System, Show System**

Power On Elapsed Time

`:SYSTem:PON:ETIMe?`

Returns the number of seconds that have elapsed since the analyzer was turned on for the very first time.

Front Panel Access:	**System, Show System**

Power On Time

`:SYSTem:PON:TIME?`

Returns the number of milliseconds that have elapsed since the analyzer was last turned on.

Power On Type

`:SYSTem:PON:TYPE PRESet|LAST`

`:SYSTem:PON:TYPE?`

Sets the defined instrument conditions after a power-on or **Preset**.

PRESet - The instrument settings at power-on will be either the factory preset or user preset, as set by **`:SYSTem:PRESet:TYPE FACTory|USER.`**

LAST - The instrument settings at power-on will be the settings at the time of power down.

Factory Preset and *RST:	The factory default is Preset. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**System, Power On/Preset, Power On Last Preset**

Enable IF/Video/Sweep Output Ports

`:SYSTem:PORTs:IFVSweep:ENABle OFF|ON|0|1`

`:SYSTem:PORTs:IFVSweep:ENABle?`

This command enables or disables the IF, video, and sweep output ports for analyzers having options A4J (IF, Sweep, and Video Ports) and AYX (Fast Time Domain Sweeps).

Factory Preset and *RST:	On
Example:	**`:SYST:PORT:IFVS:ENAB ON`**
Range:	On/Off

History:	Added with firmware revision A.04.00.
Remarks:	Disable the output ports for faster measurement times.

Preset

`:SYSTem:PRESet`

Returns the instrument to a set of defined conditions. The particular set is selected by `:SYSTem:PRESet:TYPE`. This command does not change any persistent parameters. The term persistent means that the command retains the setting previously selected, even through a power cycle.

Front Panel Access:	**Preset**

Persistent State Reset

`:SYSTem:PRESet:PERSistent`

Sets the persistent state values to their factory defaults. The term persistent means that the command retains the setting previously selected, even through a power cycle. Examples of persistent functions are: GPIB address, power-on type, and preset type.

Front Panel Access:	**System, Restore Sys Defaults**

Preset Type

`:SYSTem:PRESet:TYPE FACTory|USER|MODE`

Selects the preset state to be either factory-defined or user-defined preset conditions.

Factory Preset and *RST:	The factory default is `MODE`. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
History:	Changed with firmware revision A.08.00. Previous firmware revisions had default type `FACTory`.
Remarks:	`:SYSTem:PRESet:USER:SAVE` defines the *user preset*.
Front Panel Access:	**System, Power On/Preset, Preset Type**

Save User Preset

`:SYSTem:PRESet[:USER]:SAVE`

Saves the current instrument conditions as the *user preset* condition.

Front Panel Access:	**System, Power On/Preset, Save Type Preset**

Speaker Control

`:SYSTem:SPEaker[:STATe] OFF|ON|0|1`

`:SYSTem:SPEaker[:STATe]?`

Turns the internal speaker on or off.

Factory Preset and *RST:	Off
Front Panel Access:	**Det/Demod, Demod, Speaker On Off**

Set Time

`:SYSTem:TIME <hour>,<minute>,<second>`

`:SYSTem:TIME?`

Sets the time of the real-time clock of the instrument.

Hour must be an integer 0 to 23.

Minute must be an integer 0 to 59.

Second must be an integer 0 to 59.

Front Panel Access:	**System, Time/Date, Set Time**

SCPI Version Query

`:SYSTem:VERSion?`

Returns the SCPI version number with which the instrument complies.

TRACe Subsystem

The TRACe subsystem controls access to the internal trace memory of the analyzer.

NOTE Refer also to :`CALCulate` and :`MMEMory` subsystems for more trace and limit line commands.

Copy Trace

`:TRACe:COPY <source_trace>,<dest_trace>`

Transfers the source trace to the destination trace and leaves the destination trace in VIEW mode.

Source traces are: TRACE1|2|3

Destination traces are: TRACE1|2|3

Example: `:TRAC:COPY TRACE2,TRACE1`

Front Panel Access: **View/Trace, Operations, 1 –> 3**

View/Trace, Operations, 2 –> 3

Transfer Trace Data

`:TRACe[:DATA] <trace_name>|RAWTRACE,<definite_length_ block>|<comma_separated_ASCII_data>`

`:TRACe[:DATA]? <trace_name> |RAWTRACE|LLINE1|LLINE2`

This command transfers trace data from the controller to the instrument. The data format is set by the command :`FORMat [:TRACe][:DATA]`. The data is comma-separated ASCII values in ASCII formatting, and a definite length block in REAL, INTeger, and UINTeger formatting.

The query returns the current values of the designated trace. The data is terminated with <NL><END> (for GPIB that is newline, or linefeed, followed by EOI set true; for RS-232 this is newline only.)

`LLINE1` and `LLINE2` can only be queried; they cannot be set.

<trace_name> is `TRACE1|2|3`

NOTE

This command does not allow setting all trace points to the same amplitude value by sending just a single value. If you need to set all trace points to the same value, you must send the same value to each trace point.

Rawtrace data is available with **`UINT,16`** or **`INT,32`** formatting. It is unitless, returns uncorrected ADC values, and is the fastest method of obtaining measurement data.

Example: **`:TRAC:DATA TRACE1,#41604<binary trace data><LF-EOI>`**

Remarks: Commands **`:MMEM:STOR:TRAC`** and **`:MMEM:LOAD:TRAC`** are used to transfer trace data to, or from, the internal hard drive or floppy drive of the instrument.

The number of points in a trace is specified by **`[:SENSe]:SWEep:POINts`**. The trace data format is determined by **`:FORMat[:TRACe][:DATA]`**, and the binary data byte order is determined by **`:FORMat:BORDer`**.

If the parameter to the query is **`LLINE1`** or **`LLINE2`**, a very large positive or negative value is returned at any point outside the range of limit values. A large positive number is returned for an upper limit, and a large negative value for lower limits. There is no SCPI short form for parameters **`LLINE1|LLINE2.`**

Exchange Traces

`:TRACe:EXCHange <trace_1>,<trace_2>`

Exchanges 2 traces, point by point and leaves both in VIEW mode.

Trace_1 choices are: TRACE1|2|3

Trace_2 choices are: TRACE1|2|3

Example: **`:TRAC:EXCH TRACE3,TRACE2`**

Front Panel Access: **View/Trace, Operations, 1 <–> 3**

View/Trace, Operations, 2 <–> 3

Trace Math Add

`:TRACe:MATH:ADD <destination_trace>,<source_trace1>,<source_trace2>`

Adds the magnitudes of the two source traces and places the result in the destination trace.

Destination traces are: TRACE1|2|3

Source traces are: TRACE1|2|3

Example: `:TRAC:MATH:ADD TRACE2,TRACE1,TRACE3` is equivalent to : (trace 2 = trace 1 + trace 3)

Mean Trace Data

`:TRACe:MATH:MEAN? <trace>`

Returns the mean of the amplitudes of the trace amplitude elements in measurement units.

Traces are: TRACE1|2|3

Query the Signal Peaks

`:TRACe:MATH:PEAK[:DATA]?`

Outputs the signal peaks by frequency or by amplitude. This command uses only trace1 data.

The sort mode is determined by the command :TRACe:MATH:PEAK:SORT. The commands :CALCulate:MARKer:PEAK:EXCursion and :CALCulate:MARKer:PEAK:THReshold are used to determine what is a signal peak. To get the number of signals found meeting the specified limits, use the query :TRACe:MATH:PEAK:POINts?

Query Number of Peaks Found

`:TRACe:MATH:PEAK:POINts?`

Outputs the number of signal peaks identified. The amplitude of the peaks can then be queried with :`TRACe:MATH:PEAK:DATA?` This command uses only trace1 data.

Peak Sorting

`:TRACe:MATH:PEAK:SORT AMPLitude|FREQuency`

`:TRACe:MATH:PEAK:SORT?`

Determines if the signals in the :`TRACe:MATH:PEAK:DATA?` query are sorted by frequency or amplitude.

Amplitude sorts the identified peaks by descending amplitude.

Frequency sorts the identified peaks by increasing frequency.

Smooth Trace Data

`:TRACe:MATH:SMOoth <trace>`

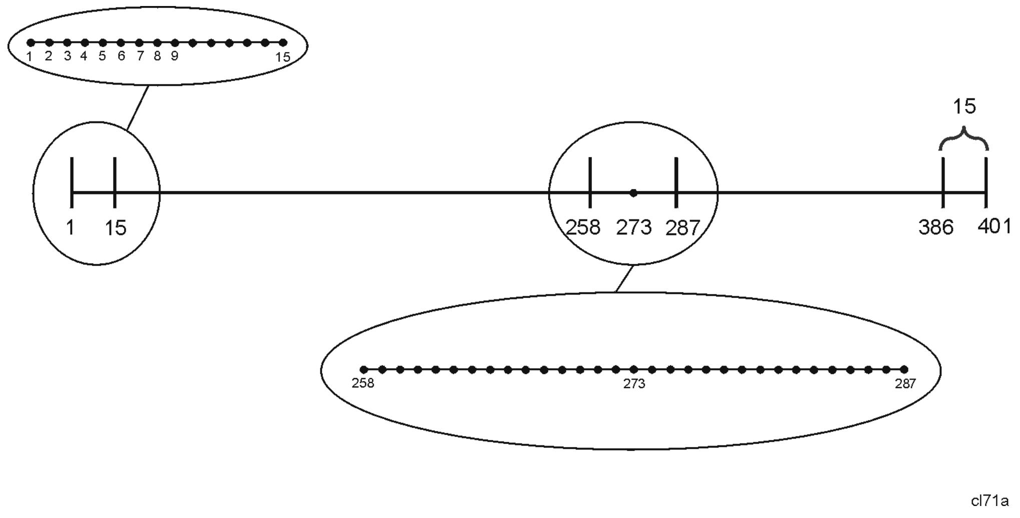
Smooths the trace according to the number of points specified in :`TRACe:MATH:SMOoth:POINts`. There is no equivalent front panel function.

Traces are: TRACE1|2|3, and RAWTRACE commands.

The purpose of this function is to perform a spatial video averaging as compared to the temporal version supplied by the video-average command `[:SENSe]:AVERage:TYPE VIDeo`. The functions of `:TRACe:MATH:SMOoth <trace>` and `[:SENSe]:AVERage:TYPE VIDeo|POWer` are not interchangeable.

Each point value is replaced with the average of the values of the selected number of points, with half of those points located on each side of any particular point (when possible). Refer to Figure 5-5. This figure illustrates a 401 point trace with a smoothing number of 31. Think of the trace points as “buckets” of data. To smooth (arbitrary) point 273, the analyzer averages buckets 258 through 287 and applies that value to point 273.

Figure 5-5 Smoothing With 401 Trace Points and 31 Smoothing Points



Increasing the number of points increases smoothing at the cost of decreasing resolution.

The amount of smoothing decreases at the end points. Because `:TRACe:MATH:SMOoth <trace>` averages values that occur before and after the data point in time, display irregularities can be caused at the start and stop frequencies. To avoid possible irregularities (signal distortion) at the ends of the trace, use small values for the smooth parameter.

Refer to Figure 5-5 for a discussion of this end-point smoothing phenomena. With 31 smoothing points and a 401 point trace, point 16 will be the first point to have full 31-bucket smoothing. Likewise, point 385 will be the last point with full 31-bucket smoothing. Under the conditions stated, points 2 through 15 will be smoothed as follows: Point 2 is derived from averaging buckets 1 through 3. Point 3 is derived from averaging buckets 1 through 5, Point 4 is derived from averaging buckets 1 through 7, and so forth until point 16 is reached. The quantity of buckets used for the smoothing running average increases at the rate of 2 buckets per point, from point 1 to point ([smoothing number/2]

+ 1), at which time the full number of smoothing points is utilized. The same characteristic occurs at the completion of the trace, beginning at point 386, when the number of averaging buckets begins to decrease until point 401 is reached.

By replacing the value of each point in a trace with the average of the values of a number of points centered about that point, any rapid variations in noise or signals are smoothed into more gradual variations. It thereby performs a function similar to reducing the video bandwidth without the corresponding changes in sweep time; as such, frequency resolution is decreased. Also, signal peaks are reduced with large smoothing values; and this can cause the amplitude to appear to be less than its actual value.

Number of Points for Smoothing

`:TRACe:MATH:SMOoth:POINts <integer>`

`:TRACe:MATH:SMOoth:POINts?`

Specifies the number of points that will be smoothed in **`:TRACe:MATH:SMOoth`**. See that command for an explanation of how smoothing is performed.

Increasing the number of points increases smoothing at the cost of decreasing resolution. If the number of points is an even number, then the number of points is increased by one. If the number of points is larger than the number of sweep points, then the number of sweep points is used, unless the number of sweep points is even, in which case the number of points will be the sweep points minus one. The number of points smoothed is always an odd number.

Range:	Integer, 3 to current number of sweep points

Trace Math Subtract

`:TRACe:MATH:SUBTract <destination_trace>,<source_trace1>,<source_trace2>`

Subtracts the magnitude of the two source traces (trace 1 – trace 2) and places the result in the destination trace.

Destination traces are: TRACE1|2|3

Source traces are: TRACE1|2|3

Example:	**`:TRAC:MATH:SUBT TRACE3,TRACE3,TRACE2`** is equivalent to: (trace 3 = trace 3 – trace 2)

Trace Math Subtract From Display Line

`:TRACe:MATH:SUBTract:DLINe <trace>`

Subtracts the magnitude of the display line from the selected trace and places the result back in the selected trace.

Trace is: TRACE1|2|3

Example: `:TRAC:MATH:SUBT:DLIN TRACE1`

is equivalent to: (trace1 = trace 1 – display line)

Front Panel Access: **View/Trace, Operations, 2 – DL –> 2**

Select Trace Display Mode

`:TRACe1|2|3:MODE WRITe|MAXHold|MINHold|VIEW|BLANk`

`:TRACe1|2|3:MODE?`

Selects the display mode for the selected trace.

Write puts the trace in the normal mode, updating the data.

Maximum hold displays the highest measured trace value for all the data that has been measured since the function was turned on.

Minimum hold displays the lowest measured trace value for all the data that has been measured since the function was turned on.

View turns on the trace data so that it can be viewed on the display.

Blank turns off the trace data so that it is not viewed on the display.

Remarks: Whenever the number of sweep points change, the following functions are affected:

- All trace data is erased
- Any traces in view mode will go to blank mode

Front Panel Access: **View/Trace, Clear Write**

View/Trace, Max Hold

View/Trace, Min Hold

View/Trace, View

View/Trace, Blank

View/Trace, Normalize, Ref Trace View Blank

TRIGger Subsystem

The TRIGger subsystem is used to set the controls and parameters associated with triggering the data acquisitions. Other trigger-related commands are found in the INITiate and ABORt subsystems.

External Trigger, Line Trigger Delay Value

`:TRIGger[:SEQuence]:DELay <delay>`

`:TRIGger[:SEQuence]:DELay?`

This command sets the amount of trigger delay when using the rear panel external trigger input or the line trigger.

Factory Preset and *RST:	1 µs
Range:	0.3 µs to 429 seconds
Default Unit:	seconds

External Trigger, Line Trigger Delay Enable

`:TRIGger[:SEQuence]:DELay:STATe OFF|ON|0|1`

`:TRIGger[:SEQuence]:DELay:STATe?`

This command allows you to turn on or off a delay, during which the analyzer will wait to begin a sweep after receiving an external trigger signal or a line trigger.

Factory Preset and *RST:	Off
Default Unit:	seconds
Remarks:	Free-run activates the trigger condition that allows the next sweep to start as soon as possible after the last sweep. This function is not available when **Gate** is on.
Front Panel Access:	**Trig, Trig Delay On Off**

External Trigger Slope

`:TRIGger[:SEQuence]:EXTernal[1]:SLOPe POSitive|NEGative`

`:TRIGger[:SEQuence]:EXTernal[1]:SLOPe?`

This command activates the trigger condition that allows the next sweep to start when the external voltage (connected to **GATE TRIG/EXT TRIG IN** on the rear panel) passes through approximately 1.5 volts. The external trigger signal must be a 0 V to +5 V TTL signal. This function only controls the trigger polarity (for positive or negative-going signals).

Factory Preset and *RST:	Positive
Front Panel Access:	**Trig, External Pos Neg**

Trigger Offset

`:TRIGger[:SEQuence]:OFFSet <64 bit floating point value>`

`:TRIGger[:SEQuence]:OFFSet?`

This command sets the trigger offset.

Factory Preset and *RST:	0 seconds
Example:	**`:TRIG:SEQ:OFFS 1.0s`**
Range:	Hardware specific; dependent upon the ADC being used, current state, and the number of sweep points.
Default Unit:	seconds

Remarks: Trigger offset refers to the specified time interval before or after the trigger event from which data is to be written to the trace, and then displayed. Ordinarily, the trigger offset value is zero, and trace data is displayed beginning at the trigger event. A negative trigger offset value results in the display of trace data prior to the trigger event. A positive trigger offset value results in an effective delay in the display of trace data after the trigger event.

The trigger offset value used when the feature is enable will depend on the following parameters:

- Normal trigger offset value originally entered
- Specific instrument hardware in use
- Sweep time
- Number of sweep points

The effective trigger offset value will be re-calculated whenever any of these parameters change.

Front Panel Access: **Trig, Trig Offset**

Trigger Source

`:TRIGger[:SEQuence]:SOURce IMMediate|VIDeo|LINE|EXTernal`

`:TRIGger[:SEQuence]:SOURce?`

Specifies the source (or type) of triggering used to start a measurement.

Immediate is free-run triggering

Video triggers on the video signal level

Line triggers on the power line signal

External allows you to connect an external trigger source

Factory Preset and *RST:	Immediate (free-run triggering)
Remarks:	Free-run activates the trigger condition that allows the next sweep to start as soon as possible after the last sweep.
Front Panel Access:	**Trig, Free Run** **Trig, Video** **Trig, Line** **Trig, External Pos Neg**

NOTE Trigger Delay is not available in Free Run, so turning Free Run on turns off Trigger Delay, but preserves the value of Trigger Delay.

Video Trigger Level Amplitude

`:TRIGger[:SEQuence]:VIDeo:LEVel <ampl>`

`:TRIGger[:SEQuence]:VIDeo:LEVel?`

Specifies the level at which a video trigger will occur.

Factory Preset and *RST:.	2.5 divisions below reference level
Range:.	10 display divisions below reference level to reference level
Default Unit:.	current amplitude units
Remarks:.	Video is adjusted using this command, but must also be selected using the command **`:TRIGger[:SEQuence]:SOURce VIDeo`**. When in FM Demod and Demod View is on, the Video Trigger level is adjusted/queried using the command **`:TRIGger[:SEQuence]:VIDeo:LEVel:FREQuency <freq>`**.

. NOTE Trigger Delay is not available in Video trigger mode, so turning Video on turns off Trigger Delay, but preserves the value of Trigger Delay.

Front Panel Access:.	**Trig, Video**

Video Trigger Level Frequency

`:TRIGger[:SEQuence]:VIDeo:LEVel:FREQuency <freq>`

`:TRIGger[:SEQuence]:VIDeo:LEVel:FREQuency?`

This command is used to adjust the Video Trigger level when in FM Demod, and Demod View is on.

Default Unit:	Hz
Remarks:	Video is adjusted using this command, but must also be selected using the command **`:TRIGger[:SEQuence]:SOURce VIDeo`**. When not in FM Demod, the Video Trigger level is adjusted/queried using the command **`:TRIGger[:SEQuence]:VIDeo:LEVel <ampl>`**.

NOTE Trigger Delay is not available in Video trigger mode, so turning Video on turns off Trigger Delay, but preserves the value of Trigger Delay.

UNIT Subsystem

Select Power Units of Measure

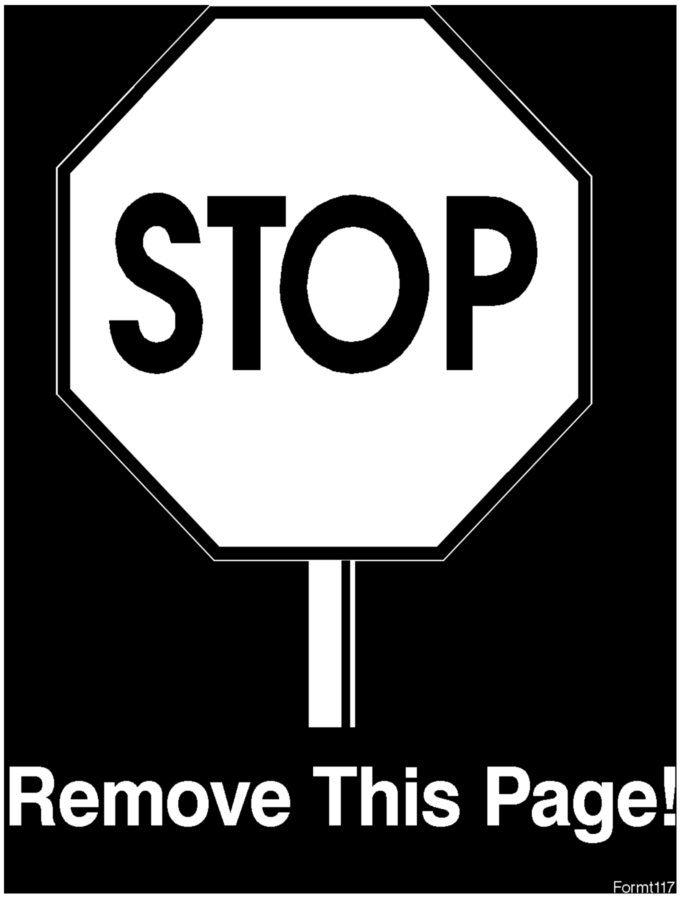
`:UNIT:POWer DBM|DBMV|DBUV|DBUA|V|W|A`

`:UNIT:POWer?`

Specifies amplitude units for the input, output and display.

Factory Preset and *RST:	dBm in log amplitude scale
	volts in linear amplitude scale
History:	Ampere and decibel microampere units are available only with instruments having firmware revision A.06.00 and later.
Front Panel Access:	**AMPLITUDE/Y Scale, Amptd Units**
	AMPLITUDE/Y Scale, Amptd Units, dBm
	AMPLITUDE/Y Scale, Amptd Units, dBmV
	AMPLITUDE/Y Scale, Amptd Units, dBμV
	AMPLITUDE/Y Scale, Amptd Units, Volts
	AMPLITUDE/Y Scale, Amptd Units, Watts
	AMPLITUDE/Y Scale, Amptd Units, Amps
	AMPLITUDE/Y Scale, Amptd Units, dBμA

6 Agilent 8590/EMC Analyzers Programming Conversion Guide



NOTE Please remove this page and insert the wire-O bound Programming Conversion Guide here, Agilent Part Number E7401-90035.

Index

Symbols
*CLS command, 59
*ESE command, 70
*SRE command, 67
*STB? command, 67
*TST? query, 197

Numerics
3 dB bandwidth command, 199

A
abort command, 198, 250
ac input coupling command, 252
Agilent Technologies URL, 4
alignment
  all assemblies, 197, 229
  commands, 229
  FM demodulation assembly, 230
AM demodulation, 280
  on/off, 281
amplitude
  correction
    data, merging, 277
    data, setting, 276
    deleting, 276
    external amplifier, 279
    set, deleting, 278
    set, turning on/off, 278
    setting interpolation, 278
    turning on/off, 276
  maximizing input signal, 297
  scaling, 240
annotation display, 237
ASCII data format, 243
assemblies, aligning all, 229
attenuation
  input, resetting protection, 253
  setting, 295
  tracking generator, 303
  tracking generator, auto on/off, 304
automeasure
  average on/off, 287
  margin on/off, 289
  peak on/off, 287
  quasi-peak on/off, 287
average detection, 285
averaging
  dwell time, setting, 288
  number of averages, 269
  on/off
    automatic, 270
    automeasure, 287
    averaging, 269
  restart, 269
  smoothing trace data, 330
  type, setting, 271

B
background alignment, 230
band markers, 219
  set start frequency, 225
  set stop frequency, 226
  start frequency, 227
  stop frequency, 227
bandpower marker, 217
bandwidth
  measurement command, 200
  measurement command, NdB results, 199
  NdB points, 199
  resolution BW
    couple to video BW, 273
    coupling, 272
    setting, 272
    type, 275
  resolution BW automatic on/off, 272
  video BW
    auto on/off, 273
    ratio to resolution BW, 273
    ratio, auto/manual, 273
    values, 273
baud rate
  RS-232 bus, 52
  serial bus receive, 320
binary data order, 243
blanking the trace, 333
brightness, display angle, 235
bus configuration, 319
byte order of data, 243

C
C language
  addressing sessions, 102
  closing sessions, 103
  compiling and linking, 97
  creating, 96
  example, 99
  opening session, 100
  sessions, 100
  using VISA library, 96
  using VISA transition library, 97, 99
CALCulate subsystem commands, 199
calibration
  align all assemblies, 229
  align now, 193
  automatic, 230
  automatic mode, 229
  command introduction, 229

Index

corrections on/off, 231
defaults, 230
frequency reference adjustment, 231
frequency reference query, 230
IEEE command, 193
reference on/off, 232
RF, 231
tracking generator, 232
catalog memory/disk, 260
center frequency
setting, 290
step size, 290, 291
clear status, IEEE command, 193
CLS command, description, 61
color printing, 246
command parameters, 42
commands
See command list following Contents
See Also specific command
comment set
all signals, 205
current signal, 205
marked signals, 205
compiling C with VTL, 97
condition register, definition, 59
continuous vs. single measurement mode, 249, 259
control measurement commands, 249
controller (GPIB), 50
copy trace command, 328
correction
amplitude
deleting, 276
deleting correction set, 278
merging data, 277
on/off, 276
setting correction set, 278
setting data, 276
turning set on/off, 278
constant default, 230
external amplifier, 279
frequency, 200
impedance mismatch, 279
internal tracking generator, 303
set (amplitude), deleting, 278
set (amplitude), setting, 278
corrections table, loading from file, 261
corrections table, storing, 264
coupling, ac/dc, 252
coupling, none (manual)/all (automatic), 233
cursor to signal list, 207
custom printer, 245

D

data
append to list, 201
exchange trace, 329
format, numeric, 243
limit line, merging, 213
mean of trace, 330
moving to file, 261
placing in output buffer, 255
testing against limit line, 214
trace, normalize, 228
transferring trace, 328
date
display format, 235
display on/off, 235
setting, 322
dc input coupling command, 252
defaults, preset, 326
defaults, setting, 254
delay time
demodulation, 281
gate trigger to open, 300
delta markers, 219
demodulation, 280
displaying signal, 281
on/off, 281
squelch, 280
time, 281
type, AM/FM, 280
detection
type, auto on/off, 282
detection type
EMI modes, 285
selecting, 284
detector ranging, 286
detector, restore previous settings, 286
diagnostic commands, 229
directories
creating, 263
deleting, 264
of memory/disk, 260
display
angle, 235
annotation on/off, 237
auto ranging on/off, 238
commands, 235
date format, 235
date on/off, 235
display line amplitude, 238
display line on/off, 238
full screen on/off, 236
graticule on/off, 237
hardware configuration, 321
intensity (angle), 235

Index

number printed per page, 247
on/off, 236
reference level, 240
reference level offset, 241
reference level, normalized, 239
reference level, position, 239
scaling, 240
scaling, vertical axis, 242
signal list parameters, 208
system configuration, 322
time format, 235
time on/off, 235
title, 236
trace, 237
units, 338
display line, subtraction from trace, 333
DTR for serial bus, 319
dwell time
average, 288
range, 289
setting, 288

E

EMI detection modes, 285
EMI view, 286
errors
data transfer, R-S232, 53
locate SCPI, 323
monitoring, 196
querying, 322
ESE command, description, 61
event enable register, definition, 59
event register, definition, 59
event status
enable, 193
enable and read byte, 193
register, query and clear, 194
external trigger, 336
delay on/off, 334
delay value, 334
slope, 334

F

factory defaults, 230
factory preset, 326
fetch
and add features, 202
commands, 255
commands, field descriptions, 202
files
copy, 260
correction table, loading, 261
correction table, storing, 264
deleting, 261
instrument state, loading, 262
instrument state, storing, 265
limit line, storing, 265
limit lines, loading, 261
listing, 260
measurement results, storing, 265
moving data, 261
screen image, storing, 265
signal list, loading, 263
signal list, storing, 263
trace, loading from file, 262
trace, storing, 266
FM
demodulation type, 280
deviation, 280
FM demodulation on/off, 281
form feed printer, 247
format, data, 243
frequency
calculate correction at, 200
center, 290
configure, 257
domain type, 298
optimization, phase noise, 293
phase noise auto/manual, 294
reference
adjustment, coarse, 231
adjustment, fine, 231
calibration on/off, 231
scaling, 240
span
full, 292
marker type, 219
previous, 292
value, 291
start, 292
step size, 290, 291
stop, 293

G

gate trigger
delay, trigger to open, 300
edge/level, 302
length (EDGE triggering), 300
level, 296
level (level triggering), 301
on/off, 301
polarity (EDGE triggering), 301
preset (EDGE triggering), 301
GPIB
address, 319
command statements, 50
instrument types, 50
overview, 50
graticule display, 237

Index

H

handshake, RS-232 bus, 52
hardcopy output, 245
hardware configuration query, 321
hardware configuration, display, 321
host identification query, 323

I

identification of host, 323
identification query, 194
IEEE commands
  *ESR?, 194
  *IDN?, 194
  *SAV, 196
  common commands, 193
IEEE common commands, 48
IF sweep output on/off, 325
impedance mismatch correction, 279
initiate measurement, 197
input
  attenuation, 295
  configuration, 319
  coupling command, 252
  power, maximizing, 297
  signal, measuring, 268
input attenuation, 295
instrument
  GPIB type, 50
  preset, 326
  reset, 196
  state, loading from file, 262
  state, storing, 265
  status, monitoring, 196
intensity, display angle, 235
internal source. *See* tracking generator
IP, instrument preset, 326

L

landscape printing, 247
Language Reference, 189
license key, deleting, 324
license key, entering, 324
limit lines
  adding values, 213
  amplitude interpolation, 209
  correction sets, delete all, 209
  define values, 211
  deleting, 213
  displaying, 213
  fixed/relative, 209
  frequency interpolation, 211
  from memory, 261
  margin size, setting, 214
  margin, displaying, 214
  maximum test, 215
  on/off, 215
  sorting, 207
  storing, 265
  testing, 200
  testing data, 214
  type upper/lower, 215
  x-axis units, 210
line trigger, 336
linear scaling, 242
linking C C with VTL, 97
listener (GPIB), 50
logarithmic scaling, 242
LRN, IEEE command, 194

M

marker list, add to, 201
markers
  add to signal list, 201
  all off, 216
  bandpower/noise/off, 217
  band-type
    set start frequency, 225
    set stop frequency, 226
    start frequency, 227
    stop frequency, 227
  center frequency, 221
  commands, 216
  configure to measure at, 258
  continuous peaking on/off, 216
  delta, 221
  frequency
    span, 222
    start, 222
    step size, 222
    stop, 222
  frequency counter
    automatic resolution on/off, 217
    on/off, 217
    query, 217
    resolution, 216
  marker table on/off, 223
  measure and add to list, 258
  NdB bandwidth command, 199
  on/off, 222
  peak search
    define, 220
    left, 218
    minimum, 219
    next, 218
    perform, 218
    right, 219
  peak, definition of, 220
  peak, threshold, 221

Index

peak-to-peak, 221
reference level, 221
span pair, 224, 225
span-type, set values, 226
trace assignment, 223
trace assignment, auto, 223
tracking signal, 223
type, 219
x-axis location, 224
x-axis readout, 226
y-axis query, 227

marks, clear all, 204

max hold trace, 333

max/min view
on/off, 258
select max, 258

maximum power to mixer, 296

mean of trace data, 330

measurement
abort, 250
at marker and add to list, 258
bandwidth command, 200
commands, 254, 255
control of, 249
frequency, 257
increasing speed, 45
input signal, 268
marker configuration, 258
pause, 250
peaks, 259
restart, 251
results, storing, 265
resume, 251
return results
of last, 257
single/continuous, 249, 259
stopping, 254
trigger, 250

min hold trace, 333

mixer
maximum input power, 296

monitoring errors, 196

monitoring status, 196

N

NdB bandwidth command, 199

NdB command, 199

NdBstate command, 200

negative peak detection, 284

negative transition register, definition, 59

noise marker, 217

normal marker, 219

normalize on/off, 228

normalized referencel level position, 239

O

offsets
trigger, 335

OPC command, description, 61

operation complete command, IEEE command, 195

operation event query, 308

options query, 195, 324

options, IEEE command, 195

outputs
configuration, 319
on/off
IF sweep, 325
sweep, 325
tracking generator, 267
video, 325

overall status register system, diagram, 60

P

PACE for serial bus receive, 321

PACE for serial bus transmit, 321

page orientation (printing), 247

parameters in commands, 42

pass/fail limit line test, 200

peak detector dwell time, 288

peak search
define, 220
left, 218
minimum, 219
next, 218
peak threshold, 221
perform, 218
right, 219

peaks
automeasure on/off, 287
list of, 330
measuring, 259
number of points, 330
sorting, 330

persistent state values, preset to default, 326

phase noise auto/manual, 294

phase noise optimization, 293

polling method, accessing status register, 62

portrait printing, 247

positive peak detection, 284

positive transition register, definition, 59

power
tracking, 307
tracking generator, 304
tracking generator, fixed/swept, 305
tracking, peak, 307

Index

units, 338
power sweep
  tracking generator
    on/off, 305
    range, 305, 307
    start amplitude, 306
    step auto on/off, 306
    step size, 306
power-on
  conditions, preset/last, 325
  time (milliseconds), 325
  time (seconds), 325
preamplifier on/off, 296
preselector centering on/off, 288
preselector peaking, 297
preset, 196, 326
  customized, 327
  to factory defaults, 326
  type at power-on, 325
  type, factory/user, 325, 326
printer type, 245
printing
  abort, 245
  color, 246
  commands, 245
  equipment, 54
  form feed, 247
  interconnections, 54
  page orientation, 247
  page size, 248
  print now, 247
  printer type, 245
  prints per page, 247
  report, 245
  report content, 245
  screen dump/report, 248
  setup, 54
  signal list, 246
  signal list, delta, 246
  testing setup, 56
programming commands
  creating valid, 39
  multiple commands on one line, 48
  notation syntax, 40
  parameters, 42
  separator syntax, 48
  special characters, 41
  terminations, 48
programming errors, debug information, 323
programming examples
  list of, 94
  system requirements, 95
  using C language, 96, 99

Q
quasi-peak
  automeasure on/off, 287
  detection, 284, 285
  dwell time, 288
  fetch command, 203
queries
  frequency reference, 230
  hardware configuration, 321
  host identification, 323
  identification, 194
  marker frequency counter, 217
  marker y-axis, 227
  operation event, 308
  peaks
    frequency/amplitude, 330
    number of, 330
    sort, frequency/amplitude, 330
  SCPI version, 327
  standard event status register, 194
  state, 194
  system configuration, instrument, 322
  system options, 324
questionable condition registers, 73
questionable condition registers/queries, 310
quit command, 198

R
read command (string format), 257
read commands, description, 256
real number data format, 243
recall, IEEE command, 195
reference
  calibration control, 232
  level
    normalized, 239
    offset, 241
    position, normalized, 239
    y-axis amplitude, 240
registers
  event enable, definition, 59
  event, definition, 59
  negative transition, definition, 59
  positive transition, definition, 59
  questionable condition, 310
  service request enable, description, 67
  status byte, description, 66
remeasure signal list, 259
report content, 245
reset, IEEE command, 196
resolution bandwidth
  automatic on/off, 272
  coupling, 272
  ratio auto/manual, 273

Index

ratio to video BW, 273
type on/off, 275
value, 272

RF
alignment on/off, 229
calibration, 231
overload reset, 253

RS-232 bus
baud rate, 52
character format parameters, 52
data transfer errors, 53
handshake, 52
modem handshaking, 53
overview, 52
serial interface settings, 52

RTS for serial bus, 320

S

sample detection, 284

save command, 196

scale type, linear/logarithmic, 298

scaling, 242

SCPI
command subsystems, locating, 191
errors during execution, 323
errors, locating, 323
version query, 327

screen image, storing, 265

self-test query, 197

separator syntax, 48

serial bus
DTR, 319
receive baud, 320
receive PACE, 321
RTS, 320
transmit PACE, 321

serial bus overview, 52

serial interface settings, 52

service request
accessing status register, method, 62
enable register, 67
generating, 63
IEEE command, 196
using, 63

service request enable register
bit descriptions, 68
description, 67

signal list
add marker, 258
add marker frequency/amplitude, 201
add peaks, 259
append data, 201
cursor to, 207
display parameters, 208
loading from file, 263
on/off, 206
remeasure all signals, 259
sort key, 207
storing, 263

signal peaks
number of, 330
query, 330
sorting, 330

signals
clear marks, 204
comment set
for all, 205
for current, 205
for marked, 205
cursor to signal list, 207
delete all, 205
demodulated, displaying, 281
marking, 206
retrieve current/specified, 206
retrieve from signal list, 206
signal list parameters, 208
sort key, 207
tracking, 223

single vs. continuous measurement mode, 249, 259

sort key, specifying, 207

source
calibration, internal, 232
internal tracking generator commands, 303
See also tracking generator

span
full scale, 292
marker type, 219
markers values, 226
to previous, 292
value, 291

speaker, on/off, 327

special characters in commands, 41

squelch, 280

SRE command, description, 61

SRQ command, 196

SRQ, using, 63

standard event
status byte, enable and read, 193
status event enable register, description, 70
status register query, 194
status register, description, 68

start frequency, 292

start measurement, 197

states, setting
amplitude correction, 276
amplitude correction set, 278
automatic averaging, 270
automatic detection type, 282

Index

averaging on/off, 269
loading from file, 262
max/min view, 258
query, 194
recall, 195
resolution BW automatic, 272
resolution BW type, 275
save, 196
tracking generator output, 267

status byte
clearing, 193
IEEE command, 196
register, 66

status command, using, 64

status registers
commands, 61
descriptions, 57, 59
operation condition register, description, 73
operation condition register, description of bits, 72
operation register, description, 70
operation register, diagram, 71
querying, 65
questionable condition, 73
questionable power register, bit descriptions, 77
questionable power register, diagram, 76
questionable register, diagram, 75
questionable status power register, diagram, 76
questionable status power, bit descriptions, 77
setting, 65
standard event, 68
using, 58

STB command, description, 61

stop
command, 198
frequency, 293

subdirectories, creating, 263

subdirectories, deleting, 264

sweep output on/off, 325

sweep points, setting number, 298

sweep time
auto on/off, 299
stimulus response/spectrum analyzer, 300
time, 299

synchronization, 195, 197

syntax of programming commands, 40

system configuration
commands, 319
display, 322
hardware query, 321
instrument query, 322
options query, 324

T

talker (GPIB), 50

terminations in commands, 48

test limits
maximum, 215
pass/fail, 200

test, IEEE command, 197

throughput, improving, 45

time
display format, 235
display on/off, 235
setting, 327
since power-on (milliseconds), 325
since power-on (seconds), 325

timing control, 195, 197

title display, 236

trace
adding magnitudes, 329
averaging
type, 271
averaging on/off, 270
blank, 333
commands, 328
copying, 328
data, normalize, 228
display, 237
exchange data, 329
format, 243
max hold, 333
mean of amplitudes, 330
min hold, 333
mode, 333
smoothing, number of points, 332
smoothing, select, 330
storing, 266
subtraction, 332
subtraction of display line, 333
transferring data, 328
view, 333
writing, 333

trace, loading from file, 262

tracking generator
attenuation, 303
attenuation, auto on/off, 304
calibration, 232
commands, 303
correction offset, 303
fixed/swept, 305
output on/off, 267
power
level, 304
tracking, 307
tracking, peak, 307
power sweep
on/off, 305
range, 305, 307
start amplitude, 306
step auto on/off, 306

Index

step size, 306

trigger
- commands, 334
- external
  - delay on/off, 334
  - delay value, 334
  - slope, 334
- IEEE command, 197
- level, gate, 296
- measurement, 250
- offset, 335
- source, video, 336
- video, level amplitude, 336
- video, level frequency, 337

U

units, setting power, 338

URL (Agilent Technologies), 4

user preset, 326

user preset, saving, 327

V

video bandwidth
- auto on/off, 273
- couple to resolution bandwidth, 273
- ratio, auto/manual, 273
- ration to resolution bandwidth, 273
- values, setting, 273

video output on/off, 325

video trigger
- level amplitude, 336
- level frequency, 337
- source, 336

view commands, 235

VISA library, 97, 99

W

wait, IEEE command, 197

Y

y-axis
- amplitude scaling, 240
- frequency scaling, 240
- units, 338