

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Keysight / Agilent E1419A* at our website: ***Click HERE***

HP E1419A Table of Contents

1. Getting Started

About this Chapter . . . 13
Configuring the HP E1419A . . . 13
Setting the Logical Address Switch . . . 14
Installing Signal Conditioning Plug-ons . . . 15
Disabling the Input Protect Feature (optional) . . . 18
Disabling Flash Memory Access (optional) . . . 19
Instrument Drivers . . . 21
About Example Programs . . . 21
Verifying a Successful Configuration . . . 21

2. Field Wiring

About This Chapter . . . 23
Planning Your Wiring Layout . . . 23
SCP Positions and Channel Numbers . . . 23
SCP Types and Signal Paths . . . 24
Pairing Sense and Source SCPs for Resistance Measurements . . . 25
Planning for Thermocouple Measurements . . . 26
Faceplate Connector Pin-Signal Lists . . . 27
Optional Terminal Modules . . . 28
The SCPs and Terminal Module Connections . . . 28
Option 11 Terminal Module Layout . . . 29
Option 12 Terminal Module Layout . . . 30
Reference Temperature Sensing with the HP E1419A . . . 31
Configuring the On-board/Remote Reference Jumpers . . . 32
Preferred Measurement Connections . . . 34
Wiring and Attaching the Terminal Module . . . 37
Attaching/Removing the HP E1419 Terminal Module . . . 39
Adding Components to the Option 12 Terminal Module . . . 41
Option 11 Terminal Module Wiring Map . . . 42
Option 12 Terminal Module Wiring Map . . . 43
The Option A3F . . . 44

3. Programming the HP E1419A Multifunction*Plus*

About This Chapter . . . 47
Overview of the HP E1419 Multifunction*Plus* . . . 48
Multifunction*Plus*? . . . 48
Operating Model . . . 52
Executing The Programming Model . . . 53
Power-on and *RST Default Settings . . . 53
Setting up Analog Input and Output Channels . . . 57
Configuring Programmable Analog SCP Parameters . . . 57
Setting Filter Cutoff Frequency . . . 58

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Linking Channels to EU Conversion . . . 59
Linking Output Channels to Functions . . . 67
Setting up Digital Input and Output Channels . . . 67
Setting up Digital Inputs . . . 67
Setting up Digital Outputs . . . 68
Performing Channel Calibration (Important!) . . . 72
Defining C Language Algorithms . . . 74
Global variable definition . . . 74
Algorithm definition . . . 75
Pre-setting Algorithm Variables . . . 75
Defining Data Storage . . . 76
Specifying the Data Format . . . 76
Selecting the FIFO Mode . . . 77
Setting up the Trigger System . . . 78
Arm and Trigger Sources . . . 78
Programming the Trigger Timer . . . 80
Setting the Trigger Counter . . . 80
Outputting Trigger Signals . . . 81
INITiating/Running Algorithms . . . 81
Starting Algorithms . . . 81
The Operating Sequence . . . 82
Retrieving Algorithm Data . . . 83
Modifying Running Algorithm Variables . . . 86
Updating the Algorithm Variables and Coefficients . . . 86
Enabling and Disabling Algorithms . . . 87
Setting Algorithm Execution Frequency . . . 88
Example Command Sequence . . . 88
Using the Status System . . . 90
Enabling Events to be Reported in the Status Byte . . . 93
Reading the Status Byte . . . 94
Clearing the Enable Registers . . . 95
The Status Byte Group's Enable Register . . . 95
Reading Status Groups Directly . . . 95
HP E1419A Background Operation . . . 96
Updating the Status System and VXIbus Interrupts . . . 97
Creating and Loading Custom EU Conversion Tables . . . 98
Compensating for System Offsets . . . 100
Special Considerations . . . 102
Detecting Open Transducers . . . 103
More On Auto Ranging . . . 104
Settling Characteristics . . . 104
Background . . . 104
Checking for Problems . . . 105
Fixing the Problem . . . 105

4. The Algorithm Language and Environment

About This Chapter . . . 109
Overview of the Algorithm Language . . . 110
Example Language Usage . . . 111
The Algorithm Execution Environment . . . 112
The Main Function . . . 112
How Your Algorithms Fit In . . . 112

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Accessing the E1419A's Resources . . . 113
Accessing I/O Channels . . . 114
Defining and Accessing Global Variables . . . 115
Determining First Execution (First_loop) . . . 115
Initializing Variables . . . 116
Sending Data to the CVT and FIFO . . . 116
Setting a VXIbus Interrupt . . . 117
Calling User Defined Functions . . . 118
Operating Sequence . . . 118
Overall Sequence . . . 118
Algorithm Execution Order . . . 120
Defining Algorithms (ALG:DEF) . . . 120
ALG:DEFINE in the Programming Sequence . . . 121
ALG:DEFINE's Two Data Formats . . . 121
Changing an Algorithm While it's Running . . . 122
A Very Simple First Algorithm . . . 124
Writing the Algorithm . . . 124
Running the Algorithm . . . 124
Non-Control Algorithms . . . 125
Data Acquisition Algorithm . . . 125
Process Monitoring Algorithm . . . 125
Algorithm Language Reference . . . 126
Standard Reserved Keywords . . . 126
Special HP E1419A Reserved Keywords . . . 126
Identifiers . . . 126
Special Identifiers for Channels . . . 127
Operators . . . 127
Intrinsic Functions and Statements . . . 128
Program Flow Control . . . 128
Data Types . . . 129
Data Structures . . . 130
Using Type Float in Integer Situations . . . 131
Language Syntax Summary . . . 133
Program Structure and Syntax . . . 138
Declaring Variables . . . 138
Assigning Values . . . 138
The Operations Symbols . . . 139
Conditional Execution . . . 139
Comment Lines . . . 141
Overall Program Structure . . . 142

5. HP VEE Programming Examples

About This Chapter . . . 143
Wiring Connections and File Locations for the Examples . . . 147
Example File Location . . . 147
Installing Example Files . . . 147
Virtual Front Panel Program . . . 148
Calibration . . . 151
Function Test . . . 152
Programming Model Example . . . 153
Error Checking . . . 156
Configuration Display . . . 157

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Engineering Unit Conversion . . . 158
Custom Function Generation . . . 160
Custom EU/Function Example . . . 162
Curve Fitting and EU Generation . . . 164
Interrupt Handling . . . 165
Simple Data Logger Example . . . 167
Modification of Variables and Arrays . . . 170
Algorithm Modification . . . 172
Driver Download . . . 174
Firmware-Update Download . . . 175

6. HP E1419 Command Reference

Using This Chapter . . . 177
Overall Command Index . . . 177
Command Fundamentals . . . 182
Common Command Format . . . 182
SCPI Command Format . . . 182
Linking Commands . . . 185
C-SCPI Data Types . . . 187
SCPI Command Reference . . . 188
ABORt . . . 189
ALGorithm . . . 190
ALGorithm[:EXPLicit]:ARRay . . . 191
ALGorithm[:EXPLicit]:ARRay? . . . 192
ALGorithm[:EXPLicit]:DEFine . . . 192
ALGorithm[:EXPLicit]:SCALar . . . 196
ALGorithm[:EXPLicit]:SCALar? . . . 197
ALGorithm[:EXPLicit]:SCAN:RATio . . . 197
ALGorithm[:EXPLicit]:SCAN:RATio? . . . 198
ALGorithm[:EXPLicit]:SIZe? . . . 199
ALGorithm[:EXPLicit][:STATe] . . . 199
ALGorithm[:EXPLicit][:STATe]? . . . 200
ALGorithm[:EXPLicit]:TIME? . . . 201
ALGorithm:FUNCtion:DEFine . . . 201
ALGorithm:OUTPut:DELay . . . 203
ALGorithm:OUTPut:DELay? . . . 204
ALGorithm:UPDate[:IMMediate] . . . 204
ALGorithm:UPDate:CHANnel . . . 205
ALGorithm:UPDate:WINDow . . . 206
ALGOrithm:UPDate:WINDow? . . . 207
ARM . . . 208
ARM[:IMMediate] . . . 209
ARM:SOURce . . . 209
ARM:SOURce? . . . 210
CALibration . . . 211
CALibration:CONFigure:RESistance . . . 212
CALibration:CONFigure:VOLTage . . . 213
CALibration:SETup . . . 214
CALibration:SETup? . . . 214
CALibration:STORe . . . 215
CALibration:TARE . . . 216

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CALibration:TARE:RESet . . . 218
CALibration:TARE? . . . 218
CALibration:VALue:RESistance . . . 219
CALibration:VALue:VOLTage . . . 219
CALibration:ZERO? . . . 220
DIAGnostic . . . 222
DIAGnostic:CALibration:SETup[:MODE] . . . 223
DIAGnostic:CALibration:SETup[:MODE]? . . . 223
DIAGnostic:CALibration:TARE[:OTDetect]:MODE . . . 224
DIAGnostic:CALibration:TARE[:OTDetect]:MODE? . . . 224
DIAGnostic:CHECksum? . . . 225
DIAGnostic:CUSTom:LINear . . . 225
DIAGnostic:CUSTom:PIECewise . . . 226
DIAGnostic:CUSTom:REFerence:TEMPerature . . . 226
DIAGnostic:IEEE . . . 227
DIAGnostic:IEEE? . . . 227
DIAGnostic:INTerrupt[:LINe] . . . 228
DIAGnostic:INTerrupt[:LINe]? . . . 228
DIAGnostic:OTDetect[:STATe] . . . 228
DIAGnostic:OTDetect[:STATe]? . . . 229
DIAGnostic:QUERy:SCPREAD? . . . 230
DIAGnostic:VERSion? . . . 230
FETCh? . . . 231
FORMat . . . 233
FORMat[:DATA] . . . 233
FORMat[:DATA]? . . . 234
INITiate . . . 236
INITiate[:IMMediate] . . . 236
INPut . . . 237
INPut:DEBounce:TIME . . . 237
INPut:FILTer[:LPASs]:FREQuency . . . 238
INPut:FILTer[:LPASs]:FREQuency? . . . 239
INPut:FILTer[:LPASs][:STATe] . . . 240
INPut:FILTer[:LPASs][:STATe]? . . . 240
INPut:GAIN . . . 241
INPut:GAIN? . . . 242
INPut:LOW . . . 242
INPut:LOW? . . . 243
INPut:POLarity . . . 243
INPut:POLarity? . . . 243
INPut:THReshold:LEVel? . . . 244
MEMory . . . 245
MEMory:VME:ADDRess . . . 246
MEMory:VME:ADDRess? . . . 246
MEMory:VME:SIZE . . . 246
MEMory:VME:SIZE? . . . 247
MEMory:VME:STATe . . . 247
MEMory:VME:STATe? . . . 248
OUTPut . . . 249
OUTPut:CURRent:AMPLitude . . . 249
OUTPut:CURRent:AMPLitude? . . . 250
OUTPut:CURRent[:STATe] . . . 251

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OUTPut:CURRent[:STATe]? . . . 251
OUTPut:POLarity . . . 252
OUTPut:POLarity? . . . 252
OUTPut:SHUNt . . . 253
OUTPut:SHUNt? . . . 253
OUTPut:TTLTrg:SOURce . . . 254
OUTPut:TTLTrg:SOURce? . . . 254
OUTPut:TTLTrg<n>[:STATe] . . . 255
OUTPut:TTLTrg<n>[:STATe]? . . . 255
OUTPut:TYPE . . . 255
OUTPut:TYPE? . . . 256
OUTPut:VOLTage:AMPLitude . . . 256
OUTPut:VOLTage:AMPLitude? . . . 257
ROUTe . . . 258
ROUTe:SEQuence:DEFine? . . . 258
ROUTe:SEQuence:POINts? . . . 259
SAMPle . . . 260
SAMPle:TIMer . . . 260
SAMPle:TIMer? . . . 260
[SENSe] . . . 262
[SENSe:]CHANnel:SETTling . . . 263
[SENSe:]CHANnel:SETTling? . . . 264
[SENSe:]DATA:CVTable? . . . 264
[SENSe:]DATA:CVTable:RESet . . . 265
[SENSe:]DATA:FIFO[:ALL]? . . . 266
[SENSe:]DATA:FIFO:COUNt? . . . 266
[SENSe:]DATA:FIFO:COUNt:HALF? . . . 267
[SENSe:]DATA:FIFO:HALF? . . . 267
[SENSe:]DATA:FIFO:MODE . . . 268
[SENSe:]DATA:FIFO:MODE? . . . 269
[SENSe:]DATA:FIFO:PART? . . . 269
[SENSe:]DATA:FIFO:RESet . . . 270
[SENSe:]FREQuency:APERture . . . 270
[SENSe:]FREQuency:APERture? . . . 271
[SENSe:]FUNCtion:CONDition . . . 271
[SENSe:]FUNCtion:CUSTom . . . 272
[SENSe:]FUNCtion:CUSTom:REFerence . . . 273
[SENSe:]FUNCtion:CUSTom:TCouple . . . 274
[SENSe:]FUNCtion:FREQuency . . . 275
[SENSe:]FUNCtion:RESistance . . . 275
[SENSe:]FUNCtion:STRain:FBENding . . . 277
[SENSe:]FUNCtion:STRain:FBPoisson . . . 277
[SENSe:]FUNCtion:STRain:FPOisson . . . 277
[SENSe:]FUNCtion:STRain:HBENding . . . 277
[SENSe:]FUNCtion:STRain:HPOisson . . . 277
[SENSe:]FUNCtion:STRain[:QUARter] . . . 277
[SENSe:]FUNCtion:TEMPerature . . . 278
[SENSe:]FUNCtion:TOTalize . . . 280
[SENSe:]FUNCtion:VOLTage[:DC] . . . 280
[SENSe:]REFerence . . . 281
[SENSe:]REFerence:CHANnels . . . 283
[SENSe:]REFerence:TEMPerature . . . 284

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]STRain:EXCitation . . . 284
[SENSe:]STRain:EXCitation? . . . 285
[SENSe:]STRain:GFACtor . . . 285
[SENSe:]STRain:GFACtor? . . . 286
[SENSe:]STRain:POISson . . . 286
[SENSe:]STRain:POISson? . . . 287
[SENSe:]STRain:UNSTrained . . . 287
[SENSe:]STRain:UNSTrained? . . . 288
[SENSe:]TOTalize:RESet:MODE . . . 288
[SENSe:]TOTalize:RESet:MODE? . . . 289
SOURce . . . 290
SOURce:FM[:STATe] . . . 290
SOURce:FM:STATe? . . . 291
SOURce:FUNCtion[:SHAPe]:CONDition . . . 291
SOURce:FUNCtion[:SHAPe]:PULSe . . . 292
SOURce:FUNCtion[:SHAPe]:SQUare . . . 292
SOURce:PULM[:STATe] . . . 292
SOURce:PULM:STATe? . . . 293
SOURce:PULSe:PERiod . . . 293
SOURce:PULSe:PERiod? . . . 294
SOURce:PULSe:WIDTh . . . 294
SOURce:PULSe:WIDTh? . . . 294
STATus . . . 296
STATus:OPERation:CONDition? . . . 298
STATus:OPERation:ENABle . . . 299
STATus:OPERation:ENABle? . . . 300
STATus:OPERation[:EVENt]? . . . 300
STATus:OPERation:NTRansition . . . 300
STATus:OPERation:NTRansition? . . . 301
STATus:OPERation:PTRansition . . . 302
STATus:OPERation:PTRansition? . . . 302
STATus:PRESet . . . 303
STATus:QUEStionable:CONDition? . . . 304
STATus:QUEStionable:ENABle . . . 304
STATus:QUEStionable:ENABle? . . . 305
STATus:QUEStionable[:EVENt]? . . . 305
STATus:QUEStionable:NTRansition . . . 305
STATus:QUEStionable:NTRansition? . . . 306
STATus:QUEStionable:PTRansition . . . 307
STATus:QUEStionable:PTRansition? . . . 307
SYSTem . . . 308
SYSTem:CTYPe? . . . 308
SYSTem:ERRor? . . . 308
SYSTem:VERSion? . . . 309
TRIGger . . . 310
TRIGger:COUNt . . . 312
TRIGger:COUNt? . . . 312
TRIGger[:IMMediate] . . . 313
TRIGger:SOURce . . . 313
TRIGger:SOURce? . . . 314
TRIGger:TIMer[:PERiod] . . . 314
TRIGger:TIMer[:PERiod]? . . . 315

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Common Command Reference . . . 316
*CAL? . . . 316
*CLS . . . 317
*DMC <*name*>,<*cmd_data*> . . . 317
*EMC . . . 317
*EMC? . . . 317
*ESE <*mask*> . . . 317
*ESE? . . . 318
*ESR? . . . 318
*GMC? <*name*> . . . 318
*IDN? . . . 318
*LMC? . . . 319
*OPC . . . 319
*OPC? . . . 319
*PMC . . . 320
*RMC <*name*> . . . 320
*RST . . . 320
*SRE <*mask*> . . . 321
*SRE? . . . 321
*STB? . . . 321
*TRG . . . 321
*TST? . . . 322
*WAI . . . 325
Command Quick Reference . . . 326

A. Specifications . . . 333

B. Error Messages . . . 363

C. Glossary . . . 371

D. Wiring and Noise Reduction Methods

Separating Digital and Analog SCP Signals . . . 375
Recommended Wiring and Noise Reduction Techniques . . . 376
Wiring Checklist . . . 376
HP E1419 Guard Connections . . . 377
Common Mode Voltage Limits . . . 377
When to Make Shield Connections . . . 377
Noise Due to Inadequate Card Grounding . . . 377
HP E1419 Noise Rejection . . . 378
Normal Mode Noise (Enm) . . . 378
Common Mode Noise (Ecm) . . . 378
Keeping Common Mode Noise out of the Amplifier . . . 378
Reducing Common Mode Rejection Using Tri-Filar Transformers . . . 379

E. Generating User Defined Functions

Introduction. . . . 381
Haversine Example. . . . 382
Limitations. . . . 384

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Certification

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology (formerly National Bureau of Standards), to the extent allowed by that organization's calibration facility, and to the calibration facilities of other International Standards Organization members.

Warranty

This Hewlett-Packard product is warranted against defects in materials and workmanship for a period of three years from date of shipment. Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) other HP products. During the warranty period, Hewlett-Packard Company 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 Hewlett-Packard (HP). Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with a product will execute its programming instructions when properly installed on that product. HP does not warrant that the operation of the product, 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 products or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

The design and implementation of any circuit on this product is the sole responsibility of the Buyer. HP does not warrant the Buyer's circuitry or malfunctions of HP products that result from the Buyer's circuitry. In addition, HP does not warrant any damage that occurs as a result of the Buyer's circuit or any defects that result from Buyer-supplied products.

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

Exclusive Remedies

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

Notice

The information contained in this document is subject to change without notice. HEWLETT-PACKARD (HP) 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. HP 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. This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Hewlett-Packard Company. HP assumes no responsibility for the use or reliability of its software on equipment that is not furnished by HP.

Restricted Rights Legend

U.S. Government Restricted Rights. The Software and Documentation have been developed entirely at private expense. They are delivered and licensed as "commercial computer software" as defined in DFARS 252.227- 7013 (Oct 1988), DFARS 252.211-7015 (May 1991) or DFARS 252.227-7014 (Jun 1995), as a "commercial item" as defined in FAR 2.101(a), or as "Restricted computer software" as defined in FAR 52.227-19 (Jun 1987)(or any equivalent agency regulation or contract clause), whichever is applicable. You have only those rights provided for such Software and Documentation by the applicable FAR or DFARS clause or the HP standard software agreement for the product involved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Documentation History

All Editions and Updates of this manual and their creation date are listed below. The first Edition of the manual is Edition 1. The Edition number increments by 1 whenever the manual is revised. Updates, which are issued between Editions, contain replacement pages to correct or add additional information to the current Edition of the manual. Whenever a new Edition is created, it will contain all of the Update information for the previous Edition. Each new Edition or Update also includes a revised copy of this documentation history page.

Edition 1 . . . . . April 1997

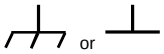
Safety Symbols



Instruction manual symbol affixed to product. Indicates that the user must refer to the manual for specific WARNING or CAUTION information to avoid personal injury or damage to the product.



Indicates the field wiring terminal that must be connected to earth ground before operating the equipment—protects against electrical shock in case of fault.



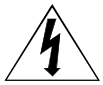
Frame or chassis ground terminal—typically connects to the equipment's metal frame.



Alternating current (AC).



Direct current (DC).



Indicates hazardous voltages.

WARNING Calls attention to a procedure, practice, or condition that could cause bodily injury or death.

CAUTION Calls attention to a procedure, practice, or condition that could possibly cause damage to equipment or permanent loss of data.

WARNINGS

The following general safety precautions must be observed during all phases of operation, service, and repair of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the product. Hewlett-Packard Company assumes no liability for the customer's failure to comply with these requirements.

Ground the equipment: For Safety Class 1 equipment (equipment having a protective earth terminal), an uninterruptible safety earth ground must be provided from the mains power source to the product input wiring terminals or supplied power cable.

DO NOT operate the product in an explosive atmosphere or in the presence of flammable gases or fumes.

For continued protection against fire, replace the line fuse(s) only with fuse(s) of the same voltage and current rating and type. DO NOT use repaired fuses or short-circuited fuse holders.

Keep away from live circuits: Operating personnel must not remove equipment covers or shields. Procedures involving the removal of covers or shields are for use by service-trained personnel only. Under certain conditions, dangerous voltages may exist even with the equipment switched off. To avoid dangerous electrical shock, DO NOT perform procedures involving cover or shield removal unless you are qualified to do so.

DO NOT operate damaged equipment: Whenever it is possible that the safety protection features built into this product have been impaired, either through physical damage, excessive moisture, or any other reason, REMOVE POWER and do not use the product until safe operation can be verified by service-trained personnel. If necessary, return the product to a Hewlett-Packard Sales and Service Office for service and repair to ensure that safety features are maintained.

DO NOT service or adjust alone: Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT substitute parts or modify equipment: Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the product. Return the product to a Hewlett-Packard Sales and Service Office for service and repair to ensure that safety features are maintained.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Declaration of Conformity

according to ISO/IEC Guide 22 and EN 45014

Manufacturer's Name: Hewlett-Packard Company
Loveland Manufacturing Center

Manufacturer's Address: 815 14th Street S.W.
Loveland, Colorado 80537

declares, that the product:

Product Name: HP Multifunction*Plus*

Model Number: HP E1419A

Product Options: All

conforms to the following Product Specifications:

Safety: IEC 1010-1 (1990) Incl. Amend 1 (1992)/EN61010-1 (1993)
CSA C22.2 #1010.1 (1992)
UL 3111-1 (1994)

EMC: CISPR 11:1990/EN55011 (1991): Group1 Class A
IEC 801-2:1991/EN50082-1 (1992): 4kVCD, 8kVAD
IEC 801-3:1984/EN50082-1 (1992): 3 V/m
IEC 801-4:1988/EN50082-1 (1992): 1kV Power Line
.5kV Signal Lines

Supplementary Information: The product herewith complies with the requirements of the Low Voltage Directive 73/23/EEC and the EMC Directive 89/336/EEC (inclusive 93/68/EEC) and carries the "CE" mark accordingly.

Tested in a typical configuration in an HP C-Size VXI mainframe.



March, 1997 **Jim White, QA Manager**

European contact: Your local Hewlett-Packard Sales and Service Office or Hewlett-Packard GmbH, Department HQ-TRE, Herrenberger Straße 130, D-71034 Böblingen, Germany (FAX +49-7031-14-3143).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Notes

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 1
Getting Started

About this Chapter

This chapter will explain hardware configuration before installation in a VXIbus mainframe. By attending to each of these configuration items, your HP E1419A won't have to be removed from its mainframe later. Chapter contents include:

- Configuring the HP E1419A 13
- Instrument Drivers 21
- About Example Programs 21
- Verifying a Successful Configuration 21

Configuring the HP E1419A

There are several aspects to configuring the module before installing it in a VXIbus mainframe. They are:

- Setting the Logical Address Switch 14
- Installing Signal Conditioning Plug-ons 15
- Disabling the Input Protect Feature 18
- Disabling Flash Memory Access 19

For most applications you will **only need to change the Logical Address switch and install SCPs** prior to installation in a mainframe. The other settings can be used as delivered.

Switch/Jumper	Setting
Logical Address Switch	208
Input Protect Jumper	Protected
Flash Memory Protect Jumper	PROG

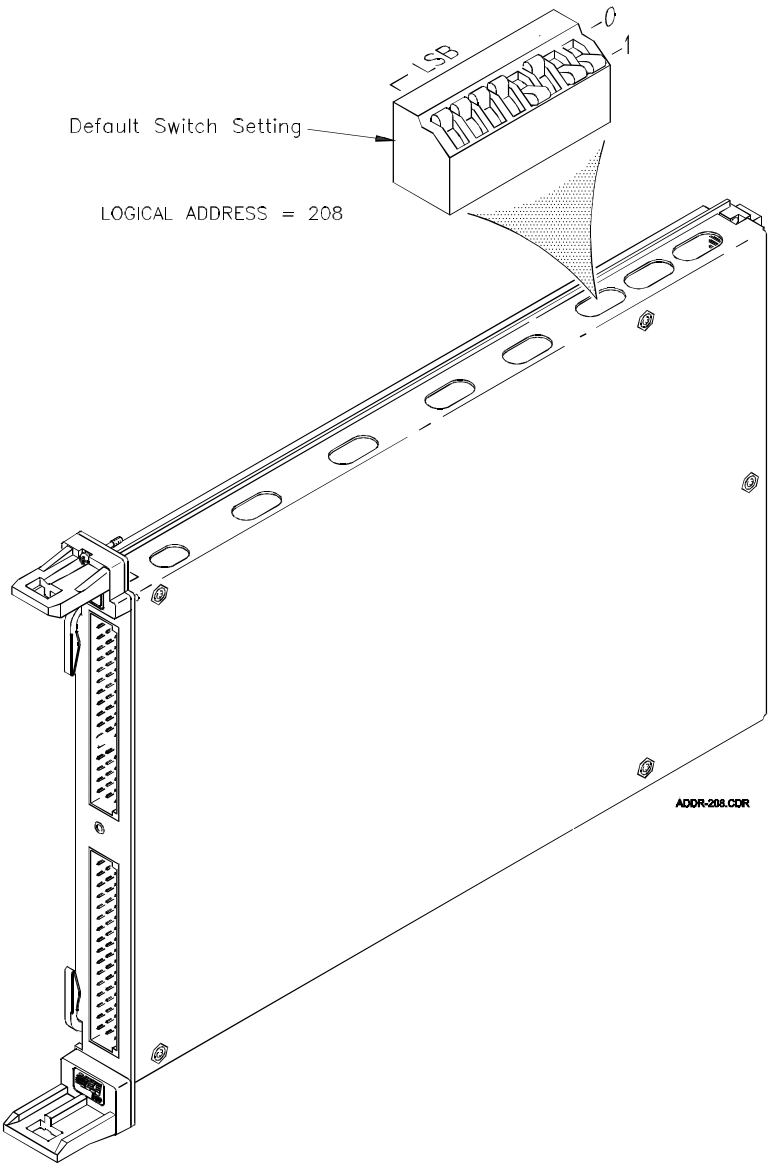
NOTE Setting the VXIbus Interrupt Level: The HP E1419A uses a default VXIbus interrupt level of 1. The default setting is in effect at power-on and after a *RST command. You can change the interrupt level by executing the DIAGnostic:INTerrupt[:LINe] command in your application program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Setting the Logical Address Switch

Follow the next figure and ignore any switch numbering printed on the Logical Address switch. When installing more than one HP E1419A in a single VXIbus Mainframe, set each instrument to a different Logical Address.

Setting Logical Address Switch **E1419A**



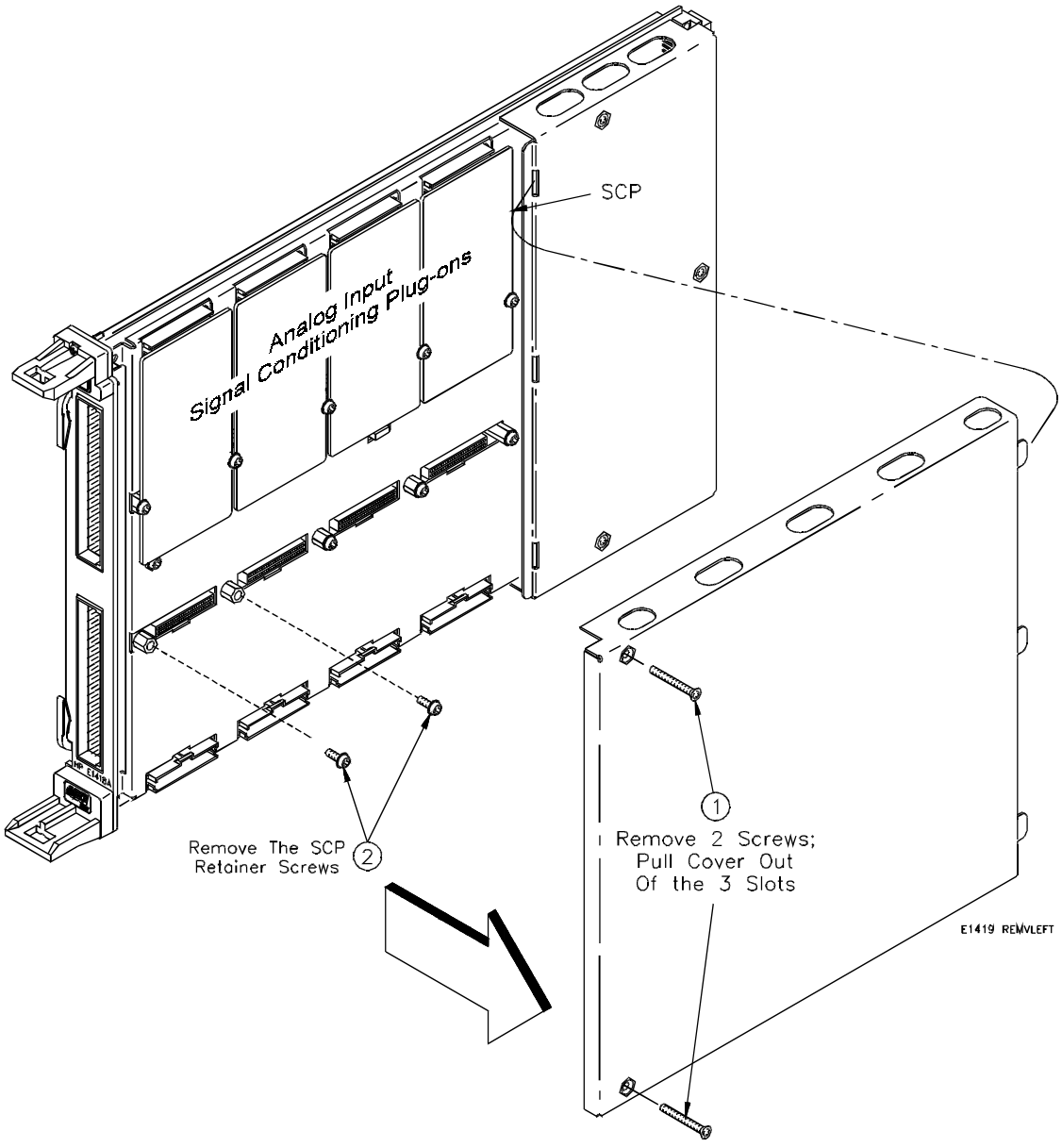
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Installing Signal Conditioning Plug-ons

The following illustrations show the steps you'll use to install Signal Conditioning Plug-ons. The HP E1419A supports only non-programmable analog input SCPs in positions 0 through 3. Any mix of SCP types can be installed in SCP positions 4 through 7.

CAUTION Use approved Static Discharge Safe handling procedures anytime you have the covers removed from the HP E1419A or are handling SCPs.

1 Installing SCPs: Removing the Cover **E1419A**

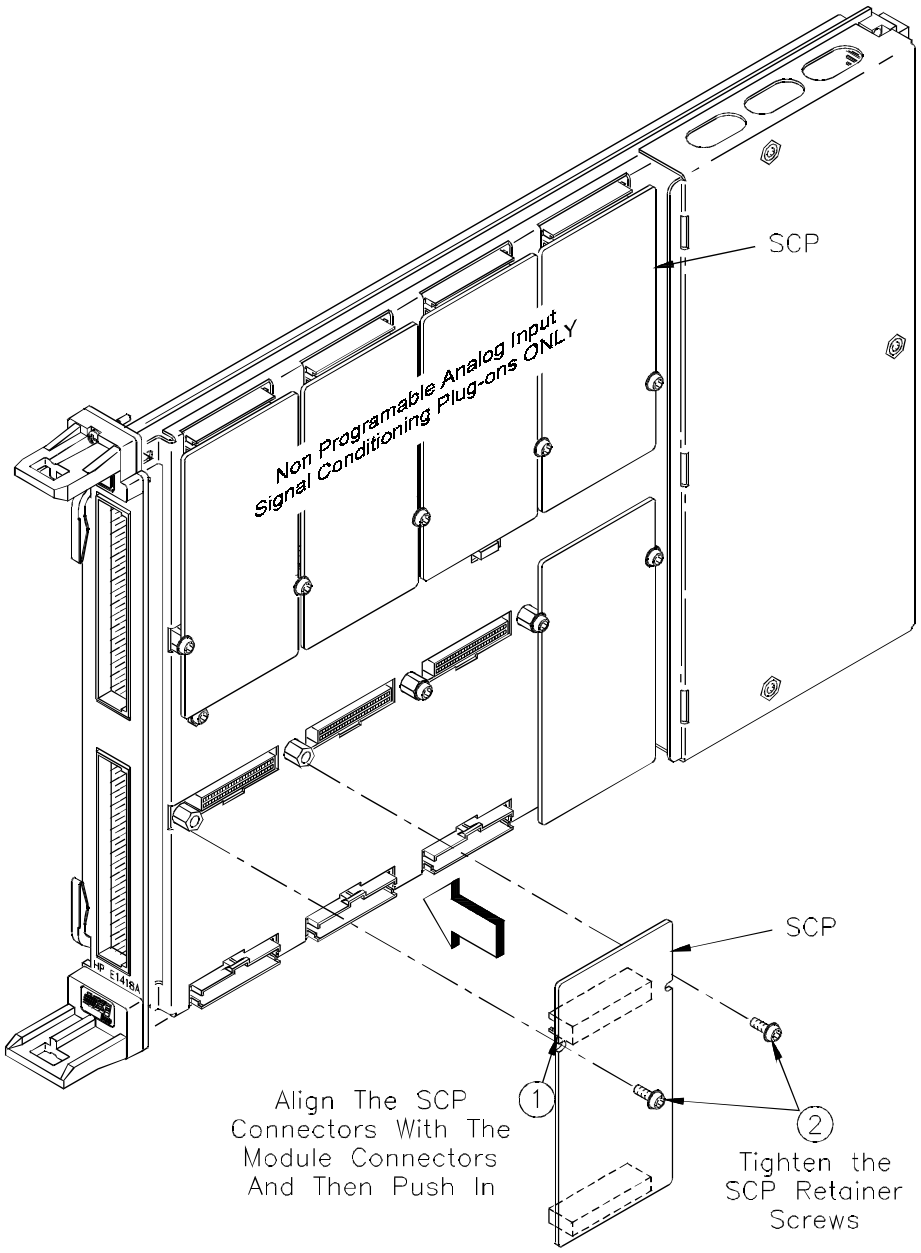


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note The only SCPs supported in SCP positions 0 through 3 are:

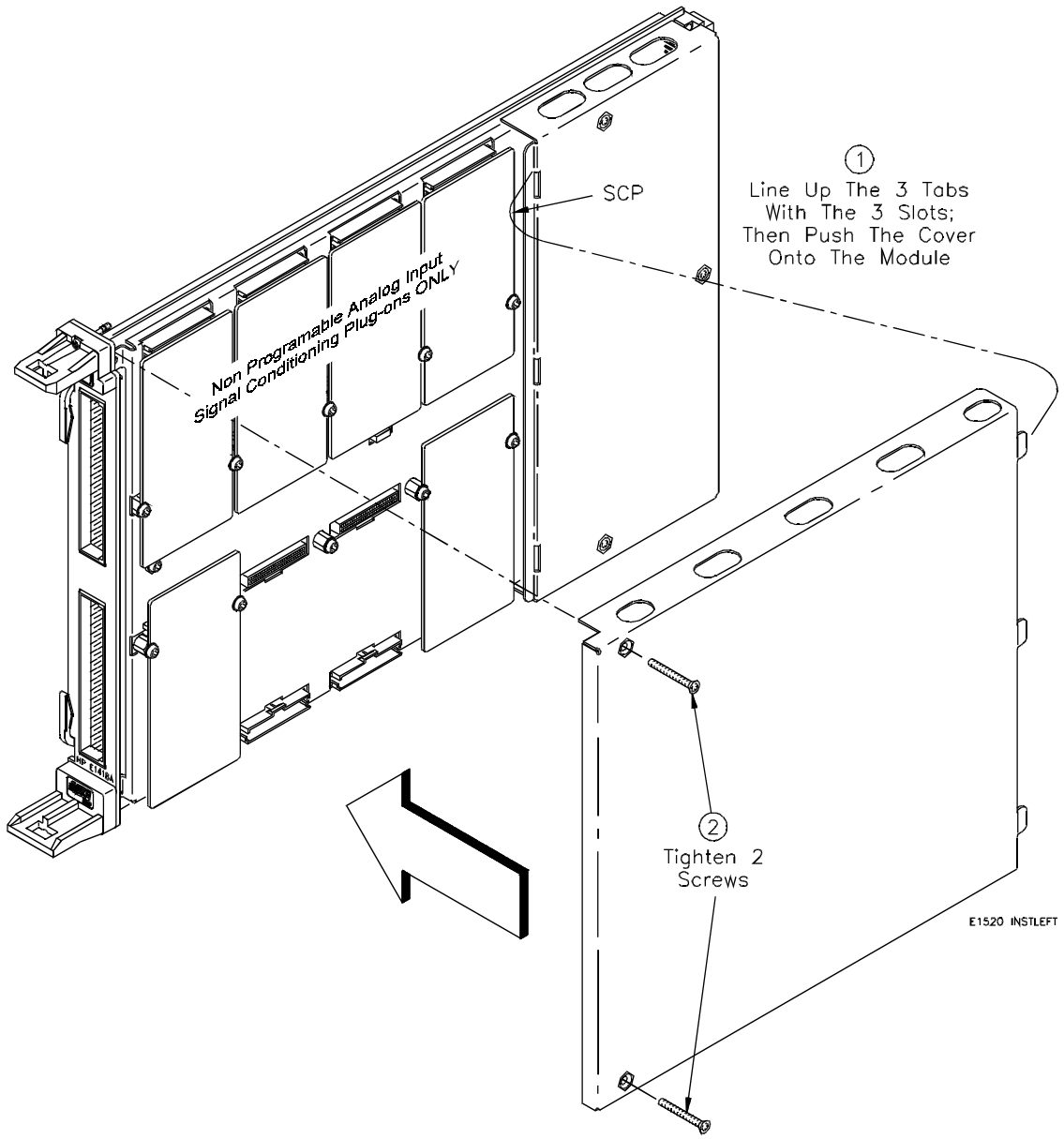
HP E1501	HP E1513
HP E1502	HP E1514
HP E1508	HP E1515
HP E1509	HP E1516
HP E1512	HP E1517

2 Installing SCPs **E1419A**



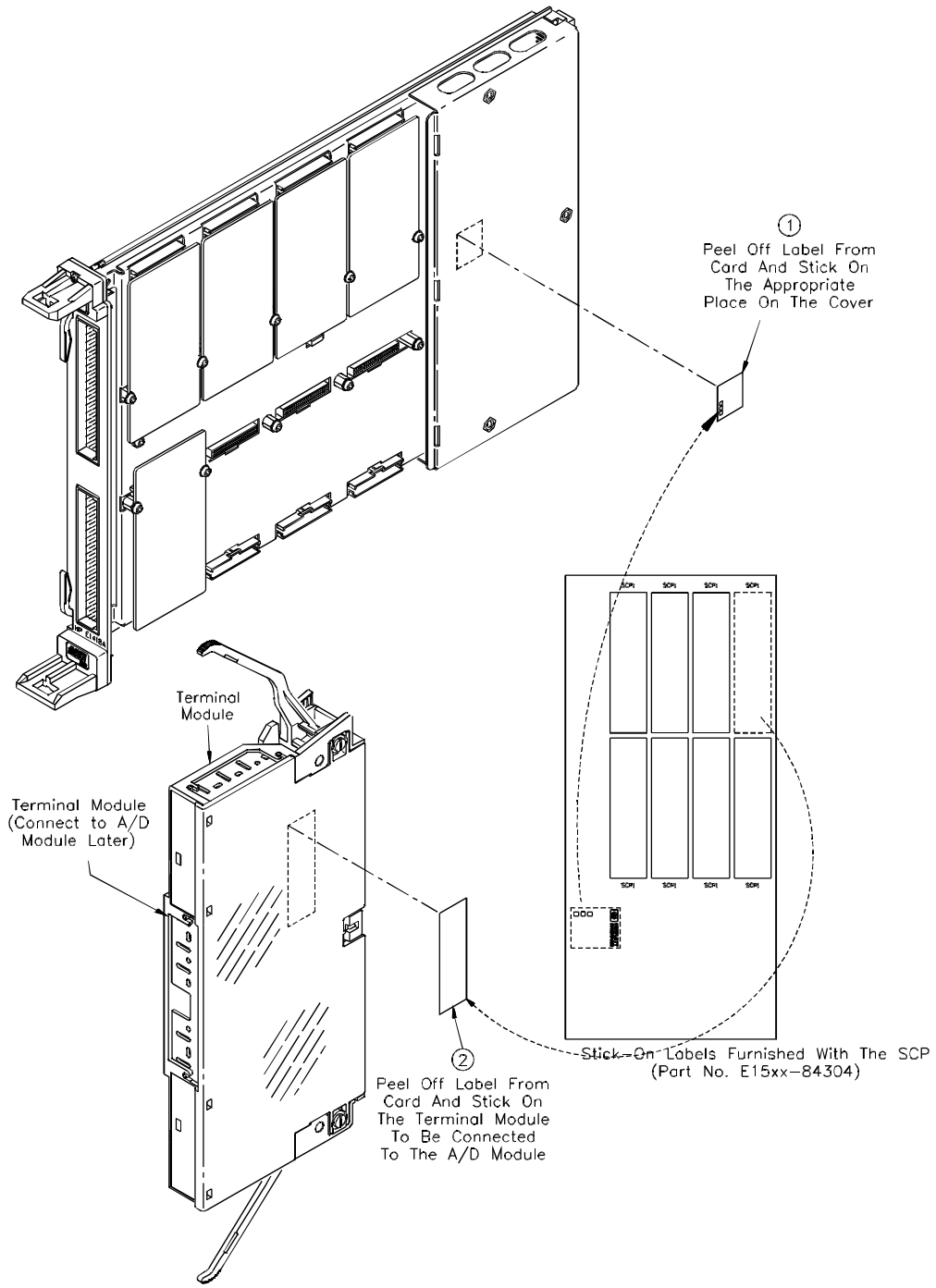
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3 Installing SCPs: Reinstalling the Cover **E1419A**



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4 Installing SCPs: Labeling **E1419A**



Disabling the Input Protect Feature (optional)

Disabling the Input Protect feature voids the HP E1419A's warranty. The Input Protect feature allows the HP E1419A to open all channel input relays if any input's voltage exceeds ±19 volts (±6 volts for non-isolated digital

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

I/O SCPs). This feature will help to protect the card's Signal Conditioning Plug-ons, input multiplexer, ranging amplifier, and A/D from destructive voltage levels. The level that trips the protection function has been set to provide a high probability of protection. The voltage level that is certain to cause damage is somewhat higher. **If in your application the importance of completing a measurement run outweighs the added risk of damage to your HP E1419A, you may choose to disable the Input Protect feature.**

VOIDS WARRANTY

Disabling the Input Protection Feature voids the HP E1419A's warranty.

To disable the Input Protection feature, locate and cut JM2202. Make a single cut in the jumper and bend the adjacent ends apart. See following illustration for location of JM2202.

Disabling Flash Memory Access (optional)

The Flash Memory Protect Jumper (JM2201) is shipped in the "PROG" position. We recommend that you leave the jumper in this position so that all of the calibration commands can function. Changing the jumper to the protect position will mean you won't be able to execute:

- The SCPI calibration command CAL:STORE ADC | TARE
- The register-based calibration commands STORECAL, and STORETAR
- Any application that installs firmware-updates or makes any other modification to Flash Memory through the A24 window.

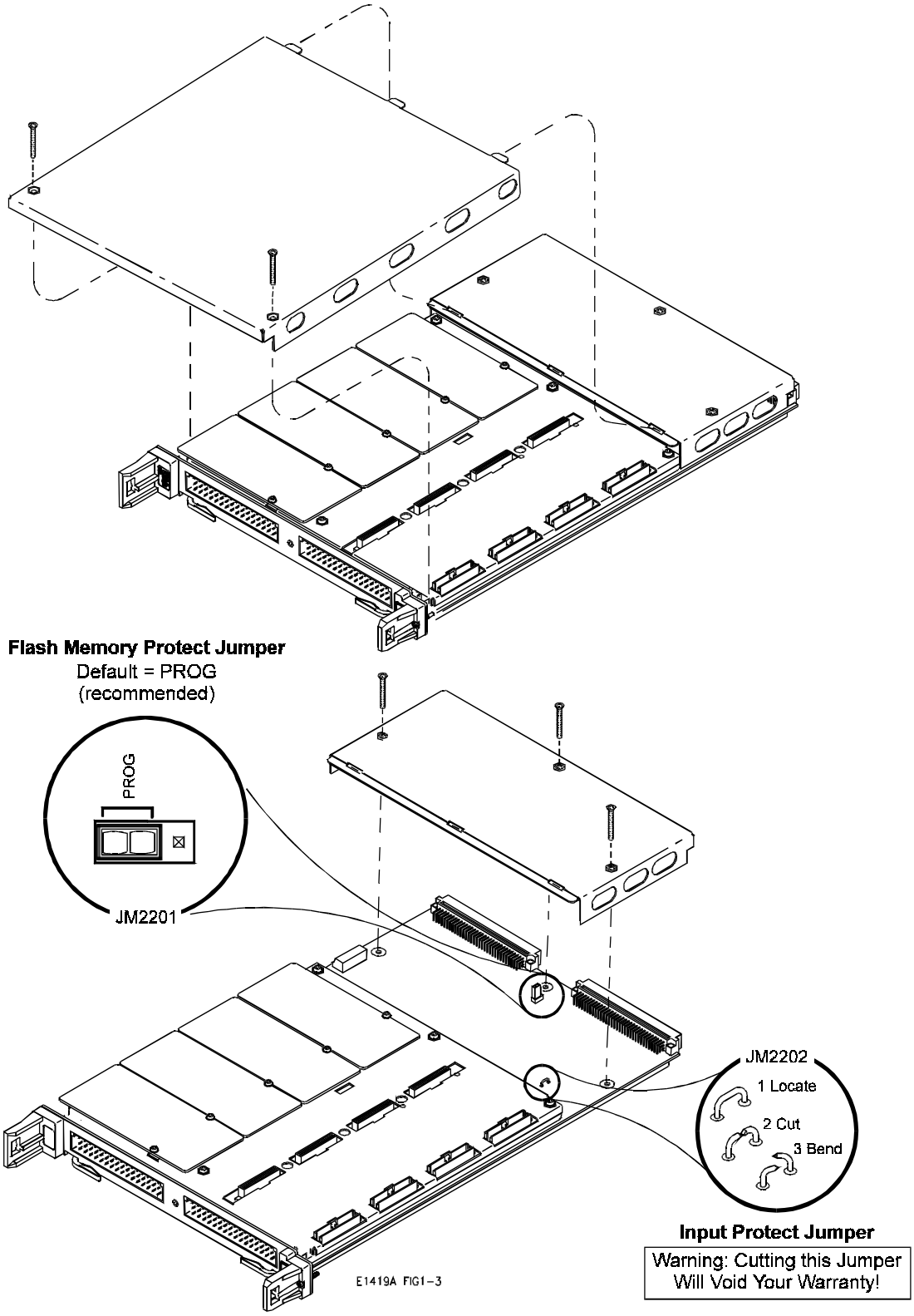
With the jumper in the "PROG" position, you can completely calibrate one or more HP E1419As without removing them from the application system. An HP E1419A calibrated in its working environment will in general be better calibrated than if it were calibrated separate from its application system.

The multimeter you use during the periodic calibration cycle should be considered your calibration transfer standard. Have your Calibration Organization control unauthorized access to its calibration constants. See the *HP E1419A Service Manual* for complete information on HP E1419A periodic calibration.

If you must limit access to the HP E1419A's calibration constants, you can place JM2201 in the protected position and cover the shield retaining screws with calibration stickers. See following illustration for location of JM2201.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Accessing and Locating JM2201 and JM2202 **E1419A**



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Instrument Drivers

The HP E1405B/E1406, down-loadable driver is supplied with your HP E1419A on the "HP Instrument Drivers" CD ROM. To download this driver into your HP E1406A Command module, run the Setup program on the CD, click on "continue", then click on "Perform Installations", then select "Download SCPI Drivers into Command Modules". This sequence starts the HP VIC installation. HP VIC will help you configure your VXI system, including downloading the HP E1419's driver.

About Example Programs

Examples on CD

All example programs mentioned by file name in this manual are available on the "Driver CD ROM" supplied with your HP E1419A. See the HP VEE Program Examples Chapter, page 147 for specific location of files on the CD.

Example Command Sequences

Where programming concepts are discussed in this manual, the commands to send to the HP E1419A are shown in the form of command sequences. These are not example programs because they are not written in any computer language. They are meant to show the HP E1419A SCPI commands in the sequence they should be sent. Where necessary, these sequences include comments to describe program flow and control such as loop - end loop, and if - end if. See the code sequence on page 3-85 for an example.

Verifying a Successful Configuration

Among the HP VEE example programs supplied with your HP E1419A is a program (file name "panl1419.vee") that can be used to check out your HP E1419A configuration and installation. When you start this "Front Panel" program, it communicates with your HP E1419A and executes instructions to determine and display the installed SCP types. It also simulates a strip chart recorder so you can select input channels to monitor and display. "Buttons" are included that will run the HP E1419A's self-test as well as well as perform an "auto-calibration". Self-test and Cal can take 3 to 15 minutes to coplete depending upon the number and type of SCPs you have installed in your HP E1419A.

Note

When you load the first HP VEE program that communicates with your HP E1419A, HP VEE will display a dialog box asking for the HP-IB address string to use. The typical HP-IB address is "70926" for an HP E1419A at logical address 208, and an HP E1406 at HP-IB address 09.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 2
Field Wiring

About This Chapter

This chapter shows how to plan and connect field wiring to the HP E1419's Terminal Module. The chapter explains proper connection of analog signals to the HP E1419, both two-wire voltage type and four-wire resistance type measurements. Connections for other measurement types (e.g., strain using the Bridge Completion SCPs) refer to specific SCP manual in the "SCP Manuals" section. Chapter contents include:

- Planning Wiring Layout for the HP E1419 23
- Faceplate Connector Pin-Signal List 27
- Optional Terminal Modules 28
- Reference Temperature Sensing with the HP E1419 31
- Configuring the On-Board/Remote Reference Jumpers 32
- Preferred Measurement Connections. 35
- Wiring and Attaching the Terminal Modules 37
- Attaching/Removing the Terminal Modules 39
- Adding Components to the Option 12 Terminal Module . . . 41
- Option 11 Terminal Module Wiring Map 42
- Option 12 Terminal Module Wiring Map 43
- The Option A3F 44

Planning Your Wiring Layout

To help you plan field wiring connections to your HP E1419A this section will provide you with a high-level overview of the HP E1419A's signal paths between the face plate connectors and the Control Processor (DSP). To eliminate any surprises after your system is wired, we will also cover any configuration interdependencies or other limiting situations (there are very few with the HP E1419A).

SCP Positions and Channel Numbers

The HP E1419A has a fixed relationship between Signal Conditioning Plug-on positions and their channel assignments. See Figure 2-1 for this discussion. Each of the eight SCP positions can connect to eight channels. Each channel signal path consists of both a High and Low signal path (for differential analog signals). Some SCP models will connect to fewer of these eight channels and those left unconnected cannot be used for other purposes. The HP E1533 Digital I/O SCP on the other hand will use each High and Low channel to provide 16 digital bits from a single SCP position.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

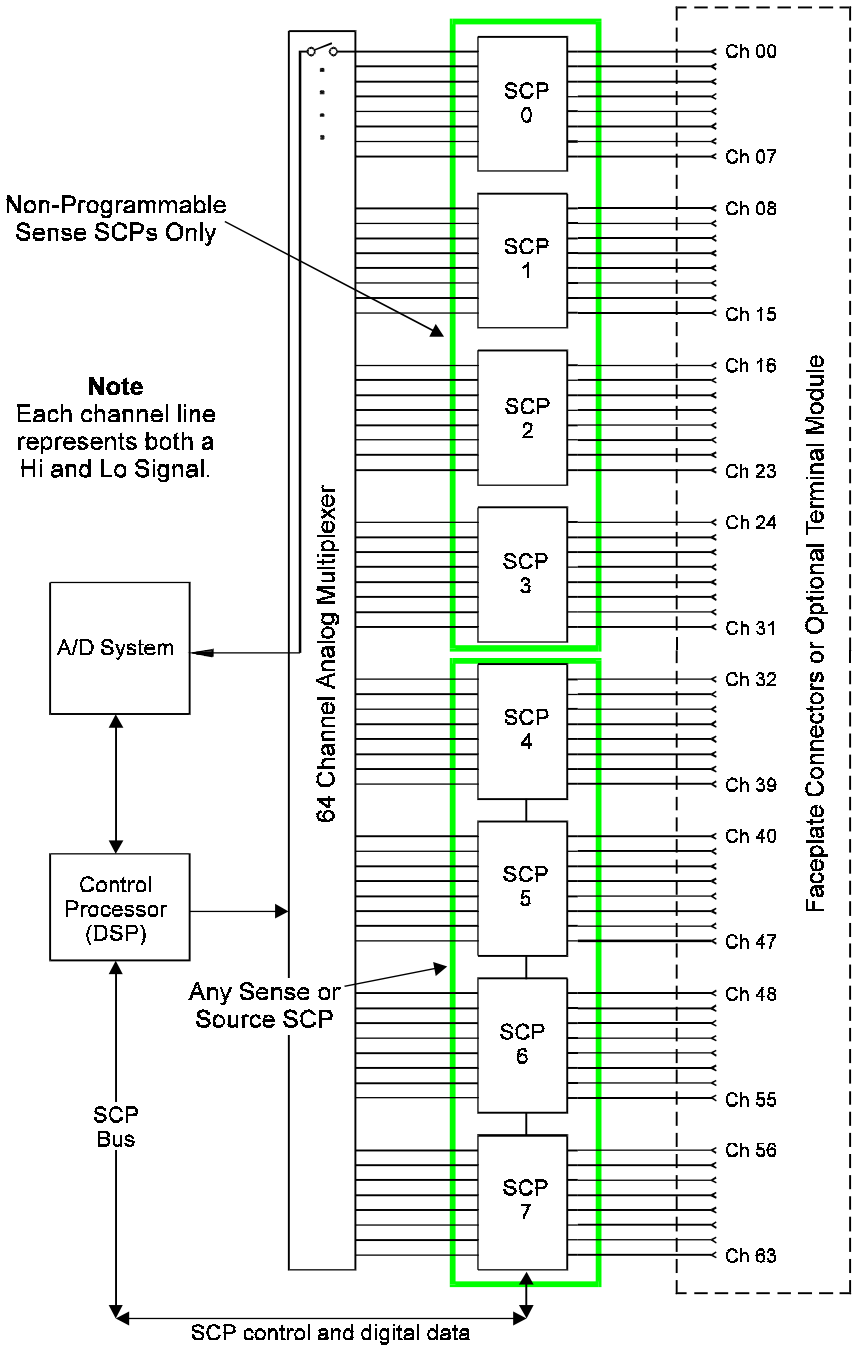


Figure 2-1 Channel Numbers at SCP Positions

SCP Types and Signal Paths

Different SCP types (analog sense, analog source, digital I/O) use different signal paths in the HP E1419A. We'll discuss each of these basic types.

Analog Sense SCPs

Analog sense SCPs connect signals at the faceplate connector and pass these signals (most with signal amplification and/or filtering) to the analog multiplexer and thus to the A/D for measurement. Here the primary signal

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

path is along the analog Hi and Lo lines. The SCP Bus carries digital signals to control the programmable parameters on the HP E1503 and HP E1510.

Analog Source SCPs

The primary signal path for analog source SCPs like the HP E1505 Resistance Current Source, the HP E1531 Voltage DAC, and the HP E1532 Current DAC is along the Hi and Lo lines from the SCP to the face plate connectors. The path from the SCP to the analog multiplexer can be used to read and verify the approximate output (although this path is not calibrated). The SCP Bus carries digital signals to these SCPs to control their output levels.

Combined Analog Source and Sense SCPs

The HP E1506, HP E1507, and HP E1511 Strain Completion SCPs as well as the HP E1518 Resistance Measurement SCP combine analog sense and analog sources in a single SCP. With these SCPs, some channels will be used to sense measurement values while others will be used to carry analog excitation voltage or current. Again the SCP Bus carries digital signals to control SCP source level and/or measurement configuration.

Digital SCPs

With digital SCPs, the signal path to and from the face plate connectors and the SCP is as always, the Hi and Lo signal paths. The HP E1534, HP E1536, and HP E1538 digital SCPs provide one digial bit per Hi and Lo pair while the HP E1533 provides 16 digital bits from a single SCP position by connecting 8 bits to the channel Hi lines and another 8 bits to the channel Lo lines. With digital SCPs, the SCP Bus is the only data path between the Control Processor and the SCP for both data and configuration control.

Pairing Sense and Source SCPs for Resistance Measurements

Resistance measurements and resistance-temperature measurements require supplying an excitation current to the resistive element to be measured. With the HP E1419A, two channels are required for each resistance to be measured. Resistances are always measured in a Four-Wire configuration. The HP E1505 Current Source SCP provides 8 excitation supplies that can be paired with any available analog sense channels to complete the measurement circuit. The HP E1518 Resistance Measurement SCP provides 4 excitation supplies and 4 amplified sense channels on a single SCP. In either case the source and sense channels must be paired together to make the resistance measurement. Figure 2-2 illustrates an example of "pairing" source SCP channels with sense SCP channels.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

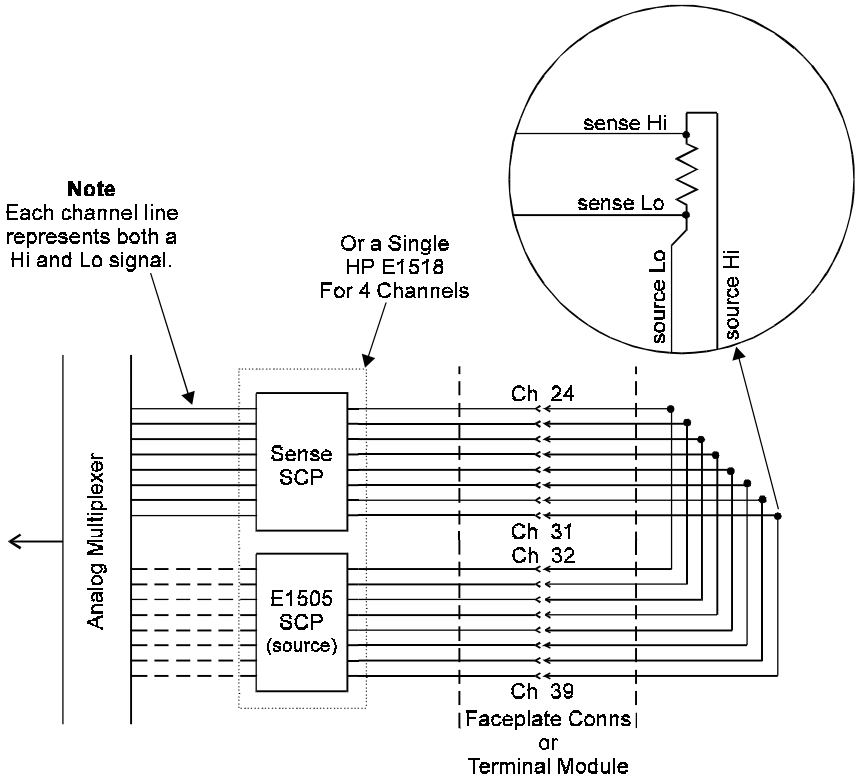


Figure 2-2 Pairing Source and Sense SCP Channels

Planning for Thermocouple Measurements

You can wire your thermocouples and your thermocouple reference temperature sensor to any of the HP E1419's channels. When you execute your scan list, you only have to make sure that the reference temperature sensor is specified in the channel sequence before any of the associated thermocouple channels (see the [SENSe:]REF:CHAN command).

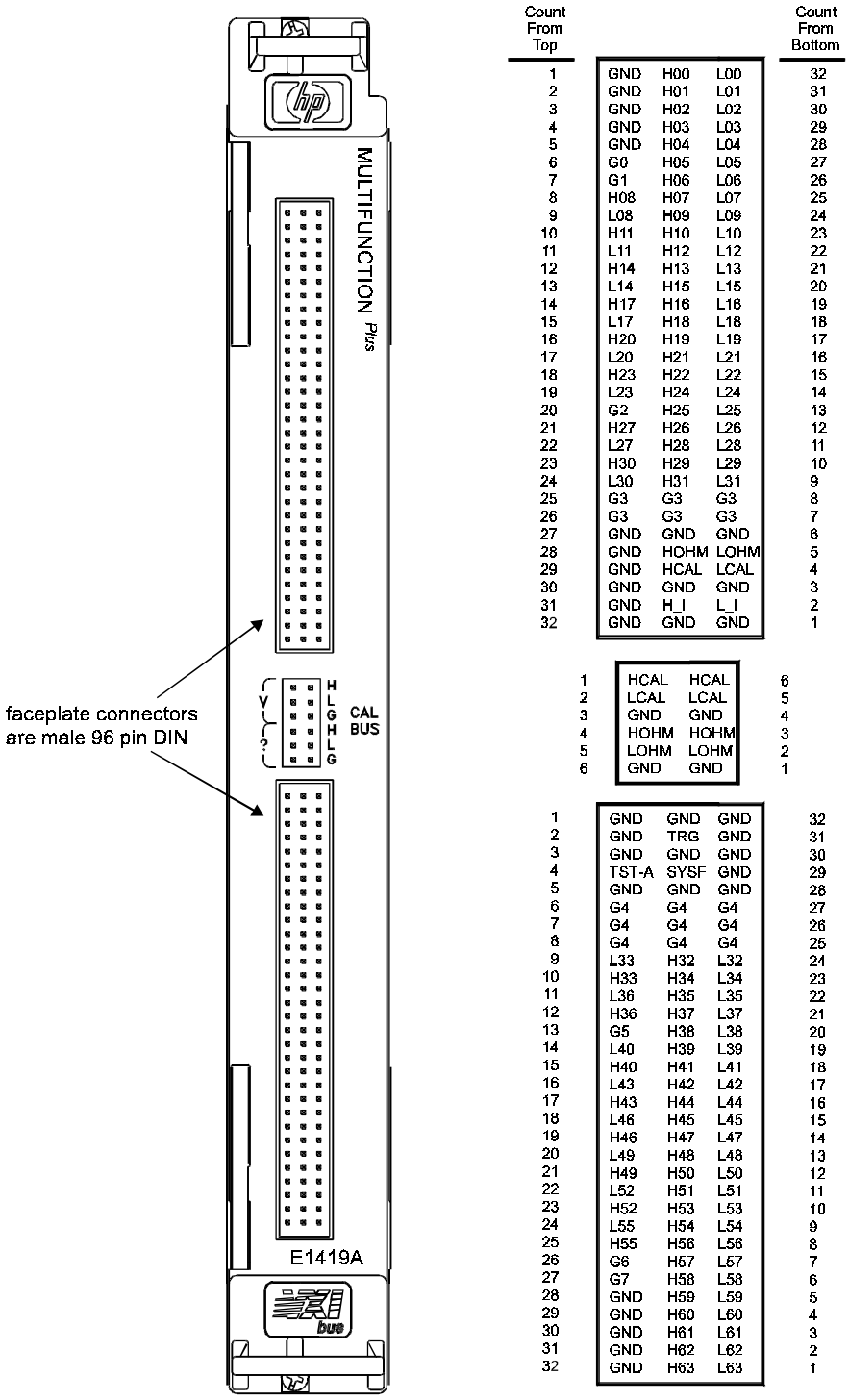
External wiring and connections to the HP E1419 are made using the Terminal Module (see page 2-37).

NOTE The isothermal reference temperature measurement made by an HP E1419 applies only to thermocouple measurements made by that instrument and through the terminal blocks associated with the reference temperature sensor (for increased isothermal reference accuracy the HP 1586 Rack Mount Terminal Panel has three reference temperature thermistors). In systems with multiple HP E1419s, each instrument must make its own reference measurements. The reference measurement made by one HP E1419 can not be used to compensate thermocouple measurements made by another HP E1419.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Faceplate Connector Pin-Signal Lists

Figure 2-3 shows the Faceplate Connector Pin Signal List for the HP E1419A.



Count From Top				Count From Bottom
1	GND	H00	L00	32
2	GND	H01	L01	31
3	GND	H02	L02	30
4	GND	H03	L03	29
5	GND	H04	L04	28
6	G0	H05	L05	27
7	G1	H06	L06	26
8	H08	H07	L07	25
9	L08	H09	L09	24
10	H11	H10	L10	23
11	L11	H12	L12	22
12	H14	H13	L13	21
13	L14	H15	L15	20
14	H17	H16	L16	19
15	L17	H18	L18	18
16	H20	H19	L19	17
17	L20	H21	L21	16
18	H23	H22	L22	15
19	L23	H24	L24	14
20	G2	H25	L25	13
21	H27	H26	L26	12
22	L27	H28	L28	11
23	H30	H29	L29	10
24	L30	H31	L31	9
25	G3	G3	G3	8
26	G3	G3	G3	7
27	GND	GND	GND	6
28	GND	HOHM	LOHM	5
29	GND	HCAL	LCAL	4
30	GND	GND	GND	3
31	GND	H_I	L_I	2
32	GND	GND	GND	1

1	HCAL	HCAL	6
2	LCAL	LCAL	5
3	GND	GND	4
4	HOHM	HOHM	3
5	LOHM	LOHM	2
6	GND	GND	1

1	GND	GND	GND	32
2	GND	TRG	GND	31
3	GND	GND	GND	30
4	TST-A	SYSF	GND	29
5	GND	GND	GND	28
6	G4	G4	G4	27
7	G4	G4	G4	26
8	G4	G4	G4	25
9	L33	H32	L32	24
10	H33	H34	L34	23
11	L36	H35	L35	22
12	H36	H37	L37	21
13	G5	H38	L38	20
14	L40	H39	L39	19
15	H40	H41	L41	18
16	L43	H42	L42	17
17	H43	H44	L44	16
18	L46	H45	L45	15
19	H46	H47	L47	14
20	L49	H48	L48	13
21	H49	H50	L50	12
22	L52	H51	L51	11
23	H52	H53	L53	10
24	L55	H54	L54	9
25	H55	H56	L56	8
26	G6	H57	L57	7
27	G7	H58	L58	6
28	GND	H59	L59	5
29	GND	H60	L60	4
30	GND	H61	L61	3
31	GND	H62	L62	2
32	GND	H63	L63	1

Figure 2-3 HP E1419A Faceplate Connector Pin Signals

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Optional Terminal Modules

The HP E1419A Option 11 Terminal Module has screw type terminal blocks. The HP E1419A Option 12 Terminal Module has spring clamp type terminal blocks. Both of these Terminal Modules provide:

- Terminal block connections to field wiring.
- Strain relief for the wiring bundle.
- Reference junction temperature sensing for thermocouple measurements.

The HP E1419A Option A3F Terminal Module is available to interface the HP E1419A to an HP E1586A rack mount terminal panel (see Page 44).

The SCPs and Terminal Module Connections

The same Terminal is used for all field wiring regardless of which Signal Conditioning Plug-on (SCP) is used. Each SCP includes a set of labels to map that SCP's channels to the Terminal Module's terminal blocks. See step 4 in "Installing Signal Conditioning Plug-ons" in Chapter 1 page 18 for HP E1419 Terminal Modules.

Note The SCPs HP E1531 through HP E1537 do not include wiring labels for the Option 11 terminal module. For these SCPs use the connection tables in the SCP's manual along with the Option 11 wiring map on page 42.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Option 11 Terminal Module Layout

Figure 2-4 shows the Option 11 Screw Terminal Module feature and connector locations.

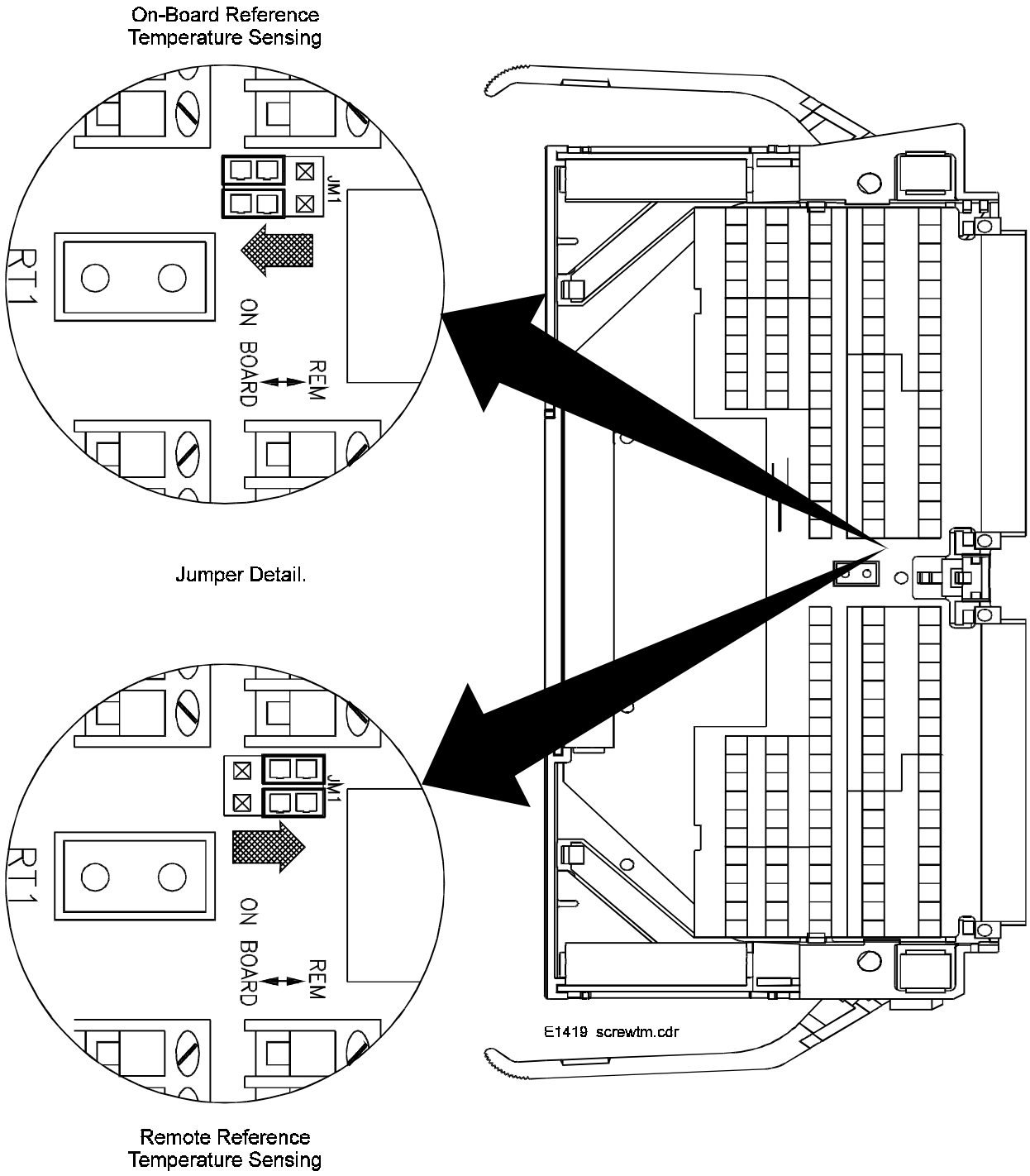


Figure 2-4 The Option 11 Screw Terminal Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Option 12 Terminal Module Layout

Figure 2-5 shows the Option 12 Spring Terminal Module features and connector locations.

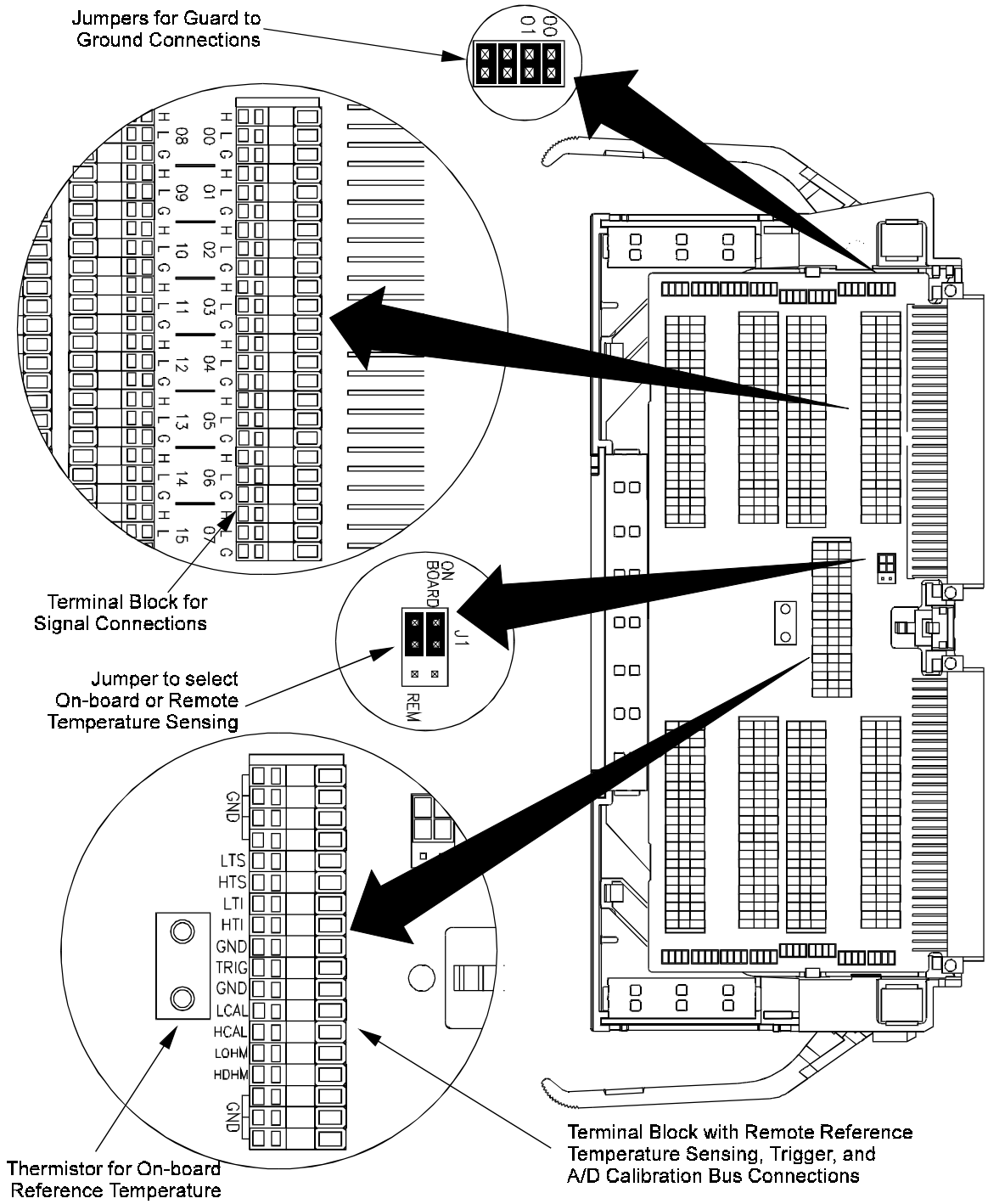


Figure 2-5 The Option 12 Spring Terminal Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Reference Temperature Sensing with the HP E1419A

The Terminal Modules provide an on-board thermistor for sensing isothermal reference temperature of the terminal blocks. Also provided is a jumper set (JM1 in Figures 2-5 and 2-4) to route the HP E1419A's on-board current source to a thermistor or RTD on a remote isothermal reference block. Figures 2-6 and 2-7 show connections for both local and remote sensing.

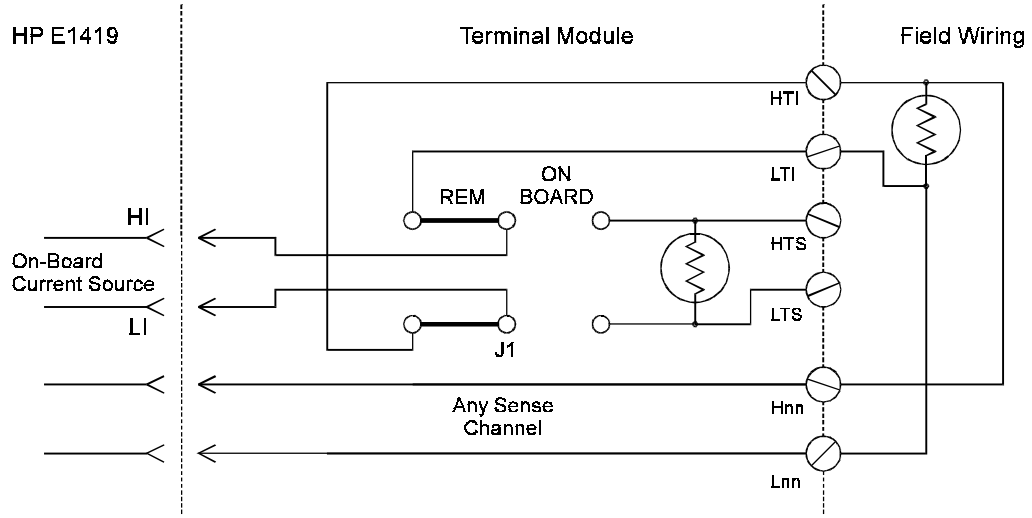


Figure 2-6 Remote Thermistor or RTD Connections

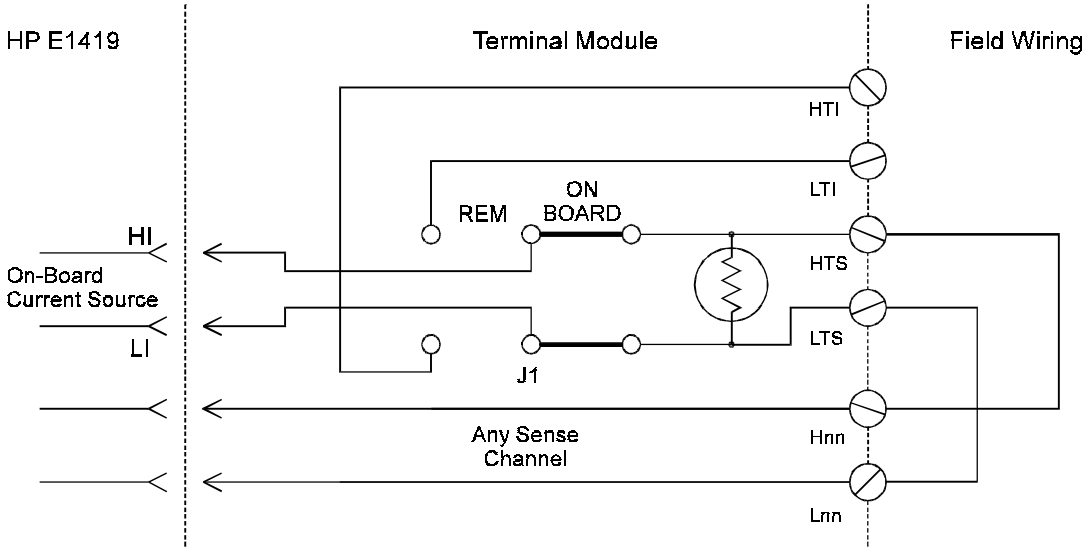
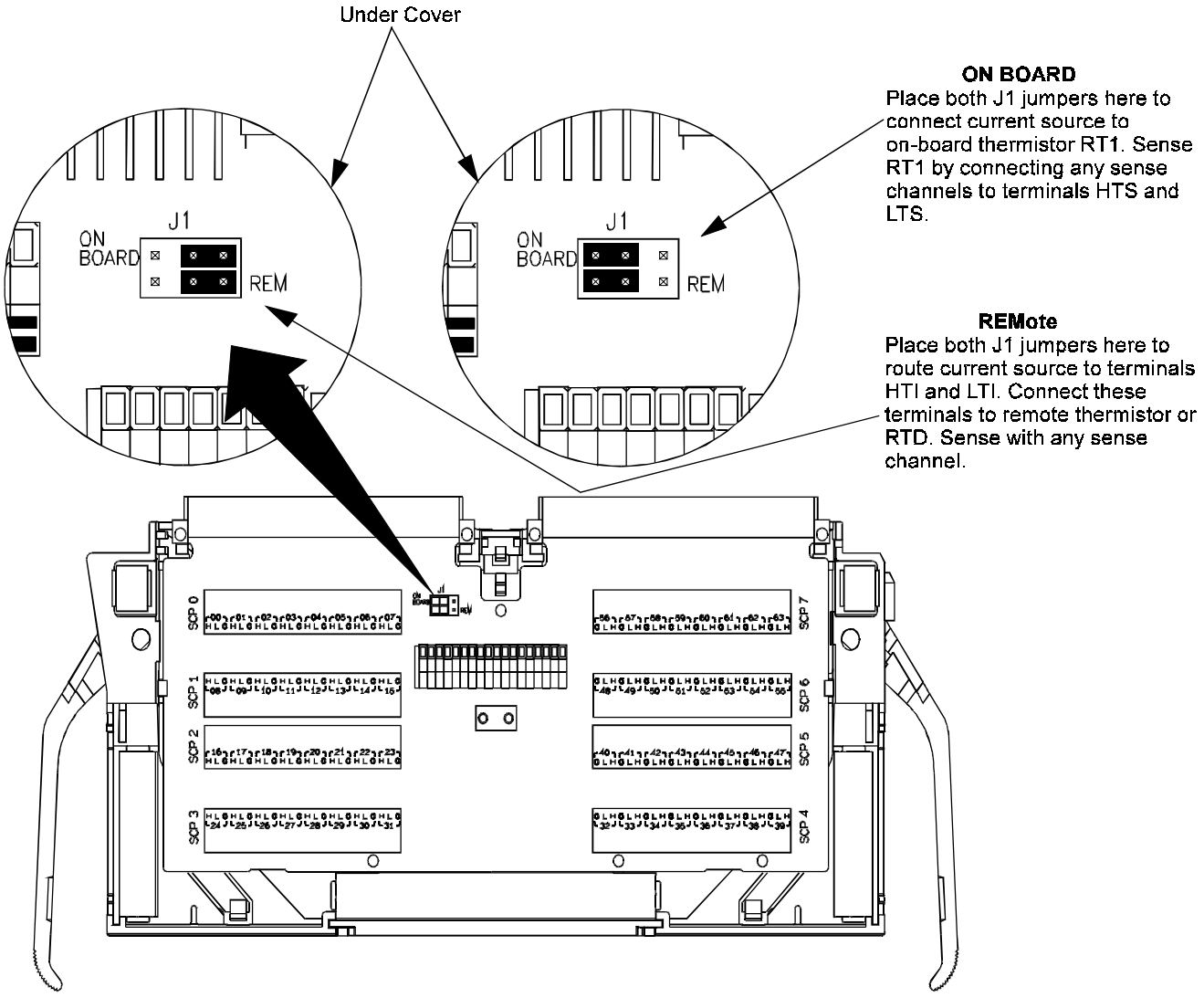


Figure 2-7 On-Board Thermistor Connections

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Configuring the On-board/Remote Reference Jumpers

Figure 2-8 shows how to set the Option 12's jumpers for on-board and remote thermocouple reference temperature measurement. Figure 2-2 shows the jumpers on the Option 11 Terminal Module. The Thermistor is used for reference junction temperature sensing for thermocouple measurements.



See figure on page 37 to remove the cover

Figure 2-8 Temperature Sensing for HP E1419 Terminal Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Terminal Module Considerations for Thermocouple Measurements

The isothermal characteristics of the Terminal Modules are crucial for good TC readings and can be affected by any of the following factors:

1. The clear plastic cover must be on the Terminal Module.
2. The thin white mylar thermal barrier must be inserted over the Terminal Module connector (Option 12 only). This prevents airflow from the HP E1419A into the Terminal Module.
3. The Terminal Module must also be in a fairly stable temperature environment, and it is best to minimize the temperature gradient between the HP E1419 and the Terminal Module.
4. The VXI mainframe cooling fan filters must be clean and there should be as much clear space in front of the fan intakes as possible.
5. Recirculating warm air inside a closed rack cabinet can cause a problem if the Terminal Module is suspended into ambient air that is significantly warmer or cooler. If the mainframe recess is mounted in a rack with both front and rear doors, closing both doors helps keep the entire HP E1419 at a uniform temperature. If there is no front door, try opening the back door to allow the mainframe to cool to the temperature of the Terminal Module.
6. HP recommends that the cooling fan switch on the back of the of an HP E1401 Mainframe is in the "High" position. The normal variable speed cooling fan control can make the internal HP E1419 module temperature cycle up and down, which affects the amplifiers with these uV level signals.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Preferred Measurement Connections

IMPORTANT!

For any A/D Module to scan channels at high speeds, it must use a very short sample period (<10µsecond for the HP E1419). If significant normal mode noise is presented to its inputs, that noise will be part of the measurement. To make quiet, accurate measurements in electrically noisy environments, use properly connected shielded wiring between the A/D and the device under test. Figure 2-9 shows recommended connections for powered transducers, thermocouples, and resistance transducers. (See Appendix D for more information on Wiring Techniques).

HINTS

1. Try to install Analog SCPs relative to Digital I/O as shown in "Separating Digital and Analog Signals" in Appendix D.
2. Use individually shielded, twisted-pair wiring for each channel.
3. Connect the shield of each wiring pair to the corresponding Guard (G) terminal on the Terminal Module (see Figure 2-10 for schematic of Guard to Ground circuitry on the Terminal Module).
4. The Terminal Module is shipped with the Ground-Guard (GND-GRD) shorting jumper installed for each channel. These may be left installed or removed (see Figure 2-11 to remove the jumper), dependent on the following conditions:
   a. **Grounded Transducer with shield connected to ground at the transducer:** Low frequency ground loops (DC and/or 50/60Hz) can result if the shield is also grounded at the Terminal Module end. To prevent this, remove the GND-GRD jumper for that channel (Figure 2-9 A/C).
   b. **Floating Transducer with shield connected to the transducer at the source:** In this case, the best performance will most likely be achieved by leaving the GND-GRD jumper in place (Figure 2-9 B/D).
5. In general, the GND-GRD jumper can be left in place unless it is necessary to remove to break low frequency (below 1 kHz) ground loops.
6. Use good quality foil or braided shield signal cable.
7. Route signal leads as far as possible from the sources of greatest noise.
8. In general, don’t connect Hi or Lo to Guard or Ground at the HP E1419.
9. It is best if there is a D.C. path somewhere in the system from Hi or Lo to Guard/Ground.
10. The impedance from Hi to Guard/Ground should be the same as from Lo to Guard/Ground (balanced).
11. Since each system is different, don’t be afraid to experiment using the suggestions presented here until you find an acceptable noise level.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

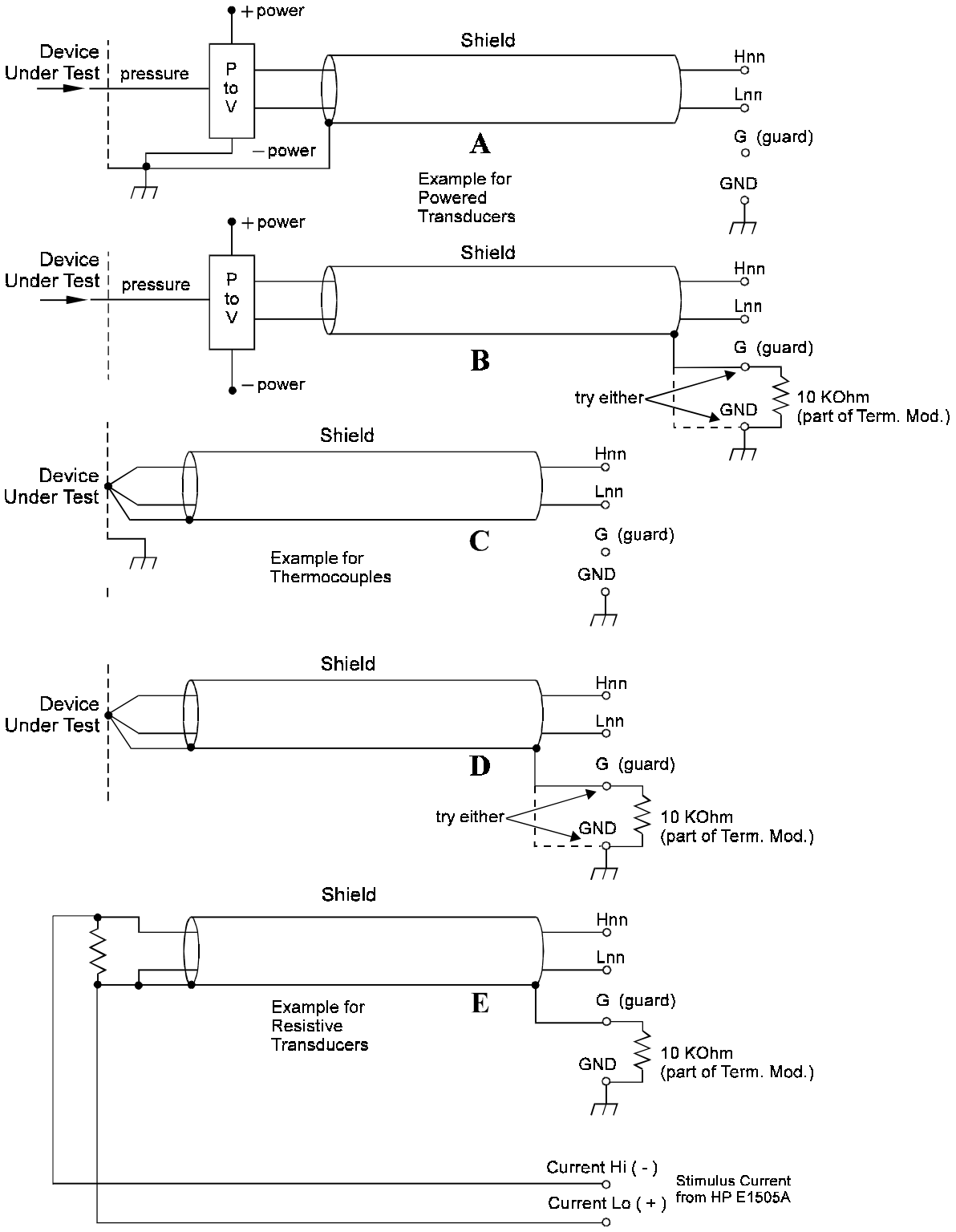


Figure 2-9 Preferred Signal Connections

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

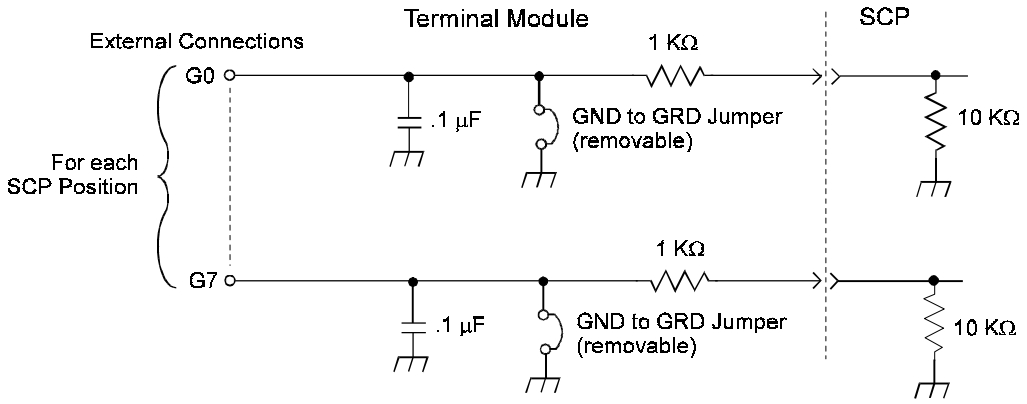


Figure 2-10 GRD/GND Circuitry Opt. 12 Terminal Module

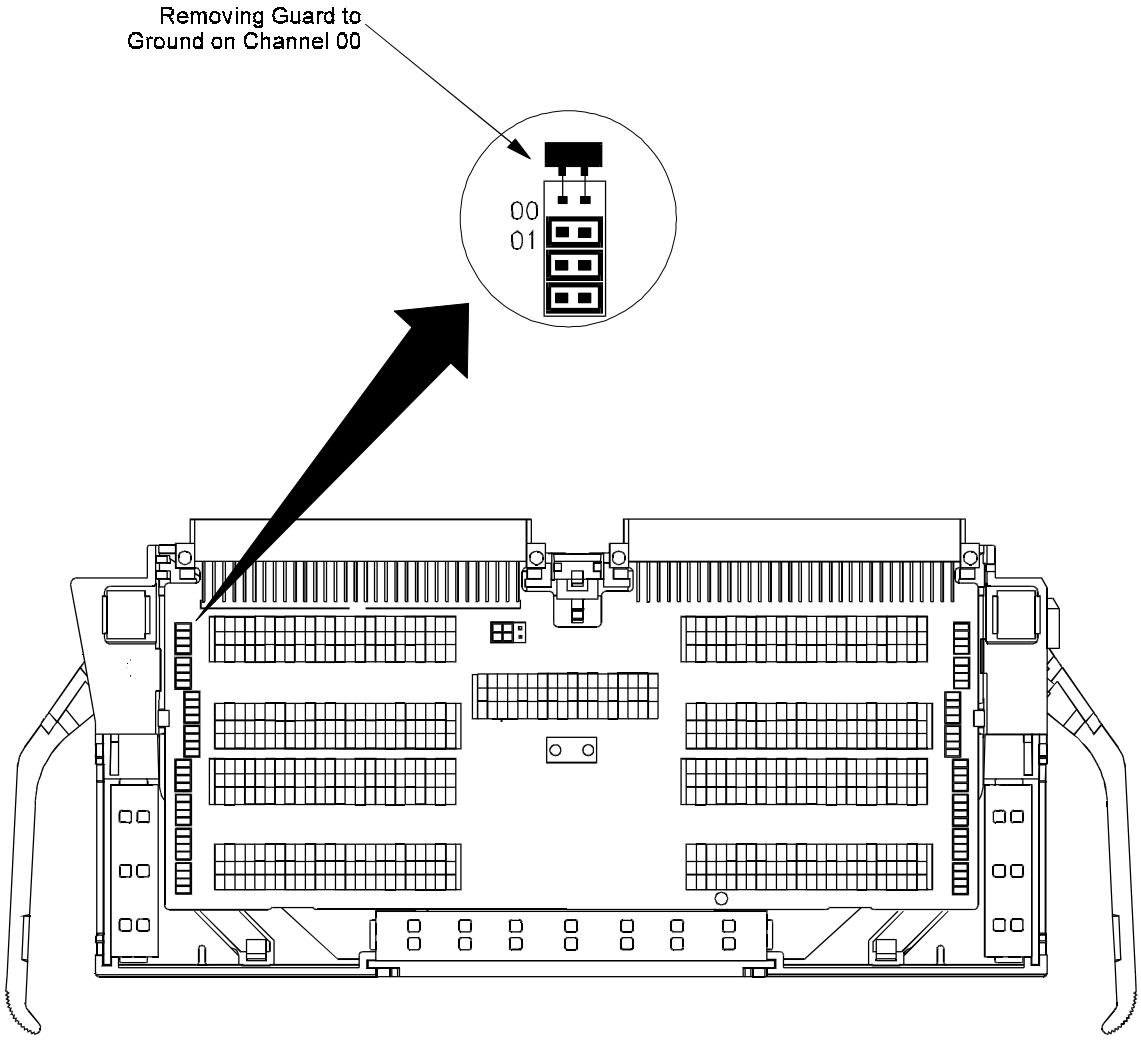


Figure 2-11 Grounding Option 12 Guard Terminals

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Wiring and Attaching the Terminal Module

Figures 2-12 and 2-13 show how to open, wire, and attach the terminal module to an HP E1419.

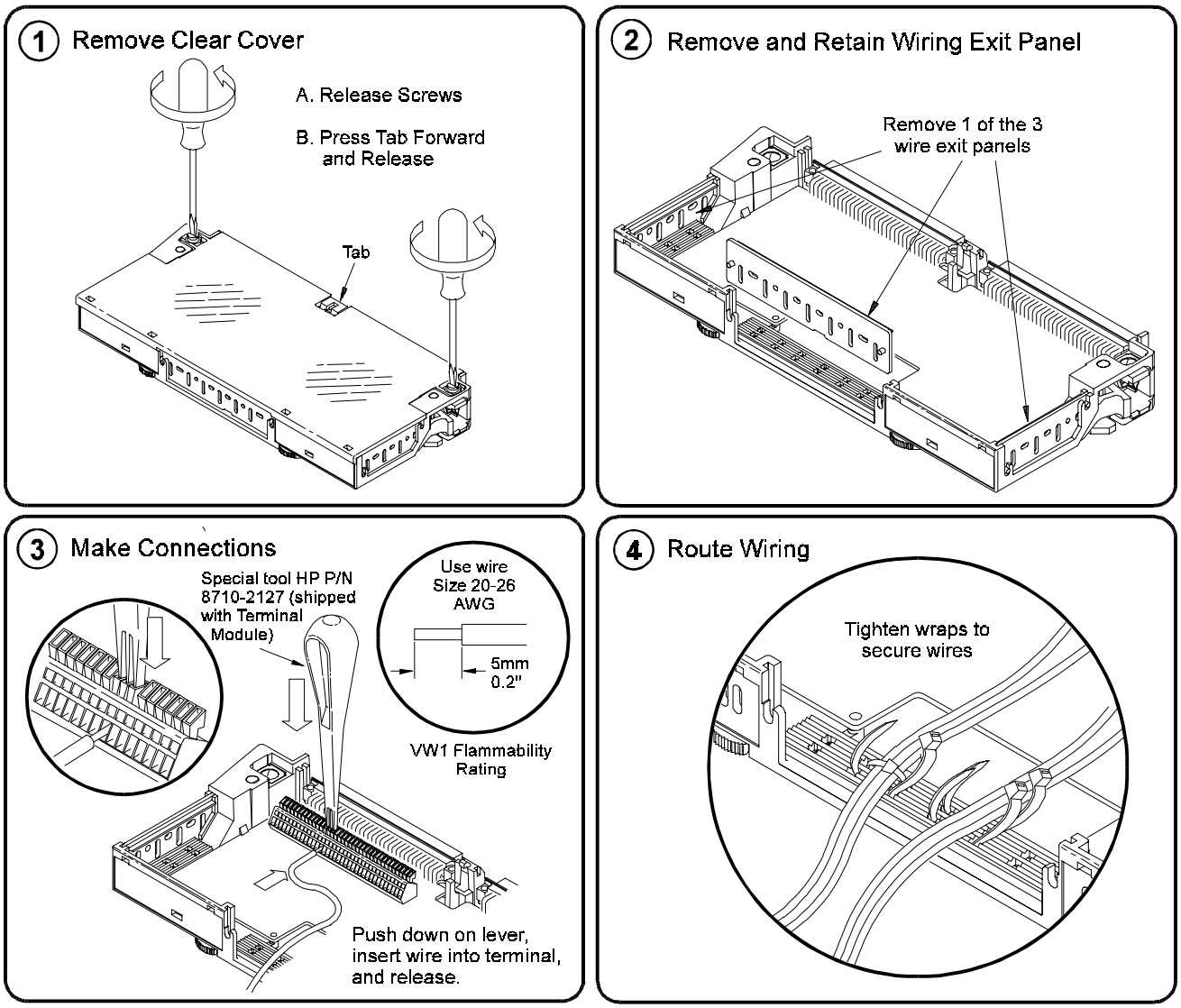


Figure 2-12 Wiring and Connecting the E1419 Terminal Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

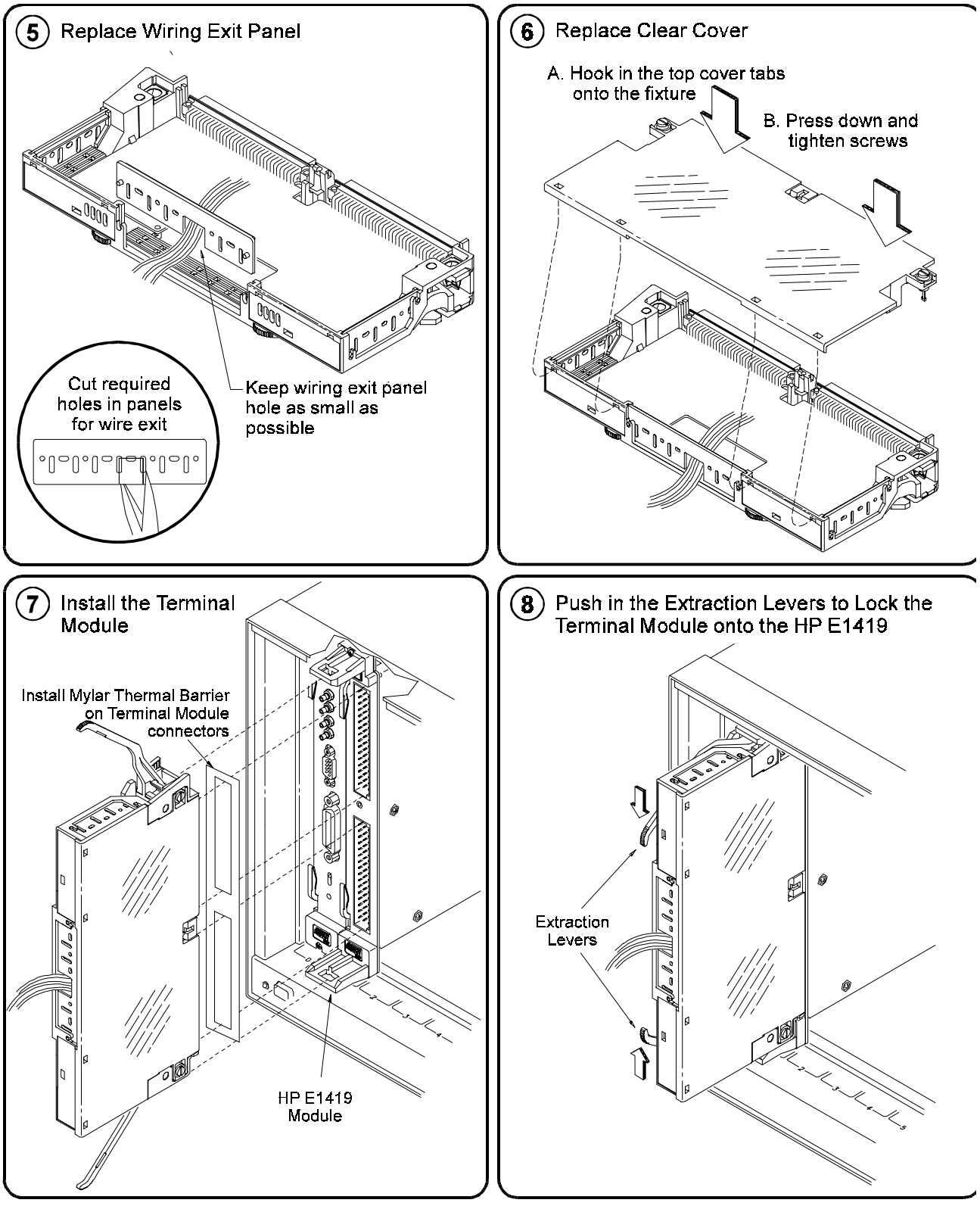


Figure 2-13 Wiring and Connecting the E1419 Terminal Module (Cont.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Attaching/Removing the HP E1419 Terminal Module

Figure 2-14 shows how to attach the terminal module to the HP E1419 and Figure 2-15 shows how to remove it.

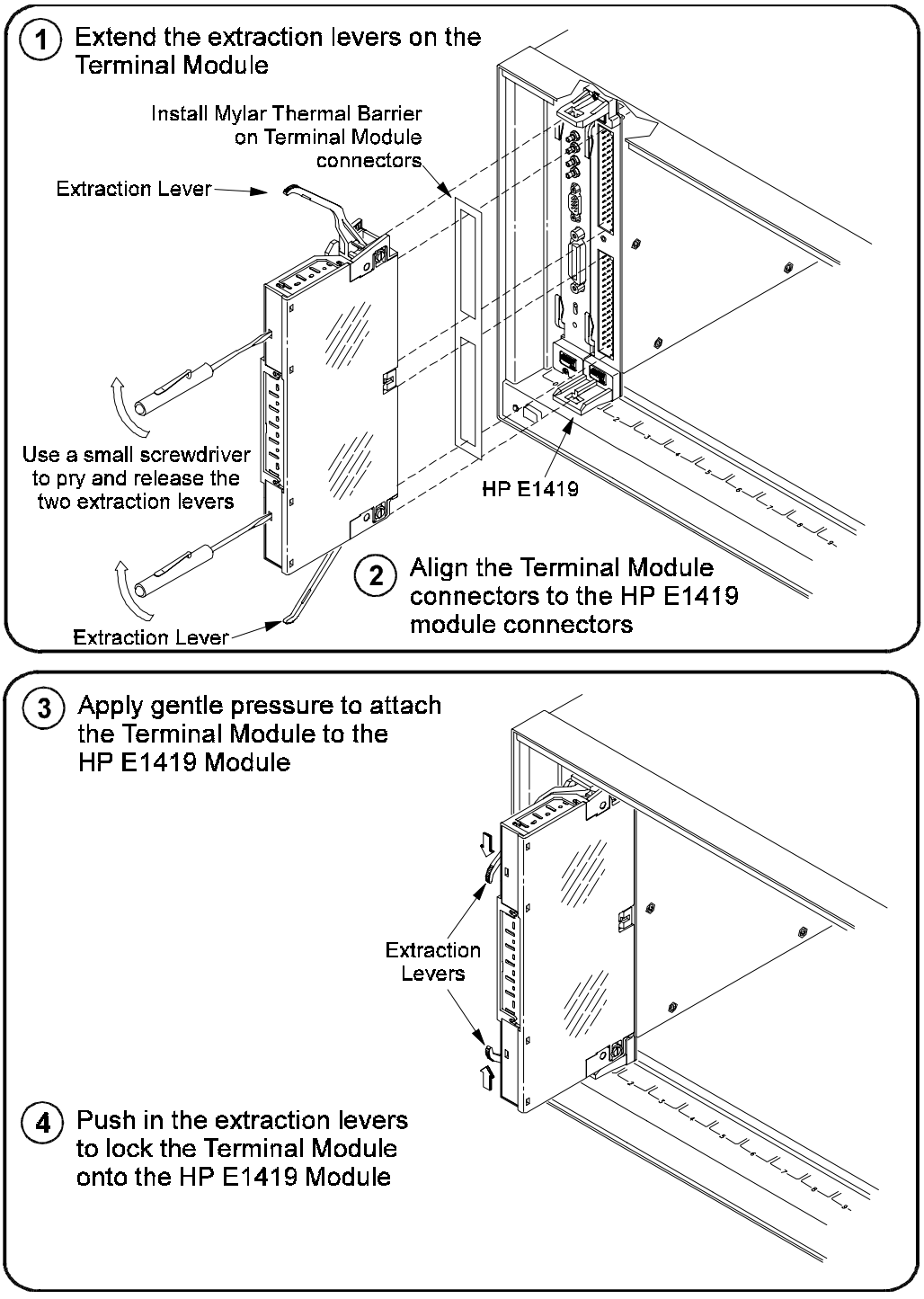


Figure 2-14 Attaching the HP E1419 Terminal Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

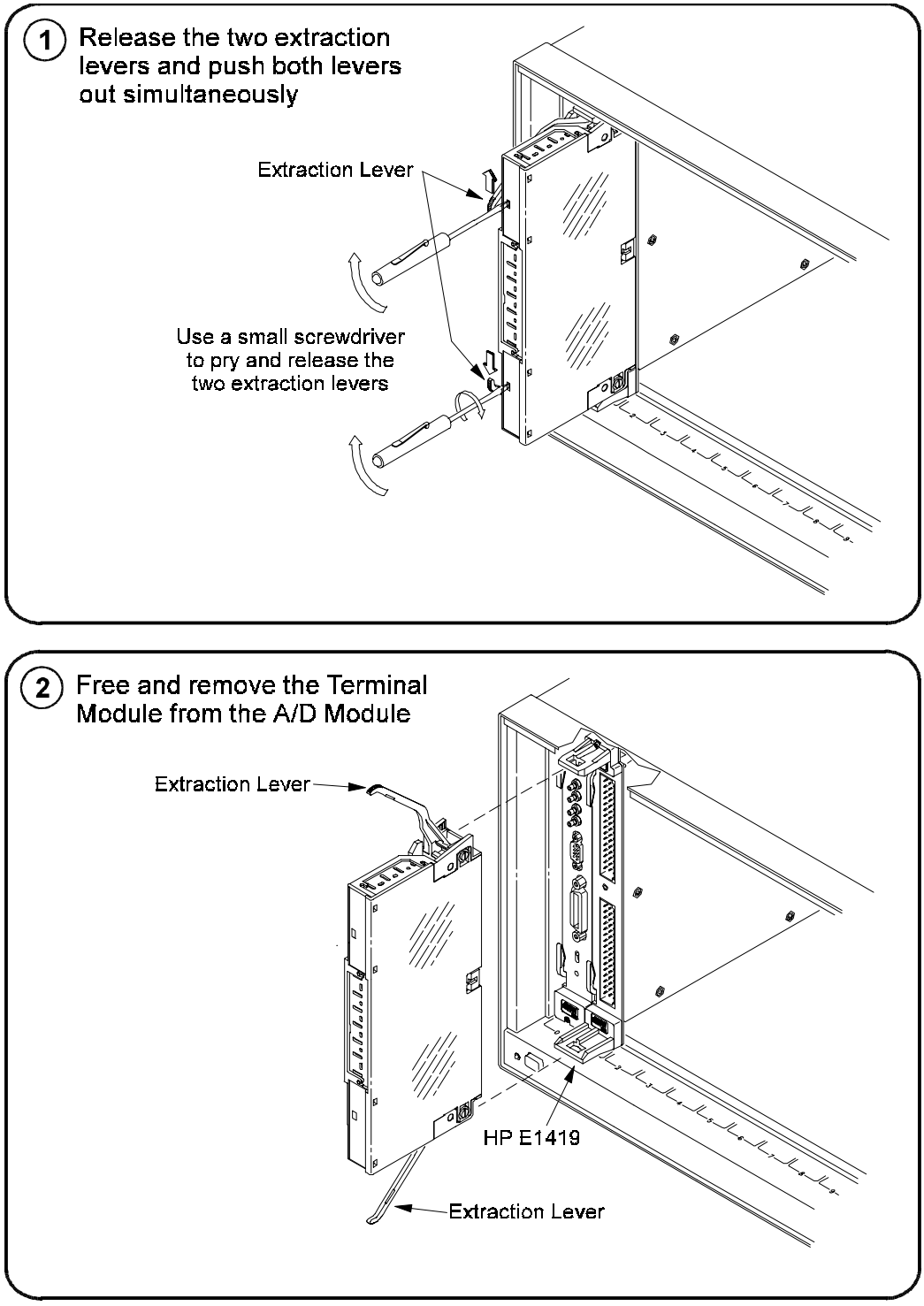


Figure 2-15 Removing the HP E1419 Terminal Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Adding Components to the Option 12 Terminal Module

The back of the terminal module P.C. board provides surface mount pads which you can use to add serial and parallel components to any channel's signal path. Figure 2-16 shows additional component locator information (see the schematic and pad layout information on the back of the teminal module P.C. board). Figure 2-17 shows some usage example schematics.

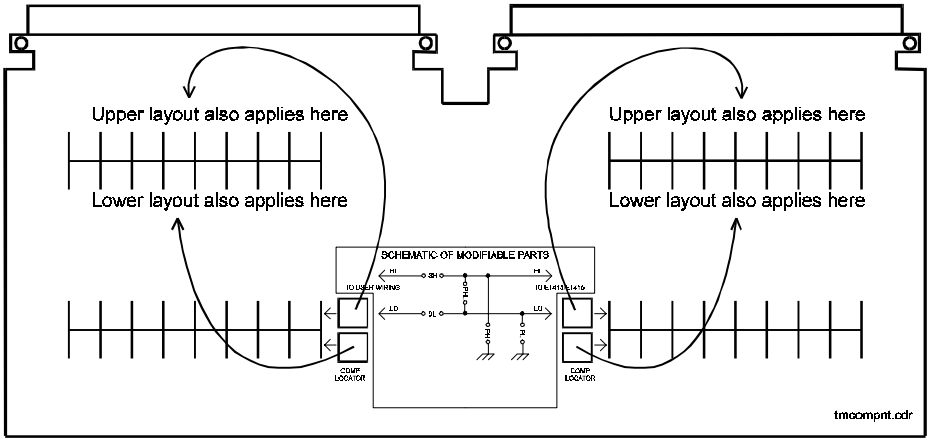


Figure 2-16 Additional Component Location Information

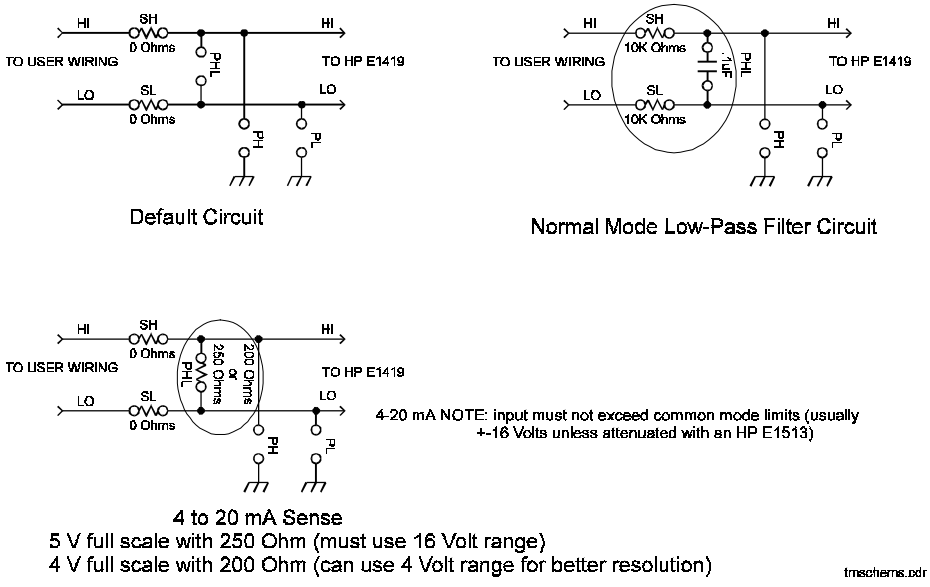


Figure 2-17 Series & Parallel Component Examples

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Option 11 Terminal Module Wiring Map

Figure 2-0 shows the Terminal Module map for the HP E1419.

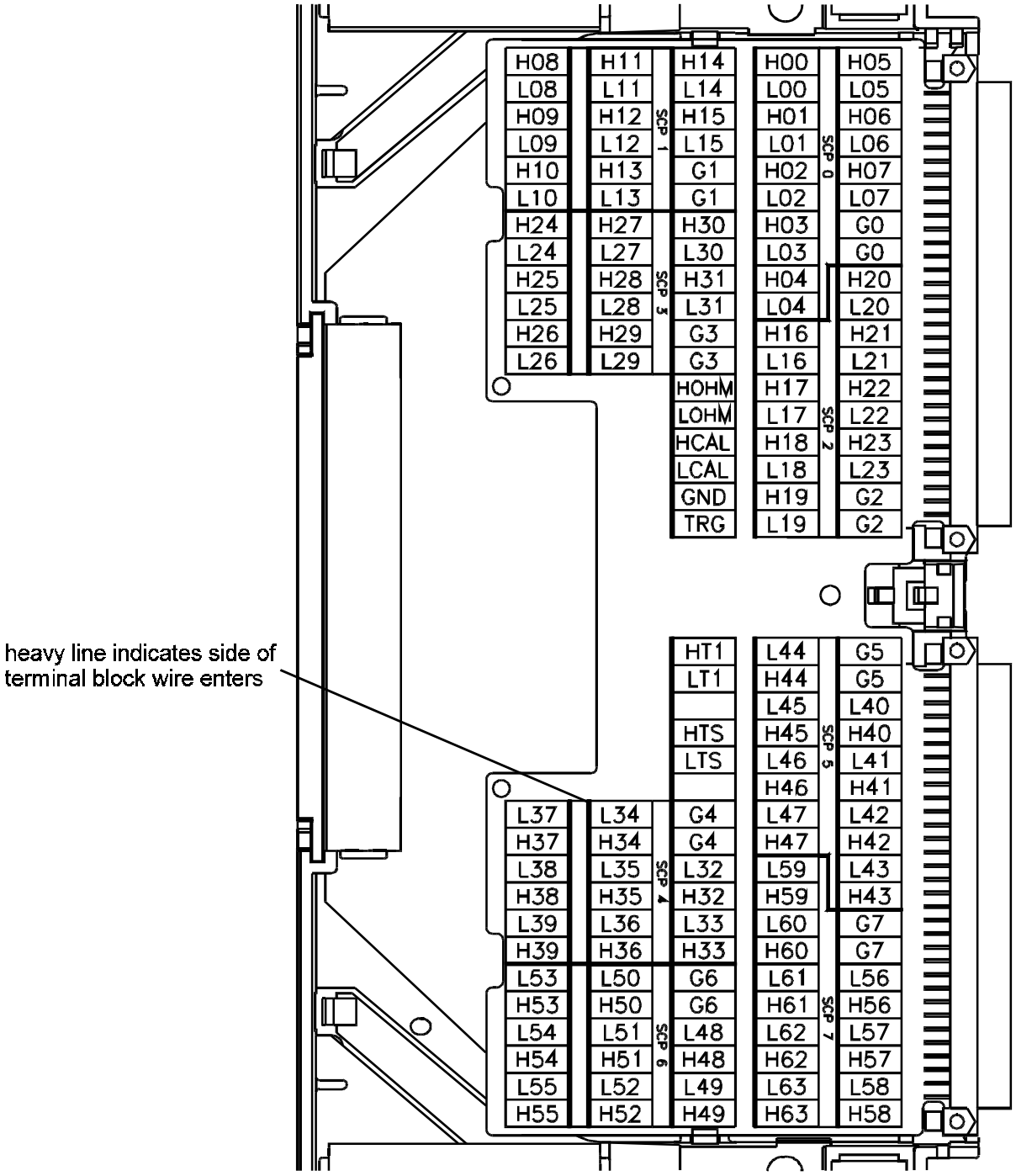


Figure 2-18 HP E1419A Option 11 Terminal Module Map

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Option 12 Terminal Module Wiring Map

Figure 2-0 shows the Terminal Module map for the HP E1419.

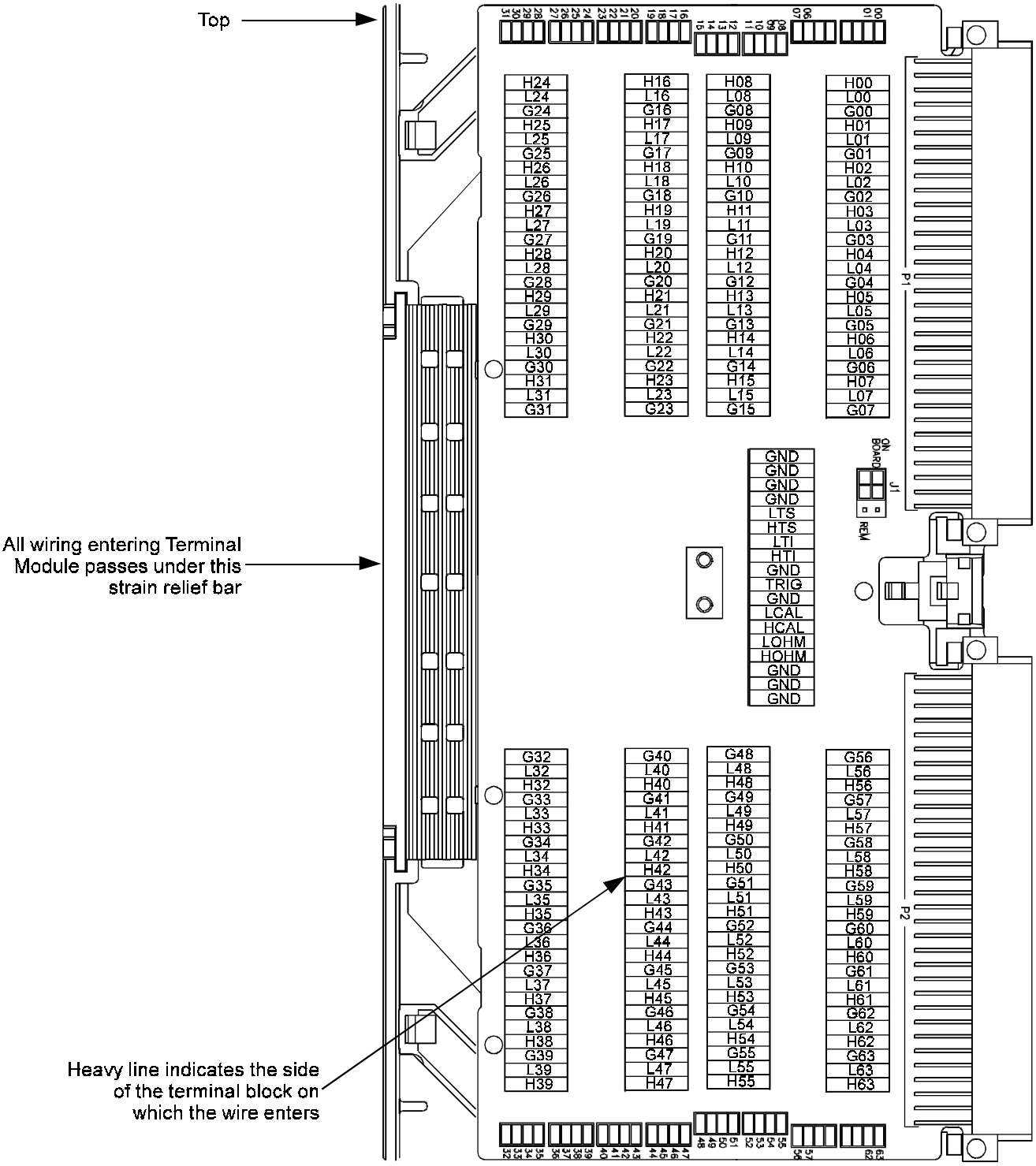


Figure 2-19 HP E1419A Option 12 Terminal Module Map

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Option A3F

Option A3F allows an HP E1419 to be connected to an HP E1586 Rack Mount Terminal Panel. The option provides 4 SCSI plugs on a Terminal Module to make connections to the Rack Mount Terminal Panel using 4 separately ordered SCSI cables. Option A3F Is shown in Figure 2-0.

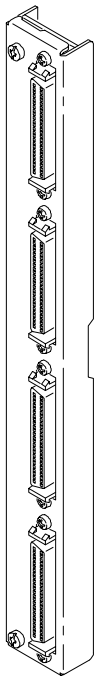


Figure 2-20 Option A3F

Rack Mount Terminal Panel Accessories

There are two different cables available to connect the HP E1586 Rack Mount Terminal Panel to the HP E1419 Option A3F. In both cases, four cables are required if all 64-channels are needed. These cables do not come with the HP E1419A Option A3F and must be ordered separately.

Standard Cable

This cable (HP E1588A) is a 16-channel twisted pair cable with an outer shield. This cable is suitable for relatively short cable runs.

Custom Length Cable

This cable (HP Z2220A Option 050) is available in custom lengths. It is a 16-channel twisted pair cable with each twisted pair individually shielded to provide better quality shielding for longer cable runs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HF Common Mode Filters

Optional High Frequency Common Mode Filters are on the HP E1586 Rack Mount Terminal Panel's input channels (HP E1586 Option 001, RF Filters). They filter out AC common mode signals present in the cable that connects between the terminal panel and the device under test. The filters are useful for filtering out small common mode signals below 5 Vp-p. To order these filters, order HP E1586 Option 001.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 3
Programming the HP E1419A Multifunction*Plus*

About This Chapter

The focus of this chapter is to show the programming model of the HP E1419 Multifunction*Plus* Data Acquisition and Control System. You will be introduced to the concept of configuring the E1419 using SCPI, organizing 'C' programs that execute directly on the E1419 VXI card, using those 'C' programs to make high-speed decisions, and acquiring data from the E1419's sophisticated FIFO and Current Value Table to display within the powerful HP VEE graphical environment. This chapter contains:

- Overview of the HP E1419A Multifunction*Plus* 48
  - Operatinging Model 52
- Executing the Programming Model 53
  - Programming Overview Diagram 56
  - Setting up Analog Input and Output Channels 57
    - Configuring Programmable SCP Parameters 57
    - Linking Input Channels to EU Conversion 59
    - Linking Output Channels to Functions 67
  - Setting up Digital Input and Output Channels 67
    - Digital Input Channels 67
    - Digital Output Channels 68
  - Performing Channel Calibration (Important!) 72
  - Defining C Language Algorithms 74
  - Pre-setting Algorithm variables and coefficients 75
  - Defining Data Storage 76
    - Specifying the Data Format 76
    - Selecting the FIFO Mode 77
  - Setting up the Trigger System 78
    - Arm and Trigger Sources 78
    - Programming the Trigger Timer 80
  - INITiating/Running Algorithms 81
  - Retrieving Algorithm Data 83
    - Reading Algorithm Variables 83
  - Modifying Algorithm Variables 86
    - Updating Algorithm Variables 86
    - Enabling/Disabling Algorithms 87
    - Setting Algorithm Execution Frequency 88
- Using the Status System 90
- HP E1419A Background Operation 96
- Updating the Status System and VXI Interrupts 97
- Creating and Loading Custom EU Tables 98
- Compensating for System Offsets 100
- Detecting Open Transducers 103
- More on Auto Ranging 104
- Settling Characteristics 104

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Overview of the HP E1419 Multifunction*Plus*

This section describes how the HP E1419 gathers input data, executes its 'C' algorithms, and sends its output data. Figure 3-1 shows a simplied functional block diagram.

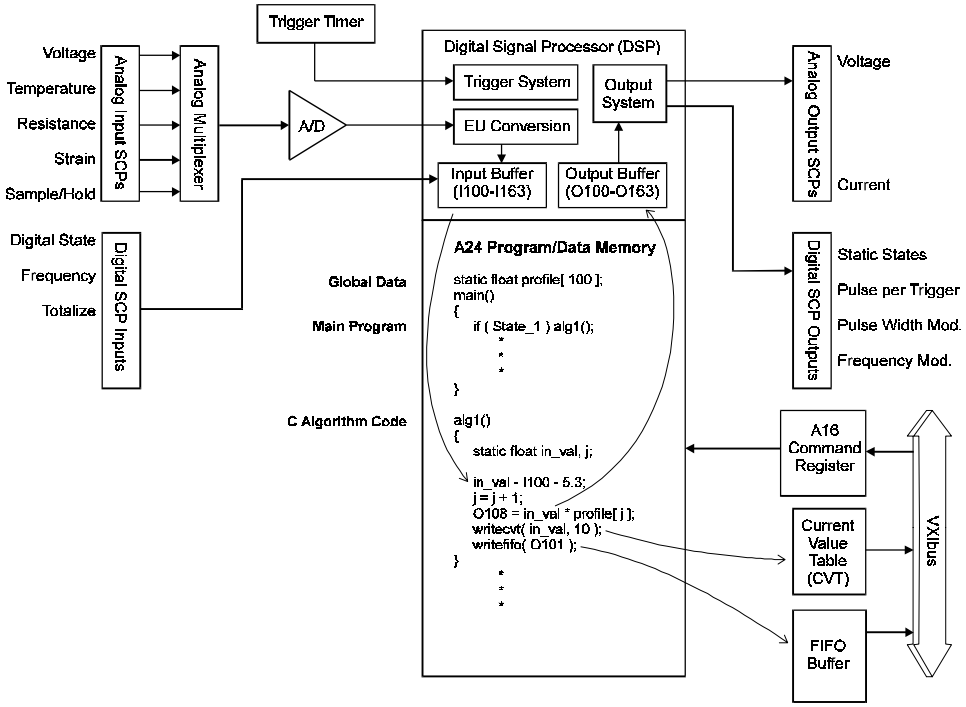


Figure 3-1 Simplified Functional Block Diagram

Multifunction*Plus*?

The HP E1419 is a complete data acqusition and control system on a single VXI card. It is multifunction because it uses the Signal Conditioning Plug-on(SCP) concept whereby you can mix and match your analog input/output and digital input/output channels to meet various application needs. It is Multifunction*Plus* because it has local intelligence to permit the card to run stand-alone with very little interaction required with the supervisory computer.

The HP E1419 has eight SCP slots with each SCP slot capable of addressing up to 8 channels of input or output channels for a total of 64 channels. The first 4 SCP slots(which represent channels numbers 100-131) can mix and match any non-programmable analog input SCP such as fixed gain, fixed filter, straight-through, etc. The standard configuration of the HP E1419 is four straight-through E1501 SCP's that provide high-level signal input capabilities. The remaining the 4 SCP slots can be used for any of the twenty-plus analog/digital SCP's available for the E1419 which cover most data acquisition and control needs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You configure the input and output SCP's with the SCPI programming language. Analog SCP's are measured with the E1419's A/D. Configuring the analog SCP's includes specifying what type of Engineering Unit(EU) conversion you want for each analog input channel. For example, one channel may require a type T thermocouple conversion and another may be a resistance measurement. The on-board Digital Signal Processor(DSP) converts the voltage read across the analog input channel and applies a high-speed conversion which results in temperature, resistance, etc. Digital input SCP's perform their own conversions as configured by the SCPI language.

When the Trigger System is configured and either generates its own trigger or accepts a trigger from an external source, all digital input SCP's latch their current input state and the A/D starts scanning the analog channels. All measurement data is represented as 32-bit real numbers even if the input channel is inherently integer. The EU-converted numbers such as temperature, strain, resistance, volts, state, frequency, etc. are stored in an Input Buffer and later accessed by 'C' programs executing on the E1419 card. Approximately 2000 lines of user-written 'C' code can be downloaded into the E1419's memory and can be split among up to 32 algorithms. HP refers to these as algorithms because an algorithm is a step-by-step procedure for solving some problem or accomplishing some end. Though the documentation continues to refer to the 'C' code as algorithms, you may think in traditional terms as each algorithm representing a 'C' function with a main() program which calls them.

The user-written 'C' algorithms execute after all analog/digital inputs have been stored in the Input Buffer. The 'C' code accesses the measurement data like constants with the names of I100-I163 representing the 32-bit real EU-converted numbers. As seen in Figure 3-1, the algorithms have access to both local and global variables and arrays. The I-variables are inherently global and accessible by any algorithm. Local variables are only visible to the particular algorithm(just like in 'C' functions). Declared global variables can be shared by any algorithm.

HP VEE can read or write any local or global variable in any algorithm by using SCPI syntax that actually identifies the variable by name, but a more efficient means of reading data is available through the E1419's FIFO and Current Value Table(CVT). As seen in Figure 3-1, any algorithm can write any expression or constant to the FIFO/CVT. HP VEE can then read the FIFO/CVT to characterize what's happening inside the E1419 and to provide an operator view of any input/output channel, variable, or constant.

Output SCP's derive their channel values from O-variables that are written by the algorithms. O100-O163 are read/write global variables that are read after all algorithms have finished executing. The 32-bit real values are converted to the appropriate units as defined by the SCPI configuration commands and written to the various output SCP's by channel number.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

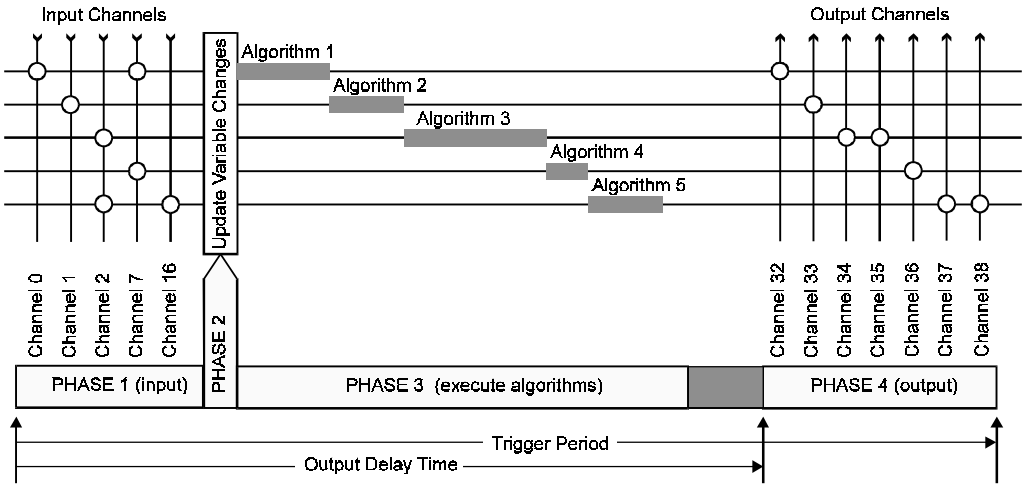


Figure 3-2 HP E1419A Cycle Phases

Figure 3-2 illustrates the timing of all these operations and describes the E1419's input-update-execute algorithms-output phases. This cycle-based design is desirable because it results in deterministic operation of the E1419. That is, the input channels are always scanned, and the output channels are always written at pre-defined intervals. Note too that any number of input channels or output channels are accessible by any of up to 32 user-written algorithms. The algorithms are named ALG1-ALG32 and execute in numerical order.

Notice the Update Window (phase 2) illustrated in Figure 3-2. This window has a user-specified length and is used to accept and make changes to local and global variables from the supervisory computer. Up to 512 scalar or array changes can be made while executing algorithms. Special care was taken to make sure all changes take place at the same time so that any particular algorithm or group of algorithms all operate on the new changes at a user-specified time. This does not mean that all scalar and array changes have to be received during one cycle to become effective at the next cycle. On the contrary, it may take a number of cycles to download new values, especially when trying to re-write 1024 element arrays and especially when the trigger cycle time is very short.

There are multiple times between the base triggers where scalar and array changes can be accepted from the supervisory computer, and these changes are held in a holding buffer until the supervisory computer instructs the changes to take effect. These changes then take place during the Update window and take effect BEFORE algorithms start executing. The "do-update-now" signal can be sent by command(ALG:UPD) or by a

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

change in a digital input state(ALG:UPD:CHAN). In either case, the programmer has control over when the new changes take effect.

The E1419's ability to execute programs directly on the card and its fast execution speed give the programmer real-time response to changing conditions. And, programming the card has been made very easy to understand. HP chose C as the language used to write user programs since that language is already considered the industry standard. Choosing C allows you to write algorithms on PC's or UNIX workstations that have C compilers, so you can debug algorithms before execution on the card. The E1419 also provides good debugging tools that permit you to determine worst-case execution speed, monitor variables while running, and selectively enable/disable any of the E1419's 32 algorithms.

HP created a limited and simplified version of C since most applications need only basic operations: add, subtract, multiply, divide, scalar variables, arrays, and programming constructs. The programming constructs are limited to if-then-else to allow conditional evaluation and response to input changes. Since all algorithms have an opportunity to execute after each time-base trigger, the if-then-else constructs permit conditional skipping of cycle intervals so that some code segments or algorithms can execute at multiples of the cycle time instead of every cycle.

Looping constructs such as for or while are purposely left out of the language so that user programs are deterministic. Note that looping is not really needed for most applications since the cycle interval execution(via the trigger system) of every algorithm has inherent repeat looping. With no language looping constructs, the HP E1419's C compiler can perform a worst-case branch analysis of user programs and return the execution time for determining the minimum time-base interval. Making this timing query available allows the programmer to know exactly how much time may be required to execute any/all phases before attempting to set up physical test conditions.

Note the darker shaded portion at the end of the Execute Algorithms Phase in Figure 3-2 . The conditional execution of code can cause the length of this phase to move back and forth like an accordion. This can cause undesirable output jitter when the beginning of the output phase starts immediately after the last user algorithm executes. The HP E1419's design allows the user to specify when output signals begin relative to the start of the trigger cycle. Outputs then always occur at the same time, every time.

The programming task is further made easy with this design because all the difficult structure of handling input and output channels is done automatically. This is not true of many other products that may have several ways to acquire measurement data or write results to its I/O channels. When the E1419's user-written C algorithms are compiled, input channels and

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

output channels are detected in the algorithms and are automatically grouped and configured for the Input and Output phases as seen in Figure 3-2 . Each algorithm simply accesses input channels as variables and writes to output channels as variables. The rest is handled and optimized by the Input and Output phases. You're left to think of solving your application in terms of input and output values variables rather than worrying about how to deal with each SCP's differences.

Operating Model

The E1419 card operates in one or two states: either the "idle" state or the "running" state. The "idle" can be referred to as "Before INIT" and the "running" state can be referred to as "After INIT". See Figure 3-3 for the following discussion.

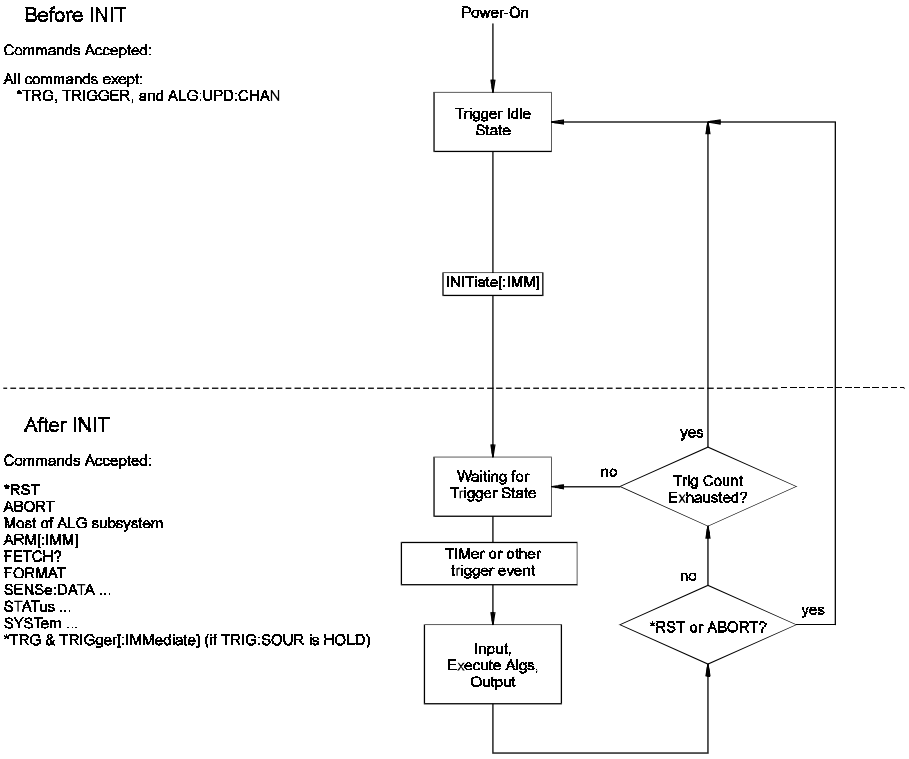


Figure 3-3 Module States

"Before INIT" positions the card in the Trigger Idle State, and its DSP chip is ready to accept virtually all SCPI commands. This is the time when configuring and set up operations are performed. This would include linking Engineering Unit conversions to channels, designating digital input/output channels, downloading algorithms, etc.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

"After INIT" (and when trigger events are taking place), the DSP is busy measuring input channels, executing algorithms, and updating outputs. However, there are periods of time between trigger events where the DSP is waiting for I/O or just waiting for the next trigger event. This time is utilized to accept a limited command set from the supervisory program(HP VEE, for example) to change scalars, arrays/elements, or to download new algorithms. HP VEE communicates with the E1419's driver, and the driver then interfaces with the DSP, FIFO, CVT, etc., in cooperation with the operating state. The "When Accepted:" comments in the Command Reference Chapter specify which commands may be accepted before or after INIT.

Executing The Programming Model

This section shows the sequence of programming steps that should be used for the HP E1419. Within each step, most of the available choices are shown using example command sequences. Further details about various SCPI commands can be found in the Command Reference Chapter 6. A "command sequence" example can be found on page 85 of this chapter. Many HP VEE programming examples can be found in Chapter 5.

Important

Most programming difficulties can be resolved if you know what's wrong. It is very important while developing your application that you include error checking at least at the end of each major programming step by using the SYST:ERR? query command. More often may be desirable for complex sequences of commands in any particular step.

Power-on and *RST Default Settings

Some of the programming operations that follow may already be set after Power-ON or after a *RST command. Where these default settings coincide with the configuration settings you require, you do not need to explicitly execute a command to set them. These are the default settings:

- No algorithms are defined and therefore no channels will be scanned
- Programmable SCP's are configured to their Power-ON defaults (see the SCP's manual for these defaults)
- All analog input channels are linked to EU conversion for voltage
- All non-isolated digital I/O channels are set to input static digital state
- HP E1536 Isolated digital I/O channels are switch configured and wake up as such
- Trigger subsystem set to: ARM:SOURCE IMM, TRIG:SOUR TIMER, TRIG:COUNT INFinite, and TRIG:TIMER 0.010.
- FIFO/CVT data returned in ASCII format
- FIFO set to BLOCking mode to disallow overwriting of unread data

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 3-4 shows a comprehensive step-by-step programming sequence that may be required for your application. As stated earlier, many of these steps need only minimal attention since the most common configurations are defaulted for you at power-ON or *RST. Figure 3-5 shows a block diagram of the HP E1419A with the numbered programming steps and various SCPI commands associated with those steps.

Keep in mind that the first four SCP positions (0 through 3) can only be configured with non-programmable SCP's. SCP's with programmable gain/filter, digital input/output, analog output, strain gage, etc., are NOT supported in these slots. This restriction encompasses channels 100-131.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

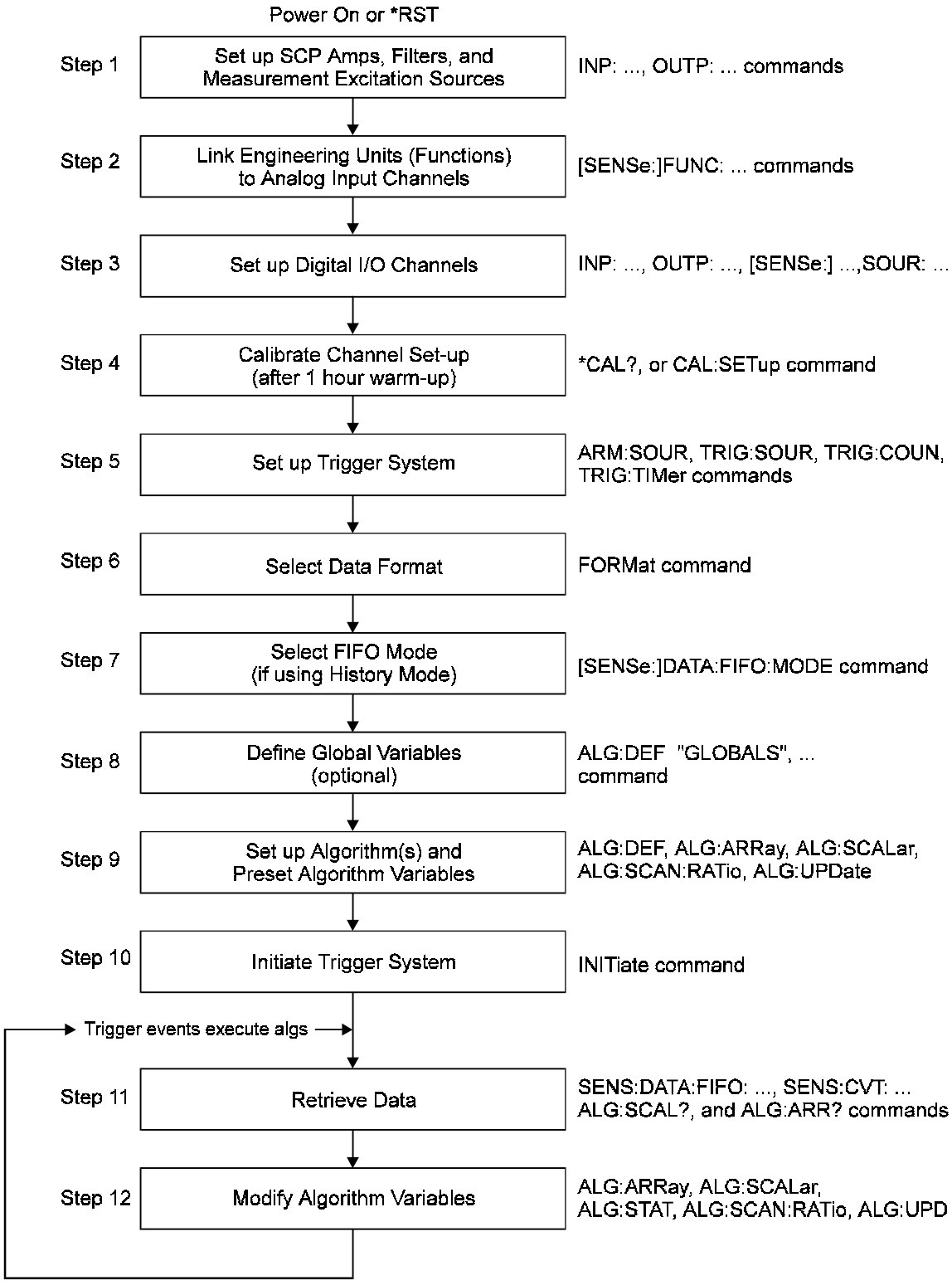
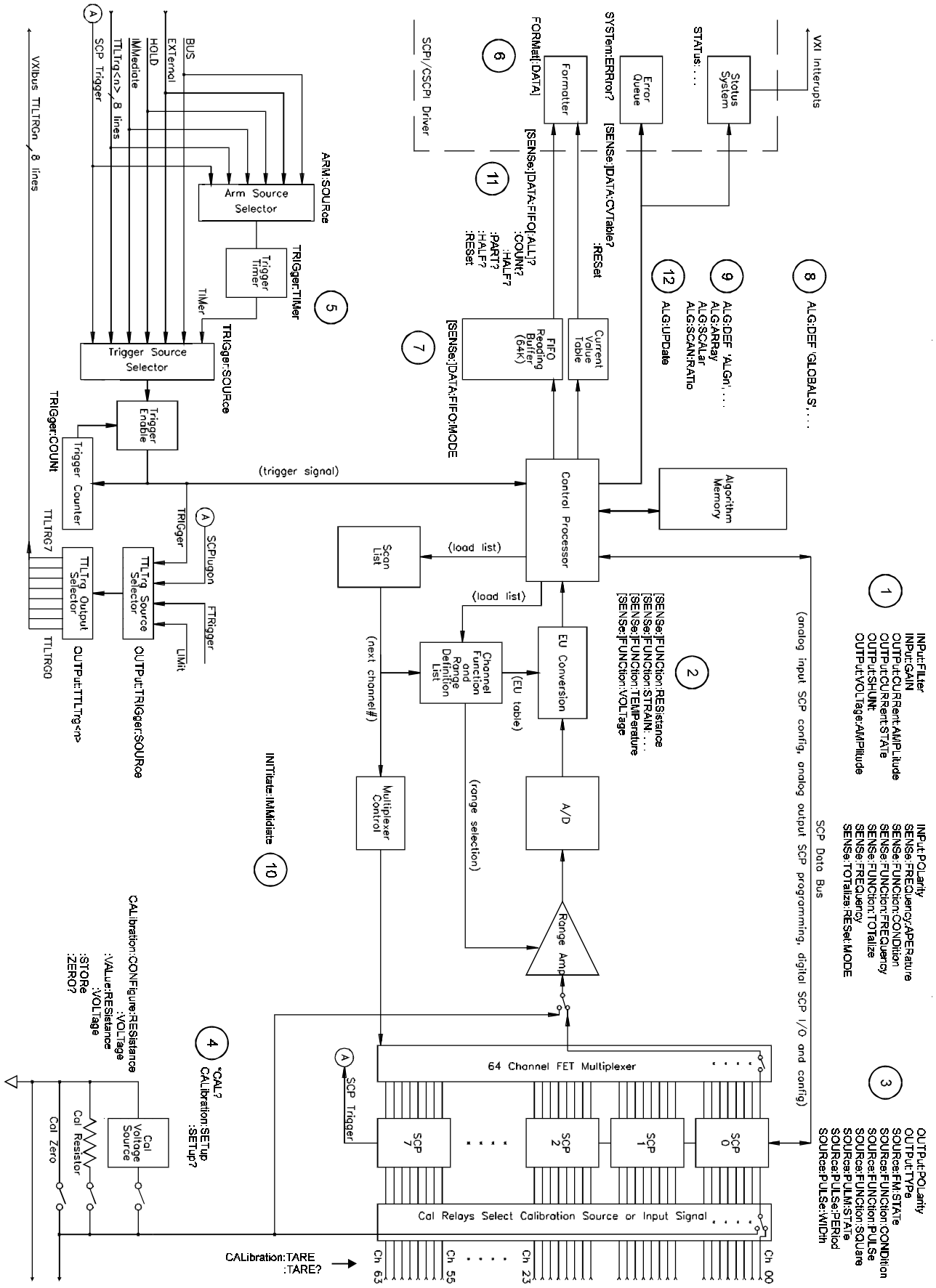


Figure 3-4 Programming Sequence

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 3-5 Programming Model Block Diagram



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Setting up Analog Input and Output Channels

This section covers configuring input and output channels to provide the measurement values and output characteristics that your algorithms need to operate.

Configuring Programmable Analog SCP Parameters

This step applies only to programmable Signal Conditioning Plug-ons such as the HP E1503 Programmable Amplifier/Filter SCP, the HP E1505 Current Source SCP, the HP E1518 Resistance Measurement SCP, the HP E1510 Sample and Hold SCP, and the HP E1511 Transient Strain SCP. See the particular SCP's User's manual to determine the gain, filter cutoff frequency, or excitation amplitude selections that it may provide.

Note The HP E1419A only supports these programmable analog SCPs on SCP positions 4 through 7.

Setting SCP Gains

An important thing to understand about input amplifier SCPs is that given a fixed input value at a channel, changes in channel gain do not change the value your algorithm will receive from that channel. The DSP chip (Digital Signal Processor) keeps track of SCP gain and Range Amplifier settings, and "calculates" a value that reflects the signal level at the input terminal. The only time this in not true is when the SCP gain chosen would cause the output of the SCP amplifier to be too great for the selected A/D range. As an example; with SCP gain set to 64, an input signal greater than ±0.25 volts would cause an over-range reading even with the A/D set to its 16 volt range.

The gain command for SCPs with programmable amplifiers is:

INPut:GAIN <*gain*>,(@<*ch_list*>) to select SCP channel gain.

The gain selections provided by the SCP can be assigned to any channel individually or in groups. Send a separate command for each gain selection. An example for the HP E1503 programmable Amp&Filter SCP:

To set the SCP gain to 8 for channels 40, 44, 46, and 48 through 51 send:

```
INP:GAIN  8,(@140,144,146,148:151)
```

To set the SCP gain to 16 for channels 56 through 59, and to 64 for channels 60 through 63 send:

```
INP:GAIN  16,(@156:159)
INP:GAIN  64,(@160:163)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Setting Filter Cutoff Frequency

The commands for programmable filters are:

INPut:FILTer[:LPASs]:FREQuency <*cutoff_freq*>,(@<*ch_list*>) to select cutoff frequency

INPut:FILTer[:LPASs][:STATe] ON | OFF,(@<*ch_list*> to enable or disable input filtering

The cutoff frequency selections provided by the SCP can be assigned to any channel individually or in groups. Send a separate command for each frequency selection. For example:

To set 10 Hz cutoff for channels 40, 44, 46, and 48 through 51 send:

```
INP:FILT:FREQ  10,(@140,144,146,148:151)
```

To set 10 Hz cutoff for channels 56 through 59, and 100 Hz cutoff for channels 60 through 633 send:

```
INP:FILT:FREQ  10,(@156:159)
INP:FILT:FREQ  100,(@160:163)
```

By default (after *RST or at power-on) the filters are enabled (ON). To disable or re-enable individual (or all) channels, use the INP:FILT ON | OFF, (@<ch_list>) command. For example, to program filters for channels 56 and 57 off, send:

```
INP:FILT:STAT  Off,(@156:157)
```

Setting the HP E1505 and HP E1518 Current Source SCPs

The Current Source SCP supplies excitation current for resistance type measurements. These include resistance, and temperature measurements using resistance temperature sensors. The commands to control the HP E1505 Current Source and HP E1518 Resistance Measurement SCPs are:
OUTPut:CURRent:AMPLitude <*amplitude*>,(@<*ch_list*>) and OUTPut:CURRent[:STATe] <*enable*>.

- The *amplitude* parameter sets the current output level. It is specified in units of Amps DC and can take on the values 30e-6 (or MIN), and 488e-6 (or MAX). Select 488µA for measuring resistances of less than 8,000 Ohms. Select 30µA for resistances of 8,000 Ohms and above.
- The *ch_list* parameter specifies the Current Source SCP channels that will be set.

To set channels 40 and 41 to output 30 µA and channels 42 and 43 to output 488 µA:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
OUTP:CURR  30e-6,(@140,141)
OUTP:CURR  488e-6,(@142,143)
```
separate command per output level

Notes

1. The OUTPut:CURRent:AMPLitude command is only for programming excitation current used in resistance measurement configurations. It is does not program output DAC SCPs like the HP E1532.
2. The HP E1518 Current Measurement SCP is a combinaation of 4 channels of current source (same as the HP E1505), and 4 channels of amplified analog input (same as the HP E1508). The current source channels are on the lower four channels of the HP E1518.

Setting the HP E1511 Strain Bridge SCP Excitation Voltage

The HP E1511 Strain Bridge Completion SCP has a programmable bridge excitation voltage source. The command to control the excitation supply is OUTPut:VOLTage:AMPLitude <*amplitude*>,(@<*ch_list*>)

- The <*amplitude*> parameter can specify 0, 1, 2, 5, or 10 volts for the HP E1511's excitation voltage.
- The <*ch_list*> parameter specifies the SCP and bridge channel excitation supply that will be programmed. There are four excitation supplies in each HP E1511.

To set the excitation supplies for channels 40 through 43 to output 2 volts:

```
OUTP:VOLT:AMPL  2,(@140:143)
```

NOTE

The OUTPut:VOLTage:AMPLitude command is only for programming excitation voltage used measurement configurations. It is does not program output DAC SCPs like the HP E1531 and HP E1537.

Linking Channels to EU Conversion

This step links each of the module's channels to a specific measurement type. For analog input channels this "tells" the on-board control processor which EU conversion to apply to the value read on any channel. The processor is creating a list of conversion types vs. channel numbers.

The commands for linking EU conversion to channels are:

[SENSe:]FUNCtion:RESistance <*excite_current*>,[<*range*>,](@<*ch_list*>)
for resistance measurements

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]FUNCtion:STRain:… <*excite_current*>,[<*range*>,](@<*ch_list*>) for strain bridge measurements

[SENSe:]FUNCtion:TEMPerature <*type*>,<*sub_type*>,[<*range*>,](@<*ch_list*>) for temperature measurements with thermocouples, thermistors, or RTDs

[SENSe:]FUNCtion:VOLTage <*range*>,(@<*ch_list*>) for voltage measurements

[SENSe:]FUNCtion:CUSTom <*range*>,(@<*ch_list*>) for custom EU conversions.

NOTE At Power-on and after *RST, the default EU Conversion is autorange voltage for analog input channels.

Linking Voltage Measurements

To link channels to the voltage conversion send the [SENSe:]FUNCtion:VOLTage [<*range*>,] (@<*ch_list*>) command.

- The *ch_list* parameter specifies which channels to link to the voltage EU conversion.
- The optional *range* parameter can be used to choose a fixed A/D range. Valid values are: .0625, .25, 1, 4, 16, or AUTO. When not specified, the module uses auto-range (AUTO).

To set channels 0 through 15 to measure voltage using auto-range:

```
SENS:FUNC:VOLT  AUTO,(@100:115)
```

To set channels 0 and 23 to the 16 volt range, and 28 through 31 to the .0625 volt range:

```
SENS:FUNC:VOLT  16,(@100,123)
SENS:FUNC:VOLT  .625,(@128:131)        must send a command per range
```

Note When using manual range in combination with amplifier SCPs, the EU conversion will try to return readings which reflect the value of the input signal. However, it is up to you to choose range values that will provide good measurement performance (avoiding over-ranges and select ranges that provide good resolution based on the input signal). In general, measurements can be made at full speed using auto-range.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Linking Resistance Measurements

To link channels to the resistance EU conversion send the [SENSe:]FUNCtion:RESistance <*excite_current*>,[<*range*>,](@<*ch_list*>) command.

Resistance measurements assume that there is at least one Current Source SCP installed (eight current sources per SCP). See Figure 3-6.

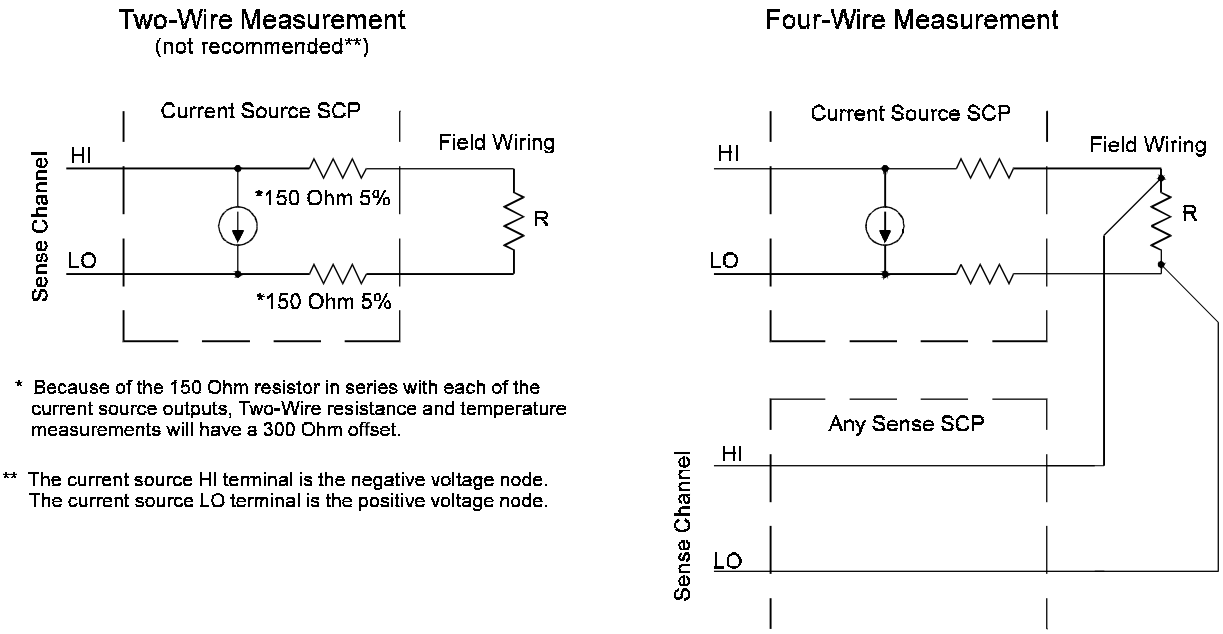


Figure 3-6 Resistance Measurement Sensing

- The *excite_current* parameter is used only to tell the EU conversion what the Current Source SCP channel is now set to. *Excite_current* is specified in Amps DC and the choices for the HP E1505 SCP are 30e-6 (or MIN) and 488e-6 (or MAX). Select 488µA for measuring resistances of less than 8,000 Ohms. Select 30µA for resistances of 8,000 Ohms and above.

- The optional *range* parameter can be used to choose a fixed A/D range. When not specified (defaulted), the module uses auto-range.

- The *ch_list* parameter specifies which channel(s) to link to the resistance EU conversion. These channels will sense the voltage across the unknown resistance. Each can be a Current Source SCP channel (a two-wire resistance measurement) or a sense channel separate from the Current Source SCP channel (a four-wire resistance measurement). See figure 3-6 for diagrams of these measurement connections.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To set channels 0 through 3 to measure resistances greater than 8,000 Ohms and set channels 4 through 7 to measure resistances less than 8K (in this case paired to current source SCP channels 32 through 39):

OUTP:CURR:AMPL 30e-6, (@132:135)

set 4 channels to output 30μA for 8KΩ or greater resistances

SENS:FUNC:RES 30e-6, (@100:103)

link channels 0 through 4 to resistance EU conversion (8KΩ or greater)

OUTP:CURR:AMPL 488e-6, (@136:139)

set 4 channels to output 488μA for less than 8KΩ resistances

SENS:FUNC:RES 488e-6, (@104:107)

link channels 4 through 7 to resistance EU conversion (less than 8KΩ)

Linking Temperature Measurements

To link channels to temperature EU conversion send the [SENSe:]FUNCtion:TEMPerature <*type*>, <*sub_type*>, [<*range*>,](@<*ch_list*>) command.

- The *ch_list* parameter specifies which channel(s) to link to the temperature EU conversion.
- The *type* parameter specifies RTD, THERmistor, or TC (for ThermoCouple)
- The optional *range* parameter can be used to choose a fixed A/D range. When not specified (defaulted), the module uses auto-range.

RTD and Thermistor Measurements

Temperature measurements using resistance type sensors involve all the same considerations as resistance measurements discussed in the previous section. See the discussion of Figure 3-6 in "Linking Resistance Measurements".

For resistance temperature measurements the *sub_type* parameter specifies:

- For RTDs; "85" or "92" (for 100 Ohm RTDs with 0.003**85** or 0.003**92** Ohms/Ohm/Degree C temperature coefficients respectively)
- For Thermistors; 2250, 5000, or 10000 (the nominal value of these devices at 25 degrees C)

NOTES

1. Resistance temperature measurements (RTDs and THERmistors) require the use of Current Source Signal Conditioning Plug-Ons. The following table shows the Current Source setting that must be used for the following RTDs and Thermistors:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Required Current Amplitude	Temperature Sensor Types and Subtypes
MAX (488µA) MIN (30µA)	RTD,85 | 92 and THER,2250 THER,5000 | 10000

1. *sub_type* values of 2250, 5000, and 10000 refer to thermistors that match the Omega 44000 series temperature response curve. These 44000 series thermistors have been selected to match the curve within 0.1 or 0.2°C.

To set channels 0 through 7 to measure temperature using 2,250 Ohm thermistors (in this case paired to current source SCP channels 32 through 39):

```
OUTP:CURR:AMPL 488e-6,(@132:139)
```

set excite current to 488µA on current SCP channels 32 through 39

```
SENS:FUNC:TEMP THER, 2250, (@100:107)
```

link channels 0 through 7 to temperature EU conversion for 2,250Ω thermistor

To set channels 8 through 15 to measure temperature using 10,000 Ohm thermistors (in this case paired to current source SCP channels 40 through 47):

```
OUTP:CURR:AMPL 30e-6,(@140:147)
```

set excite current to 30µA on current SCP channels 40 through 47

```
SENS:FUNC:TEMP THER, 10000, (@108:115)
```

link channels 8 through 15 to temperature EU conversion for 10,000Ω thermistor

To set channel 7 to measure temperature using 100 Ohm RTD with a TC of .00385 Ohm/Ohm/°C (in this case paired to current source SCP channel 39):

```
OUTP:CURR:AMPL 488e-6,(@139)
```

set excite current to 488µA on current SCP channels 32 through 47

```
SENS:FUNC:TEMP RTD, 85, (@107)
```

link channel 7 to temperature EU conversion for 100Ω RTDs with .00385 TC.

Thermocouple Measurements

Thermocouple measurements are voltage measurements that the EU conversion changes into temperature values based on the *sub_type* parameter and latest reference temperature value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- For Thermocouples the *sub_type* parameter can specify CUSTom, E, EEXT, J, K, N, R, S, T (CUSTom is pre-defined as Type K, no reference junction compensation. EEXT is the type E for extended temperatures of 800°F or above).

To set channels 24 through 31 to measure temperature using type E thermocouples:

SENS:FUNC:TEMP TC, E, (@124:131)

(see following section to configure a TC reference measurement)

Thermocouple Reference Temperature Compensation

The isothermal reference temperature is required for thermocouple temperature EU conversions. The Reference Temperature Register must be loaded with the current reference temperature before thermocouple channels are scanned. The Reference Temperature Register can be loaded two ways:

1. By measuring the temperature of an isothermal reference junction during an input scan.
2. By supplying a constant temperature value (that of a controlled temperature reference junction) before a scan is started.

Setting up a Reference Temperature Measurement

This operation requires two commands, the [SENSe:]REFerence command and the [SENSe:]REFerence:CHANnels command.

The [SENSe:]REFerence <*type*>, <*sub_type*>,[<*range*>,](@<*ch_list*>) command links channels to the reference temperature EU conversion.

- The *ch_list* parameter specifies the sense channel that you have connected to the reference temperature sensor.
- The *type* parameter can specify THERmistor, RTD, or CUSTom. THER and RTD, are resistance temperature measurements and use the on-board 122 µA current source for excitation. CUSTom is pre-defined as a Type E thermocouple which has a thermally controlled ice point reference junction.
- The *sub_type* parameter must specify:
  - For RTDs; "85" or "92" (for 100 Ohm RTDs with 0.003**85** or 0.003**92** Ohms/Ohm/Degree C temperature coefficients respectively)
  - For Thermistors; only "5000" (See previous note on page 3-63)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

– For CUSTom; only "1"

- The optional *range* parameter can be used to choose a fixed A/D range. When not specified (defaulted), or set to AUTO, the module uses auto-range.

Reference Measurement Before Thermocouple Measurements

At this point we are going to introduce you to the concept of the HP E1419A's Scan List. As you define each algorithm, the HP E1419A places any reference to an analog input channel into the Scan List. When you run algorithms, the scan list tells the HP E1419A which analog channels to scan during the Input Phase. Since the algorithm has no way to specify that an input channel is a reference temperature channel, the command: [SENSe:]REFerence:CHANnels (@<*ref_chan*>),(@<*meas_ch_list*>) is used to place the <*ref_chan*> channel in the scan list before the related thermocouple measuring channels. Now when analog channels are scanned, the HP E1419A will include the reference channel in the scan list and will scan it before the specified thermocouples are scanned. The reference measurement will be stored in the Reference Temperature Register. The reference temperature value is applied to the thermocouple EU conversions for thermcouple channel measurements that follow.

A Complete Thermocouple Measurement Command Sequence

The command sequence performs these functions:

- Configures reference temperature measurement on channel 15.
- Configures thermocouple measurements on channels 16 through 23.
- Instructs the HP E1419A to add channel 15 to the Scan List and order channels so channel 15 will be scanned before channels 16 through 23.

SENS:REF THER, 5000, (@115)	*5K thermistor temperature for channel 15*
SENS:FUNC:TEMP TC,J,(@116:123)	*Type J thermocouple temperature for channels 16 through 23*
SENS:REF:CHAN (@115),(@116:123)	*configure reference channel to be scanned before channels 16 - 23*

Supplying a Fixed Reference Temperature

The [SENSe:]REFerence:TEMPerature <*degrees_c*> command immediately stores the temperature of a controlled temperature reference junction panel in the Reference Temperature Register. The value is applied to all subsequent thermocouple channel measurements so there is no need to use SENS:REF:CHANNELS when using SENS:REF:TEMP.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To specify the temperature of a controlled temperature reference panel:

```
SENS:REF:TEMP 50                    reference temp = 50 °C
Now begin scan to measure thermocouples
```

Linking Strain Measurements

Strain measurements usually employ a Strain Completion and Excitation SCP (HP E1506,E1507,E1511). To link channels to strain EU conversions send the [SENSe:]FUNCtion:STRain:<*bridge_type*> [<*range*>,](@<*ch_list*>)

- <*bridge_type*> is not a parameter but is part of the command syntax. The following table relates the command syntax to bridge type. See the HP E1506 and HP E1507, and HP E1511 SCPs' user's manual for bridge schematics and field wiring information.

Command	Bridge Type
:FBENding	Full Bending Bridge
:FBPoisson	Full Bending Poisson Bridge
:FPOisson	Full Poisson Bridge
:HBENding	Half Bending Bridge
:HPOisson	Half Poisson Bridge
[:QUARter]	Quarter Bridge (default)

- The *ch_list* parameter specifies which sense SCP channel(s) to link to the strain EU conversion. *ch_list* does not specify channels on the HP E1506, and 07 Strain Bridge Completion SCPs but does specify one of the lower four channels of an HP E1511 SCP.

- The optional *range* parameter can be used to choose a fixed A/D range. When not specified (defaulted), the module uses auto-range.

To link channels 40 through 43 to the quarter bridge strain EU conversion:

```
SENS:FUNC:STR:QUAR (@140:143)       uses autorange
```

Other commands used to set up strain measurements are:

```
[SENSe:]STRain:POISson
[SENSe:]STRain:EXCitation
[SENSe:]STRain:GFACtor
[SENSe:]STRain:UNSTrained
```

For more detailed programming information, see the individual SCP manual.

NOTE Because of the number of possible strain gage configurations, the driver must generate any Strain EU conversion tables and download them to the instrument when INITiate is executed. This can cause the time to complete the INIT command to exceed 1 minute.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Custom EU Conversions

See "Creating and Loading Custom EU Conversion Tables" on page 3-98.

Linking Output Channels to Functions

Analog outputs are implemented either by an HP E1531 or HP E1537 Voltage Output SCP or an HP E1532 Current Output SCP. Channels where these SCPs are installed are automatically considered outputs. No SOURce:FUNCtion command is required since the HP E1531 and HP E1537 can only output voltage, while the HP E1532 can only output current. The only way to control the output amplitude of these SCPs is through the HP E1419A's Algorithm Language.

Setting up Digital Input and Output Channels

Setting up Digital Inputs

Digital inputs can be configured for polarity and depending on the SCP model, a selection of input functions as well. The following discussion will explain which functions are available with a particular Digital I/O SCP model. For Digital SCPs who's data direction is programmable, setting a digital channel's input function is what defines it as an input channel. The HP E1536 Isolated Digital I/O SCP's data direction is set by configuration switches so the SENSe:FUNCtion and SOURce:FUNCtion commands do not apply.

Setting Input Polarity

To specify the input polarity (logical sense) for digital channels use the command INPut:POLarity <*mode*>,(@<*ch_list*>). This capability is available on all digital SCP models. This setting is valid even while the specified channel in not an input channel. If and when the channel is configured for input (an input FUNCtion command), the setting will be in effect. For the HP E1536 the INP:POL command is disallowed for output channels.

- The <*mode*> paramter can be either NORMal or INVerted. When set to NORM, an input channel with 3v applied will return a logical 1. When set to INV, a channel with 3v applied will return a logic 0.
- The <*ch_list*> parameter specifies the channels to configure. The HP E1533 has 2 channels of 8 bits each. All 8 bits in a channel take on the configuration specified for the channel. The HP E1534 has 8 I/O bits that are individually configured as channels.

To configure the lower 8 bit channel of an HP E1533 for inverted polarity:

```
INP:POLARITY  INV,(@156)          SCP in SCP position 7
```

To configure the lower 4 bits of an HP E1534 for inverted polarity:

```
INP:POL  INV,(@148:151)           SCP in SCP position 6
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Setting Input Function

Both the HP E1533 Digital I/O SCP and HP E1534 Frequency/Totalizer SCP can input static digital states. The HP E1534 Frequency/Totalizer SCP can also input Frequency measurements and Totalize the occurrence of positive or negative digital signal edges.

Static State (CONDition) Function

To configure digital channels to input static states, use the [SENSe:]FUNCtion:CONDition (@<*ch_list*>) command. Examples:

To set the lower 8 bit channel of an HP E1533 in SCP position 4 to input

SENS:FUNC:COND (@132)

To set the upper 4 channels (bits) of an HP E1534 in SCP pos 5 to input states

SENS:FUNC:COND (@144:147)

Frequency Function

The frequency function uses two commands. For more on this HP E1534 capability see the SCP's User's Manual.

To set the frequency counting gate time execute:

[SENSe:]FREQuency:APERature <*gate_time*>,(@<*ch_list*>)

Sets the digital channel function to frequency

[SENSe:]FUNCtion:FREQuency (@<*ch_list*>)

Totalizer Function

The totalizer function uses two commands also. One sets the channel function, and the other sets the condition that will reset the totalizer count to zero. For more on this HP E1534 capability see the SCP's User's Manual.

To set the HP E1534's totalize reset mode

[SENSe:]TOTalize:RESet:MODE INIT | TRIG,(@<*ch_list*>)

To configure HP E1534 channels to the totalizer function

[SENSe:]FUNCtion:TOTalize (@<*ch_list*>)

Setting up Digital Outputs

Digital outputs can be configured for polarity, output drive type, and depending on the SCP model, a selection of output functions as well. The following discussion will explain which functions are available with a particular Digital I/O SCP model. Setting a digital channel's output function is what defines it as an output channel.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Setting Output Polarity

To specify the output polarity (logical sense) for digital channels use the command OUTPut:POLarity <*mode*>,(@<*ch_list*>). This capability is available on all digital SCP models. This setting is valid even while the specified channel in not an output channel. If and when the channel is configured for output (an output FUNCtion command), the setting will be in effect.

- The <*mode*> paramter can be either NORMal or INVerted. When set to NORM, an output channel set to logic 0 will output a TTL compatible low. When set to INV, an output channel set to logic 0 will output a TTL compatible high.
- The <*ch_list*> parameter specifies the channels to configure. The HP E1533 has 2 channels of 8 bits each. All 8 bits in a channel take on the configuration specified for the channel. The HP E1534 has 8 I/O bits that are individually configured as channels.

To configure the higher 8 bit channel of an HP E1533 for inverted polarity:

```
OUTP:POLARITY INV,(@133)          SCP in SCP position 4
```

To configure the upper 4 bits of an HP E1534 for inverted polarity:

```
OUTP:POL INV,(@136:139)           SCP in SCP position 4
```

Setting Output Drive Type

The HP E1533 and HP E1534 use output drivers that can be configured as either active or passive pull-up. To configure this, use the command OUTPut:TYPE <*mode*>,(@<*ch_list*>). This setting is valid even while the specified channel in not an output channel. If and when the channel is configured for output (an output FUNCtion command), the setting will be in effect.

- The <*mode*> parameter can be either ACTive or PASSive. When set to ACT (the default), the output provides active pull-up. When set to PASS, the output is pulled up by a resistor.
- The <*ch_list*> parameter specifies the channels to configure. The HP E1533 has 2 channels of 8 bits each. All 8 bits in a channel take on the configuration specified for the channel. The HP E1534 has 8 I/O bits that are individually configured as channels.

To configure the higher 8 bit channel of an HP E1533 for passive pull-up:

```
OUTP:TYPE PASS,(@156)             SCP in SCP position 7
```

To configure the upper 4 bits of an HP E1534 for active pull-up:

```
OUTP:TYPE ACT,(@148:155)          SCP in SCP position 6
```

Setting Output Functions

Both the HP E1533 Digital I/O SCP, and HP E1534 Frequency/Totalizer SCP can output static digital states. The HP E1534 Frequency/Totalizer SCP can also output single pulses per trigger, continuous pluses that are width

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

modulated (PWM, and continuous pulses that are frequency modulated (FM).

Static State (CONDition) Function

To configure digital channels to output static states, use the SOURce:FUNCtion:CONDition (@<*ch_list*>) command. Examples:

To set the upper 8 bit channel of an HP E1533 in SCP position 7 to output

```
SOUR:FUNC:COND (@157)
```

To set the lower 4 channels (bits) of an HP E1534 in SCP pos 6 to output states

```
SOUR:FUNC:COND (@156:159)
```

Variable Width Pulse Per Trigger

This function sets up one or more HP E1534 channels to output a single pulse per trigger (per algorithm execution). The width of the pulse from these channels is controlled by Algorithm Language statements. Use the command SOURce:FUNCtion[:SHAPe]:PULSe (@<*ch_list*>). Example command sequence:

To set HP E1534 channel 2 at SCP position 6 to output a pulse per trigger

```
SOUR:FUNC:PULSE (@149)
```

Example algorithm statement to control pulse width to 1 msec

```
O149 = 0.001
```

Variable Width Pulses at Fixed Frequency (PWM)

This function sets up one or more HP E1534 channels to output a train of pulses. A companion command sets the period for the complete pulse (rising edge to rising edge). This of course fixes the frequency of the pulse train. The width of the pulses from these channels is controlled by Algorithm Language statements.

Use the command SOURce:FUNCtion[:SHAPe]:PULSe (@<*ch_list*>). Example command sequence:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To enable pulse width modulation for HP E1534's third channel at SCP position 6

```
SOUR:PULM:STATE  ON,(@150)
```

To set pulse period to 0.5 msec (which sets the signal frequency 2 KHz)

```
SOUR:PULSE:PERIOD  0.5e-3,(@150)
```

To set function of HP E1534's third channel in SCP position 6 to PULSE

```
SOUR:FUNCTION:PULSE  (@150)
```

Example algorithm statement to control pulse width to .1 msec (20% duty-cycle)

```
O150 = 0.1e-3;
```

Fixed Width Pulses at Variable Frequency (FM)

This function sets up one or more HP E1534 channels to output a train of pulses. A companion command sets the width (↑ edge to ↓ edge) of the pulses. The frequency of the pulse train from these channels is controlled by Algorithm Language statements.

Use the command SOURce:FUNCtion[:SHAPe]:PULSe (@<*ch_list*>). Example command sequence:

To enable frequency modulation for HP E1534's fourth channel at SCP position 6

```
SOUR:FM:STATE  ON,(@151)
```

To set pulse width to 0.3333 msec

```
SOUR:PULSE:WIDTH  0.3333e-3,(@151)
```

To set function of HP E1534's fourth channel in SCP position 6 to PULSE

```
SOUR:FUNCTION:PULSE  (@151)
```

Example algorithm statement to control frequency to 1000 Hz

```
O151 = 1000;
```

Variable Frequency Square-Wave Output (FM)

To set function of HP E1534's fifth channel in SCP position 6 to output a variable frequency square-wave.

```
SOUR:FUNCTION:SQUare  (@152)
```

Example Algorithm Language statement to set output to 20KHz

```
O152 = 20e3;
```

For complete HP E1534 capabilities, see the SCP's User's Manual.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Performing Channel Calibration (Important!)

The *CAL? (also performed using CAL:SETup then CAL:SETup?) is a very important step. *CAL? generates calibration correction constants for all analog input and output channels. *CAL? must be performed in order for the HP E1419A to deliver its specified accuracy. Wait for the module to thoroughly warm-up (1 hour) before you execute *CAL?. See the guidlines and notes on the following page.

The "Front Panel" example program shown in Chapter 5 provides a calibration function that executes *CAL? and also performs the CAL:STORE ADC command to store the results of the calibration to the HP E1419A's non-volatile Flash memory. "cal_1419.vee" can be merged into your HP VEE application to perform the calibration function.

Operation and Restrictions

*CAL? generates calibration correction constants for each analog input channel for offset and gain at all 5 A/D range settings. For programmable input SCPs, these calibration constants are only valid for the current configuration (gain, and filter cut-off frequency). This means that *CAL? calibration is no longer valid if you change channel gain or filter settings (INP:FILT or INP:GAIN), but is still valid for changes of channel function or range (using SENS:FUNC ...). The calibration becomes invalid if you move these SCPs to different SCP locations.

For analog output channels (both measurement excitation SCPs as well as control output SCPs) *CAL? also generates calibration correction constants. These calibration constants are valid only for the specific SCPs in the positions they are currently in. The calibration becomes invalid if you move these SCPs to different SCP locations.

How to Use *CAL?

When you turn power on to the HP E1419A after you have first installed your SCPs (or after you have moved SCPs), the module will use approximate values for calibration constants. This means that input and output channels will function although the values will not be very accurate relative to the HP E1419A's specified capability. At this point, make sure the module is firmly anchored to the mainframe (front panel screws are tight), and let it warm up for a full hour. After it has warmed up, execute *CAL?.

What *CAL? Does

The *CAL? command causes the module to calibrate A/D offset and gain, and all channel offsets. This may take many minutes to complete. The actual time it will take your HP E1419A to complete *CAL? depends on the mix of SCPs installed. *CAL? performs literally hundreds of measurements of the internal calibration sources for each channel and must allow 17 time constants of settling wait each time a filtered channel's calibration source changes value. The *CAL? procedure is internally very sophisticated and results in an extremely well calibrated module.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When *CAL? finishes, it returns a +0 value to indicate success. The generated calibration constants are now in volatile memory as they always are when ready to use. If the configuration just calibrated is to be fairly long-term, you should now execute the CAL:STORE ADC command to store these constants in non-volatile memory. That way the module can restore calibration constants for this configuration in case of a power failure. After power returns, and after the module warms up, these constants will be relatively accurate.

When to Execute *CAL?

- After a 1 hour warm-up from the time you turn on your mainframe if it has been off for more than a few minutes.
- When you change the channel gain and/or filter cut-off frequency on programmable SCPs (using INPut:GAIN, or INPut:FILTer ...)
- When you change output current amplitude on the HP E1505 or HP E1518 SCPs.
- When you re-configure SCPs to different locations. This is true even if you replace an SCP with an identical model SCP because the calibration constants are specific to each SCP channel's individual performance.
- When the ambient temperature within the mainframe changes significantly. Temperature changes affect accuracy much more than long-term component drift. See temperature coefficients in Appendix A "Specifications".

Notes

1. To save time when performing channel calibration on multiple HP E1419As in the same mainframe, use the CAL:SETup and CAL:SETup? commands (see Chapter 6 for details).
2. You do not have to execute *CAL? or CAL:SETup each time you run your algorithms. See "When to Execute *CAL?" above for guidlines.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Defining C Language Algorithms

This section is an overview of how to write and download C algorithms into the E1419's memory. The assumption is that you have some programming experience in C, but since the E1419's version of C is limited, just about any experience with a programming language will suffice. See Chapter 4 for a complete description of the E1419's C language and functionality.

Arithmetic Operators: add **+**, subtract **-**, multiply *****, divide **/**
Assignment Operator: =
Comparison Functions: less than **<**, less than or equal **<=**, greater than **>**, greater than or equal **>=**, equal to **==**, not equal to **!=**
Boolean Functions: and **&&**, or **||**, not **!**
Variables: scalars of type **`static float`**, and single dimensioned arrays of type **`static float`** limited to 1024 elements.
Constants:
32-bit decimal integer; **`Dddd...`** where **`D`** and **`d`** are decimal digits but **`D`** is not zero. No decimal point or exponent specified.
32-bit octal integer; **`0oo...`** where **`0`** is a leading zero and **`o`** is an octal digit. No decimal point or exponent specified.
32-bit hexadecimal integer; **`0Xhhh...`** or **`0xhhh...`** where **`h`** is a hex digit.
32-bit floating point; **`ddd.`**, **`ddd.ddd`**, **`ddde±dd`**, **`dddE±dd`**, **`ddd.dddedd`**, or **`ddd.dddEdd`** where d is a decimal digit.
Flow Control: conditional construct **`if(){ } else { }`**
Intrinsic Functions:
Return the absolute value; **`abs(<`***expr***`>)`**
Return minimum; **`min(<`***expr1***`>,<`***expr2***`>)`**
Return maximum; **`max(<`***expr1***`>,<`***expr2***`>)`**
User defined function; **`<`***user_name***`>(<`***expr***`>)`**
Write value to CVT element; **`writecvt(<`***expr***`>,<`***expr***`>)`**
Write value to FIFO buffer; **`writefifo(<`***expr***`>)`**
Write value to both CVT and FIFO; **`writeboth(<`***expr***`>,<`***expr***`>)`**

Global variable definition

Global variables are necessary when you need to communicate information from one algorithm to another. Globals are initialized to 0 unless specifically assigned a value at define time. The initial value is only valid at the time of definition. That is, globals remain around and may be altered by other SCPI commands or algorithms. Globals are removed only by power-ON or *RST. The following string output is valid for strings of 256 characters or less.

```
ALG:DEF 'globals','static float output_max = 1, coefficients[ 10 ];'
```

If the global definition exceeds 256 characters, you need to download an indefinite block header, the definitions, and terminated by a LF/EOI sequence:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
ALG:DEF 'globals',#0static float output_max = 1, ..... LF/EOI
```

The LF/EOI sequence is part of the I/O and Instrument Manager in HP VEE. You must edit the E1419 I/O device for direct I/O and purposely select EOI to be sent with the EOL terminator.

Algorithm definition

Algorithms are similar in nature to global definitions. Both scalars and arrays can be defined for local use by the algorithm. If less than 256 characters, you need only place the algorithm code within string quotes:

```
ALG:DEF 'alg1','static float a = 1; if ( I100 > a ) writecvt( I100,10);'
```

If the algorithm exceeds 256 characters, you need to download an indefinite block header, the algorithm code, and terminated by a LF/EOI sequence:

```
ALG:DEF 'alg2',#0static float a = 1; ... ;LF/EOI
```

Algorithms remain around and cannot be altered once defined unless you specify a fixed size for the algorithm(see Chapter 4). Algorithms are removed from memory only by issueing a *RST or power-ON condition.

HP VEE text boxes are the best tool for storing the algorithm code and will be used extensively by this manual. See the "temp1419.vee" example program in Chapter 5 illustrates downloading algorithms to the HP E1419A.

Pre-setting Algorithm Variables

As you may have noticed in the examples above, you can initialize a variable to a particular value. However, that value is a one-time initialization. Later program execution may alter the variable, and re-issueing an INIT command to re-start program execution will NOT re-initialize that variable. Instead, you can alter any scalar or array using SCPI commands prior to issueing the INIT command, or you can rely upon the intrinsic variable First_loop to conditionally preset variables after receiving the INIT command. First_loop is a variable that is preset to non-zero due to the execution of the INIT command. With the occurrence of the first scan trigger and when algorithms execute for the first time, First_loop's value will be non-zero. Subsequent triggers will find this variable cleared. Here's an example of how First_loop can be used:

```
ALG:DEF 'alg1',#0static float a,b,c, start, some_array[ 4 ]; if ( First_loop )
{ a = 1; b = 2; c = 3; } * * LF/EOI
```

To pre-set variables under program control before issueing the INIT command, the ALG:SCALAR and ALG:ARRAY commands can be used. Assume the example algorithm above has already been defined. To preset the scalar **start** and the array **some_array**, you can use the following commands:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
ALG:SCAL 'alg1','start',1.2345
ALG:ARR 'alg1','some_array',#232.....LF/EOI
ALG:UPD
```

The ALG:SCAL command designates the name of the algorithm of where to find the local variable start and assigns that variable the value of 1.2345. Likewise, the ALG:ARRAY command designates the name of the algorithm, the name of the local array, and a definite length block for assigning the four real number values. As you can see, the scalar assignment uses ASCII and the array assignment uses binary. The later makes for a much faster transfer especially for large arrays. The format used is IEEE-754 8-byte binary real numbers. The header is #232 which states "the next 2 bytes are to be used to specify how many bytes are comming". In this case, 32 bytes represent the four 8-byte elements of the array. A 100 element array would have a header of #3800. If you wanted to pre-initialize a global scalar or array, the word 'globals' must be used instead of the algorithm name. The name simply specifies the memory space of where to find those elements.

As stated earlier in the chapter, all updates (changes) are held in a holding buffer until the computer issues the update command. The ALG:UPD is that command. Executing ALG:UPD before INIT does not make much difference since there is no concern as to how long it takes or how it is implemented. After INIT forces the buffered changes to all take place during the next Update Phase in the trigger cycle after reception of the ALG:UPD command..

Defining Data Storage

Specifying the Data Format

The format of the values stored in the FIFO buffer and CVT never changes. They are always stored as IEEE 32-bit Floating point numbers. The FORMat <*format*>[,<*length*>] command merely specifies whether and how the values will be converted as they are transferred from the CVT and FIFO to the host computer.

- The <*format*>[,<*length*>] parameters can specify:

PACKED	Same as REAL,64 except for the values of IEEE -INF, IEEE +INF, and Not-a-Number (NaN). See FORMat command in Chapter 5 for details.
REAL,32	means real 32-bit (no conversion, fastest)
REAL	same as above
REAL,64	means real 64-bit (values converted)
ASCii,7	means 7-bit ASCII (values converted)
ASCii	same as above (the *RST condition)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To specify that values are to remain in IEEE 32-bit Floating Point format for fastest transfer rate:

```
FORMAT REAL,32
```

To specify that values are to be converted to 7-bit ASCII and returned as a 15 character per value comma separated list:

```
FORMAT  ASC,7          The *RST, *TST? and power-on default format
```

or

```
FORM ASC               same operation as above
```

Turning Off IEEE +/-INF and NaN Values

The E1419 stores data in its FIFO and CVT in a data format adhering to the IEEE-754. This format yields +/-INF and NaN numbers for those values that indicate an out-of-bound condition(overange reading) or some uninitialized number(CVT element has not been written), respectively. Normal data queries for HP VEE do not permit these numbers to go unnoticed during a transaction. HP VEE wants to make sure you deal with valid numbers to avoid making calculations that can eventually cause errors. Therefore, any transaction that involves these numbers will cause an error in HP VEE and will abort the transaction.

To avoid this condition, the E1419 SCPI command DIAG:IEEE OFF can be issued to the E1419 to force it to never output +/-INF or NaN. The default power-ON or *RST condition is DIAG:IEEE ON, so you will have to explicitly send this command to avoid the condition. Keep in mind that this condition ONLY occurs when selecting the FORM REAL command. FORM PACKED is another way to avoid the numbers, but that is limited to the 8-byte data format. For speed, you will want to use FORM REAL,32 which is only four bytes for each element.

HP VEE 4.0 does include in its Main Properties the ability to detect the inifinity numbers generated by IEEE-754 and to force 9.9E37 numbers, but it will be more efficient to let the E1419 keep from generating the IEEE-754 numbers.

Selecting the FIFO Mode

The HP E1419A's FIFO can operate in two modes. One mode is for reading FIFO values while algorithms are executing, the other mode is for reading FIFO values after algorithms have been halted (ABORT sent).

- BLOCking; The BLOCking mode is the default and is used to read the FIFO while algorithms are executing. Your application program must read FIFO values often enough to keep it from overflowing (see "Continuously Reading the FIFO" on page 3-84). The FIFO stops accepting values when it becomes full (65,024 values). Values sent by algorithms after the FIFO is full are discarded. The first value to exceed 65,024 sets the STAT:QUES:COND? bit 10 (FIFO

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Overflowed), and an error message is put in Error Queue (read with SYS:ERR? command).

- Overwrite; When the FIFO fills, the oldest values in the FIFO are overwritten by the newest values. Only the latest 65,024 values are available. In OVERwrite mode the module must be halted (ABORT sent) before reading the FIFO (see "Reading the Latest FIFO Values" on page 3-86). This mode is very useful when you want to view an algorithm's response to a disturbance.

To set the FIFO mode (blocking is the *RST/Power-on condition):

[SENSe:]DATA:FIFO:MODE OVERWRITE	*select overwrite mode*
[SENSe:]DATA:FIFO:MODE BLOCK	*select blocking mode*

Setting up the Trigger System

Arm and Trigger Sources

Figure 3-7 shows the trigger and arm model for the HP E1419A. Note that when the Trigger Source selected is TIMer(the default), the remaining sources become Arm Sources. Using ARM:SOUR allows you to specify an event that must occur in order to start the Trigger Timer. The default Arm source is IMMediate (always armed).

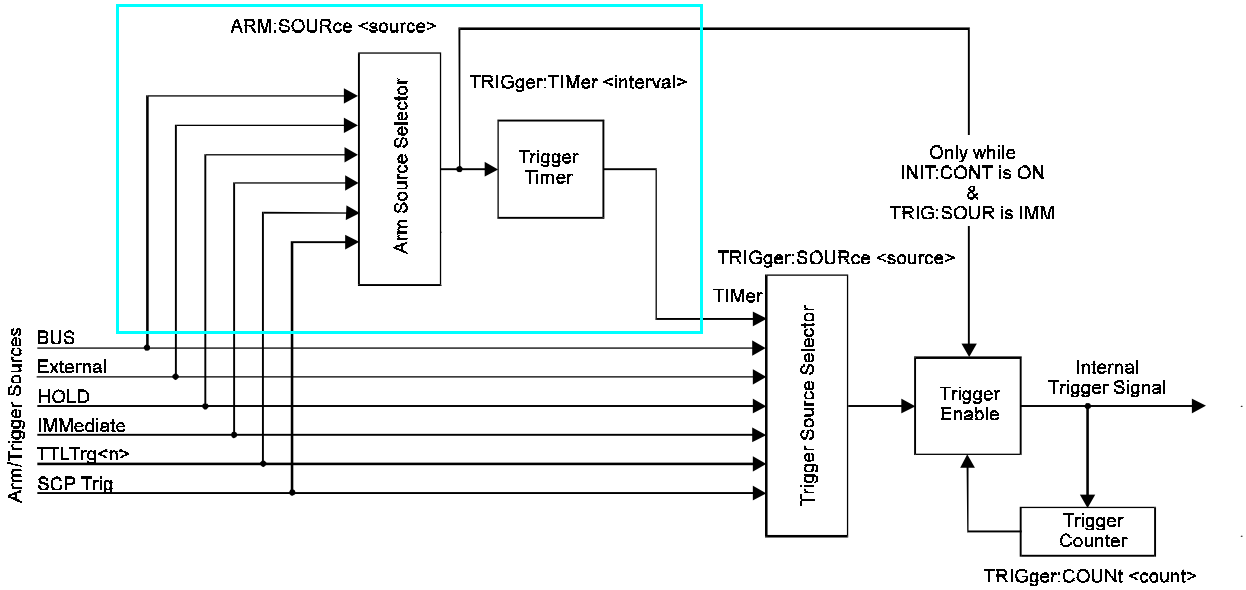


Figure 3-7 Logical Arm and Trigger Model

Selecting the Trigger Source

In order to start an algorithm execution cycle, a trigger event must occur. The source of this event is selected with the TRIGger:SOURce <*source*> command. The following table explains the possible choices for <*source*>.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter Value	Source of Trigger (after INITiate:… command)
BUS	TRIGger[:IMMediate], *TRG, GET (for HP-IB)
EXTernal	"TRG" signal input on terminal module
HOLD	TRIGger[:IMMediate]
IMMediate	The trigger signal is always true (scan starts when an INITiate:… command is received).
SCP	SCP Trigger Bus (future HP or SCP Breadboard)
TIMer	The internal trigger interval timer (must set Arm source)
TTLTrg<n>	The VXIbus TTLTRG lines (n=0 through 7)

NOTES

1. When TRIGger:SOURce is not TIMer, ARM:SOURce must be set to IMMediate (the *RST condition). If not, the INIT command will generate an error -221,"Settings conflict".
2. When TRIGger:SOURce is TIMer, the trigger timer interval (TRIG:TIM <*interval*>) must allow enough time to scan all channels, execute all algorithms and update all outputs or a +3012, "Trigger Too Fast" error will be generated during the algorithm cycle. See the TRIG:TIM command on page 6-314 for details.

To set the trigger source to the internal Trigger Timer (the default):

```
TRIG:SOUR TIMER                    now select ARM:SOUR
```

To set the trigger source to the External Trigger input connection:

```
TRIG:SOUR EXT                      an external trigger signal
```

To set the trigger source to a VXIbus TTLTRG line:

```
TRIG:SOUR TTLTRG1                  the TTLTRG1 trigger line
```

Selecting Trigger Timer Arm Source

Figure 3-7 shows that when the TRIG:SOUR is TIMer, the other trigger sources become Arm sources that control when the timer will start. The command to select the arm source is ARM:SOURce <*source*>.

- The <*source*> parameter choices are explained in the following table

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter Value	Source of Arm (after INITiate:… command)
BUS	ARM[:IMMediate]
EXTernal	"TRG" signal input on terminal module
HOLD	ARM[:IMMediate]
IMMediate	The arm signal is always true (scan starts when an INITiate:… command is received).
SCP	SCP Trigger Bus (future HP or SCP Breadboard)
TTLTrg<n>	The VXIbus TTLTRG lines (n=0 through 7)

NOTE When TRIGger:SOURce is not TIMer, ARM:SOURce must be set to IMMediate (the *RST condition). If not, the INIT command will generate an error -221,"Settings conflict".

To set the external trigger signal as the arm source:

```
ARM:SOUR  EXT                    trigger input on connector module
```

Programming the Trigger Timer

When the HP E1419A is triggered, it begins its algorithm execution cycle. The time it takes to complete a cycle is the minimum interval setting for the Trigger Timer. If programmed to a shorter time, the module will generate a "Trigger too fast" error. So, how can you determine this minimum time? After you have defined all of your algorithms, you send the ALG:TIME? command with its <*alg_name*> parameter set to 'MAIN'. This causes the HP E1419A's driver to analyze the time required for all four phases of the execution cycle; Input, Update, Execute Algorithm, and Output. The value returned from ALG:TIME? 'MAIN' is the minimum allowable Trigger Timer interval. With this information you now execute the command TRIGger:TIMer <*interval*> and set <*interval*> to the desired time that is equal to or greater than the minimum.

Setting the Trigger Counter

The Trigger Counter controls how many trigger events will be allowed to start an input-calculate-output cycle. When the number of trigger events set with the TRIGger:COUNt command is reached, the module returns to the Trigger Idle State (needs to be INITiated again). The default Trigger Count is 0 which is the same as INF (can be triggered an unlimited number of times). This setting will be used most often because it allows un-interrupted execution of control algorithms.

To set the trigger count to 50 (perhaps to help debug an algorithm):

```
TRIG:COUNT 50                    execute algorithms 50 times then return to Trig Idle State.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Outputting Trigger Signals

The HP E1419A can output trigger signals on any of the VXIbus TTLTRG lines. Use the OUTPut:TTLTrg<n>[:STATe] ON | OFF command to select one of the TTLTRG lines and then choose the source that will drive the TTLTRG line with the command OUTPut:TTLTrg:SOURce command. For details see OUTP:TTLTRG commands starting on page 6-254

To output a signal on the TTLTRG1 line each time the Trigger Timer cycles execute the commands:

```
TRIG:SOUR TIMER                 select trig timer as trig source
OUTP:TTLTRG1 ON                 select and enable TTLTRG1 line
OUTP:TTLTRG:SOUR TRIG           each trigger output on TTLTRG1
```

INITiating/Running Algorithms

When the INITiate[:IMMediate] command is sent, the HP E1419A builds the input Scan List from the input channels you referenced when you defined the algorithm with the ALG:DEF command above. The module also enters the Waiting For Trigger State (see Figure 3-3). In this state, all that is required to run the algorithm is a trigger event for each pass through the input-calculate-output cycle. To initiate the module, send the command:

```
INIT                            module to Waiting for Trigger State
```

When an INIT command is executed, the driver checks several interrelated settings programmed in the previous steps. If there are conflicts in these settings an error message is placed in the Error Queue (read with the SYST:ERR? command). Some examples:

- If TRIG:SOUR is not TIMer then ARM:SOUR must be IMMediate.
- The time it would take to execute all algorithms is longer than the TRIG:TIMER interval currently set.

Starting Algorithms

Once the module is INITiated it can accept triggers from any source specified in TRIG:SOUR.

```
TRIG:SOUR  TIMER                (*RST default)
ARM:SOUR  IMM                   (*RST default)
INIT                            INIT starts Timer triggers
    or
TRIG:SOUR  TIMER
ARM:SOUR  HOLD
INIT                            INIT  readies module
ARM                             ARM starts Timer triggers.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

... and the algorithms start to execute.

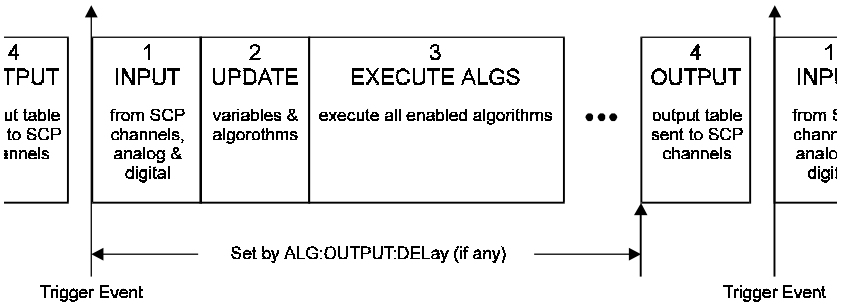


Figure 3-8 Sequence of Loop Operations

The Operating Sequence

The HP E1419A has four major operating phases. Figure 3-8 shows these phases. A trigger event starts the sequence:

1. (INPUT); the state of all digital inputs are captured and each analog input channel that is linked to an algorithm variable is scanned.
2. (UPDATE); The update phase is a window of time made large enough to process all variables and algorithm changes made after INIT. Its width is specified by ALG:UPDATE:WINDOW. This window is the only time variables and algorithms can be changed. Variable and algorithm changes can actually be accepted during other phases, but the changes don't take place until an ALG:UPDATE command is received and the update phase begins. If no ALG:UPDATE command is pending, the update phase is simply used to accept variable and algorithm changes from the application program (using ALG:SCAL, ALG:ARR, ALG:DEF). Data acquired by external specialized measurement instruments can be sent to your algorithms at this time.
3. (EXECUTE ALGS); all INPUT and UPDATE values have been made available to the algorithm variables and each enabled algorithm is executed. The results to be output from algorithms are stored in the Output Channel Buffer.
4. (OUTPUT); each Output Channel Buffer value stored during (EXECUTE ALGS) is sent to its assigned SCP channel. The start of the OUTPUT phase relative to the Scan Trigger can be set with the SCPI command ALG:OUTP:DELay.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Retrieving Algorithm Data

The most efficient means of acquiring data from the E1419 is to have its algorithms store real-number results in the FIFO or CVT. The algorithms use the writefifo(), writecvt(), and writeboth() intrinsic functions to perform this operation as seen in Figure 3-9.

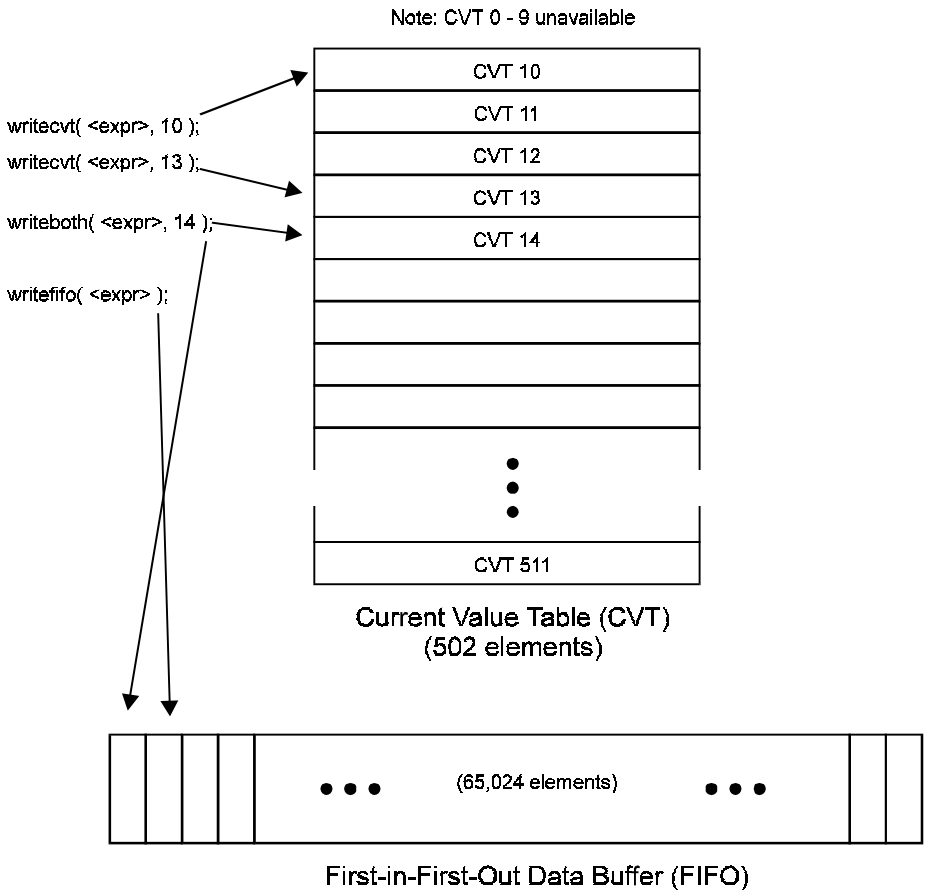


Figure 3-9 Writing Algorithm Data to FIFO and CVT

Note that the first 10 elements of the CVT are unavailable. These are used by the driver for internal data retrieval. However, all algorithms have access to the remaining 502 elements. Data is retrieved from the CVT with:

DATA:CVT? (@10,12,14:67)

The format of data comming from the CVT is determined by the FORMat command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The FIFO can store up to 65024 real numbers. Each writefifo() or writeboth() cause that expression to be placed into the FIFO. With a FIFO this large, you can store many seconds worth of data, dependent upon the volume of writes and the trigger cycle time. The FIFO's most valuable service is to keep your computer from having to spend too much time acquiring data from the E1419. This is ideal for HP VEE which has many other operator interactions and analysis to perform. HP VEE can quickly read the buffered data when required. Data is retrieved from the FIFO with:

DATA:FIFO:PART? <*count*>

<*count*> can be a number larger than the FIFO(up to 2.1 billion) if you want to read data continuously with HP VEE READ transactions. And, you can query how much data is in the FIFO with the DATA:FIFO:COUNT? command.

Read Variables Directly

To directly read algorithm variables that are not stored in the FIFO or CVT, you only need specify the memory space(algorithm name or globals) and the name of the variable. To read the values of scalar variables or single array elements, you use the command ALG:SCALar?. To read an entire array, you use ALG:ARRay? The former returns data in ASCII, and the later returns data in REAL,64(8-byte IEEE-754 format). This coincides with the ALG:SCAL and ALG:ARR commands form writing data to these variables. Here are some examples:

```
ALG:SCAL? 'globals','my_var'
ALG:SCAL? 'alg1','my_array[6]'
ALG:ARR? 'alg2','my_other_array'
```

The ALG:ARR? response data will consist of a block header and real-64 data bytes. For example, if **my_other_array** was 10 elements, the block header would be #280 which says there are two bytes of count that specify 80 bytes of data to follow. Data from the E1419 is terminated with the HP-IB EOI signal.

Which FIFO Mode?

The way you will read the FIFO depends on how the FIFO mode was set in the programming step "Setting the FIFO Mode" on page 3-77.

Continuously Reading the FIFO (FIFO mode BLOCK)

If you are going to read the FIFO while algorithms are running, you must set the FIFO mode to SENS:DATA:FIFO:MODE BLOCK. In this mode if the FIFO fills up, it stops accepting values from algorithms. The algorithms continue to execute, but the latest data is lost. To avoid losing any FIFO data, your application needs to read the FIFO often enough to keep it from overflowing. Here's a flow diagram to show you where and when to use the FIFO commands.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

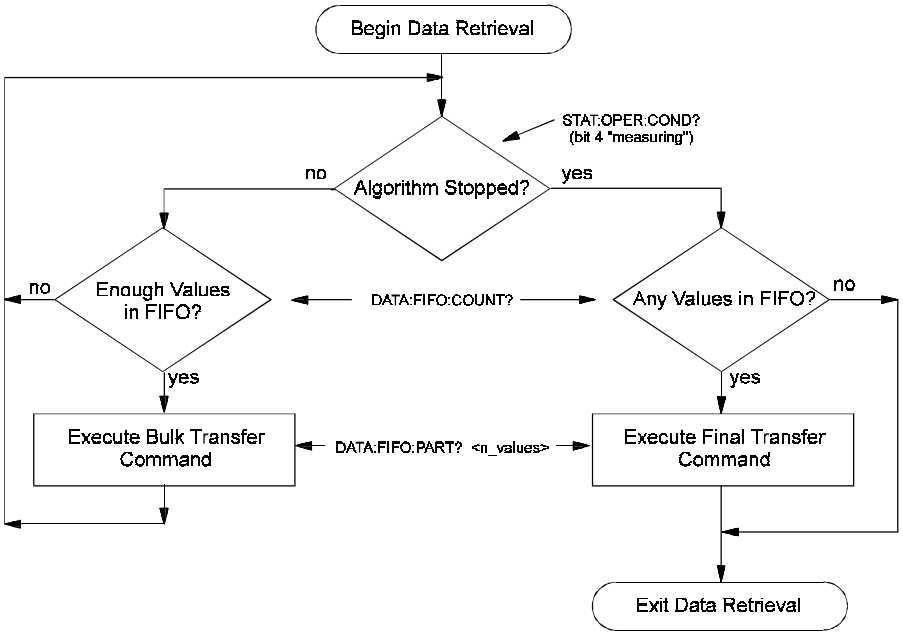


Figure 3-10 Controlling Reading Count

Here's an example command sequence for Figure 3-10. It assumes that the FIFO mode was set to BLOCK and that at least one algorithm is sending values to the FIFO.

```
    following loop reads number of values in FIFO while algorithms executing
loop while "measuring" bit is true          see STAT:OPER:COND bit 4
    SENS:DATA:FIFO:COUNT?                   query for count of values in FIFO
    input n_values here
    if n_values >= 16384                    Set minimum block size you want
                                            to transfer
        SENS:DATA:FIFO:PART? n_values       ask for n_values
        input read_data here                Format depends on FORMat cmd
    end if
end while loop
    following checks for values remaining in FIFO after "measuring" false
SENS:DATA:FIFO:COUNT?                       query for values still in FIFO
input n_values here
if n_values                                 if any values...
    SENS:DATA:FIFO:PART? n_values
    input read_data here                    get remaining values from FIFO
end if
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Reading the Latest FIFO Values (FIFO mode OVER)

In this mode the FIFO always contains the latest values (up to the FIFO's capacity of 65,024 values) from running algorithms. In order to read these values the algorithms must be stopped (use ABORT).This forms a record of the algorithm's latest performance. In the OVERwrite mode, the FIFO must not be read while it is accepting data from algorithms. Algorithm execution must be stopped before your application program reads the FIFO.

Here is an example command sequence you can use to read values from the FIFO after algorithms are stopped (ABORT sent).

```
SENS:DATA:FIFO:COUNT?                  query count of values in FIFO
input n_values here
if n_values                            if any values...
    SENS:DATA:FIFO:PART?  n_values     Format of values set by FORMat
    input read_data here               get remaining values from FIFO
end of if
```

Modifying Running Algorithm Variables

Updating the Algorithm Variables and Coefficients

The values sent with the ALG:SCALAR command are kept in the Update Queue until an ALGorithm:UPDate command is received.

```
ALG:UPD                                cause changes to take place
```

Updates are performed during phase 2 of the algorithm execution cycle (see figure 3-8 on page 3-82). The UPDate:WINDow <*num_updates*> command can be used to specify how many updates you need to perform during phase 2 (UPDATE phase) and assigns a constant window of time to accomplish all of the updates you will make. The default value for <*num_updates*> is 20. Fewer updates (shorter window) means slightly faster loop execution times. Each update takes approximately 1.4 µseconds.

To set the Update Window to allow 10 updates in phase 2:

```
ALG:UPD:WIND 10                        allows slightly faster execution
                                       than default of 20 updates
```

A way to synchronize variable updates with an external event is to send the ALGorithm:UPDate:CHANnel '<*dig_chan/bit*>' command.

- The <*dig_chan/bit*> parameter specifies the digital channel/bit that controls execution of the update operation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When the ALG:UPD:CHAN command is received, the module checks the current state of the digital bit. When the bit next changes state, pending updates are made in the next UPDATE Phase.

```
ALG:UPD:CHAN 'I133.B0'          perform updates when bit zero of
                                HP E1533 at channel 133
                                changes state
```

Enabling and Disabling Algorithms

An algorithm is enabled by default when it is defined. However, the ALG:STATe <*alg_name*>, ON | OFF command is provided to allows you to enable or disable algorithms. When an individual algorithm is enabled, it will execute when the module is triggered. When disabled, the algorithm will not execute.

NOTE The command ALG:STATE <*alg_name*>, ON | OFF does not take effect until an ALG:UPDATE command is received. This allows you to send multiple ALG:STATE commands and then synchronize their effect.

To enable ALG1 and ALG2, and disable ALG3 and ALG4:

```
ALG:STATE  'ALG1',ON            enable algorithm ALG1
ALG:STATE  'ALG2',ON            enable algorithm ALG2
ALG:STATE  'ALG3',OFF           disable algorithm ALG3
ALG:STATE  'ALG4',OFF           disable algorithm ALG4
ALG:UPDATE                      changes take effect at next update
                                phase
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Setting Algorithm Execution Frequency

The ALGorithm:SCAN:RATio '<*alg_name*>',<*num_trigs*> command sets the number of trigger events that must occur before the next execution of algorithm <*alg_name*>. If you wanted 'ALG3' to execute only every 20 triggers, you would send ALG:SCAN:RATIO 'ALG3',20, followed by an ALG:UPDATE command. 'ALG3' would then execute on the first trigger after INIT, then the 21st, then the 41st, etc. This can be useful to adjust the response time of one algorithm relative to others. The *RST default for all algorithms is to execute on every trigger event.

Example Command Sequence

This example command sequence puts together all of the steps discussed so far in this chapter.

```
*RST                                        Reset the module
    Setting up Signal Conditioning (only for programmable SCPs in pos 4-7)
INPUT:FILTER:FREQUENCY 2,(@140:143)
INPUT:GAIN 64,(@140:143)
INPUT:GAIN 8,(@144:147)
    set up digital channel characteristics
INPUT:POLARITY  NORM,(@156)                 (*RST default)
OUTPUT:POLARITY  NORM,(@157)                (*RST default)
OUTPUT:TYPE  ACTIVE,(@157)
    link channels to EU conversions (measurement functions)
SENSE:FUNCTION:VOLTAGE AUTO,(@100:107) (*RST default)
SENSE:REFERENCE THER,5000,AUTO,(@108)
SENSE:FUNCTION:TEMPERATURE TC,T,AUTO,(@109:123)
SENSE:REFERENCE:CHANNELS  (@108),(@109:123)
    configure digital output channel for "alarm channel"
SOURCE:FUNCTION:CONDITION (@157)
    execute channel calibration
*CAL?                                       can take several minutes
    Configure the Trigger System
ARM:SOURCE IMMEDIATE                        (*RST default)
TRIGGER:COUNT INF                           (*RST default)
TRIGGER:TIMER .010                          (*RST default)
TRIGGER:SOURCE TIMER                        (*RST default)
    specify data format
FORMAT ASC,7                                (*RST default)
    select FIFO mode
SENSE:DATA:FIFO:MODE BLOCK                  may read FIFO while running
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Define algorithm

```
ALG:DEFINE 'ALG1','static float a,b,c, div, mult, sub;
                    if ( First_loop )
                        {
                         a = 1; b = 2; c = 3;
                        writecvt( a, 10 ); writefifo( b, 11 ); writefifo( c, 12 );
                        }
                    writecvt( a / div, 13 );
                    writecvt( b * mult, 14 );
                    writecvt( c - sub, 15 );'
```

Pre-set coefficients

```
ALG:SCAL 'ALG1','div',5
ALG:SCAL 'ALG1','mult',5
ALG:SCAL 'ALG1','sub',0
ALG:UPDATE
```

initiate trigger system (start algorithm)

```
INITIATE
```

retrieve data

```
SENSE:DATA:CVT? (@10:15)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the Status System

The HP E1419A's Status System allows you to quickly poll a single register (the Status Byte) to see if any internal condition needs attention. Figure 3-11 shows that the three Status Groups (Operation Status, Questionable Data, and the Standard Event Groups) and the Output Queue all send summary information to the Status Byte. By this method the Status Byte can report many more events than its eight bits would otherwise allow. Figure 3-12 shows the Status System in detail.

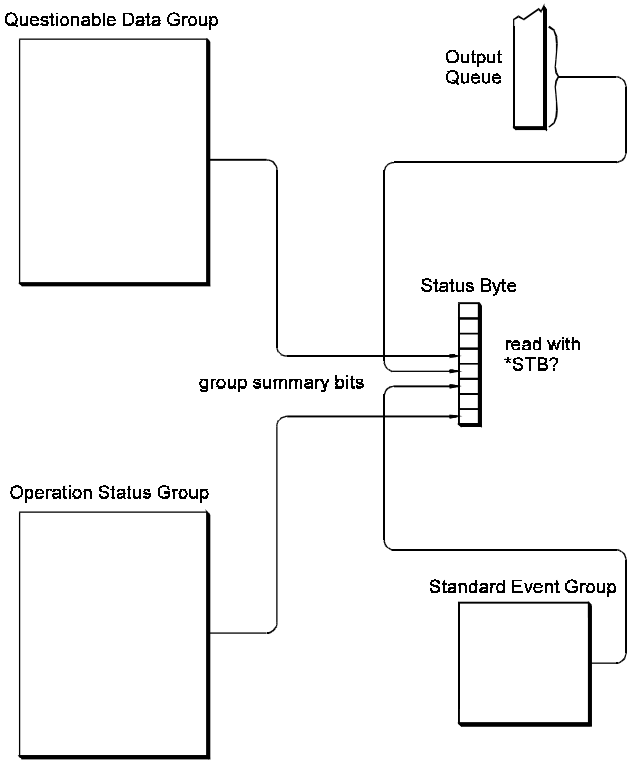


Figure 3-11 Simplified Status System Diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

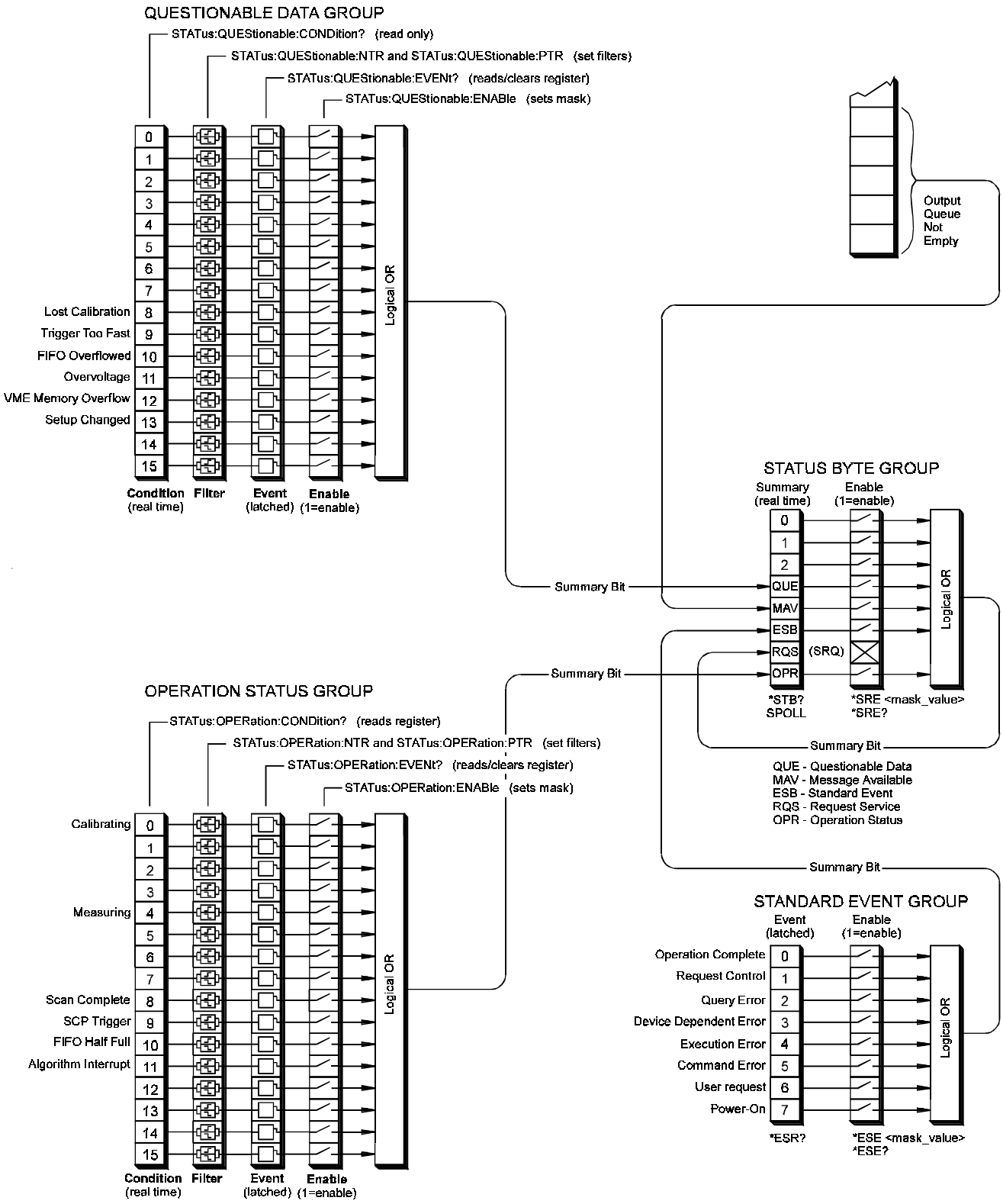


Figure 3-12 HP E1419 Status System

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Status Bit Descriptions

Questionable Data Group			
Bit	**Bit Value**	**Event Name**	**Description**
8	256	Lost Calibration	At *RST or Power-on Control Processor has found a checksum error in the Calibration Constants. Read error(s) with SYST:ERR? command and re-calibrate areas that lost constants.
9	512	Trigger Too Fast	Scan not complete when another trigger event received.
10	1024	FIFO Overflowed	Attempt to store more than 65,024 values in FIFO.
11	2048	Overvoltage (Detected on Input)	If the input protection jumper has not been cut, the input relays have been opened and *RST is required to reset the module. Overvoltage will also generate an error.
12	4096	VME Memory Overflow	The number of values taken exceeds VME memory space.
13	8192	Setup Changed	Channel Calibration in doubt because SCP setup may have changed since last *CAL? or CAL:SETup command. (*RST always sets this bit.)

Operation Status Group			
Bit	**Bit Value**	**Event Name**	**Description**
0	1	Calibrating	Set by CAL:TARE, and CAL:SETup. Cleared by CAL:TARE?, and CAL:SETup?. Set while *CAL? executing, then cleared.
4	16	Measuring	Set when instrument INITiated. Cleared when instrument returns to Trigger Idle State.
8	256	Scan Complete	Set when each pass through a Scan List is completed
9	512	SCP Trigger	Reserved for future HP SCPs
10	1024	FIFO Half Full	FIFO contains at least 32,768 values
11	2048	Algorithm Interrupt	The interrupt() function was called in an executing algorithm

Standard Event Group			
Bit	**Bit Value**	**Event Name**	**Description**
0	1	Operation Complete	*OPC command executed and instrument has completed all pending operations.
1	2	Request Control	Not used by HP E1419A
2	4	Query Error	Attempting to read empty output queue or output data lost.
3	8	Device Dependent Error	A device dependent error occurred. See Appendix B.
4	16	Execution Error	Parameter out of range, or instrument cannot execute a proper command because it would conflict with another instrument setting.
5	32	Command Error	Unrecognized command or improper parameter count or type.
6	64	User Request	Not used by HP E1419A
7	128	Power-On	Power has been applied to the instrument

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Enabling Events to be Reported in the Status Byte

There are two sets of registers that individual status conditions must pass through before that condition can be recorded in a group's Event Register. These are the Transition Filter Registers and the Enable registers. They provide selectivity in recording and reporting module status conditions.

Configuring the Transition Filters

Figure 3-12 shows that the Condition Register outputs are routed to the input of the Negative Transition and Positive Transition Filter Registers. For space reasons they are shown together but are controlled by individual SCPI commands. Here is the truth table for the Transition Filter Registers:

Condition Reg Bit	PTRansition Reg Bit	NTRansition Reg Bit	Event Reg Input
0→1	0	0	0
1→0	0	0	0
0→1	1	0	1
1→0	1	0	0
0→1	0	1	0
1→0	0	1	1
0→1	1	1	1
1→0	1	1	1

The Power-on default condition is: All Positive Transition Filter Register bits set to one and all Negative Transition Filter Register bits set to 0. This applies to both the Operation and Questionable Data Groups.

An Example using the Operation Group

Suppose that you wanted the module to report via the Status System when it had completed executing *CAL?. The "Calibrating" bit (bit 0) in the Operation Condition Register goes to 1 when *CAL? is executing and returns to 0 when *CAL? is complete. In order to record only the negative transition of this bit in the STAT:OPER:EVEN register you would send:

```
STAT:OPER:PTR 32766      All ones in Pos Trans Filter register except bit 0=0
STAT:OPER:NTR 1          All zeros in Neg Trans Filter register except bit 0=1
```

Now when *CAL? completes and Operation Condition Register bit zero goes from 1 to 0, Operation Event Register bit zero will become a 1.

Configuring the Enable Registers

In Figure 3-12 you will note that each Status Group has an Enable Register. These control whether or not the occurrence of an individual status condition will be reported by the group's summary bit in the Status Byte.

Questionable Data Group Examples

If you only wanted the "FIFO Overflowed" condition to be reported by the QUE bit (bit 3) of the Status Byte, you would execute;

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STAT:QUES:ENAB 1024 *1024=decimal value for bit 10*

If you wanted the "FIFO Overflowed" and "Setup Changed" conditions to be reported you would execute;

STAT:QUES:ENAB 9216 *9216=decimal sum of values for bits 10 and 13*

Operation Status Group Examples

If you only wanted the "FIFO Half Full" condition to be reported by the OPR bit (bit 7) of the Status Byte, you would execute;

STAT:OPER:ENAB 1024 *1024=decimal value for bit 10*

If you wanted the "FIFO Half Full" and "Scan Complete" conditions to be reported you would execute;

STAT:OPER:ENAB 1280 *1280=decimal sum of values for bits 10 and 8*

Standard Event Group Examples

If you only wanted the "Query Error", "Execution Error", and "Command Error" conditions to be reported by the ESB bit (bit 5) of the Status Byte, you would execute;

*ESE 52 *52=decimal sum of values for bits 2, 4, and 5*

Reading the Status Byte

To check if any enabled events have occurred in the status system, you first read the Status Byte using the *STB? command. If the Status Byte is all zeros, there is no summary information being sent from any of the status groups. If the Status Byte is other than zero, one or more enabled events have occurred. You interpret the Status Byte bit values and take further action as follows:

Bit 3 (QUE) bit value 8_{10}

Read the Questionable Data Group's Event Register using the STAT:QUES:EVENT? command. This will return bit values for events which have occurred in this group. After reading, the Event Register is cleared.

Note that bits in this group indicate error conditions. If bit 8, 9 or 10 is set, error messages will be found in the Error Queue. If bit 7 is set, error messages will be in the error queue following the next *RST or cycling of power. Use the SYST:ERR? command to read the error(s).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bit 4 (MAV) bit value 16_{10}	There is a message available in the Output Queue. You should execute the appropriate query command.
Bit 5 (ESB) bit value 32_{10}	Read the Standard Event Group's Event Register using the *ESR? command. This will return bit values for events which have occurred in this group. After reading, this status register is cleared. Note that bits 2 through 5 in this group indicate error conditions. If any of these bits are set, error messages will be found in the Error Queue. Use the SYST:ERR? command to read these.
Bit 7 (OPR) bit value 128_{10}	Read the Operation Status Group's Event Register using the STAT:OPER:EVENT? command. This will return bit values for events which have occurred in this group. After reading, the Event Register is cleared.

Clearing the Enable Registers

To clear the Enable Registers execute:

STAT:PRESET	*for Operation Status and Questionable Data Groups*
*ESE 0	*for the Standard Event Group*
*SRE 0	*for the Status Byte Group*

The Status Byte Group's Enable Register

The Enable Register for the Status Byte Group has a special purpose. Notice in Figure 3-12 how the Status Byte Summary bit wraps back around to the Status Byte. The summary bit sets the RQS (request service) bit in the Status Byte. Using this Summary bit (and those from the other status groups) you can poll the Status Byte and check the RQS bit to determine if there are any status conditions which need attention. In this way the RQS bit is like the HP-IB's SRQ (Service Request) line. The difference is that while executing an HP-IB serial poll (SPOLL) releases the SRQ line, executing the *STB? command does not clear the RQS bit in the Status Byte. You must read the Event Register of the group who's summary bit is causing the RQS.

Reading Status Groups Directly

You may want to directly poll status groups for instrument status rather than poll the Status Byte for summary information.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Reading Event Registers

The Questionable Data, Operation Status, and Standard Event Groups all have Event Registers. These Registers log the occurrence of even temporary status conditions. When read, these registers return the sum of the decimal values for the condition bits set, then are cleared to make them ready to log further events. The commands to read these Event Registers are:

STAT:QUES:EVENT?	*Questionable Data Group Event Register*
STAT:OPER:EVENT?	*Operation Status Group Event Register*
*ESR?	*Standard Event Group Event Register*

Clearing Event Registers

To clear the Event Registers without reading them execute:

*CLS	*clears all group's Event Registers*

Reading Condition Registers

The Questionable Data and Operation Status Groups each have a Condition Register. The Condition Register reflects the group's status condition in "real-time". These registers are not latched so transient events may be missed when the register is read. The commands to read these registers are:

STAT:QUES:COND?	*Questionable Data Group Condition Register*
STAT:OPER:COND?	*Operation Status Group Condition Register*

HP E1419A Background Operation

The HP E1419A inherently runs its algorithms and calibrations in the background mode with no interaction required from the driver. All resources needed to run the measurements are controlled by the on-board Control Processor (DSP).

The driver is required to set up the type of measurement to be run, modify algorithm variables, and to unload data from the card after it appears in the CVT or FIFO. Once the INIT[:IMM] command is given, the HP E1419A is initiated and all functions of the trigger system and algorithm execution are controlled by its on-board control processor. The driver returns to waiting for user commands. No interrupts are required for the HP E1419A to complete its measurements.

While the module is running algorithms, the driver can be queried for its status, variables and algorithms can be accessed, and data can be read from the FIFO and CVT. The ABORT command may be given to force continuous execution to complete. Any changes to the measurement set up will not be allowed until the TRIG:COUNT is reached, or an ABORT

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

command is given. Of course any commands or queries can be given to other instruments while the HP E1419A is running algorithms.

Updating the Status System and VXIbus Interrupts

The driver needs to update the status system's information whenever the status of the HP E1419A changes. This update is always done when the status system is accessed, or when CALibrate, INITiate, or ABORt commands are executed. Most of the bits in the OPER and QUES registers represent conditions which can change while the HP E1419A is measuring (initiated). In many circumstances it is sufficient to have the status system bits updated the next time the status system is accessed, or the INIT or ABORt commands are given. When it is desired to have the status system bits updated closer in time to when the condition changes on the HP E1419A, the HP E1419A interrupts can be used.

The HP E1419A can send VXI interrupts upon the following conditions:

- Trigger too Fast condition is detected. Trigger comes prior to trigger system being ready to receive trigger.
- FIFO overflowed. In either FIFO mode, data was received after the FIFO was full.
- Overvoltage detection on input. If the input protection jumper has not been cut, the input relays have all been opened, and a *RST is required to reset the HP E1419A.
- Scan complete. The HP E1419A has finished a scan list.
- SCP trigger. A trigger was received from an SCP.
- FIFO half full. The FIFO contains at least 32768 values.
- Measurement complete. The trigger system exited the "Wait-For-Arm". This clears the Measuring bit in the OPER register.
- Algorithm executes an "interrupt()" statement.

These HP E1419A interrupts are not always enabled since, under some circumstances, this could be detrimental to the users system operation. For example, the Scan Complete, SCP triggers, FIFO half full, and Measurement complete interrupts could come repetitively, at rates that would cause the operating system to be swamped processing interrupts. These conditions are dependent upon the user's overall system design, therefore the driver allows the user to decide which, if any, interrupts will be enabled.

The way the user controls which interrupts will be enabled is via the *OPC, STATUS:OPER/QUES:ENABLE, and STAT:PRESET commands.

Each of the interrupting conditions listed above, has a corresponding bit in the QUES or OPER registers. If that bit is enabled via the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STATus:OPER/QUES:ENABle command to be a part of the group summary bit, it will also enable the HP E1419A interrupt for that condition. If that bit is not enabled, the corresponding interrupt will be disabled.

Note

Once a status driven condition sets an enabled bit in one of the Event registers, that Event register must be read (STAT:OPER:EVENT? or STAT:QUES:EVENT?) in order to clear the register and re-fenable further interrupts to occurr.

Sending the STAT:PRESET will disable all the interrupts from the HP E1419A.

Sending the *OPC command will enable the measurement complete interrupt. Once this interrupt is received and the OPC condition sent to the status system, this interrupt will be disabled if it was not previously enabled via the STATUS:OPER/QUES:ENABLE command.

Note for C-SCPI and SICL

The above description is always true for a downloaded driver. In the C-SCPI driver, however, the interrupts will only be enabled if **cscpi_overlap** mode is ON when the enable command is given. If **cscpi_overlap** is OFF, the user is indicating they do not want interrupts to be enabled. Any subsequent changes to **cscpi_overlap** will not change which interrupts are enabled. Only sending *OPC or STAT:OPER/QUES:ENAB with **cscpi_overlap** ON will enable interrupts.

In addition the user can enable or disable all interrupts via the SICL calls, **iintron()** and **iintroff()**.

Creating and Loading Custom EU Conversion Tables

The HP E1419A provides for loading custom EU conversion tables. This allows you to have on-board conversion of transducers not otherwise supported by the HP E1419A.

Standard EU Operation

The EU conversion tables built into the HP E1419A are stored in a "library" in the module’s non-volatile Flash Memory. When you link a specific channel to a standard EU conversion using the [SENSe:]FUNC:… command, the module copies that table from the library to a segment of RAM allocated to the specified channel. When a single EU conversion is specified for multiple channels, multiple copies of that conversion table are

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

put in RAM, one copy into each channel's Table RAM Segment. The conversion table-per-channel arrangement allows higher speed scanning since the table is already loaded and ready to use when the channel is scanned.

Custom EU Operation

Custom EU conversion tables are loaded directly into a channel's Table RAM Segment using the DIAG:CUST:LIN and DIAG:CUST:PIEC commands. The DIAG:CUST:… commands can specify multiple channels. To "link" custom conversions to their tables you would execute the [SENSe:]FUNC:CUST <*range*>,(@<*ch_list*>) command. Unlike standard EU conversions, the custom EU conversions are already linked to their channels (tables loaded) before you execute the [SENSe:]FUNC:CUST command but the command allows you to specify the A/D range for these channels.

NOTE The *RST command clears all channel Table RAM segments. Custom EU conversion tables must be re-loaded using the DIAG:CUST:… commands.

Custom EU Tables

The HP E1419A uses two types of EU conversion tables, linear and piecewise. The linear table describes the transducer's response slope and offset (y=mx+b). The piecewise conversion table gets its name because it is actually an approximation of the transducer's response curve in the form of 512 linear segments whose end-points fall on the curve. Data points that fall between the end-points are linearly interpolated. The built-in EU conversions for thermistors, thermocouples, and RTDs use this type of table.

Custom Thermocouple EU Conversions

The HP E1419A can measure temperature using custom characterized thermocouple wire of types E, J, K, N, R, S, and T. The custom EU table generated for the individual batch of thermocouple wire is loaded to the appropriate channels using the DIAG:CUST:PIEC command (see the HP VEE example "eu_1419.vee"). Since thermocouple EU conversion requires a "reference junction compensation" of the raw thermocouple voltage, the custom EU table is linked to the channel(s) using the command [SENSe:]FUNCtion:CUSTom:TCouple <*type*>[,<*range*>],(@<*ch_list*>).

The <*type*> parameter specifies the type of thermocouple wire so that the correct built-in table will be used for reference junction compensation. Reference junction compensation is based on the reference junction temperature at the time the custom channel is measured. For more information see Thermocouple Reference Temperature Compensation on page 3-64.

Custom Reference Temperature EU Conversions

The HP E1419A can measure reference junction temperatures using custom characterized RTDs and thermistors. The custom EU table generated for the individually characterized transducer is loaded to the appropriate channel(s) using the DIAG:CUST:PIEC command (see the HP VEE example "eu_1419.vee"). Since the EU conversion from this custom EU table is to be

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

considered the "reference junction temperature", the channel is linked to this EU table using the command [SENSe:]FUNCtion:CUSTom:REFerence [<*range*>,](@<*ch_list*>).

This command uses the custom EU conversion to generate the reference junction temperature as explained in the section Thermocouple Reference Temperature Compensation on page 3-64.

Creating Conversion Tables

The HP 1419A comes with an HP VEE example program you can use to generate custom EU tables. See the "eu_1419.vee" example in Chapter 5 for more information.

Summary

The following points describe the capabilities of custom EU conversion:

- A given channel only has a single active EU conversion table assigned to it. Changing tables requires loading it with a DIAG:CUST:… command.
- The limit on the number of different custom EU tables that can be loaded in an HP E1419A is the same as the number of channels.
- Custom tables can provide the same level of accuracy as the built-in tables. In fact the built-in resistance function uses a linear conversion table, and the built -in temperature functions use the piecewise conversion table.

Compensating for System Offsets

System Wiring Offsets

The HP E1419A can compensate for offsets in your system's field wiring. Apply shorts to channels at the Unit-Under-Test (UUT) end of your field wiring, and then execute the CAL:TARE (@<*ch_list*>) command. The instrument will measure the voltage at each channel in <*ch_list*> and save those values in RAM as channel Tare constants.

Important Note for Thermocouples

- You must not use CAL:TARE on field wiring that is made up of thermocouple wire. The voltage that a thermocouple wire pair generates can not be removed by introducing a short anywhere between its junction and its connection to an isothermal panel (either the HP E1419A's Terminal Module or a remote isothermal reference block). Thermal voltage is generated along the entire length of a thermocouple pair where there is any temperature gradient along that length. To CAL:TARE thermocouple wire this way would introduce an unwanted offset in the voltage/temperature relationship for that thermocouple. If you inadvertently CAL:TARE a thermocouple wire pair, see "Resetting CAL:TARE" on page 3-102.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- You should use CAL:TARE to compensate wiring offsets (copper wire, not thermocouple wire) between the HP E1419A and a remote thermocouple reference block. Disconnect the thermocouples and introduce copper shorting wires between each channel's HI and LO, then execute CAL:TARE for these channels.

Residual Sensor Offsets

To remove offsets like those in an unstrained strain gage bridge, execute the CAL:TARE command on those channels. The module will then measure the offsets and as in the wiring case above, remove these offsets from future measurements. In the strain gage case, this "balances the bridge" so all measurements have the initial unstrained offset removed to allow the most accurate high speed measurements possible.

Operation

After CAL:TARE <*ch_list*> measures and stores the offset voltages, it then performs the equivalent of a *CAL? operation. This operation uses the Tare constants to set a DAC which will remove each channel offset as "seen" by the module's A/D converter.

The absolute voltage level that CAL:TARE can remove is dependent on the A/D range. CAL:TARE will choose the lowest range that can handle the existing offset voltage. The range that CAL:TARE chooses will become the lowest usable range (range floor) for that channel. For any channel that has been "CAL:TAREd" Autorange will not go below that range floor and selecting a manual range below the range floor will return an Overload value (see table on page 6-234).

As an example assume that the system wiring to channel 0 generates a +0.1 Volt offset with 0 Volts (a short) applied at the UUT. Before CAL:TARE the module would return a reading of 0.1 Volt for channel 0. After CAL:TARE (@100), the module will return a reading of 0 Volts with a short applied at the UUT and the system wiring offset will be removed from all measurements of the signal to channel 0. Think of the signal applied to the instrument's channel input as the *gross* signal value. CAL:TARE removes the *tare* portion leaving only the *net* signal value.

Because of settling times, especially on filtered channels, CAL:TARE can take a number of minutes to execute.

The tare calibration constants created during CAL:TARE are stored in and are usable from the instrument's RAM. If you want the Tare constants to be stored in non-volatile Flash Memory you can execute the CAL:STORE TARE command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE The HP E1419A's Flash Memory has a finite lifetime of approximately ten thousand write cycles (unlimited read cycles). While executing CAL:STOR once every day would not exceed the lifetime of the Flash Memory for approximately 27 years, an application that stored constants many times each day would unnecessarily shorten the Flash Memory's lifetime.

Resetting CAL:TARE

If you wish to "undo" the CAL:TARE operation, you can execute CAL:TARE:RESet then *CAL?/CAL:SET. If current Tare calibration constants have been stored in Flash Memory, execute CAL:TARE:RESET, then CAL:STORE TARE.

Special Considerations

Here are some things to keep in mind when using CAL:TARE.

Maximum Tare Capability

The tare value that can be compensated for is dependent on the instrument range and SCP channel gain settings. The following table lists these limits

Maximum CAL:TARE Offsets

A/D range ±V F.Scale	Offset V Gain x1	Offset V Gain x8	Offset V Gain x16	Offset V Gain x64
16	3.2213	.40104	.20009	.04970
4	.82101	.10101	.05007	.01220
1	.23061	.02721	.01317	.00297
.25	.07581	.00786	.00349	.00055
.0625	.03792	.00312	.00112	n/a

Changing Gains or Filters

If you decide to change a channel's SCP setup after a CAL:TARE operation you must perform a *CAL? operation to generate new DAC constants and reset the "range floor" for the stored Tare value. You must also consider the tare capability of the range/gain setup you are changing to. For instance if the actual offset present is 0.6 Volts and was "Tared" for a 4 Volt range/Gain x1 setup, moving to a 1 Volt range/Gain x1 setup will return Overload values for that channel since the 1 Volt range is below the range floor as set by CAL:TARE. See table on page 6-234 for more on values returned for Overload readings.

Unexpected Channel Offsets or Overloads

This can occur when your HP E1419A's Flash Memory contains CAL:TARE offset constants that are no longer appropriate for its current application. Execute CAL:TARE:RESET then *CAL? to reset the tare constants in RAM. Measure the affected channels again. If the problems go away, you can now reset the tare constants in Flash memory by executing CAL:STORE TARE.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Detecting Open Transducers

Most of the HP E1419A's analog input SCPs provide a method to detect open transducers. When Open Transducer Detect (OTD) is enabled, the SCP injects a small current into the HIGH and LOW input of each channel. The polarity of the current pulls the HIGH inputs toward +17 volts and the LOW inputs towards -17 volts. If a transducer is open, measuring that channel will return an over-voltage reading. OTD is available on a per SCP basis. All eight channels of an SCP are enabled or disabled together. See Figure 3-13 for a simplified schematic diagram of the OTD circuit.

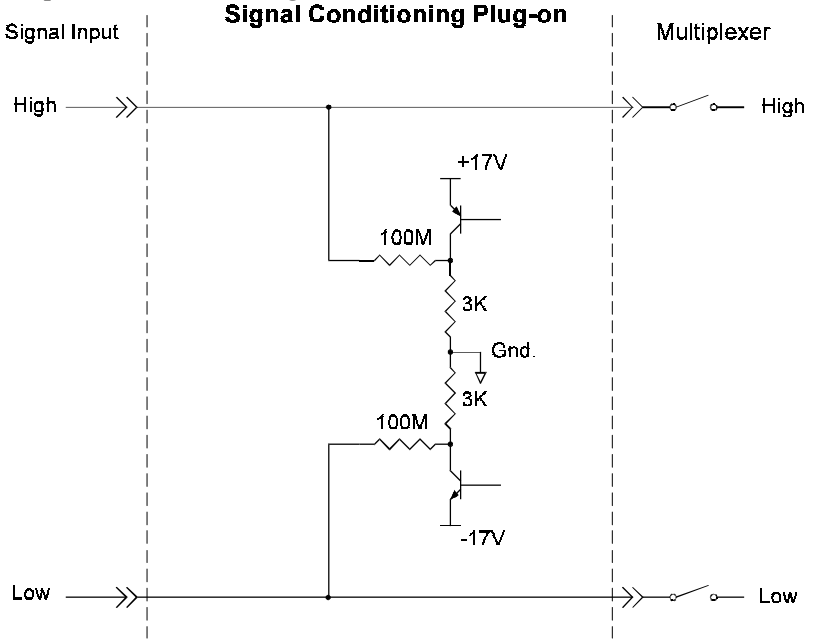


Figure 3-13 Simplified Open Transducer Detect Circuit

NOTES

1) When OTD is enabled, the inputs have up to 0.2µA injected into them. If this current will adversely affect your measurement, but you still want to check for open transducers, you can enable OTD, run your algorithms, check analog input variables for measurement values that indicate an open transducer, then disable OTD and run your algorithms without it. The HP E1419A's accuracy specifications apply only when OTD is off.

2) When a channel's SCP filtering is enabled, allow 15 seconds after turning on OTD for the filters capacitors to charge before checking for open transducers.

To enable or disable Open Transducer Detection, use the DIAGnostic:OTDetect <*enable*>, (@<*ch_list*>) command.

- The *enable* parameter can specify ON or OFF

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- An SCP is addressed when the *ch_list* parameter specifies a channel number contained on the SCP. The first channel on each SCP is:
  0, 8, 16, 24, 32, 40, 48, and 56

To enable Open Transducer Detection on all channels on SCPs 1 and 3:

```
DIAG:OTD ON, (@100,116)          0 is on SCP 1 and 16 is on SCP3
```

To disable Open Transducer Detection on all channels on SCPs 1 and 3:

```
DIAG:OTD OFF, (@100,116)
```

More On Auto Ranging

There are rare circumstances where your input signal can be difficult for the HP E1419A to auto range correctly. The module completes the range selection based on your input signal about 6 µsec before the actual measurement is made on that channel. If during that period your signal becomes greater than the selected range can handle, the module will return an overflow reading (±INFinity).

The only solution to this problem is to use manual range on channels that display this behavior.

Settling Characteristics

Some sequences of input signals as determined by their order of appearance in a scan list can be a challenge to measure accurately. This section is intended to help you determine if your system presents any of these problems and how best to eliminate them or reduce their effect.

Background

While the HP E1419A can auto-range, measure, and convert a reading to engineering units as fast as once every 10 µs, measuring a high level signal followed by a very low level signal may require some extra settling time. As seen from the point of view of the HP E1419A's Analog-to-Digital converter and its Range Amplifier, this situation is the most difficult to measure. For example lets look at two consecutive channels; the first measures a power supply at 15.5 volts, the next measures a thermocouple temperature. First the input to the Range Amplifier is at 15.5 volts (near its maximum) with any stray capacitances charged accordingly, then it immediately is switched to a thermocouple channel and down-ranged to its .0625 volt range. On this range, the resolution is now 1.91 µvolt per Least Significant Bit (LSB). Because of this sensitivity, the time to discharge these stray capacitances may have to be considered.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Thus far in the discussion, we've assumed that the low-level channel measured after a high-level channel has presented a low impedance path to discharge the A/D's stray capacitances (path was the thermocouple wire). The combination of a resistance measurement through an HP E1501 Direct Input SCP presents a much higher impedance path. A very common measurement like this would be the temperature of a thermistor. If measured through a Direct Input SCP, the source impedance of the measurement is essentially the value of the thermistor (the output impedance of the current source is in the gigohm region). Even though this is a higher level measurement than the previous example, the settling time can be even longer due to the slower discharge of the stray capacitances. The simple answer here is to always use an SCP that presents a low impedance buffered output to the HP E1419A's Range Amp and A/D. The HP E1503, 8, 9, 10, 12, and 14 through 17 SCPs all provide this capability.

Checking for Problems

The method we'll use to quickly determine if any of your system's channels needs more settling time is to simply apply some settling time to every channel. Use this procedure:

1. First run your system to make a record of its current measurement performance.
2. Then use the SAMPle:TIMer command to add a significant settling delay to every measurement in the scan list. Take care that the sample time multiplied by the number of channels in the scan list doesn't exceed the time between triggers.
3. Now run your system and look primarily for low level channel measurements (like thermocouples) who's DC value changes somewhat. If you find channels that respond to this increase in sample period, you may also notice that these channels may return slightly quieter measurements as well. The extra sample period reduces or removes the affected channels coupling to the value of the channel measured just before it.
4. If you see some improvement, increase the sample period again and perform another test. When you increase the sample period and no improvement is seen, you have found the maximum settling delay that any single channel requires.
5. If the quality of the measurements does not respond to this increase in sample period, then inadequate settling time is not likely to be causing measurement problems.

Fixing the Problem

If your system scans fast enough with the increased sample period, your problem is solved. Your system is only running as fast as the slowest channel allows but if its fast enough that's OK. If on the other hand, getting

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

quality readings has slowed your scan rate too much, there are two other methods that will, either separately or in combination, have your system making good measurements as fast as possible.

Use Amplifier SCPs

Amplifier SCPs can remove the need to increase settling delays. How? Each gain factor of 4 provided by the SCP amplifier allows the Range Amplifier to be set one range higher and still provide the same measurement resolution. Amplifier SCPs for the HP E1419A are available with gains of .5, 8, 16, 64, and 512. Lets return to our earlier example of a difficult measurement where one channel is measuring 15.5 volts on the 16 volt range, and the next a thermocouple on the .0625 range. If our thermocouple channel is amplified through an SCP with a gain of 16, the Range Amplifier can be set to the 1 volt range. On this range the A/D resolution drops to around 31 µvolt per LSB so the stray capacitances discharging after the 15.5 volt measurement are now only one sixteenth as significant and thus reduce any required settling delay. Of course for most thermocouple measurements we can use a gain of 64 and set the Range Amplifier to the 4 volt range. At this setting the A/D resolution for one LSB drops to about 122 µvolts and further reduces or removes any need for additional settling delay. This improvement is accomplished without any reduction of the overall measurement resolution.

NOTE

Filter-amplifier SCPs can provide improvements in low-level signal measurements that go beyond just settling delay reduction. Amplifying the input signal at the SCP allows using less gain at the Range Amplifier (higher range) for the same measurement resolution. Since the Range Amplifier has to track signal level changes (from the multiplexer) at up to 100 KHz, its bandwidth must be much higher than the bandwidth of individual filter-amplifier SCP channels. Using higher SCP gain along with lower Range Amplifier gain can significantly increase normal-mode noise rejection.

Adding Settling Delay for Specific Channels

This method adds settling time only to individual problem measurements as opposed to the SAMPle:TIMer command that introduces extra time for all analog input channels. If you see problems on only a few channels, you can use the SENS:CHAN:SETTLING <*num_samples*>,(@<*ch_list*>) command to add extra settling time for just these problem channels. What SENS:CHAN:SETTLING does is instruct the HP E1419A to replace single instances of a channel in the Scan List with multiple repeat instances of that channel if it is specified in (@<*ch_list*>). The number of repeats is set by <*num_samples*>.

Example:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Normal Scan List:
100, 101, 102, 103, 104

Scan List after SENS:CHAN:SETT 3,(@100,103)
100, 100, 100, 101, 102, 103, 103, 103, 104

When the algorithms are run, channels 0 and 3 will be sampled 3 times, and the final value from each will be sent to the Channel Input Buffer. This provides extra settling time while channels 1, 2, and 4 are measured in a single sample period and their values also sent to the Channel Input Buffer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 4
The Algorithm Language and Environment

Learning Hint This chapter builds upon the "HP E1419A Programming Model" information presented in Chapter 3. You should read that section before moving on to this one.

About This Chapter

This chapter describes how to write algorithms that apply the HP E1419A's measurement, calculation, and control resources. It describes these resources and how you can access them with the HP E1419A's Algorithm Language. This manual assumes that you have programming experience already. Ideally, you have programmed in the 'C' language since the HP E1419A's Algorithm Language is based on 'C'. Following the tutorial sections of this chapter is an Algorithm Language Reference. The contents of this chapter are:

- Overview of the Algorithm Language 110
- The Algorithm Execution Environment 112
- Accessing the HP E1419A's Resources 113
  - Accessing I/O Channels 114
  - Defining and Accessing Global Variables 115
  - Determining First Execution 115
  - Initializing Variables 116
  - Sending Data to the CVT and FIFO 116
  - Setting a VXIbus Interrupt 117
  - Calling User Defined Functions 118
- Operating Sequence 118
- Defining Algorithms (ALG:DEF) 120
- A Very Simple First Algorithm 124
- Non Control Algorithms 125
  - Data Acquisition Algorithm 125
  - Process Monitoring Algorithm 125
- Algorithm Language Reference 126
  - Standard Reserved Keywords 126
  - Special HP E1419A Reserved Keywords 126
  - Identifiers 126
  - Special Identifiers for Channels 127
  - Operators 127
  - Intrinsic Functions and Statements 128
  - Program Flow Control 128
  - Data Types 129
  - Data Structures 130
  - Type Float as Integer 131

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bitfield Access 131
- Language Syntax Summary 133
- Program Structure and Syntax 138

Overview of the Algorithm Language

The HP E1419A's Algorithm Language is a limited version of the 'C' programming language. It is designed to provide the necessary control constructs and algebraic operations to support measurement and control algorithms. There are no loop constructs, multi-dimensional arrays, or transcendental functions. Further, an algorithm must be completely contained within a single function subprogram 'ALGn'. The algorithm can not call another user-written function subprogram.

It is important to note, that while the HP E1419A's Algorithm Language has a limited set of intrinsic arithmetic operators, it also provides the capability to call special user defined functions "f(x)". The HP VEE example programs "fn_1419.vee" and "eufn1419.vee" in Chapter 5 will convert your functions into piece-wise linear interpolated tables and give them names you choose. The HP E1419A can extract function values from these tables in under 18µseconds regardless of the function's original complexity. This method provides faster algorithm execution by moving the complex math operations off-board.

This section assumes that you already program in some language. If you are already a 'C' language programmer, this chapter is all you'll probably need to create your algorithm. If you are not familiar with the C programming language, you should study the "Program Structure and Syntax" section before you begin to write your custom algorithms.

This section will present a quick look at the Algorithm Language. The complete language reference is provided later in this chapter.

Arithmetic Operators: add **+**, subtract **-**, multiply *****, divide **/**
NOTE: Also see "Calling User Defined Functions" on page 4-118.

Assignment Operator: =

Comparison Functions: less than **<**, less than or equal **<=**, greater than **>**, greater than or equal **>=**, equal to **==**, not equal to **!=**

Boolean Functions: and **&&**, or **||**, not **!**

Variables: scalars of type **`static float`**, and single dimensioned arrays of type **`static float`** limited to 1024 elements.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Constants:
32-bit decimal integer; **Dddd...** where **D** and **d** are decimal digits but **D** is not zero. No decimal point or exponent specified.
32-bit octal integer; **0oo...** where **0** is a leading zero and **o** is an octal digit. No decimal point or exponent specified.
32-bit hexadecimal integer; **0Xhhh...** or **0xhhh...** where **h** is a hex digit.
32-bit floating point; **ddd.**, **ddd.ddd**, **ddde±dd**, **dddE±dd**, **ddd.dddedd**, or **ddd.dddEdd** where d is a decimal digit.

Flow Control: conditional construct **if(){ } else { }**

Intrinsic Functions:
Return absulute value; **abs(<*expr1*>)**
Return minimum; **min(<*expr1*>,<*expr2*>)**
Return maximum; **max(<*expr1*>,<*expr2*>)**
User defined function; **<*user_name*>(<*expr*>)**
Write value to CVT element; **writecvt(<*expr*>,<*expr*>)**
Write value to FIFO buffer; **writefifo(<*expr*>)**
Write value to both CVT and FIFO; **writeboth(<*expr*>,<*expr*>)**

Example Language Usage

Here are examples of some Algorithm Language elements assembled to show them used in context. Later sections will explain any unfamiliar elements you see here:

Example 1;

```
/*** get input from channel 8, calculate output, check limits, output to ch 40 & 41 ***/
static float output_max = .020;          /* 20 mA max output */
static float output_min = .004;          /* 4 mA min output */
static float input_val, output_val;      /* intermediate I/O vars */

input val_ = I108;                       /* get value from input buffer channel 8*/
output_val = 12.5 * input_val;           /* calculate desired output */
if ( output_val > output_max )           /* check output greater than limit */
      output_val = output_max;           /* if so, output max limit */
else if( output_val < output_min)        /* check output less than limit */
      output_val = output_min;           /* if so, output min limit */
O140 = output_val / 2;                   /* split output_val between two SCP  */
O141 = output_val / 2;                   /* channels to get up to 20mA max    */
```

Example 2;

```
/*** same function as example 1 above but shows a different approach ***/
static float max_output = .020;          /* 20 mA max output */
static float min_output = .004;          /* 4 mA min output */

/* following lines input, limit output between min and max_output, and outputs.    */
/* output is split to two current output channels wired in parallell to provide 20mA */
O140 = max( min_output, min( max_output, (12.5 * I108) / 2 ) );
O141 = max( min_output, min( max_output, (12.5 * I108) / 2 ) );
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Algorithm Execution Environment

This section describes the execution environment that the HP E1419A provides for your algorithms. Here we describe the relationship of your algorithm to the **main()** function that calls it.

The Main Function

All ’C’ language programs consist of one or more functions. A ’C’ program must have a function called **main()**. In the HP E1419A, the **main()** function is usually generated automatically by the driver when you execute the INIT command. The **main()** function executes each time the module is triggered, and controls execution of your algorithm functions. See Figure 4-1 for a partial listing of **main()**.

How Your Algorithms Fit In

When the module is INITiated, a set of control variables and a function calling sequence is created for all algorithms you have defined. The value of variable "State_n" is set with the ALGorithm:STATe command and determines whether the algorithm will be called. The value of "Ratio_n" is set with the ALGorithm:SCAN:RATio command and determines how often the algorithm will be called (relative to trigger events).

Since the function-calling interface to your algorithms is fixed in the **main()** function, the "header" of your algorithm function is also pre-defined. This means that unlike standard ’C’ language programming, your algorithm program (a function) need not (must not) include the function declaration header, opening brace "{", and closing brace "}". You only supply the "body" of your function, the HP E1419A’s driver supplies the rest.

Think of the program space in the HP E1419A in the form of a source file with any global variables first, then the **main()** function followed by as many algorithms as you have defined. Of course what is really contained in the HP E1419A’s algorithm memory are executable codes that have been translated from your downloaded source code. While not an exact representation of the algorithm execution environment, Figure 4-1 shows the relationship between a normal ’C’ program and two HP E1419A algorithms.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

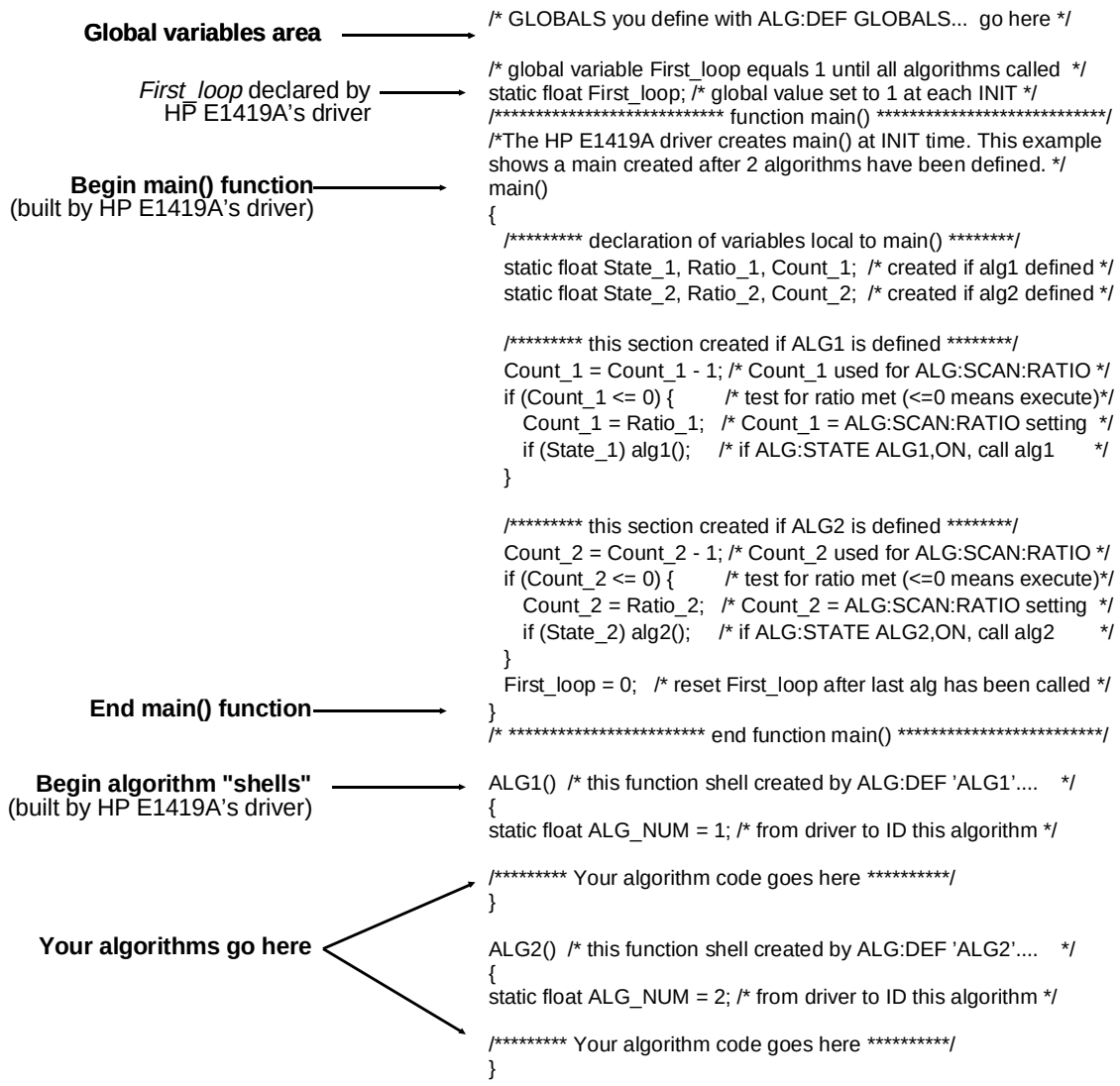


Figure 4-1 Source Listing of Function main()

Accessing the E1419A's Resources

This section describes how your algorithm accesses hardware and software resources provided by the HP E1419A. The following is a list of these resources:

- I/O channels.
- Global variables defined before your algorithm is defined.
- The value ALG_NUM which the HP E1419A makes available to your algorithm. ALG_NUM = 1 for ALG1, 2 for ALG2 etc.
- User defined functions defined with the ALG:FUNC:DEF command.
- The Current Value Table (CVT), and the data FIFO buffer (FIFO) to output algorithm data to your application program.
- VXIbus Interrupts.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Accessing I/O Channels

In the Algorithm Language, channels are referenced as pre-defined variable identifiers. The general channel identifier syntax is "Iccc" for input channels and "Occc" for output channels; where ccc is a channel number from 100 (channel 0) through 163 (channel 63). Like all HP E1419A variables, channel identifier variables always contain 32-bit floating point values even when the channel is part of a digital I/O SCP. If the digital I/O SCP has 8-bit channels (like the HP E1533), the channel's identifiers (Occc and Iccc) can take on the values 0 through 255. To access individual bit values you may append ".Bn" to the normal channel syntax; where n is the bit number (0 through 7). If the Digital I/O SCP has single-bit channels (like the HP E1534), its channel identifiers can only take on the values 0 and 1. Examples:

O148 = 1;	*assign value to output chan 0 on HP E1534.*
Inp_val = I160;	*from 8-bit channel on HP E1533 Inp_val will be 0. to 255.*
Bit_4 = I156.B4;	*assign HP E1533 chan 56 bit 4 to variable Bit_4*

Output Channels

Output channels can appear on either or both sides of an assignment operator. They can appear anywhere other variables can appear. Examples:

O132= 12.5;	*send value to output channel buffer element 0*
O156.B4 = ! O156.B4;	*compliment value found in output channel buffer element 56, bit 4 each time algorithm is executed.*
writecvt(O132,32);	*send value of output channel 132 to CVT element 32*

Input Channels

Input channel identifiers can only appear on the right side of assignment operators. It doesn't make sense to output values to an input channel. Other than that, they can appear anywhere other variables can appear. Examples:

dig_bit_value = I157.B0;	*retrieve value from Input Channel Buffer element 57, bit 0*
inp_value = I124;	*retrieve value from Input Channel Buffer element 24*
O132 = 4 * I124;	*retrieve value from Input Channel Buffer element 24, multiply by 4 and send result to Output Channel Buffer element 32*
writefifo(I124);	*send value of input channel 24 to FIFO buffer*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Defined Input and Output Channels

Your algorithm "references" channels. It can reference input or output channels. But, in order for these channels to be available to your algorithm they must be "defined". What we mean by "defined" is that an SCP must be installed, and an appropriate SOURce or SENSe :FUNCtion must explicitly (or implicitly, in the case of HP E1531, 32, and HP E1536 SCPs) be tied to the channels. If your algorithm references an input channel identifier that is not configured as an input channel, or an output channel identifier that is not configured as an output channel, the driver may generate an error when your algorithm is defined with ALG:DEF.

Defining and Accessing Global Variables

Global variables are those declared outside of the **main()** function and any algorithms (see Figure 4-1). A global variable can be read or changed by any algorithm. To declare global variables you use the command:

ALG:DEF 'GLOBALS','<*source_code*>'

where <*source_code*> is Algorithm Language source limited to constructs for declaring variables. It must not contain executable statements. Examples:

declare single variable without assignment;

```
ALG:DEF 'GLOBALS','static float glob_scal_var;'
```

declare single variable with assignment;

```
ALG:DEF 'GLOBALS','static float glob_scal_var = 22.53;'
```

declare one scalar variable and one array variable;

```
ALG:DEF 'GLOBALS','static float glob_scal_var, glob_array_var[12];'
```

You access global variables within your algorithm like any other variable.

```
glob_scal_var = P_factor * I108;
```

NOTES

1. All variables must be declared `static float`.
2. Array variables cannot be assigned a value when declared.
3. All variables declared within your algorithm are local to that algorithm. If you locally declare a variable with the same identifier as an existing global variable, your algorithm will access the local variable only.

Determining First Execution (First_loop)

The HP E1419A always declares the global variable *First_loop*. *First_loop* is set to 1 each time INIT is executed. After **main()** calls all enabled algorithms, it sets *First_loop* to 0. By testing *First_loop*, your algorithm can determine if it is being called for the first time since an INITiate command was received. Example:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
static float scalar_var;
static float array_var [ 4 ];

/* assign constants to variables on first pass only */
if ( First_loop )
{
        scalar_var = 22.3;
        array_var[0] = 0;
        array_var[1] = 0;
        array_var[2] = 1.2;
        array_var[3] = 4;
}
```

Initializing Variables

Variable initialization can be performed during three distinct HP E1419A operations.

1. When you define algorithms with the ALG:DEFINE command. A declaration initialization statement is a command to the driver's translator function and doesn't create an executable statement. The value assigned during algorithm definition is not re-assigned when the algorithm is run with the INIT command. Example statement:

   ```
   static float my_variable = 22.95;/* tells translator to allocate space for this */
                                    /* variable and initialize it to 22.95          */
   ```

2. Each time the algorithm executes. By placing an assignment statement within your algorithm. This will be executed each time the algorithm is executed. Example statment.

   ```
   my_variable = 22.95;              /* reset variable to 22.95 every pass        */
   ```

3. When the algorithm first executes after an INIT command. By using the global variable *First_loop* your algorithm can distinguish the first execution since an INIT command was sent. Example statement:

   ```
   if( First_loop ) my_variable = 22.95;  /* reset variable only when INIT starts alg */
   ```

Sending Data to the CVT and FIFO

The Current Value Table (CVT) and FIFO data buffer provide communication from your algorithm to your application program (running in your VXIbus controller).

Writing a CVT element

The CVT provides 502 addressable elements where algorithm values can be stored. To send a value to a CVT element, you will execute the intrinsic Algorithm Language statement **writecvt(<*expression*>,<*cvt_element*>)**, where <*cvt_element*> can take the value 10 through 511. The following is an example algorithm statement:

```
writecvt(O136, 330);  /* send output channel 36's value to CVT element 330 */
```

Each time your algorithm writes a value to a CVT element the previous value in that element is overwritten.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Reading CVT elements

Your application program reads one or more CVT elements by executing the SCPI command **[SENSe:]DATA:CVT? (@<*element_list*>)**, where <*element_list*> specifies one or more individual elements and/or a range of contiguous elements. The following example command will help to explain the <*element_list*> syntax.

DATA:CVT? (@10,20,30:33,40:43,330) *Return elements 10, 20, 30-33, 40-43. and element 330.*

Individual element numbers are isolated by commas. A contiguous range of elements is specified by: <starting element>colon<ending element>.

Writing values to the FIFO

The FIFO, as the name implies is a First-In-First-Out buffer. It can buffer up to 65,024 values. This capability allows your algorithm to send a continuous stream of data values related in time by their position in the buffer. This can be thought of as an electronic strip-chart recorder. Each value is sent to the FIFO by executing the Algorithm Language intrinsic statement **writefifo(<*expression*>)**. The following in an example algorithm statement:

```
writefifo(O139);  /* send output channel 39's value to the FIFO */
```

Since you can determine the actual algorithm execution rate (see "Programming the Trigger Timer" on page 3-80), the time relationship of readings in the FIFO is very deterministic.

Reading values from the FIFO

For a discussion on reading values from the FIFO, see "Retrieving Algorithm Data" on page 3-83.

Writing values to the FIFO and CVT

The **writeboth(<*expression*>,<*cvt_element*>)** statement sends the value of <*expression*> both to the FIFO and to a <*cvt_element*>. Reading these values is done the same way as mentioned for **writefifo()** and **writecvt()**.

Setting a VXIbus Interrupt

The algorithm language provides the function **interrupt()** to force a VXIbus interrupt. When **interrupt()** is executed in your algorithm, a VXIbus interrupt line (selected by the the SCPI command DIAG:INTR[:LINe]) is asserted. The following example algorithm code tests an input channel value and sets an interrupt if it is higher or lower than set limits.

```
static float upper_limit = 1.2, lower_limit = 0.2;
if( I124 > upper_limit || I124 < lower_limit ) interrupt();
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Calling User Defined Functions

Access to user defined functions is provided to avoid complex equation calculation within your algorithm. Essentially what is provided with the HP E1419A is a method to pre-compute user function values outside of algorithm execution and place these values in tables, one for each user function. Each function table element contains a slope and offset to calculate an mx+b over the interval (x is the value you provide to the function). This allows the DSP to linearly interpolate the table for a given input value and return the function's value much faster than if a transcendental function's equation were arithmetically evaluated using a power series expansion.

User functions are defined by downloading function table values with the ALG:FUNC:DEF command and can take any name that is a valid 'C' identifier like 'haversine', 'sqr', 'log10' etc. To find out how to generate table values from your function equation, see the HP VEE example program "fn_1419.vee" in Chapter 5 page 5-160. For details on the ALG:FUNC:DEF command, see page 6-201 in the Command Reference.

User defined functions are global in scope. A user function defined with ALG:FUNC:DEF is available to all defined algorithms. Up to 32 functions can be defined in the HP E1419A. You call your function with the syntax <func_name>(<expression>). Example:

for user function pre-defined as square root with name 'sqrt'

```
O132 = sqrt( I100);  /* channel 32 outputs square root of input channel 0's value */
```

NOTE A user function must be defined (ALG:FUNC:DEF) before any algorithm is defined (ALG:DEF) that references it.

Operating Sequence

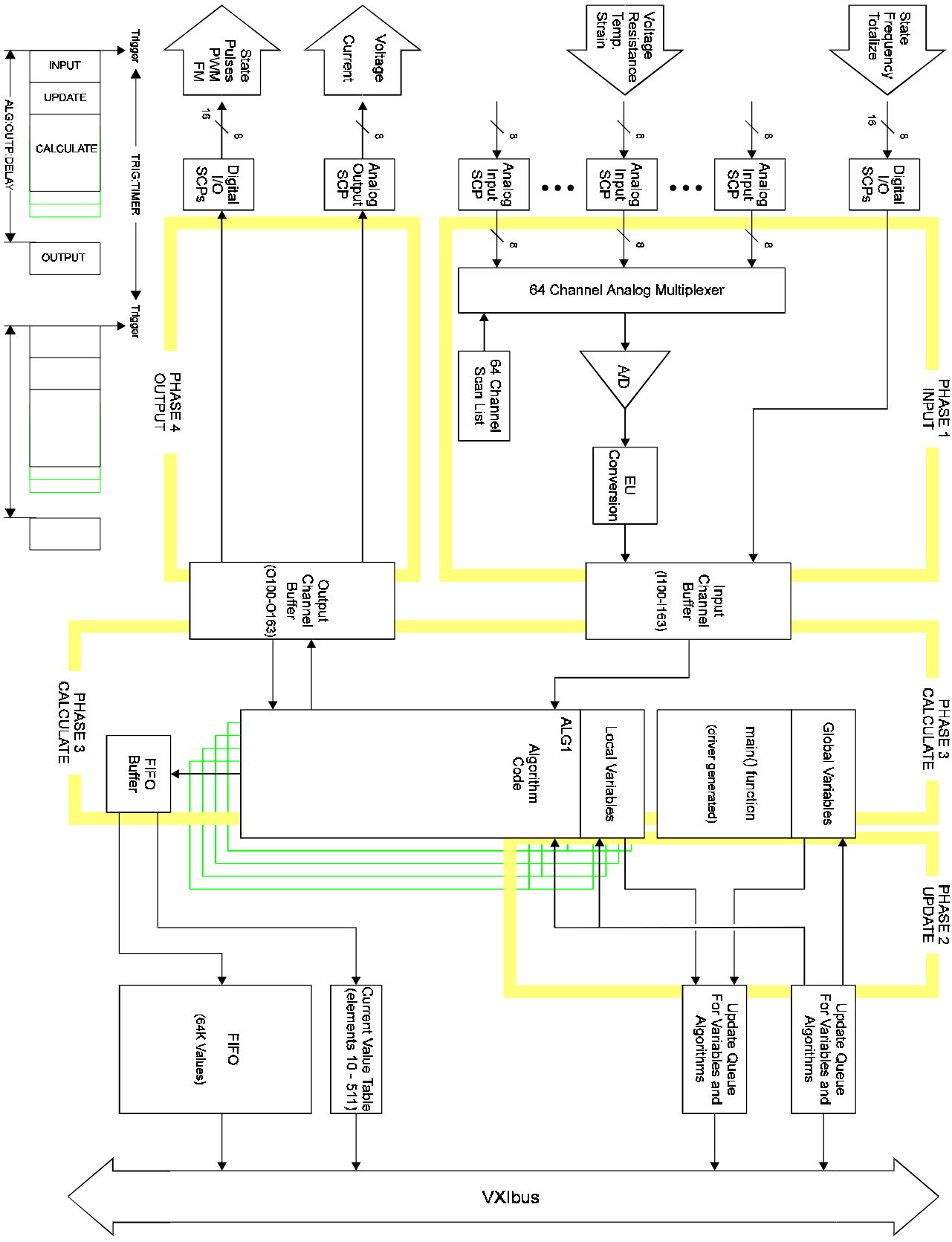
This section explains another important factor in your algorithm's execution environment. Figure 4-2 shows the same overall sequence of operations that you saw in Chapter 3, but also includes a block diagram to show you which parts of the HP E1419A are involved in each phase of the control sequence.

Overall Sequence

Here, the important things to note about this diagram are:

- All algorithm referenced input channel values are stored in the Channel Input Buffer (Input Phase) BEFORE algorithms are executed during the Calculate Phase.
- The execution of all defined algorithms (Calculate Phase) is complete BEFORE output values from algorithms, stored in the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



4-2 Algorithm Operating Sequence Diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Channel Output Buffer, are used to update the output channel hardware during the Output Phase.

In other words, algorithms don't actually read inputs at the time they reference input channels, and they don't send values to outputs at the time they reference output channels. Algorithms read channel values from an input buffer, and write (and can read) output values to/from an output buffer. Here are example algorithm statements to describe operation:

```
inp_val = I108;     /* inp_val is assigned a value from input buffer element 8   */
O137 = 22.3;        /* output buffer element 37 assigned the value 22.3          */
O133 = O132;        /* output buffer [32] is read and assigned to output buffer [33] */
```

A Common Error to Avoid

Since the "buffered input - algorithm execution - buffered output" sequence is probably not a method many are familiar with, a programming mistake associated with it is easy to make. Once you see it here, you won't do this in your programs. The following algorithm statements will help explain:

```
O156.B0 = 1;        /* digital output bit on HP E1533 in SCP position 3 */
O156.B0 = 0;
```

Traditionally you expect the first of these two statements to set output channel 56 bit 0 to a digital 1, then after the time it takes to execute the second statement, the bit would return to a digital 0. Because both of these statements are executed BEFORE any values are sent to the output hardware, only the last statement has any effect. Even if these two statements were in separate algorithms, the last one executed would determine the output value. In this example the bit would never change. The same applies to analog outputs.

Algorithm Execution Order

The buffered I/O sequence explained previously can be used to advantage. Multiple algorithms can access the very same buffered channel input value without having to pass the value in a parameter. Any algorithm can read and use as its input, the value that any other algorithm has sent to the output buffer. In order for these features to be of use, you must know the order in which your algorithms will be executed. When you define your algorithms, you give them one of 32 pre-defined algorithm names. These range from 'ALG1' to ALG32'. Your algorithms will execute in order of its name. For instance if you define 'ALG5', then 'ALG2', then 'ALG8', and finally 'ALG1', when you run them, they will execute in the order 'ALG1', 'ALG2', 'ALG5', and 'ALG8'.

Defining Algorithms (ALG:DEF)

This section discusses how to use the ALG:DEFINE command to define algorithms. Later sections will discuss "what to define".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ALG:DEFINE in the Programming Sequence

*RST erases all previously defined algorithms. You must erase all algorithms before you begin to re-define them (except in the special case described in "Changing an Algorithm While it's Running" later in this section).

ALG:DEFINE's Two Data Formats

For algorithms, the ALG:DEFINE '<*alg_name*>','<*source_code*>' command sends the algorithm's source code to the HP E1419A's driver for translation to executable code. The <*source_code*> parameter can be sent in one of three forms:

1. SCPI Quoted String: For short segments (single lines) of code, enclose the code string within single (apostrophes), or double quotes. Because of string length limitations within SCPI and some programming platforms, we recommend that the quoted string length not exceed a single program line. Example:

   ALG:DEF 'ALG1','if(First_loop) O132=0; O132=O132+.01;'

2. SCPI Indefinite Length Block Program Data: This form terminates the data transfer when it received an End Identifier with the last data byte. Use this form only when you are sure your controller platform will include the End Identifier. If it is not included, the ALG:DEF command will "swallow" whatever data follows the algorithm code. The syntax for this parameter type is:

   #0<data byte(s)><null byte with End Identifier>

   Example from "Quoted String" above:

ALG:DEF 'ALG1',#0O132=I100;LF/EOI

where LF/EOI is a line-feed character sent with End Identifier true (EOI signal for HP-IB)

NOTE for C-SCPI

For Block Program Data, the Algorithm Parser requires that the *source_code* data end with a null (0) byte. You must append the null byte to the end of the block's <data byte(s)>. If the null byte is not included within the block, the error "Algorithm Block must contain termination '\0'" will be generated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Changing an Algorithm While it's Running

The HP E1419A has a feature that allows you to specify that a given algorithm can be swapped with another even while it is executing. This is useful if, for instance, you needed to alter the function of an algorithm that is currently controlling a process and you don't want to leave that process uncontrolled. In this case, when you define the original algorithm, you can enable it to be swapped.

Defining an Algorithm for Swapping

The ALG:DEF command has an optional parameter that is used to enable algorithm swapping. The command's general form is:

ALG:DEF '<*alg_name*>'[,<*swap_size*>],'<*source_code*>'

Note the parameter <*swap_size*>. With <*swap_size*> you specify the amount of algorithm memory to allocate for algorithm <*alg_name*>. Make sure to allocate enough space for the largest algorithm you expect to define for <*alg_name*>. Here is an example of defining an algorithm for swapping:

define ALG3 so it can be swapped with an algorithm as large as 1000 words

```
ALG:DEF 'ALG3',1000,#41698<1698char_alg_source>
```

NOTE The number of characters (bytes) in an algorithm's <*source_code*> parameter is not well related to the amount of memory space the algorithm requires. Remember this parameter contains the algorithm's source code, not the executable code it will be translated into by the ALG:DEF command. Your algorithm's source might contain extensive comments, none of which will be in the executable algorithm code after it is translated.

How Does it Work?

We'll use the example algorithm definition above for this discussion. When you specify a value for <*swap_size*> at algorithm definition, the HP E1419A allocates two identical algorithm spaces for ALG3, each the size specified by <*swap_size*> (in this example 1000 words). This is called a "double buffer". We'll just call these space A and space B. The algorithm is loaded into ALG3's space A at first definition. Later, while algorithms are running you can "replace" ALG3 by again executing

```
ALG:DEF ALG3,#42435<2435char_alg_source>
```

Notice that <swap_size> is not (must not be) included this time. This ALG:DEF works like an Update Request. The HP E1419A translates and downloads the new algorithm into ALG3's space B while the old ALG3 is still running from space A. When the new algorithm has been completely loaded into space B and an ALG:UPDATE command has been sent, the HP E1419A simply switches to executing ALG3's new algorithm from space B at the next Update Phase (see Figure 3-8). If you were to send yet another ALG3, it would be loaded and executed from ALG3's space A.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Determining an Algorithm's Size

In order to define an algorithm for swapping, you will need to know how much algorithm memory to allocate for it or any of its replacements. You can query this information from the HP E1419A. Use the following sequence:

1. Define the algorithm without swapping enabled. This will cause the HP E1419A to allocate only the memory actually required by the algorithm.
2. Execute the ALG:SIZE? <*alg_name*> command to query the amount of memory allocated. You now know the minimum amount of memory required for the algorithm.
3. Repeat 1 and 2 for each of the algorithms you want to be able to swap with the original. From this you know the minimum amount of memory required for the largest.
4. Execute *RST to erase all algorithms.
5. Re-define one of the algorithms with swapping enabled and specify <swap_size> at least as large as the value from step 3 above (and probably somewhat larger because as alternate algorithms declare different variables, space is allocated for total of all variables declared).
6. Swap each of the alternate algorithms for the one defined in step 5, ending with the one you want to run now. Remember, you don't send the <swap_size> parameter with these. If you don't get an "Algorithm too big" error, then the value for <swap_size> in step 5 was large enough.
7. Define any other algorithms in the normal manner.

NOTES

1. Channels referenced by algorithms when they are defined, are only placed in the channel list before INIT. The channel list cannot be changed after INIT. If you re-define an algorithm (by swapping) after INIT, and it references channels not already in the channel list, these channels will only return a floating point zero. No error message will be generated. To make sure all required channels will be included in the channel list, define <*alg_name*> and re-define all algorithms that will replace <*alg_name*> by swapping them before you send INIT. This insures that all channels referenced in these algorithms will be available after INIT.
2. The driver only calculates overall execution time for algorithms defined before INIT. This calculation is used to set the default output delay (same as executing ALG:OUTP:DELAY AUTO). If an algorithm is swapped after INIT that take longer to execute than the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

original, the output delay will behave as if set by ALG:OUTP:DEL 0, rather than AUTO (see ALG:OUTP:DEL command). Use the same procedure from note 1 to make sure the longest algorithm execution time is used to set ALG:OUTP:DEL AUTO before INIT.

The HP VEE example program "swap1419.vee" shows how to swap algorithms while the module is running. See Chapter 5 page 172.

A Very Simple First Algorithm

This section will show you how to create and download an algorithm that simply sends the value of an input channel to a CVT element. It includes an example application program that configures the HP E1419A, downloads (defines) the algorithm, starts and then communicates with the running algorithm.

Writing the Algorithm

The most convenient method of creating your algorithm is to use your text editor or word processor to input the source code.

```
/* Example algorithm that calculates 4 Mx+B values upon
 * signal that sync == 1.  Your application sets sync with the
 * SCPI command ALG:SCALAR
 * M and B terms are also set by application
 * program.
 */
static float M, B, x, sync;
if ( First_loop ) sync = 0;
if ( sync == 1 ) {
        writecvt(  M*x+B,   10 );
        writecvt(-(M*x+B),  11 );
        writecvt( (M*x+B)/2,12 );
        writecvt( 2*(M*x+B),13 );
        sync = 2;
}
```

Running the Algorithm

The supplied HP VEE example program "temp1419.vee" shows you how to load and run algorithms. See Chapter 5 page 153.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Non-Control Algorithms

Data Acquisition Algorithm

The HP E1419A's Algorithm Language includes intrinsic functions to write values to the CVT, the FIFO, or both. Using these functions, you can create algorithms that simply perform a data acquisition function. The following example shows acquiring eight channels of analog input from SCP position 0, and one channel (8 bits) of digital input from an HP E1533 in SCP position 7. The results of the acquisition are placed in the CVT and FIFO.

```
/* Data acquisition to CVT and FIFO */
writeboth( I100, 330 ); /* channel 0 to FIFO, and CVT element 330 */
writeboth( I101, 331 ); /* channel 1 to FIFO, and CVT element 331 */
writeboth( I102, 332 ); /* channel 2 to FIFO, and CVT element 332 */
writeboth( I103, 333 ); /* channel 3 to FIFO, and CVT element 333 */
writeboth( I104, 334 ); /* channel 4 to FIFO, and CVT element 334 */
writeboth( I105, 335 ); /* channel 5 to FIFO, and CVT element 335 */
writeboth( I106, 336 ); /* channel 6 to FIFO, and CVT element 336 */
writeboth( I107, 337 ); /* channel 7 to FIFO, and CVT element 337 */
writeboth( I156, 338 ); /* channel 56 to FIFO, and CVT element 338 */
```

Using SENS:DATA:FIFO: ... and the SENS:DATA:CVT commands, your application program can access the data.

Process Monitoring Algorithm

Another function the HP E1419A performs well is monitoring input values and testing them against pre-set limits. If an input value exceeds its limit, the algorithm can be written to supply an indication of this condition by changing a CVT value, or even forcing a VXIbus interrupt. The following example shows acquiring one analog input value from channel 0, and one HP E1533 digital channel from channel 56, and limit testing them.

```
/* Limit test inputs , send values to CVT, and force interrupt when exceeded */
static float    Exceeded;   /* Exceeded is set by boolean operations to either 0 or 1 */
static float    Max_chan0, Min_chan0, Max_chan1, Min_chan1;
static float    Max_chan2, Min_chan2, Max_chan3, Min_chan3;
static float    Mask_chan56;
if ( First_loop ) Exceeded = 0;  /* initialize Exceeded on each INIT */
writecvt( I100, 330); /* write analog value to CVT */
Exceeded = ( ( I100 > Max_chan0 ) || ( I100 < Min_chan0 ) ); /* limit test analog */
writecvt( I101, 331); /* write analog value to CVT */
Exceeded = Exceeded + ( ( I101 > Max_chan1 ) || ( I101 < Min_chan1 ) );
writecvt( I102, 332); /* write analog value to CVT */
Exceeded = Exceeded + ( ( I102 > Max_chan2 ) || ( I102 < Min_chan2 ) );
writecvt( I103, 333); /* write analog value to CVT */
Exceeded = Exceeded + ( ( I103 > Max_chan3 ) || ( I103 < Min_chan3 ) );
writecvt( I156, 334); /* write 8-bit value to CVT */
Exceeded = Exceeded + ( I156 != Mask_chan56); /* limit test digital */
If ( Exceeded ) interrupt( );
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Algorithm Language Reference

This section provides a summary of reserved keywords, operators, data types, constructs, intrinsic functions and statements.

Standard Reserved Keywords

The list of reserved keywords is the same as ANSI 'C'. You may not create your own variables using these names. Note that the keywords that are shown underlined and bold are the only ANSI 'C' keywords that are implemented in the HP E1419A.

auto	double	int	struc
break	**else**	long	switch
case	enum	register	typeof
char	extern	**return**	union
const	**float**	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	**if**	**static**	while

NOTE While all of the ANSI 'C' keywords are reserved, only those keywords that are shown in bold are actually implemented in the HP E1419A.

Special HP E1419A Reserved Keywords

The HP E1419A implements some additional reserved keywords. You may not create variables using these names:

abs	interrupt	writeboth
Bn (n=0 through 9)	max	writecvt
Bnn (nn=10 through 15)	min	writefifo

Identifiers

Identifiers (variable names) are significant to 31 characters. They can include alpha, numeric, and the underscore character "_". Names must begin with an alpha character, or the underscore character.

Alpha: a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Numeric: 0 1 2 3 4 5 6 7 8 9
Other: _

NOTE Identifiers are case sensitive. The names My_array and my_array reference different identifiers.

Special Identifiers for Channels

Channel identifiers appear as variable identifiers within the algorithm and have a fixed, reserved syntax. The identifiers I100 to I163 specify input channel numbers. The "I" must be upper case. They may only appear on the right side of an assignment operator. The identifiers O100 to O163 specify output channel numbers. The "O" must be upper case. They can appear on either or both sides of the assignment operator.

NOTE Trying to declare a variable with a channel identifier will generate an error.

Operators

The HP E1419A's Algorithm Language supports the following operators:

Assignment Operator

=	(assignment)	example;	c = 1.2345

Arithmetic Operators

+	(addition)	examples;	c = a + b
-	(subtraction)		c = a - b
*	(multiplication)		c = a * b
/	(division)		c = a / b

Unary Operators

-	(unary minus)		c = a + (-b)
+	(unary plus)		c = a + (+b)

Comparison Operators

==	(is equal to)	examples;	a == b
!=	(is not equal to)		a != b
<	(is less than)		a < b
>	(is greater than)		a > b
<=	(is less than or equal to)		a <= b
>=	(is greater than or equal to)		a >= b

Logical Operators

||	(or)	examples;	(a == b) || (a == c)
&&	(and)		(a == b) && (a == c)

Unary Logical Operator

!	(not)	example;	!b

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The result of a comparison operation is a boolean value. It is still a type **float** but its value is either 0 (zero) if false, or 1 (one) if true. You may test any variable with the **if** statement. A value of zero tests false, if any other value it tests true. For example:

```
/* if my_var is other than 0, increment count_var */
 if(my_var) count_var=count_var+1;
```

Intrinsic Functions and Statements

The following functions and statements are provided in the HP E1419A's Algorithm Language:

Functions:

abs(*expression*)	return absolute value of expression
max(*expression1*,*expression2*)	return largest of the two expressions
min(*expression1*,*expression2*)	return smallest of the two expressions

Statements:

interrupt()	sets VXI interrupt
writeboth(*expression*,*cvt_loc***)**	write expression result to FIFO and CVT element specified.
writecvt(*expression*,*cvt_loc***)**	write expression result to CVT element specified.
writefifo(*expression***)**	write expression result to FIFO.

Program Flow Control

Program flow control is limited to the conditional execution construct using **if** and **else**, and **return**. Looping inside an algorithm function is not supported. The only "loop" is provided by repeatedly triggering the HP E1419A. Each trigger event (either external, or internal Trigger Timer) executes the **main()** function which calls each defined and enabled algorithm function. There is no **goto** statement.

Conditional Constructs

The HP E1419A Algorithm Language provides the **if-else** construct in the following general form:

if (*expression***)** *statement1* **else** *statement2*

If *expression* evaluates to non-zero *statement1* is executed. If *expression* evaluates to zero, *statement2* is executed. The else clause with its associated *statement2* is optional. Statement1 and/or statement2 can be compound statement. That is { *statement; statement; statement; ...* }.

Exiting the Algorithm

The **return** statement allows terminating algorithm execution before reaching the end by returning control to the main() function. The **return**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

statement can appear anywhere in your algorithm. You are not required to include a **return** statement to end an algorithm. The translator treats the end of your algorithm as an implied return.

Data Types

The data type for variables is always **static float**. However decimal constant values without a decimal point or exponent character (".", "E", or "e"), as well as Hex and Octal constants are treated as 32-bit integer values. This treatment of constants is consistent with ANSI 'C'. To understand what this can mean you must understand that not all arithmetic statements in your algorithm are actually performed within the HP E1419A's DSP chip at algorithm run-time. Where expressions can be simplified, the HP E1419A's translator (a function of the driver invoked by ALG:DEF) performs the arithmetic operations before downloading the executable code to the algorithm memory in the HP E1419A. For example look at the statement;

```
a = 5 + 8;
```

When the HP E1419A's translator receives this statement, it simplifies it by adding the two integer constants (5 and 8) and storing the sum of these as the float constant 13. At algorithm run-time, the float constant 13 is assigned to the variable "a". No surprises so far. Now analyze this statement;

```
a = ( 3 / 4 ) * 12;
```

Again the translator simplifies the expression by performing the integer divide for 3 / 4. This results in the integer value 0 being multiplied by 12 which results in the float constant 0.0 being assigned to the variable "a" at run-time. This is obviously not what you wanted but is exactly what your algorithm instructed.

You can avoid these subtle problems by specifically including a decimal point in decimal constants where an integer operation is not what you want. For example, if you had made either of the constants in the division above a float constant by including a decimal point, the translator would have promoted the other constant to a float value and performed a float divide operation resulting in the expected 0.75 * 12, or the value 9.0 So the statement;

```
a = ( 3. / 4 ) * 12;
```

will result in the value float 9.0 being assigned to the variable "a".

The Static Modifier

All HP E11419A variables, local or global, must be declared as **static**. An example:

```
static float gain_var, integer_var, deriv_var;  /* three vars declared */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In 'C', local variables that are not declared as **static** lose their values once the function completes. The value of a local **static** variable remains unchanged between calls to your algorithm. Treating all variables this way allows your algorithm to "remember" its previous state. The static variable is local in scope, but otherwise behaves as a global variable. Also note that you may not declare variables within a compound statement.

Data Structures

The HP E1419A Algorithm Language allows the following data structures:

- Simple variables of type **float**:
  Declaration
  ```
  static float simp_var, any_var;
  ```
  Use
  ```
  simp_var = 123.456;
  any_var = -23.45;
  Another_var = 1.23e-6;
  ```
  Storage
  Each simple variable requires four 16-bit words of memory.

- Single-dimensioned arrays of type **float** with a maximum of 1024 elements:
  Declaration
  ```
  static float array_var [3];
  ```
  Use
  ```
  array_var [0] = 0.1;
  array_var [1] = 1.2;
  array_var [2] = 2.34;
  array_var [3] = 5;
  ```
  Storage
  Arrays are "double buffered". This means that when you declare an array, twice the space required for the array is allocated, plus one more word as a buffer pointer. The memory required is:

  $$words\ of\ memory = (8 * num_elements) + 1$$

  This double buffered arrangement allows the ALG:ARRAY command to download all elements of the array into the "B" buffer while your algorithm is accessing values from the "A" buffer. Then an ALG:UPDATE command will cause the buffer pointer word to point to the newly loaded buffer between algorithm executions.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using Type Float in Integer Situations

There are certain situations where you would normally use integers, but with the HP E1419A you only have type float. This usually has to do with writing values to digital SCP channels. With the HP E1533 Digital I/O SCP, each channel (two per SCP) reads or writes 8 bits. With the HP E1534 and HP 1536 SCPs, each channel (eight per SCP) reads or writes 1 bit. Note the following behavior when sending values to digital channels:

- A floating point number sent to a digital channel is truncated to integer by dropping any fractional portion. So, the value 234.8 sent to an 8 bit channel will have the same effect as the integer value 234. The value 0.8 sent to a 1 bit channel will be evaluated as 0. The value 1.9 becomes 1.

- The HP E1419A treats values sent to a digital channel as signed. that is, negative and positive values are valid. Fore instance, -1 sent to an HP E1533 Digital I/O channel sets all bits to one.

- A value sent to a digital channel that is greater than the channel's bit capacity is treated in a modulo(bit width) fashion. For an 8 bit channel the formula is <channel value>MODULO 256. For example, 255 sent to an 8 bit channel sets all of its bits. But, 256 clears all eight bits because 256 is a 9 bit value. The value 257 would then set bit 0 (the 8 bit value 1).

  One bit channels behave in the same way. The formula becomes <channel value>MODULO 1. So the value 1 sent to a 1 bit channel sets the bit, but the value 2 clears the bit. For 1 bit channels all odd values set the bit while all even values clear the bit.

Bitfield Access

The HP E1419A implements bitfield syntax that allows you to manipulate individual bit values within a variable. This syntax is similar to what would be done in 'C', but doesn't require a structure declaration. Bitfield syntax is supported only for the lower 16 bits (bits 0-15) of simple (scalar) variables and channel identifiers. Values read from or written to bitfields behave as integer values as described in "Using Type Float in Integer Situations" above.

Use

```
if(word_var.B0 || word_var.B3)  /* if either bit 0 or bit 3 true ... */
    word_var.B15 = 1;           /* set bit 15                        */
```

NOTES

1. You don't have to declare a bitfield structure in order to use it. In the Algorithm Language the bitfield structure is assumed to be applicable to any simple variable including channel identifiers.

2. Unlike 'C', the Algorithm Language allows you both bit access and "whole" access to the same variable. Example:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
static float my_word_var;
my_word_var = 255       /* set bits 0 through 7 */
my_word_var.B3 = 0      /* clear bit 3 */
```

Declaration Initialization

You may only initialize simple variables (not array members) in the declaration statement:

```
static float my_var = 2;
```

NOTE!

The initialization of the variable only occurs when the algorithm is first defined with the ALG:DEF command. The first time the algorithm is executed (module INITed and triggered), the value will be as initialized. But when the module is stopped (ABORT command), and then re-INITiated, the variable will not be re-initialized but will contain the value last assigned during program execution. In order to initialize variables each time the module is re-INITialized, see "Determining First Execution" on page 4-115.

Global Variables

To declare global variables you execute the SCPI command ALG:DEF 'GLOBALS',<*program_string*>. The <*program_string*> can contain simple variable and array variable declaration/initialization statements. The string must not contain any executable source code.

Example global definition (for <*program_string*> less than 256 chars):

```
ALG:DEF 'GLOBALS','static float Scalar_global, Array_glob[10];'
```

Example global definition (for <*program_string*> gretater than 256 chars, which requires the Indefinite Block Program Data format):

```
ALG:DEF 'GLOBALS',#0static float Scalar_global, Array_glob[10].....LF/EOI
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Language Syntax Summary

This section documents the HP E1419A's Algorithm Language elements.

Identifier:

first character is A-Z, a-z, or "_", optionally followed by characters; A-Z, a-z, 0-9 or "_". Only the first 31 characters are significant. For example; a, abc, a1, a12, a_12, now_is_the_time, gain1

Decimal Constant:

first character is 0-9 or "."(decimal point). Remaining characters if present are 0-9, a "."(one only), a single "E"or"e", optional "+" or "-", 0-9. For example; 0.32, 2, 123, 123.456, 1.23456e-2, 12.34E3

NOTE Decimal constants without a decimal point character are treated by the translator as 32-bit integer values. See Data Types on page 4-129.

Hexadecimal Constant:

first characters are 0x or 0X. Remaining characters are 0-9 and A-F or a-f. No "." allowed.

Octal Constant:

first character is 0. Remaining characters are 0-7. If "." , "e", or "E" is found, the number is assumed to be a Decimal Constant as above.

Primary-expression:

constant
(*expression***)**
scalar-identifier
*scalar-identifier***.***bitnumber*
*array-identifier***[***expression***]**
abs(*expression***)**
max(*expression***,***expression***)**
min(*expression***,***expression***)**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bit-number:

B*n* where *n*=0-9
B*nn* where *nn*=10-15

Unary-expression:

primary-expression
unary-operator unary-expression

Unary-operator:

+
-
!

Multiplicative-expression:

unary-expression
multiplicative-expression multiplicative-operator unary-expression

Multiplicative-operator:

*****
/

Additive-expression:

multiplicative-expression
additive-expression additive-operator multiplicative-expression

Additive-operator:

+
-

Relational-expression:

additive-expression
relational-expression relational-operator additive-expression

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Relational-operator:

<
>
<=
>=

Equality-expression:

relational-expression
equality-expression equality-operator relational-expression

Equality-operator:

==
!=

Logical-AND-expression:

equality-expression
logical-AND-expression **&&** *equality-expression*

Expression:

logical-AND-expression
expression **||** *logical-AND-expression*

Declarator:

identifier
identifier **[** *integer-constant-expression* **]**
NOTE: integer-constant expression in array identifier above must not exceed 1023

Init-declarator:

declarator
declarator **=** *constant-expression*
NOTES: 1. May not initialize array declarator.
2. Arrays limited to single dimension of 1024 maximum.

Init-declarator-list:

init-declarator
init-declarator-list **,** *init-declarator*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Declaration:

`static float` *init-declarator-list*`;`

Declarations:

declaration
declarations declaration

Intrinsic-statement:

`interrupt ( )`
`writefifo (` *expression* **`)`**
`writecvt (` *expression* **`,`** *constant-expression* **`)`**
`writeboth(` *expression* **`,`** *constant-expression* **`)`**
`return`

Expression-statement:

scalar-identifier **`=`** *expression* **`;`**
scalar-identifier **`.`** *bit-number* **`=`** *expression* **`;`**
array-identifier **`[`** *integer-constant expression* **`]`** = expression **`;`**
intrinsic-statement **`;`**

Selection-statement:

`if (` *expression* **`)`** *statement*
`if (` *expression* **`)`** *statement* **`else`** *statement*

Compound-statement:

`{` *statement-list* **`}`**
`{ }`
NOTE: Variable declaration not allowed in compound statement

Statement:

expression-statement
compound-statement
selection-statement

Statement-list:

statement
statement-list statement

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Algorithm-definition:

declarations statement-list
statement-list

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Program Structure and Syntax

In this section you will learn the portion of the 'C' programming language that is directly applicable to the HP E1419A' Algorithm Language. To do this we will compare the 'C' Algorithm Language elements with equivalent BASIC language elements.

Declaring Variables

In BASIC you usually use the DIM statement to name variables and allocate space in memory for them. In the Algorithm Language you specify the variable type and a list of variables:

BASIC	'C'
`DIM a, var, array(3)`	`static float a, var, array[ 3 ];`

Here we declared three variables. Two simple variables; **a**, and **var**, and a single dimensioned array; **array**.

Comments:

- Note that the 'C' language statement must be terminated with the semicolon "**;**".
- Although in the Algorithm Language all variables are of type float, you must explicitly declare them as such.
- All variables in your algorithm are **static**. This means that each time your algorithm is executed, the variables "remember" their values from the previous execution. The **static** modifier must appear in the declaration.
- Array variables must have a single dimension. The array dimension specifies the number of elements. The lower bound is always zero (0) in the Algorithm Language. Therefore the variable **array** from above has three elements; **array [0]** through **array[2]**.

Assigning Values

BASIC and 'C' are the same here. In both languages you use the symbol "=" to assign a value to a simple variable or an element of an array. The value can come from a constant, another variable, or an expression. Examples:

```
a = 12.345;
a = My_var;
a = My_array[ 2 ];
a = (My_array[ 1 ] + 6.2) / My_var;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE In BASIC the assignment symbol "=" is also used as the comparison operator "is equal to". For example; IF a=b THEN ... . As you will read a little further on, 'C' uses a different symbol for this comparison.

The Operations Symbols

Many of the operation symbols are the same, and are used the same way as those in BASIC. However there are differences, and they can cause programming errors until you get used to them.

The Arithmetic Operators

The arithmetic operators available to the HP E1419A are the same as those equivalents in BASIC:

`+`	(addition)	`-`	(subtraction)
`*`	(multiplication)	`/`	(division)

Unary Arithmetic Operator

Again same as BASIC:

`-` (unary minus)
`+` (unary plus)

Examples:
```
a = b + (-c)
a = c + (+b)
```

The Comparison Operators

Here there are some differences.

BASIC		'C'	Notes
=	(is equal to)	==	Different (hard to remember)
<> or #	(is not equal to)	!=	Different but obvious
>	(is greater than)	>	Same
<	(is less than)	>	Same
>=	(is greater than or equal to)	>=	Same
<=	(is less than or equal to)	<=	Same

A common 'C' programming error for BASIC programmers is to inadvertently use the assignment operator "=" instead of the comparison operator "==" in an **if** statement. Fortunately, the HP E1419A will flag this as a Syntax Error when the algorithm is loaded.

The Logical Operators

There are three operators. They are very different from those in BASIC.

BASIC	Examples	'C'	Examples
`AND`	`IF A=B AND B=C`	`&&`	`if( ( a == b )&&( b == c ) )`
`OR`	`IF A=B OR A=C`	`||`	`if( ( a == b ) || ( a == c ) )`
`NOT`	`IF NOT B`	`!`	`if ( ! b )`

Conditional Execution

The HP E1419A Algorithm Language provides the **if - else** construct for conditional execution. The following figure compares the elements of the 'C' **if - else** construct with the BASIC **if - then - else - end if** construct. The general form of the **if - else** construct is:

`if(`*expression***`)`** *statement1* **`else`** *statement2*

where statement1 is executed if *expression* evaluates to non-zero (true), and *statement2* is executed if *expression* evaluates to zero (false). *Statement1*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

and/or *statement2* can be compound statements. That is, multiple simple statements within curly braces. See Figure 4-3

BASIC Syntax	Comments	'C' Syntax
`IF` *boolean_expression* `THEN` statement	Simplest form (used often)	`if(`*boolean_expression*`)` *statement*;
`IF` *boolean_expression* `THEN` *statement* `END IF`	Two-line form (not recommended; use multiple line form instead)	`if(`*boolean_expression*`)` *statement*;
`IF` boolean_expression `THEN` *statement* *statement* *statement* `END IF`	Multiple line form (used often)	`if(`*boolean_expression*`)` `{` *statement;* *statement;* *statement;* `}`
`IF` *boolean_expression* `THEN` *statement* *statement* `ELSE` *statement* `END IF`	Multiple line form with else (used often)	`if(`*boolean_expression*`)` `{` *statement;* *statement;* `}` `else` `{` *statement*; `}`

Figure 4-3 The if Statement 'C' versus BASIC

Note that in BASIC the *boolean_expression* is delimited by the IF and the THEN keywords. In 'C' the parentheses delimit the expression. In 'C' , the ")" is the implied THEN. In BASIC the END IF keyword terminates a multi-line IF. In 'C', the **if** is terminated at the end of the following statement when no **else** clause is present, or at the end of the statement following the **else** clause. Figure 4-4 shows examples of these forms:

Note that in 'C' "else" is part of the closest previous "if"statement. So the example:
if(x) if(y) z = 1; else z = 2;

executes like:

```
if( x ){
    if( y ){
        z = 1;
    }
    else{
        z = 2;
    }
}
```

not like:

```
if( x ){
    if ( y ){
        z = 1;
    }
}
else{
    z = 2;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

BASIC Syntax	'C' Syntax
`IF A<=0 THEN C=ABS(A)`	`if(a <= 0) c=abs(a);`
`IF A<>0 THEN` `  C=B/A` `END IF`	`if(a != 0)` `  c = b / a;`
`IF A<>B AND A<>C THEN` `  A=A*B` `  B=B+1` `  C=0` `END IF`	`if((a != b) && (a != c))` `{` `  a = a * b;` `  b = b + 1;` `  c = 0;` `}`
`IF A=5 OR B=-5 THEN` `  C=ABS(C)` `  C= 2/C` `ELSE` `  C= A*B` `END IF`	`if((a == 5) || (b == -5))` `{` `  c = abs(c);` `  c = 2 / c;` `}` `else` `{` `  c = a * b;` `}`

Figure 4-4 Examples of 'C' and BASIC if Statements

Comment Lines

Probably the most important element of programming is the comment. In older BASIC interpreters the comment line began with "REM" and ended at the end-of-line character(s) (probably carriage return then linefeed). Later BASICs allowed comments to also begin with various "shorthand" characters such as "!", or "'". In all cases a comment ended when the end-of-line is encountered. In 'C' and the Algorithm Language, comments begin with the the two characters "/*" and continue until the two characters "*/" are encountered. Examples:

```
/* this line is solely a comment line */
if ( a != b) c = d + 1;  /* comment within a code line */
/*   This comment is composed of more than one line.
     The comment can be any number of lines long and
     terminates when the following two characters appear
*/
```

About the only character combination that is not allowed within a comment is "*/", since this will terminate the comment.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Overall Program Structure

The preceding discussion showed the differences between individual statements in BASIC and 'C'. Here we will show how the HP E1419A's Algorithm Language elements are arranged into a program.

Here is a simple example algorithm that shows most of the elements discussed so far.

```
/*      Example Algorithm to show language elements in the context of a complete
        custom algorithm.

        Program variables:

            user_flag       Set this value with the SCPI command ALG:SCALAR.
            user_value      Set this value with the SCPI command ALG:SCALAR.

        Program Function:

        Algorithm returns user_flag in CVT element 330 and another value in CVT element 331
        each time the algorithm is executed.
        When user_flag = 0, returns zero in CVT 331.
        When user_flag is positive, returns user_value * 2 in CVT 331
        When user_flag is negative, returns user_value / 2 in CVT 331 and in FIFO

        Use the SCPI command ALGorithm:SCALar followed by ALGorithm:UPDate to set
        user_flag and user_value.
*/
static float user_flag = 0;             /* user_flag will be initialized to 0 when alg is defined, not when run */
static float user_value;                /* Declaration statements (end with  ; ) */

        writecvt (user_flag,330);       /* Always write user_flag in CVT (statement ends with ; ) */

        if (user_flag )                 /* if statement (note no ; ) */
        {                               /* brace opens compound statement */
            if (user_flag > 0) writecvt (user_value * 2,331);       /* one-line if statement (writecvt ends with ; ) */
            else                        /* else immediately follows complete if-statement construct */
            {                           /* open compound statement for else clause */
                    writecvt (user_value / 2,331);    /* each simple statement ends in ; (even within compound ) */
                    writefifo (user_value);           /* these two statements could combine with writeboth () */
            }                           /* close compound statement for else clause */
        }                               /* close compound statement for first if */
        else writecvt (0,331);          /* else clause goes with first if statement. Note single line else */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 5
HP VEE Programming Examples

About This Chapter

The focus of this chapter is to demonstrate a multitude of HP VEE programming examples to help you get your E1419 application running as quickly as possible. These examples are a combination of fully working programs that demonstrate various capabilities of the E1419 plus HP VEE object modules that you can merge into your HP VEE application to perform such operations as calibration, error checking, testing, custom functions, and custom EU conversion. Any operation or function that may seem hard to implement with the E1419 has been included in the spirit of HP VEE objects. You simply merge the examples and cut and paste what you want. You'll find yourself performing complex functions in a very short period of time.

These example programs are written to run on any HP VEE 3.0+ platform. This includes UNIX, Windows 3.1, Windows 95, and Windows NT. These programs are primarily designed to run from an external computer communicating over an HP-IB link to the HP E1405/6 Command Module. The Command Module holds the E1419 driver which controls the E1419 VXI card. The E1419 VXI card comes pre-configured to LADD 208(HP-IB address 70926, for example).

These HP VEE programs are also viable for the E1415 Algorithmic Closed Loop Control card since the programming architecture of both cards is very similar. The E1415 provides greater flexibility and focuses more closely on PID closed loop control applications. The E1419 is a more general purpose/lower cost data acquisition and control system solution.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The contents of this chapter are:

- **Wiring Connections and File Locations for the Examples** 147

- **Virtual Front Panel Program: panl1419.vee**
  This program performs virtually all your calibration, testing, and general wiring connection verification needs. It's a quick way to get your card up and running and making measurements. You can set analog outputs, look at all input channels, see your SCP configurations, perform strip chart comparisons among any channels, and log data to a disk. This program includes an HP VEE 4.0+ version that is compiled to load and run faster 148

- **Calibration: cal_1419.vee**
  This program operates stand-alone. However, it is easy to merge it directly into your own VEE application program to provide easy access to the calibration sequence 151

- **Functional Test: test1419.vee**
  This program operates stand-alone. However, it is easy to merge it directly into your own VEE application program to provide easy access to the card functional test sequence 152

- **Programming Model Example: temp1419.vee**
  This program operates stand-alone. It is written to follow the programming model outlined in Chapter 3. You will find examples of writing multiple algorithms, variable monitoring and modification, interrupts, temperature measurements, and data display 153

- **Error Checking: err_1419.vee**
  This program operates stand-alone. However, it is designed to be merged into your application program to provide an object that can query every error stored in the E1419's error queue. It's a good debugging tool 156
  .

- **Configuration Display: scp_1419.vee**
  This program operates stand-alone. However, it is designed to be merged into your application program to provide a means of displaying the driver and firmware revisions and identify which SCP's are loaded into the 8 SCP slots 157

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Engineering Unit Conversion: eu_1419.vee**
  This program is designed to be merged into your application program. It provides all the necessary objects to permit custom EU conversion on any of the E1419's 64 analog input channels. The program eufn1419.vee demonstrates how to use this module 158

- **Custom Function Generation: fn_1419.vee**
  This program is designed to be merged into your application program. It provides all the necessary objects to build up to 32 custom functions callable from E1419 algorithms. The program eufn1419.vee demonstrates how to use this module 160

- **Custom EU and Custom Function: eufn1419.vee**
  This program operates stand-alone. It is designed to show you how easy it is to generate complicated EU conversion and Custom C functions by simply entering in channel numbers, function names, and algebraic expressions. You need to convert volts to pressure or perform a square-root operation? You can use this program to see how easy it is to perform 162

- **Curve Fitting and EU Generation: regr1419.vee**
  This program operates stand-alone. It shows how you could use the HP VEE regression tools to generate a polynomial equation to fit volts and pressure. The generated equation can then be used in the eu_1419.vee module for converting volts to pressure during data acquisition of the E1419 164

- **Interrupt Handling: intr1419.vee**
  This program operates stand-alone. This is an example program that shows how to create multiple threads of operation in HP VEE to respond to a FIFO half-full interrupt. It teaches you the concepts of interrupt driven programming. The example temp1419.vee also incorporates a slightly different version of interrupt processing that can enhance your learning 165

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **E1419 Simple Data Logger: dlgr1419.vee**
  This program operates stand-alone. It illustrates how to configure the E1419 to collect data, store that data into its FIFO, and retrieve that data for display on a strip chart and optional logging to a file. This program can also be used to read the stored data file "aichans" generated by the panl1419.vee example, or you can use it to observe previously stored data files created with this example. This example can easily be modified to a more complicated version or you can cut and paste the pieces you need. An HP VEE 4.0+ compiled version is included with your software 167

- **Modification of Variables and Arrays. updt1419.vee**
  This program operates stand-alone. This example shows how operator interaction with running algorithms takes place and how to download changes for both scalar and array variables 170

- **Algorithm Modification: swap1419.vee**
  This program operates stand-alone. It shows you how to modify algorithms while the E1419 is running. It includes further examples on custom function generation 172

- **Driver Download: drvr1419.vee**
  This program allows you to download the E1419 driver and any other HP VXI drivers you might need into the HP E1405/6 Command Module 174

- **Firmware Update Download: flsh1419.vee**
  This program allows you to save and reprogram the FLASH memory of the E1419. 175

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Wiring Connections and File Locations for the Examples

The following illustration shows the connections you should make to the HP E1419A to allow the example programs in this chapter to operate as described. For detailed information on connecting wiring to your HP E1419A, see Chapter 2.

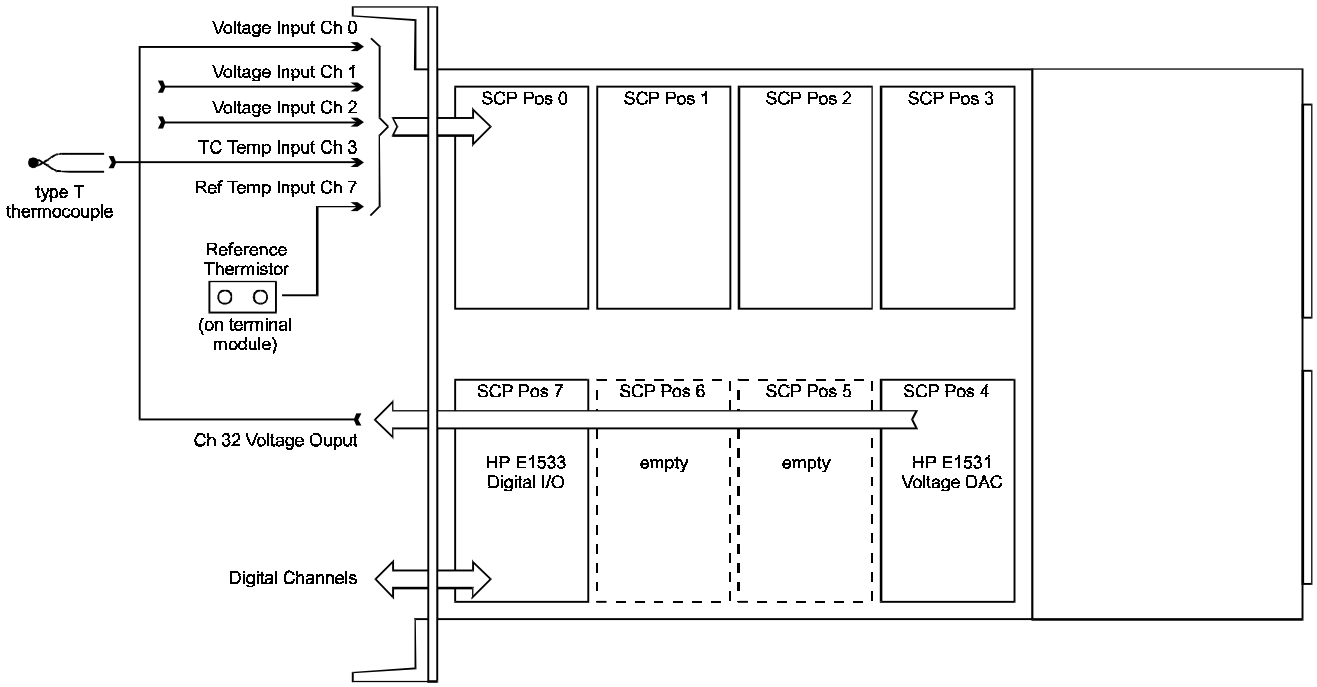


Figure 5-1 Signal Connections for Examples

Example File Location

The Supplied "HP Instrument Drivers" CD contains all of these example programs stored in the HP VEE 3.0 compatibility mode. If you are using HP VEE 4.0 or higher, you may want to compile those examples that would benefit from the increased speed (panl1419.vee, temp1419.vee, and swap1419.vee).
The Directory path to these examples is:

<cd drive letter>:\Dscpi\e1419\10322.dos\scpi_vee

Installing Example Files

In order for the example programs to run they must be installed onto your hard drive in directory **C:\dabundle**. The **Dscpi\e1419\10322.dos\scpi_vee** directory mentioned above contains a batch file "install.bat" that will copy the examples to **C:\dabundle**. Using your Windows system's File Manager or Explorer, open the CD directory **Dscpi\e1419\10322.dos\scpi_vee** and double-click on "install.bat". You can then start HP VEE and load the example programs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Virtual Front Panel Program

panl1419.vee: This program performs virtually all your calibration, testing, and general wiring connection verification needs. It's a quick way to get your card up and running and making measurements. You canset analog outputs, look all input channels, see your SCP configurations, perform strip chart comparisons among any channels, and log data to a disk.

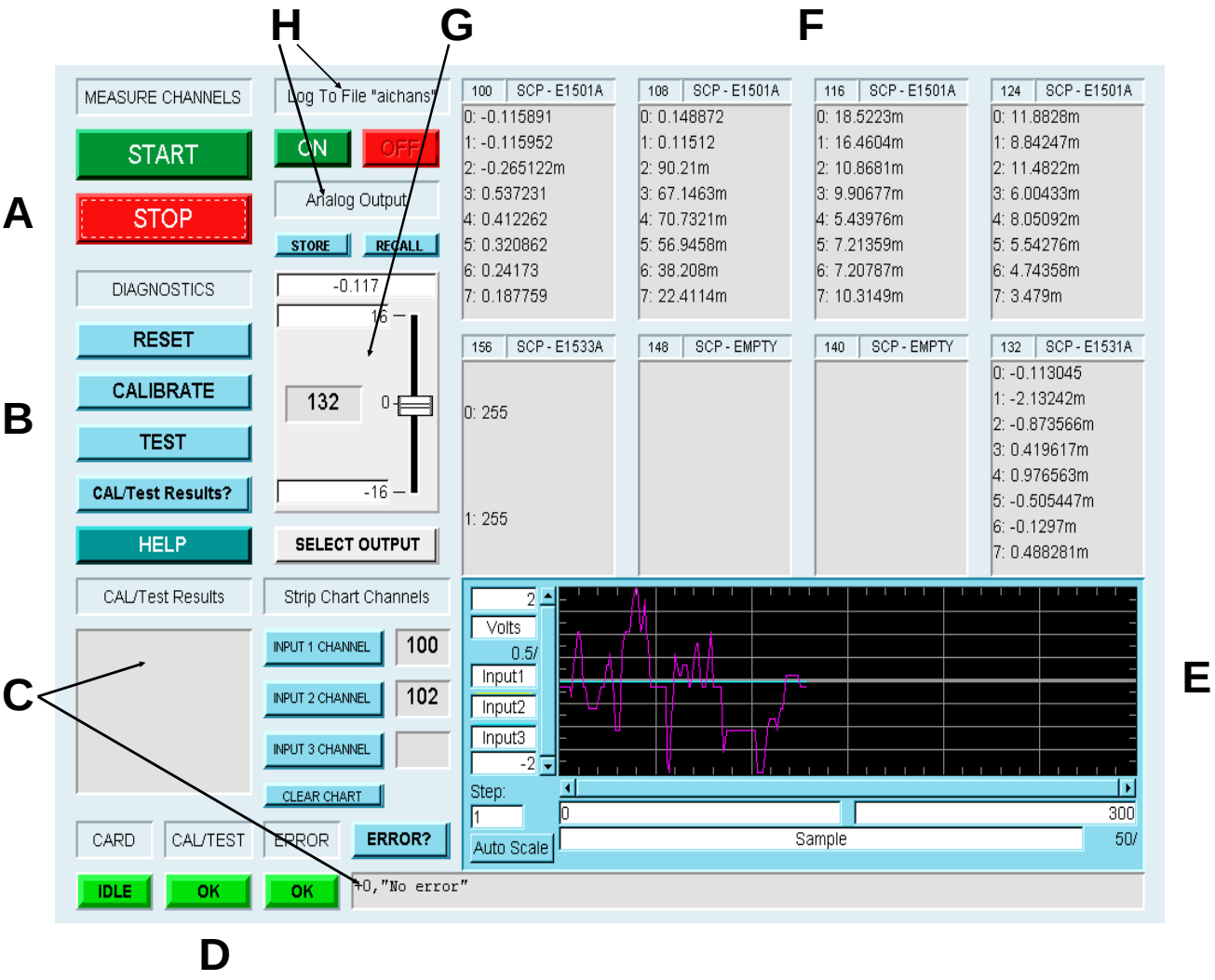


Figure 5-2 Virtual Front Panel Program

The various sections illustrated in Figure 5-2 are described as follows:

A. After executing the RUN on HP VEE, a number of operations will take place which analyzes the E1419's configuration, sets up SCP's to make measurements, and prepares for data acquisition. When the START becomes "LIT" or active, pressing START will

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

begin the processes of acquiring measurements on any and all input SCP's.

Analog input SCP's display volts and digital input SCP's display digital state information in section F. Analog output SCP's are both input and output at the same time. Pressing STOP will temporarily pause the acquisition of data.

B. This is the diagnostics section. You can RESET the card at any time to stop measurement operations, calibration, or testing. The CALIBRATE and TEST keys are not active while START in section A is active. Likewise, neither CALIBRATE nor TEST can be active at the same time. CALIBRATE should be performed after the E1419 has warmed up for 1 hour. Calibration is necessary whenever you add or move any SCP modules. TEST allows you to check the integrity of any SCP channels. If errors are present with either calibration or testing, indicators in section D will be lit and CAL/Test Results? and ERROR? can be pressed to determine what errors are present. Both TEST and CALIBRATE may take from 3 to 10 minutes depending upon the configuration of SCP's. Also, CALIBRATE performs a CAL:STORE ADC which stores the calibration constants in non-volatile RAM.

C. These two display areas will indicate errors from calibration or testing. You must press CAL/Test Results? or ERROR? to see the results.

D. These are the status monitor lights. CARD is either IDLE, MEAS, or BUSY, which indicates no operations pending, START active, or performing CALIBRATION or TEST. CAL/TEST is either OK or an ERR condition. If ERR, data will be present in section C. ERROR is either OK or ERR which indicates the need to press the ERROR? button.

E. This is the strip chart area. You can select up to 3 traces from any of the input channels you see displayed in section F. Pressing INPUT 1,2,3 will give you valid selections. The current channel will be displayed next to the strip chart. Traces are colored. The CLEAR button will erase all current traces. Auto Scale will compress all current data into the available window. Update rates vary with the processing power of your computer but can range from sub-second to several second updates.

F. This is the section that displays the type of SCP and its channel input values when START is active. Note that the beginning channel number is displayed in the upper left corner of each SCP window. Channel data is displayed as 0,1,2.... so you must add

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

that designator to the upper left corner number to obtain the actual channel number of the SCP.

G. This is the Analog Output selector. Pressing SELECT OUTPUT will give you a list of all available analog output channels. Choosing one will allow you to modify the output of any analog output channel. Since all analog output SCP's also display their output values as input channels, you can see the results of changing the output values in section F when START is active. Please note that these output values are only accurate to within 10% of the programmed value. These are sanity check readings. The actual output will be precisely what you programmed if you have calibrated the E1419, and you can connect any analog output to one of the analog input channels to see exactly what values you are setting. When returning to a previously selected output channel, the Analog Output slider will adjust itself to the last programmed value you had when you selected another channel.

H. These two sections give you some added flexibility. Under the Analog Output section you will see two buttons called STORE and RECALL. Once you have programmed the values of all your analog outputs, you can press the STORE button to save the output states. If you press RESET or re-start this program later, you can press RECALL to restore all those programmed values.

The LOG ON/OFF buttons will permit you to log the measurement results from the first 32 analog input channels to a file named "aichans". Each time you "RUN" this HP VEE example, this file is cleared. Successive ON/OFF selections while START is active will append data to the file. To view this data later, you can run the HP VEE program dlgr1419.vee and specify the filename "aichans" as the Input Data File.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Calibration

cal_1419.vee: This program operates stand-alone. However, it is easy to merge it directly into your own VEE application program to provide easy access to the calibration sequence. The HP VEE detail view is all that is developed as illustrated in Figure 5-3. A counter in the upper right-hand section of the detail gives the number of seconds elapsed so you can see if progress is being made.

This program performs a *CAL? and CAL:STORE ADC to perform a complete calibration of the E1419 card. Any errors detected will be displayed so you can identify exactly which channels may be in question. Calibration may take from 3 to 10 or more minutes to occur, dependent upon the number and type of SCP's loaded.

Calibration should be performed whenever you move or add SCP's, and you should wait 1 hour after turn-ON for the E1419 to reach temperature stabilization.

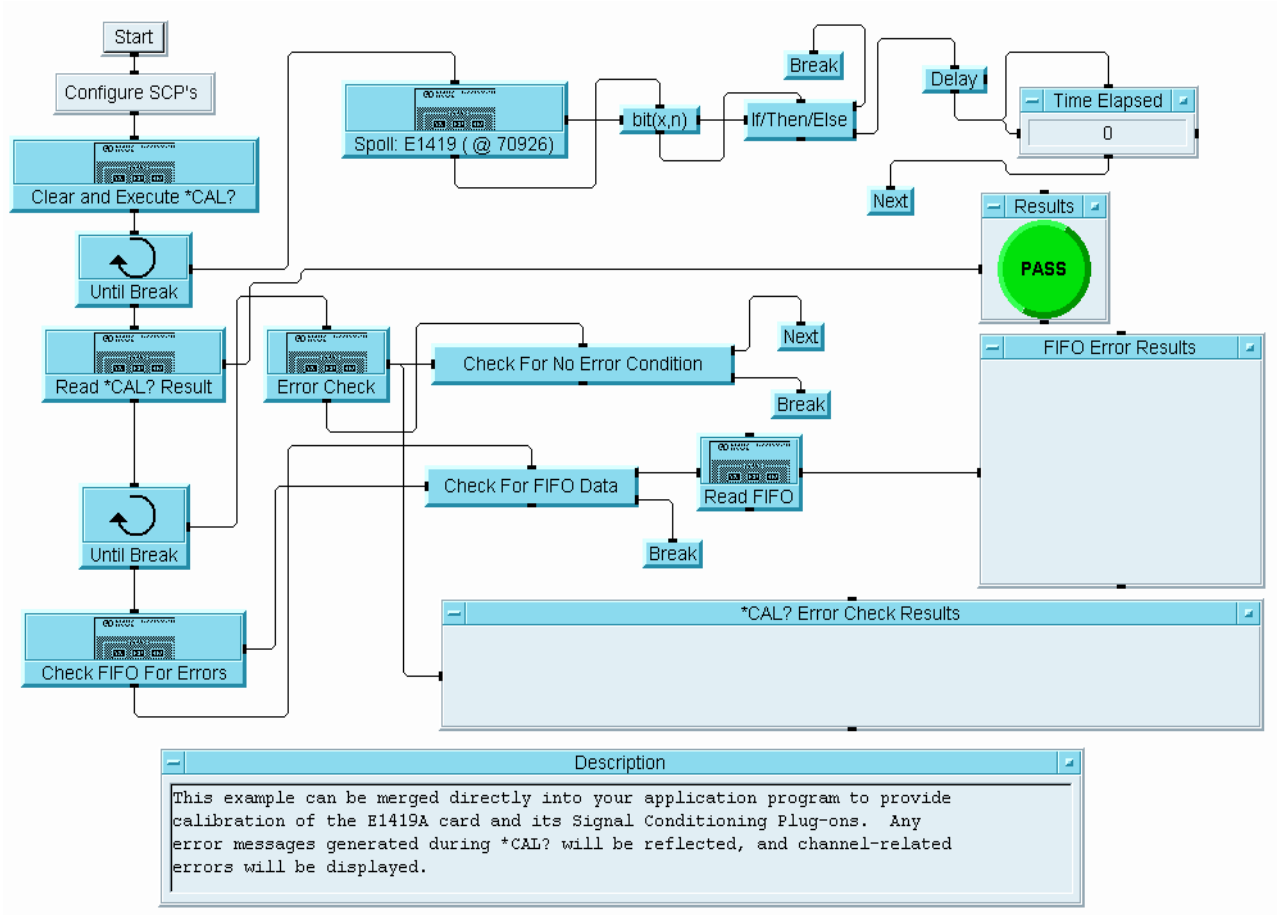


Figure 5-3 Calibration Detail View

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function Test

test1419.vee: This program operates stand-alone. However, it is easy to merge it directly into your own VEE application program to provide easy access to the testing sequence. The HP VEE detail view is all that is developed as illustrated in Figure 5-4. A counter in the upper right-hand section of the detail gives the number of seconds elapsed so you can see if progress is being made.

This program performs a *TST? Any errors detected will be displayed so you can identify exactly which channels may be in question. Testing may take from 3 to 10 or more minutes to occur, dependent upon the number and type of SCP's loaded.

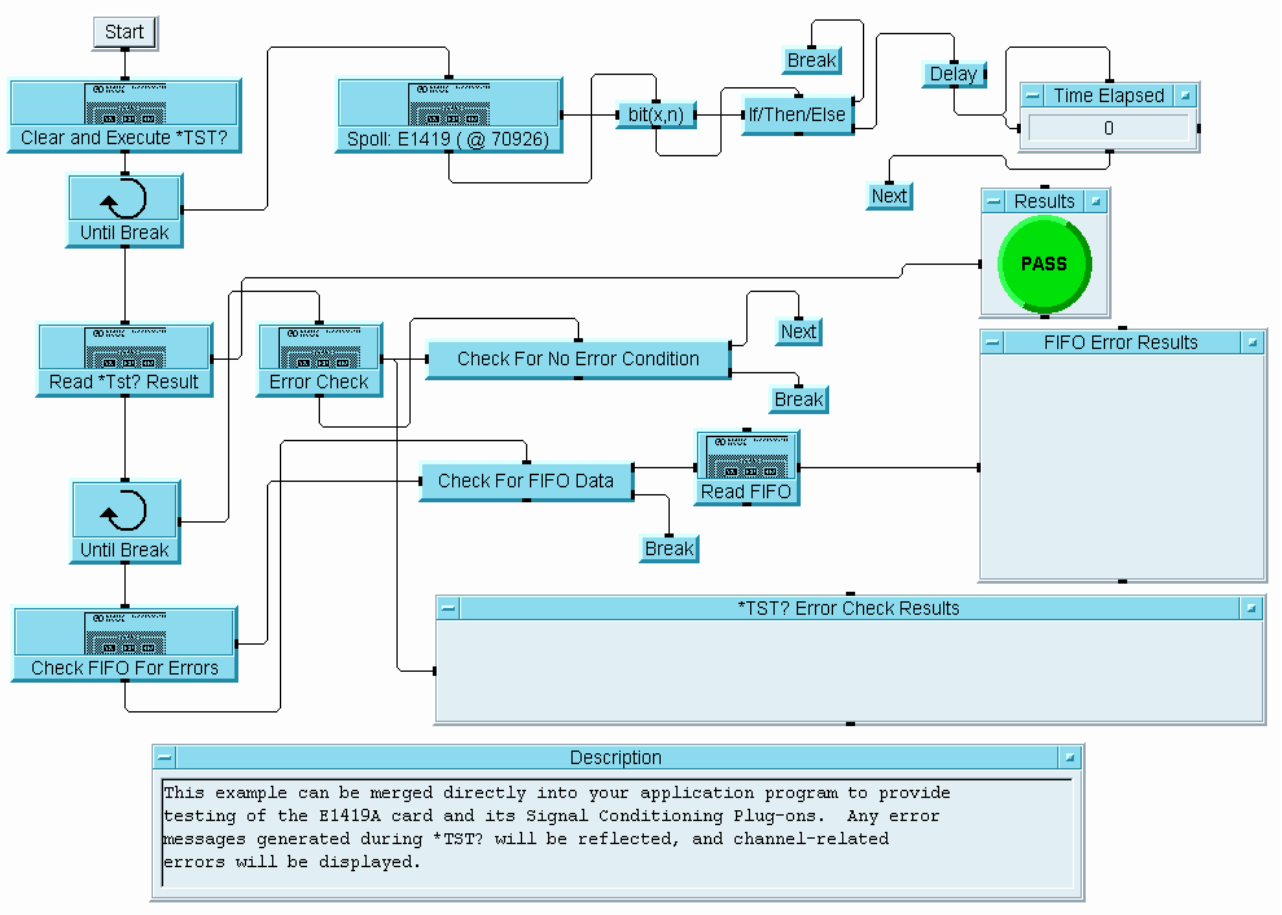


Figure 5-4 Functional Test Detail View

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Programming Model Example

temp1419.vee: This program operates stand-alone. It is written to follow the programming model outlined in Chapter 3. You will find examples of writing multiple algorithms, variable monitoring and modification, interrupts, temperature measurements, and data display. Please refer to Figure 5-5 for the remainder of the discussion.

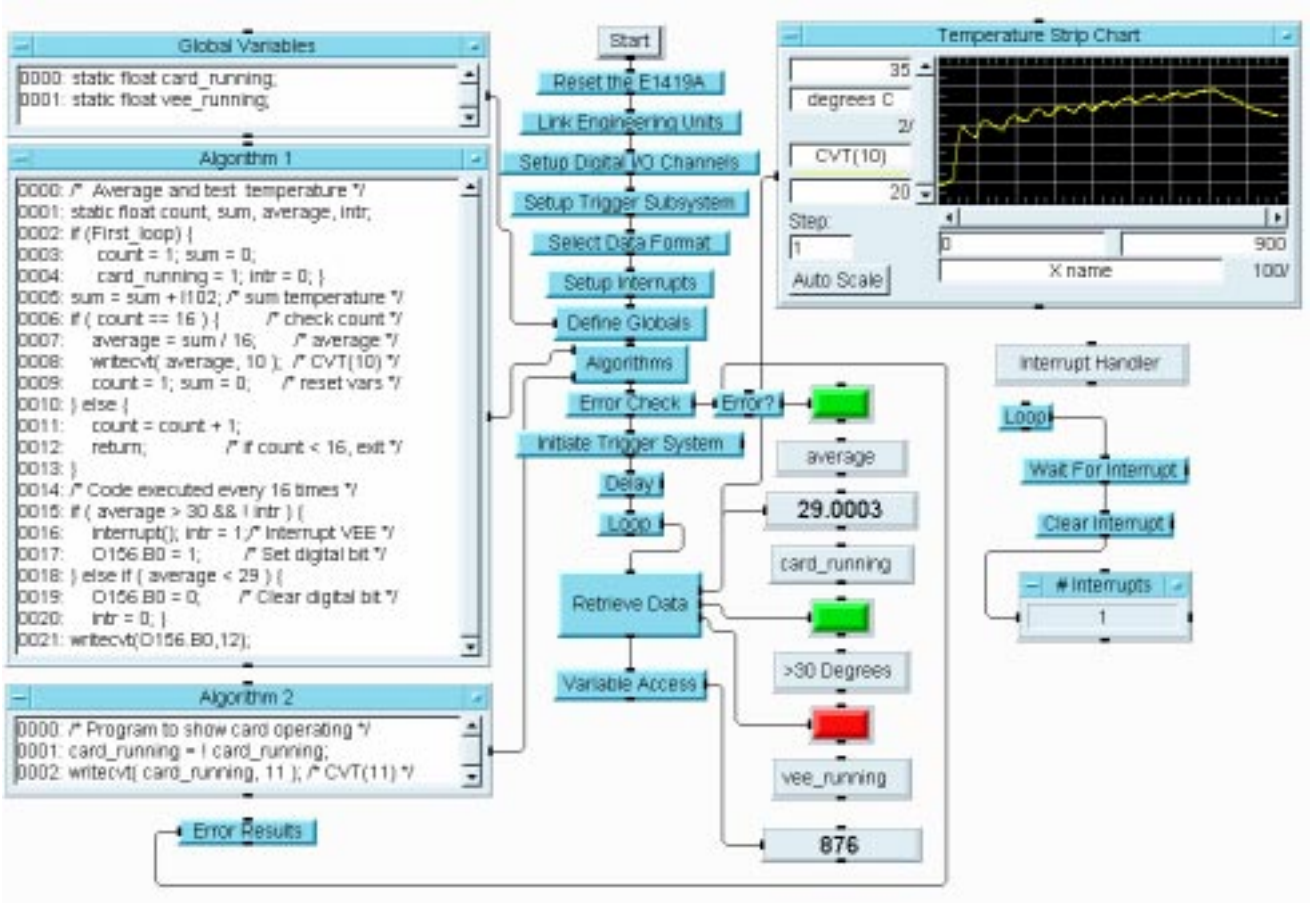


Figure 5-5 Programming Model Detail View

The hardware configuration assumes you have a type T thermocouple attached to channel 102, the thermocouple reference sensor attached to channel 103, and a digital output channel at 156. Figure 5-1 illustrates the necessary wiring connections for this and the other E1419/HP VEE examples. Of particular interest here is that the thermocouple is placed at a channel less than the reference junction channel. Since the E1419's C compiler sorts all channels in numerical order for scanning by the A/D at runtime, you must override that assignment with the SENS:REF:CHAN command as illustrated in Link Engineering Units so that the reference channel is scanned BEFORE the thermocouple channel that needs the reference junction compensation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The E1419 Algorithms are written inside HP VEE text boxes as a 1-dimension array of text lines. The Define Globals and Algorithms blocks show how these text boxes are downloaded into the E1419. This makes E1419 C program development very easy.

Note that there are two HP VEE threads of operation as indicated by the two START icons. This means that proper operation will only take place if you press the HP VEE 'RUN' button. The Interrupt Handler simply waits for the interrupt() routine in the E1419 to execute and assert the E1419's VXI interrupt line. The Interrupt Handler is simply monitoring the out-of-bound condition of the card. If the card indicates the temperature of the thermocouple rises above 30 degrees C, an interrupt is generated. The interrupt is re-enabled after its occurrence. Please note in Algorithm 1 that an interrupt is only allowed to occur once when passing through 30 degrees. After which, the card DOES NOT interrupt again until the temperature falls below 29 degrees and again passes through 30 degrees. This is done to illustrate the concept of hysteresis applied to interrupts. If you were to allow the E1419 to interrupt HP VEE constantly while above 30 degrees C, you would bombard the external computer with interrupts which would lower your overall performance. This technique achieves the needed signal to HP VEE but adds the hysteresis to avoid constant interrupts.

CVT location 12 is used to reflect the state of the digital output channel used to respond to the over-temperature condition. That condition is reflected back to HP VEE as an LED.

Other interesting features include reading and writing of variables. Algorithm 2 takes the global variable "card_running" and complements it each time it executes. It then writes that value to CVT location 11. Algorithms 2 has been configured by the ALG:SCAN:RATIO command to execute every 500 triggers, as set in the Algorithms object. Since the trigger timer is set to 2 milliseconds(Setup Trigger Subsystem), Algorithm 2 executes once every 1 second and thereby causes the card_running LED to blink at 1 second intervals. This is a good sanity check for the HP VEE program to know that the E1419 is running. If it had stopped for some reason, the LED would not be flashing.

Another check to know that HP VEE is running is performed with the Variable Access object. HP VEE reads the value of the global "vee_running", increments it by one, and re-writes that value back to the E1419. Although not included in this example, an algorithm could detect that the variable was changing and know that HP VEE was still executing. This might be a situation where if HP VEE were to be taken off-line or stopped, the E1419 could detect the situation and begin a possible shut-down of operations by itself.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note that Algorithm 1 performs an average of 16 temperature readings before writing the result to CVT location 10. Each time the algorithm executes a check is performed to see if it has executed 16 times. If not, only the sum and count is affected, and the routine exits prematurely. The average is done to provide quieter readings when trying to make temperature measurements at high speed with a non-filter/non-gain SCP. This is a caution for you. High accuracy and low drift temperature measurements are better with SCP's that have gain and filtering. However, you can get decent 1-3 degree C accuracy with the E1501 straight-through SCP's which is typically very reasonable for thermocouples.

You should spend some time opening each of the objects in this example and see what SCPI commands are used and how they relate back to concepts in Chapter 3. See Chapter 6 - the SCPI reference - for more detailed information on each command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Error Checking

err_1419.vee: This program operates stand-alone. However, it is designed to be merged into your application program to provide an object that will query every error stored in the E1419's error queue. It's a good debugging tool because it is self-contained. A good technique would be to turn this entire object into a function that you call after each major programming object in your application.

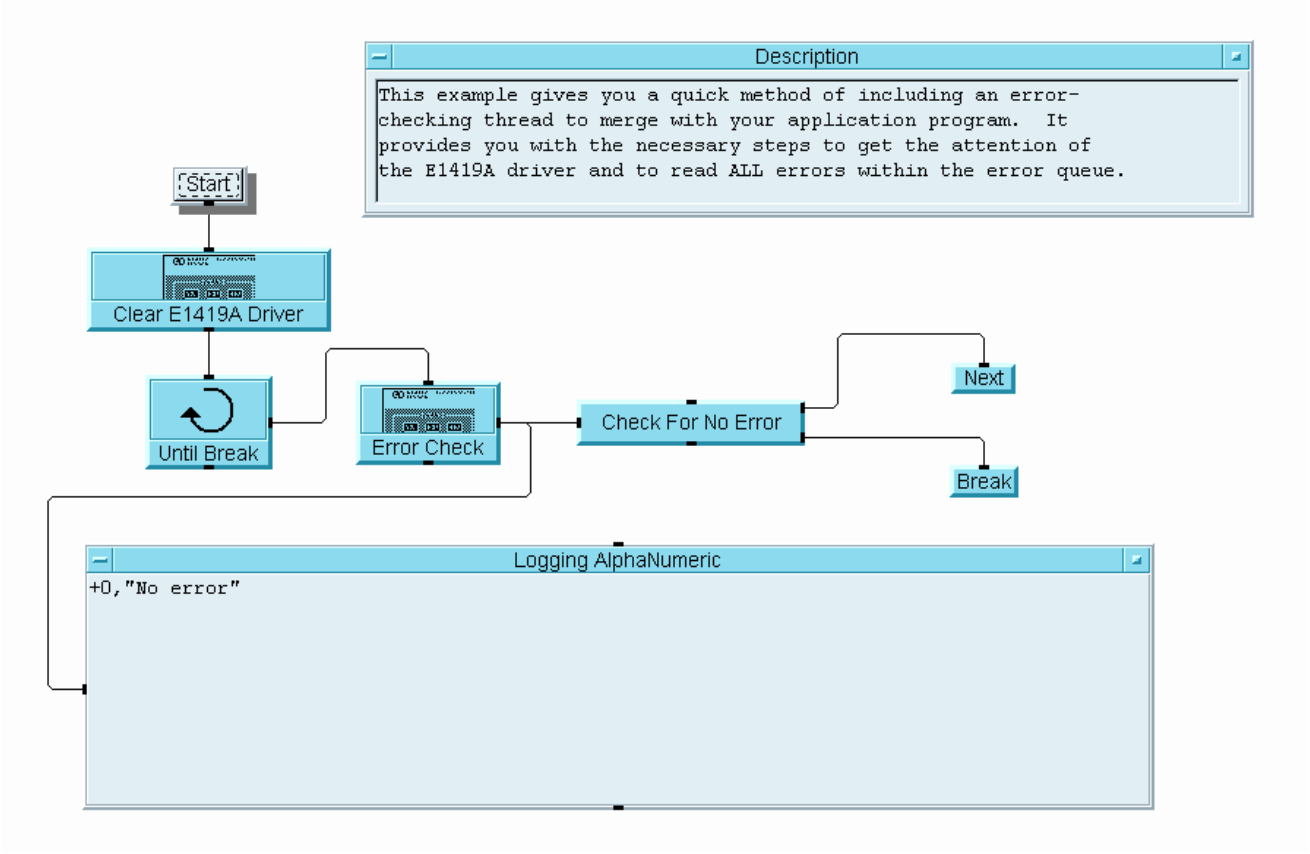


Figure 5-6 Error Checking Detail View

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Configuration Display

scp_1419.vee: This program operates stand-alone. However, it is designed to be merged into your application program to provide a means of displaying the driver and firmware revisions and identify which SCP's are loaded into the 8 SCP slots. Just like the previous error checking example, you can make this a callable function in HP VEE and insert it into your application.

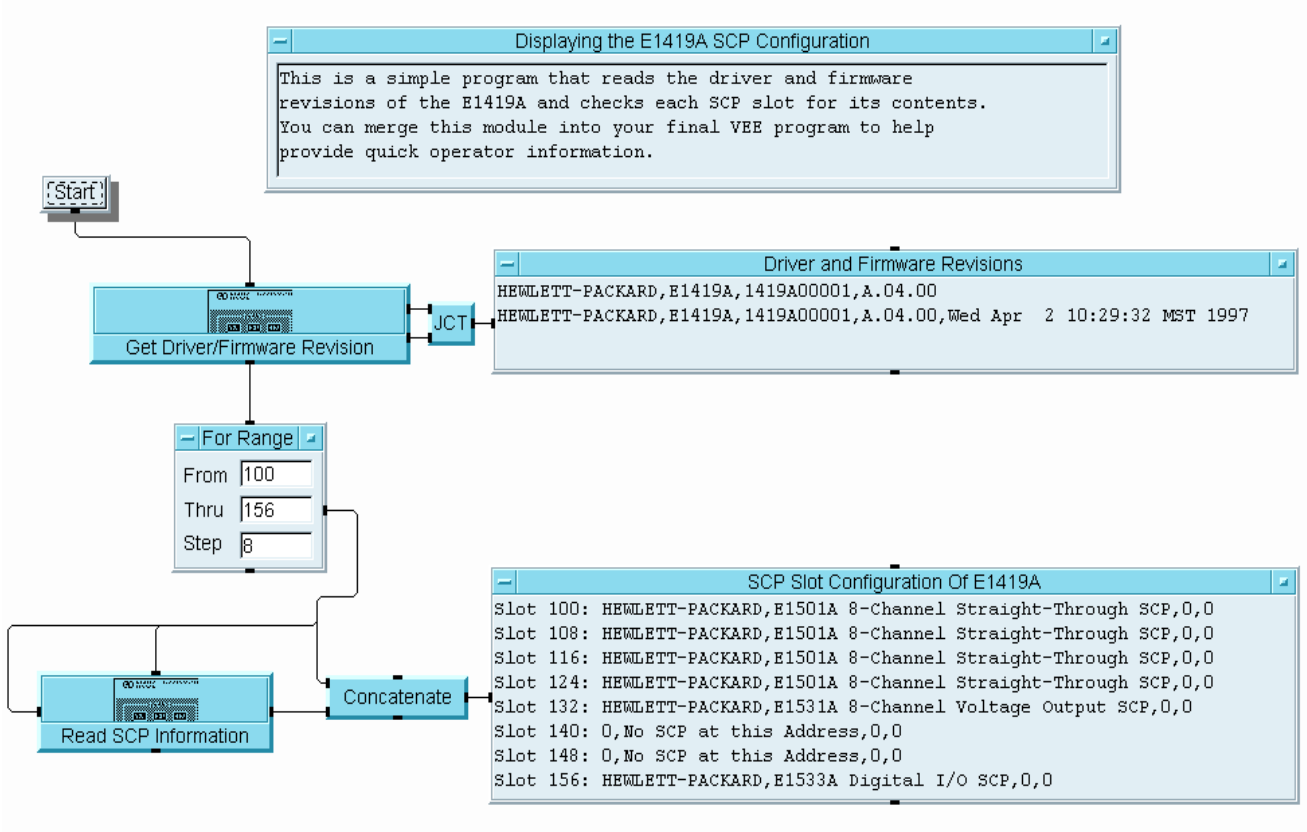


Figure 5-7 Configuration Display Detail View

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Engineering Unit Conversion

eu_1419.vee: This program is designed to be merged into your application program. It provides all the necessary objects to build custom EU table conversion on any of the E1419's 64 input channels. The program eufn1419.vee demonstrates how to use this module.

The HP VEE programming necessary to build the tables is somewhat complex and beyond the scope of this text. In fact, there is an additional program written in C that is called by this module: pc_eu.exe. Both the source code for this program and the DOS executable are included with the E1419 examples. The source code is provided so you can compile the program on other platforms where HP VEE is supported: UNIX, etc. You would issue command similar to "cc -Aa eu_141x.c -o unix_eu -lm" which would compile the program under a typical UNIX environment. Note that the name unix_eu and pc_eu have significant meaning to this module.

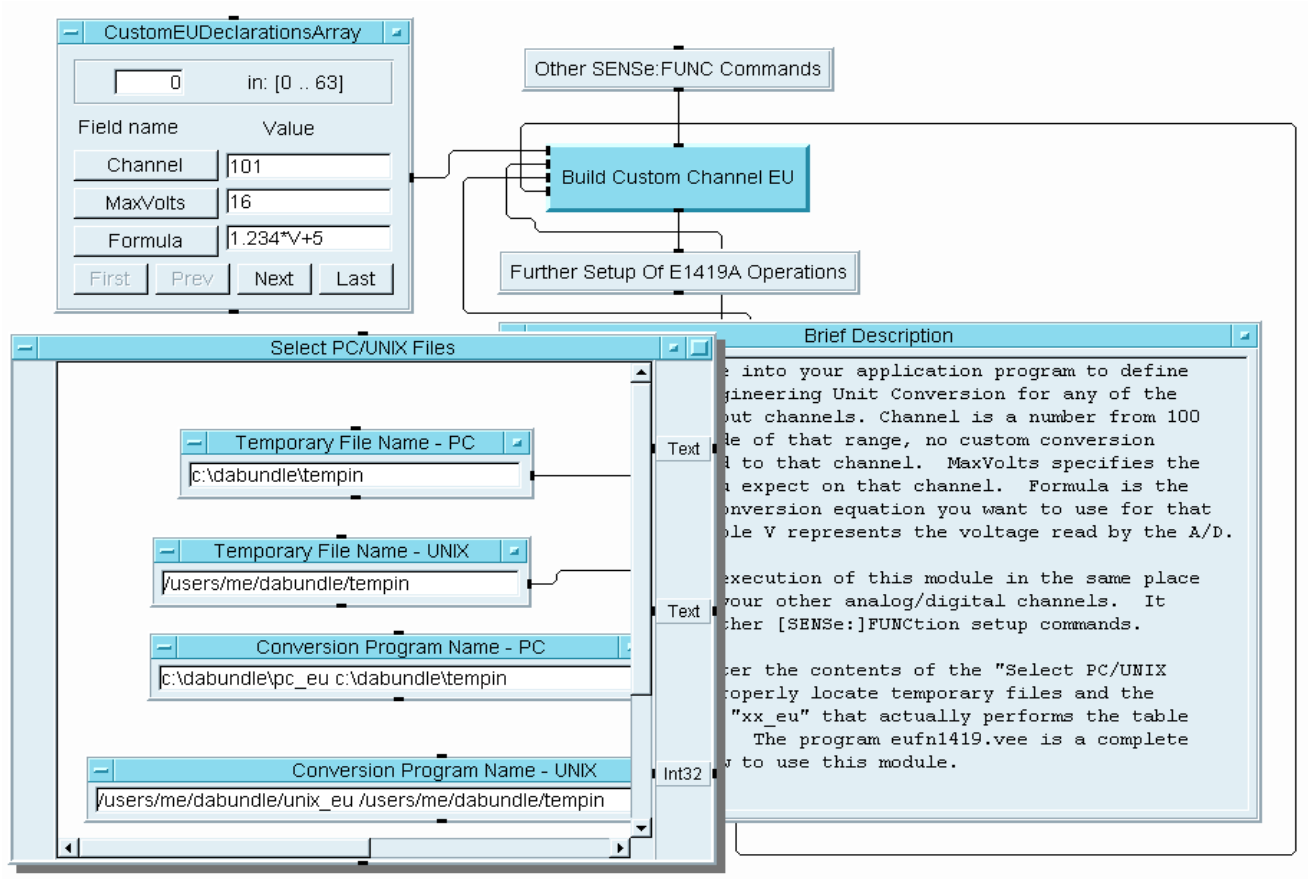


Figure 5-8 Custom EU Conversion Detail View

Figure 5-8 illustrates where this module would be integrated into your VEE application program. This is a part of the Link Engineering Units setup you

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

learned in Chapter 3. You simply select the channel, the maximum voltage you expect to see on that channel(MaxVolts represents +/- voltage), and enter any formula using the available HP VEE math functions. It's that simple!

The only restriction is that you must use the variable "V" as representing the voltage read from the channel. When the selected channel is read by the E1419's A/D, that voltage will be "inserted" into your formula just as represented in the example. Appendix F discusses custom function table generation which is based upon the same principle as EU table conversion. EU conversion executes within a few microseconds, so there is no problem with running the E1419 sample rate at 100kHz(10usec per sample).

The CustomEUDeclarationsArray can hold up to 64 channel definitions. Any valid channel number 100-163 for an Analog Input Channel will cause the associated table to be built and downloaded into the E1419's EU table memory space. Leaving the field Channel at "0" will cause that channel to be ignored by this module. Any *RST or power-ON condition will require re-execution of this module.

The object Select PC/UNIX Files contains file name and directory paths necessary to make the module execute properly on PC's or UNIX platforms. Figure 5-8 also shows that object open for your observation. You can see that the default location of the E1419 example programs is "c:\dabundle". A typical UNIX path is included for example. The example uses the HP VEE function whichOS() to determine which directory structure to use.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Custom Function Generation

fn_1419.vee: This program is designed to be merged into your application program. It provides all the necessary objects to build up to 32 C custom functions callable from E1419 algorithms. The program eufn1419.vee demonstrates how to use this module.

The HP VEE programming necessary to build the tables is somewhat complex and beyond the scope of this text. In fact, there is an additional program written in C that is called by this module: pc_fn.exe. Both the source code for this program and the DOS executable are included with the E1419 examples. The source code is provided so you can compile the program on other platforms where HP VEE is supported: UNIX, etc. You would issue command similar to "cc -Aa fn_141x.c -o unix_fn -lm" which would compile the program under a typical UNIX environment. Note that the name unix_fn and pc_fn have significant meaning to this module.

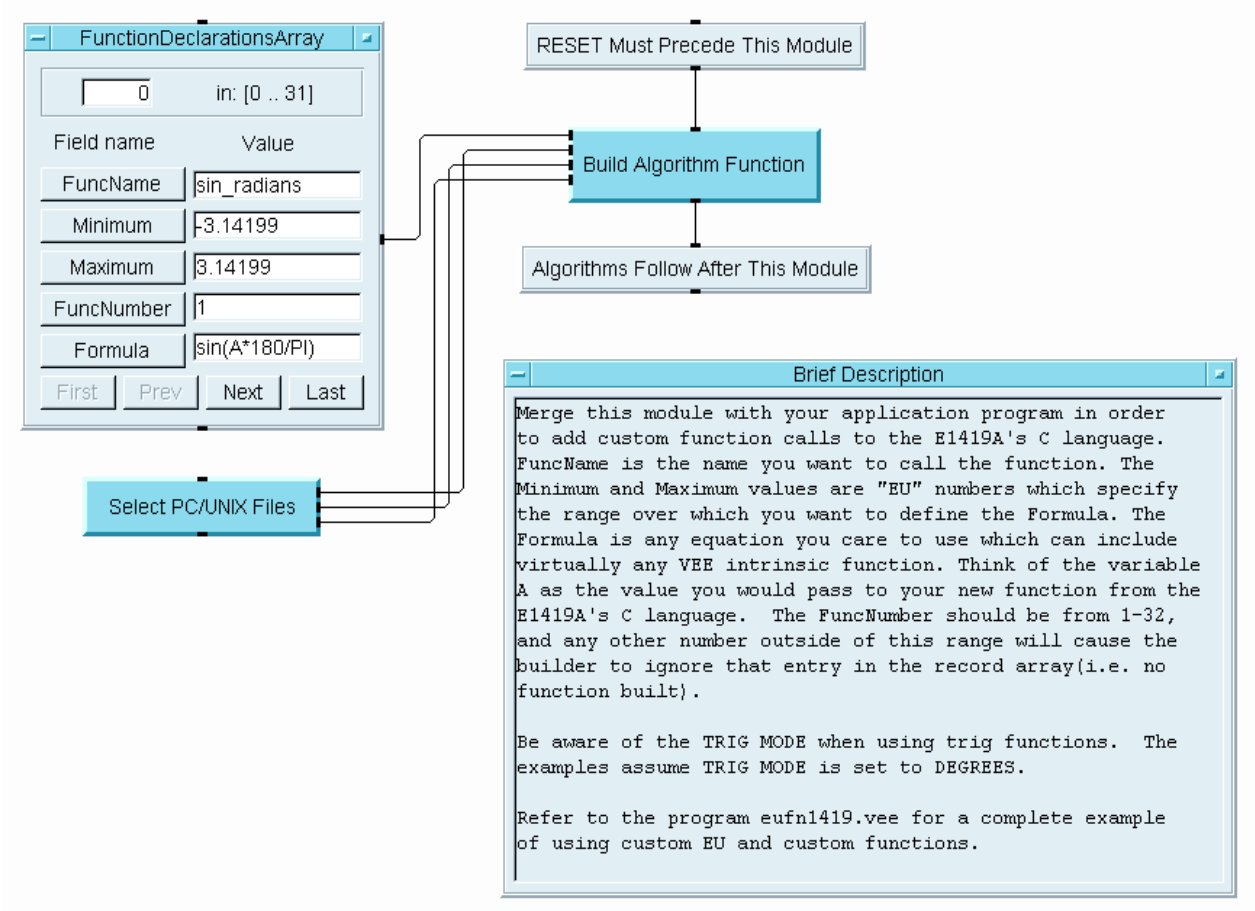


Figure 5-9 Custom Function Generation

Figure 5-9 illustrates where this module would be integrated into your VEE application program. This module must come after RESET and before any

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

algorithm is defined that would use a function. You simply pick the name of your function, the domain of input values(Minimum and Maximum), a unique function number between 1 and 32, and the formula you wish to use which includes any HP VEE math function. It's that simple!

The only restriction is that you must use the variable "A" as representing the value you will pass to the function from the C algorithm. When the function is called, that value will be "inserted" into your formula just as represented in the formula box.

Also note that the accuracy of this piece-wise linear table conversion technique is highly dependent upon the non-linearity and domain over which you try to build the tables. The table consists of 128 segments spread over a binary representation of the domain limits. Appendix F gives some background information on the capabilities and limitations of this programming technique.

The FunctionDeclarationsArray can hold up to 32 function definitions. Any valid function number 1-32 will cause the associated table to be built and downloaded into the E1419's function table memory space. A value of "0" for any FuncNumber will cause that function to be ignored and not downloaded. Any *RST or power-ON condition will require re-execution of this module.

The object Select PC/UNIX Files contains file name and directory paths necessary to make the module execute properly on PC's or UNIX platforms. Figure 5-8 of the previous example shows that object open for your observation. You can see that the default location of the E1419 example programs is "c:\dabundle". A typical UNIX path is included for example. The example uses the HP VEE function whichOS() to determine which directory structure to use.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Custom EU/Function Example

eufn1419.vee: This program operates stand-alone. It is designed to show you how easy it is to generate complicated EU conversion and Custom functions by simply entering in channel numbers, function names, and algebraic expressions. You need to convert volts to pressure or perform a square-root operation? You can use this program to see how easy it is to perform.

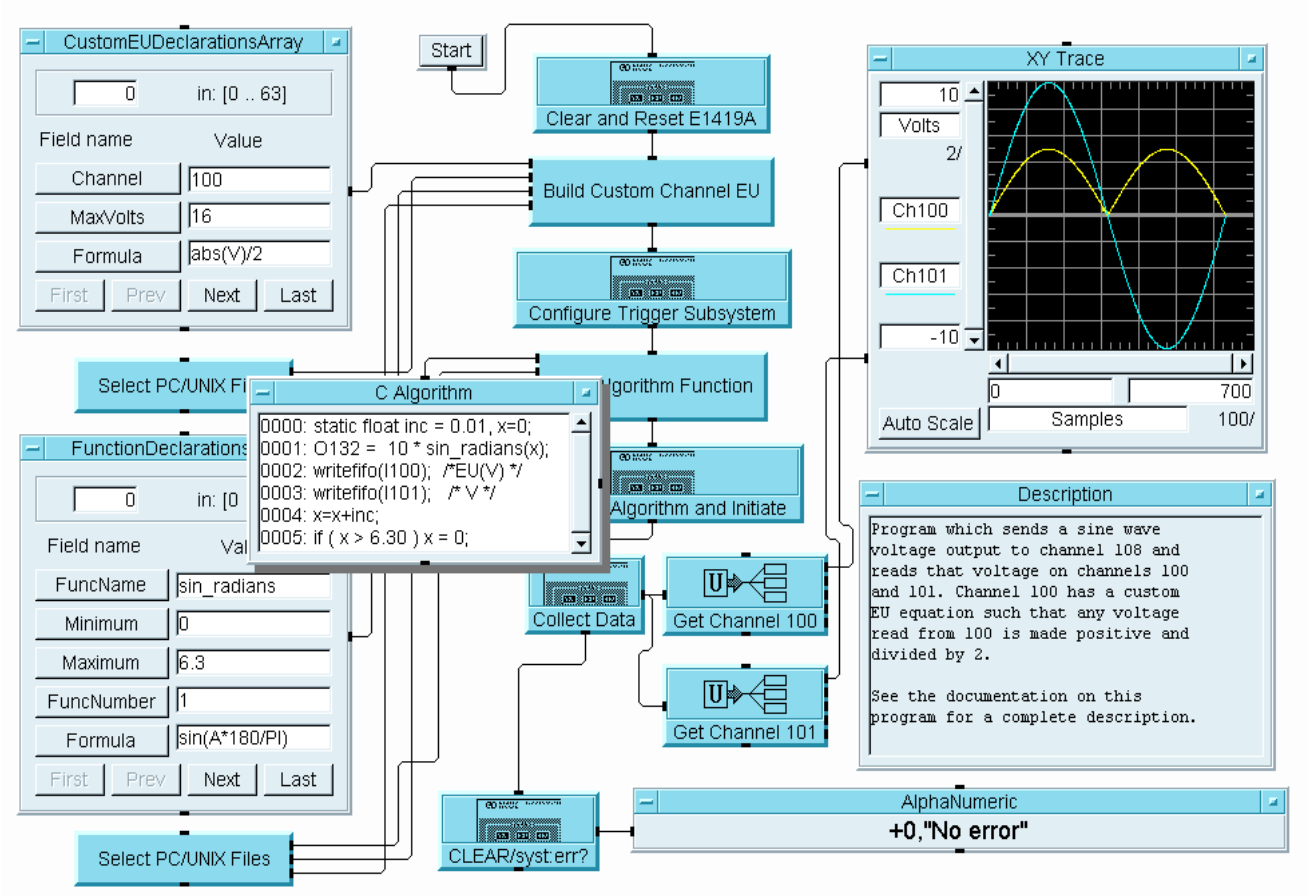


Figure 5-10 Custom EU/Function Example Detailed View

This program illustrates how to generate a sine wave from a custom function that is then used to program an analog output. The analog output (channel 132) is assumed wired to analog input channels 100 and 101. Channel 101 is the straight voltage from channel 132, and channel 100 is the same voltage but processed by the EU conversion formula for channel 100, as defined in the CustomEUDeclarationArray. The EU conversion formula simply takes the voltage read from channel 100, takes its absolute value, and divides it by 2.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Notice that the domain of the sin_radians() function is limited to 0-6.3, which represent a 0-2*PI interval. Each time the algorithm executes, it writes the new value of O132 based upon the sin_radians() function with the passed parameter "inc". "inc" is incremented once for each trigger since each trigger causes the algorithm to execute. When "inc" exceeds 6.3, it is set back to 0. Also note that the analog input voltages are sent to the FIFO after each trigger. The object Collect Data retrieves the voltage pairs and assembles them into a 2-dimension array which is then separated by Get Channel 100 and Get Channel 101. The results are passed on to the X-Y trace for display.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Curve Fitting and EU Generation

regr1419.vee: This program operates stand-alone. It shows how you could use the HP VEE regression tools to generate a polynomial equation to fit volts and pressure. The generated equation can then be used in the eu_1419.vee module for converting volts to pressure during data acquisition of the E1419.

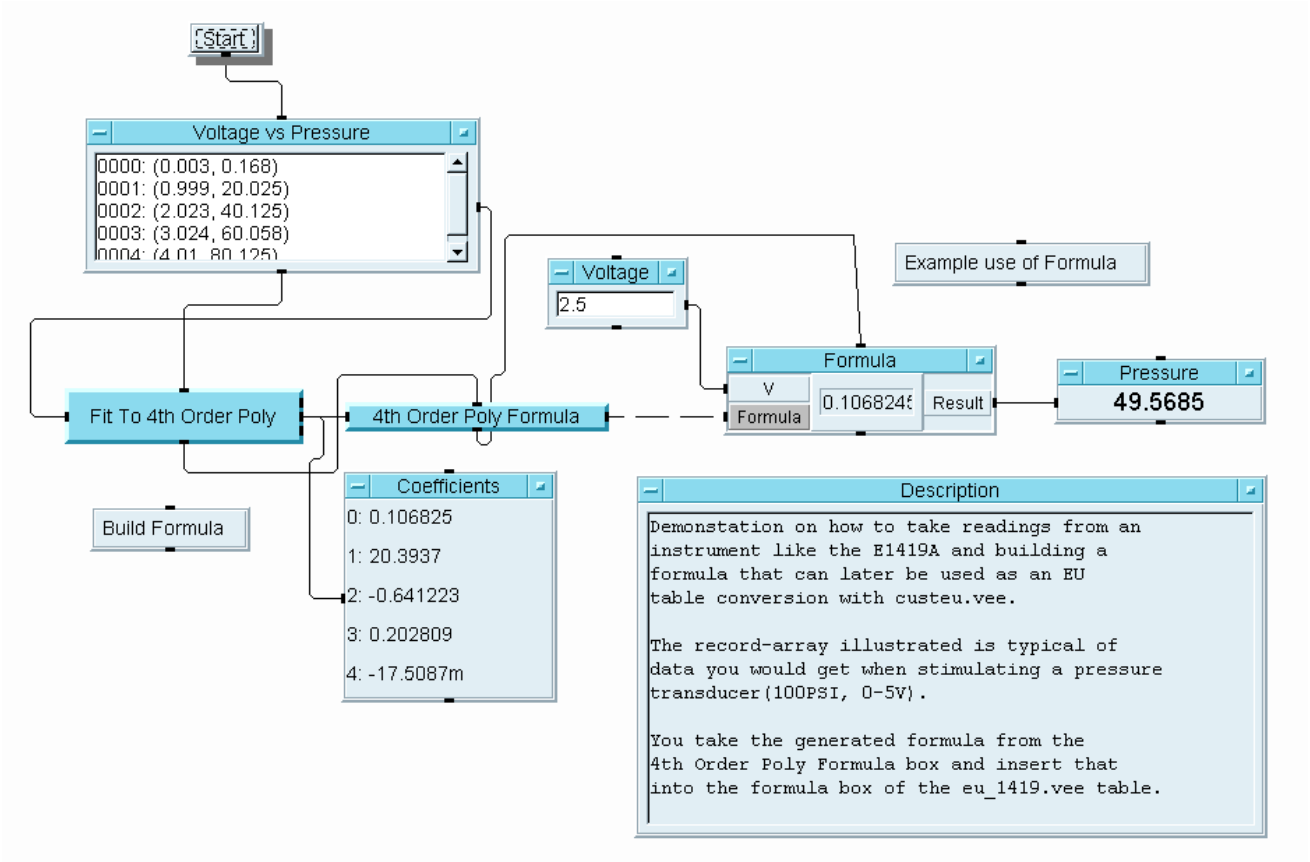


Figure 5-11 Curve Fitting and EU Generation

The "Fit To 4th Order Poly" object takes the data pairs entered into the "Voltage vs. Pressure" text object and generates the coefficients that can be entered into a 4th order polynomial. The coefficients are automatically entered into the "4th Order Poly Formula" object and a dry-lab example of a 2.5 volt input to the formula results in the 49.5685 PSI output. This is what the EU conversion would perform in the E1419 if you entered that 4th order poly into the EU Conversion table object. You simply use the same formula generated by this example along with the specified coefficients.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrupt Handling

intr1419.vee: This program operates stand-alone. This is an example program that shows how to create multiple threads of operation in HP VEE to respond to a FIFO half-full interrupt. It teaches you the concepts of interrupt driven programming. The example temp1419.vee also incorporates a slightly different version of interrupt processing that can enhance your learning.

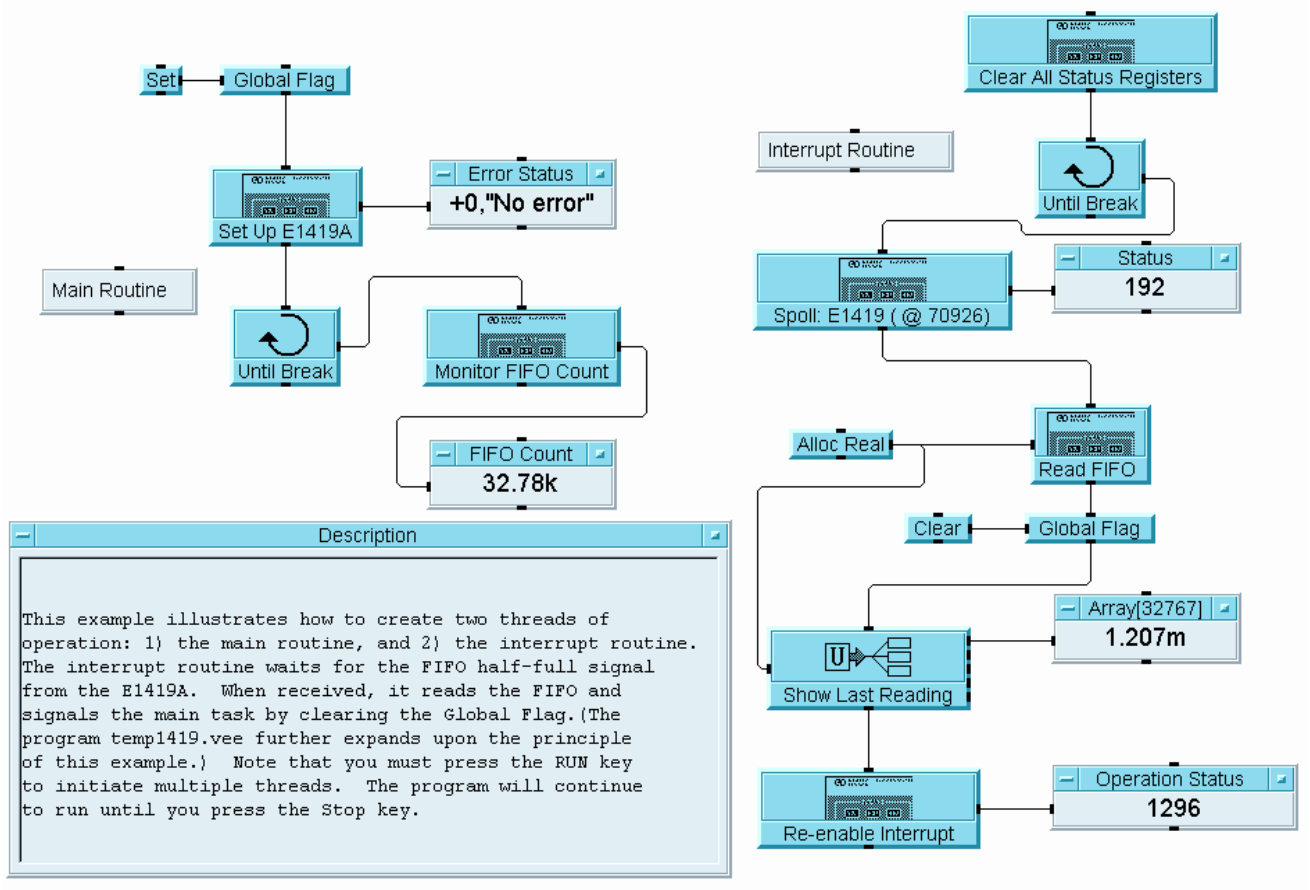


Figure 5-12 Interrupt Handling Detailed View

This example is really quite simple. It's the technique of knowing how to set up HP VEE to handle interrupts that is tricky. The Interrupt Handler has a REPEAT loop connected directly to the Spoll object which monitors the HP-IB waiting for the interrupt condition that was configured. The "Set Up E1419" object shows the necessary SCPI commands to configure for an interrupt, shows a simple algorithm that places readings into the FIFO, and configures the trigger subsystem. All these topics are covered in Chapter 3 with details about the various SCPI commands in Chapter 6.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Interrupt Handler simply waits for the FIFO-HALF-FULL interrupt, reads half the FIFO, displays the result or one reading, and re-enables the condition once again. Once you understand this example, you will easily understand how to handle other interrupts which are described in the Status Subsystem section in Chapter 3. The example temp1419.vee is another program you can load which shows interrupt handling.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Simple Data Logger Example

dlgr1419.vee: This program operates stand-alone. It illustrates how to configure the E1419 to collect data, store that data into its FIFO, and retrieve that data for display on a strip chart and optional logging to a file. This program can also be used to read stored data files generated by both this examples and the panl1419.vee example. The example can easily be modified to a more complicated version or you can cut and paste the pieces you need.

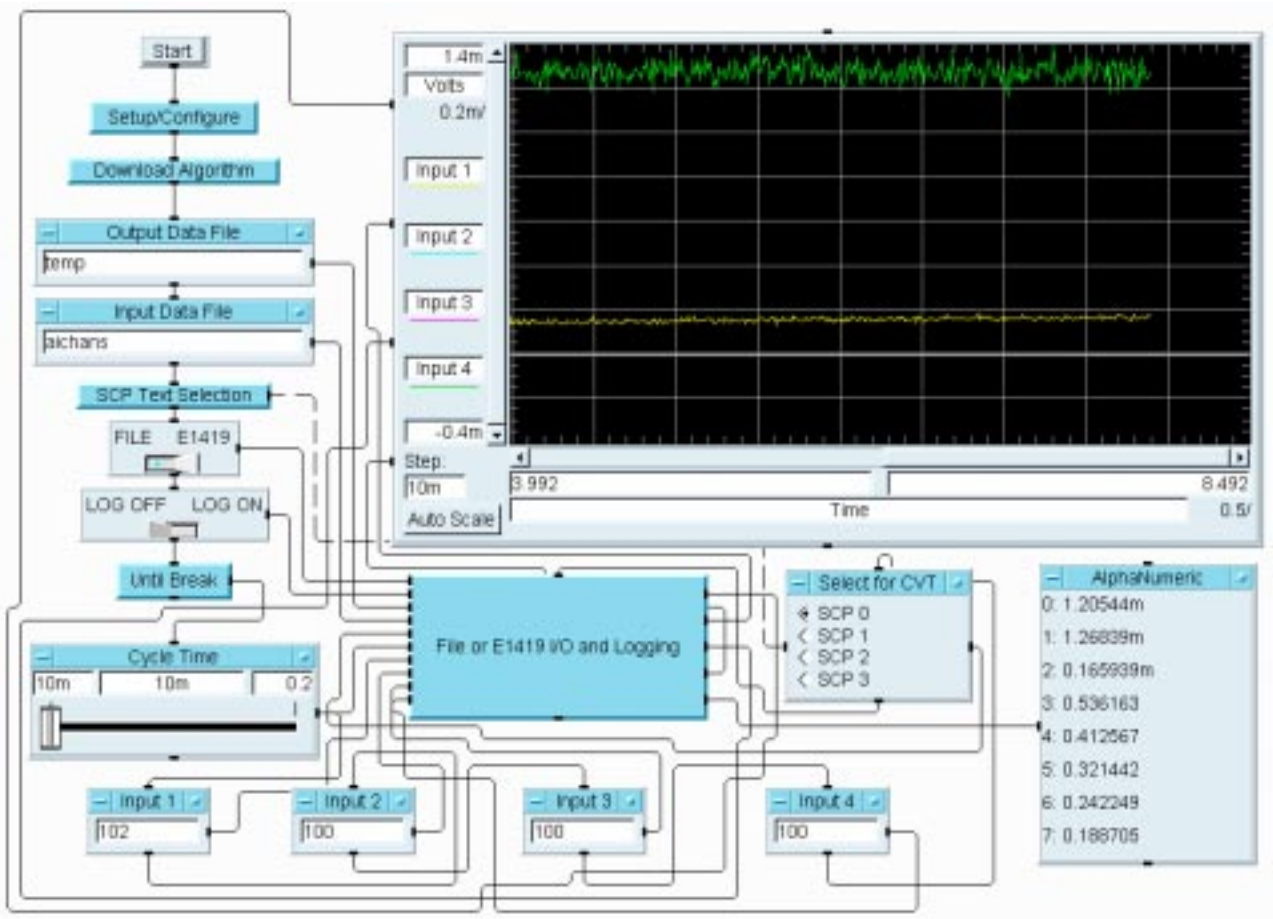


Figure 5-13 Simple Data Logger

The first object, Setup/Configure, can be used to configure the E1419 trigger subsystem, SCP's, and data formats. Since the E1419 comes pre-configured with 4 analog input SCP's in the first 4 slots, this example will concentrate only on those channels and leave them configured for voltage input. Since the example panl1419.vee also assumes this condition, data stored by that example's logging function can also be read by this example. If desired, you can refer to the example temp1419.vee to see how you can configure channels for temperature measurements using SCPI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

commands. Chapter 3 also illustrates how to configure analog channels for other measurements such as resistance, for example.

Note the "TRIG:TIMER 0.01" command will establish the scan trigger rate at which measurements are taken and C algorithms are executed. This rate was chosen purposely to illustrate the concept of slowing down data acquisition at multiples of 10msec. Also note that the data format of "FORM REAL,32" is used so the maximum rate can be achieved when reading data from the FIFO.

The second object, Download Algorithm, illustrates how to download a C program to access the channel variables for input measurements. This object consists of a REPEAT loop with a count range to automatically generate 32 of the "writefifo(Ixx);" statements. The Ixx will range from I100-I131, which represents each of the first 32 analog input channels. The only function this algorithm will have is to read all 32 analog input values for each scan trigger and place that data into the FIFO.

The Output and Input Data File text boxes allow you to specify names of data files. The Output Data File assumes the working directory unless you specify the entire path. This file will be cleared upon executing the "RUN" key of HP VEE. If you have the LOG ON switch set, all data acquired will be written to the specified data file. If you have selected the E1419 switch, then data is acquired from the actual input channels. If you select FILE, data is read from the specified Input Data File as though it were coming from the input channels. The assumption is that this data file was created with this example using the LOG ON mode or created with the logging function of the example panl1419.vee. Note that both the LOG ON/OFF and the FILE/E1419 switches come BEFORE the REPEAT loop. Therefore, you cannot modify these parameters AFTER executing the "RUN".

The REPEAT provides the rate at which HP VEE can perform the following actions:

- Reads the desired algorithm execution rate for storing data into the FIFO.
- Reads the desired channels to display on the strip chart.
- Reads 10-scans(320 values) of data from the FIFO.
- Writes 10 readings for each selected channel to the strip chart.
- Reads the desired SCP channels to display a Current Value Table(CVT) of data returned and displays that data.

The more operations you place in this loop, the more time you will place between accesses to the FIFO. The execution speed of HP VEE is dependent upon the speed of your computer is, how many I/O operations it is performing, and whether or not you are using the Compiled mode of HP VEE 4.0. If you are running a Pentium-class PC, the REPEAT loop will easily keep up with the acquisition rate of the E1419 card and provide very

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

near real-time data on the strip chart. Slower computers may fall behind. You can alter the TRIG:TIMER interval to slow down the acquisition rate or move the slider control to slow down the rate of placing data into the FIFO.

Note that you can select which SCP data you want to monitor on the alphanumeric display. Each SCP can have up to 8 channels of analog input, so selecting SCP 0-3 will allow you to see all 8 channels. The data displayed is just one of the 10 readings acquired from the 320 FIFO readings. Normally, the better choice is to use the E1419's CVT and read that directly; however, since the data was already read, adding the additional I/O statement to fetch the 8 channels from the CVT is an unnecessary performance slow-down for this application.

The four Integer input boxes labeled Input 1-4 allow you to specify which channels you want to display on the strip chart. These are scanned as part of the REPEAT loop that acquires the readings from the E1419 card. Ten readings for each of the selected channels are fetched from the FIFO data and sent to the strip chart.

The Cycle Time object allows you to slow down the rate at which data is placed in the FIFO by the E1419's C algorithm. The SCPI command ALG:SCAN:RATIO is used to cause the C algorithm to skip execution intervals established by the scan triggers. Since the "TRIG:TIMER 0.01" command was issued during Setup/Configure, this slider will convert to multiples of this rate. For example, if you select the Cycle Time of 0.04 seconds, then the C algorithm will only execute every 4 scan triggers.

The default 10msec interval of the scan trigger is also used in HP VEE's strip chart object. So, if you leave the Cycle Time set to 10msec and leave the Step size to 10msec, the strip chart data will represent the actual data acquisition time. If you modify the Cycle Time, you should also modify the Step on the strip chart. Unfortunately, the Step is not one that can be modified by adding an input terminal to the object. But, you can modify it in real-time with your keyboard entry. Keep in mind that the new step value will be assumed for all previous data too. Therefore, it's best to select a rate, program the Step value, and then "RUN" the HP VEE example for the most accurate results.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modification of Variables and Arrays

updt1419.vee: This program operates stand-alone. This example shows how operator interaction with running algorithms takes place and how to download changes for both scalar and array variables.

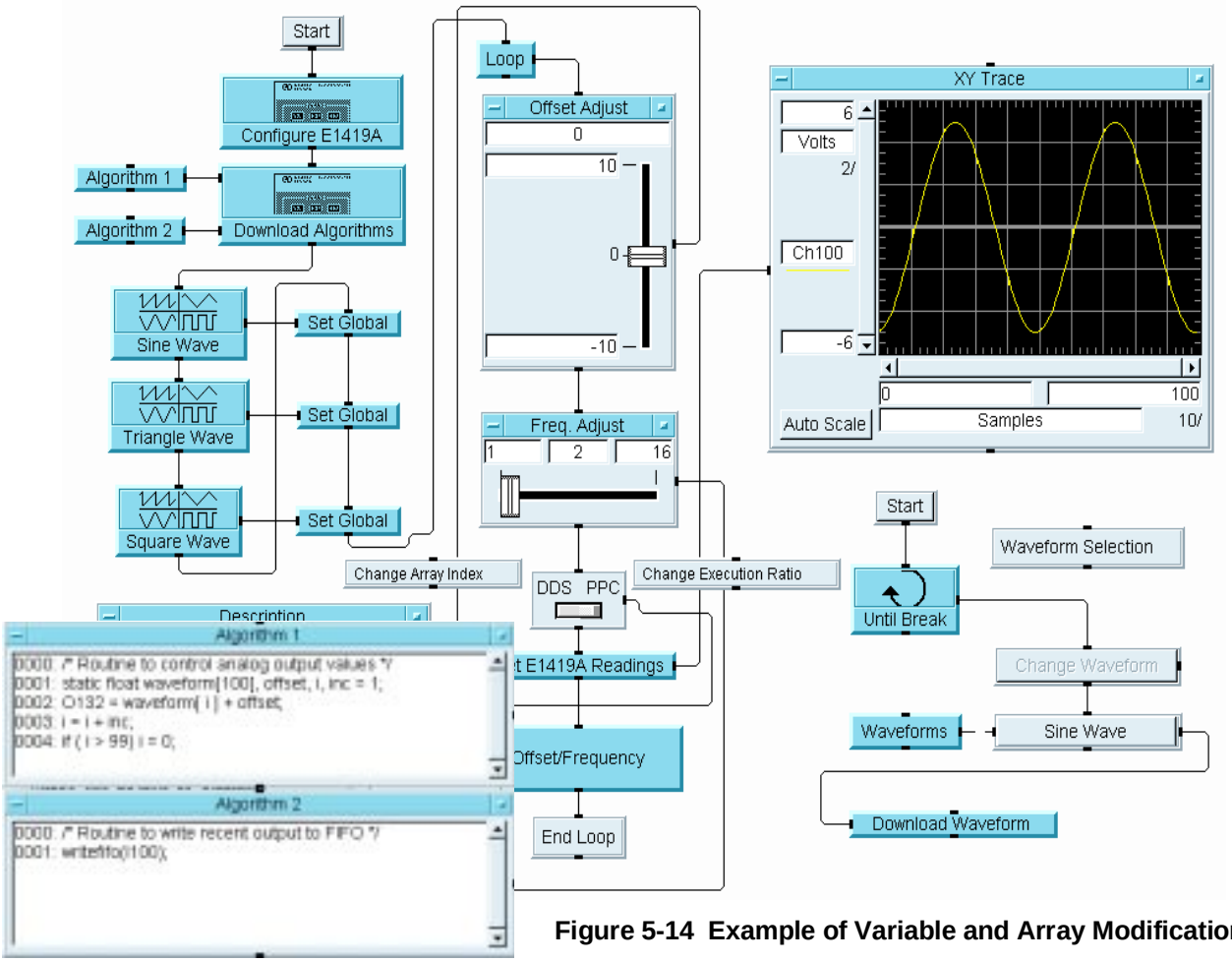


Figure 5-14 Example of Variable and Array Modification

Analog output channel 132 is assumed connected to analog input channel 100 for this example. Rather than use a custom function to generate the sine wave, HP VEE's function generator objects are used to generate a sine wave, triangle wave, and square wave. There are three 100-element arrays created that will be downloaded into the E1419's memory, dependent upon which waveform you want to generate. Algorithm 1 is expanded in the picture above and shows how it sequences through the array "waveform" to send values to the analog output. With each trigger cycle, Algorithm 1 executes and picks a value from the array dependent upon a counter variable(i). The variable "inc" is used to increment the counter so elements in the array can be skipped to generate a higher frequency waveform. Also note in Algorithm 1 that the output value to O132 consists of both the "waveform" array plus the variable "offset".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The vertical slider controls the value of "offset" and the horizontal slider controls the variable "inc". When the toggle switch is in the DDS(direct digital synthesis) mode, the horizontal slider modifies "inc" to generate lower resolution/higher frequency waveforms. When in the PPC(point per cycle) mode, the slider modifies the ALG:SCAN:RATIO command of Algorithm 1 to vary how many trigger cycles to wait before executing the algorithm and writing different data to the output channel . This has the effect of slowing down the waveform and lowering its frequency. Algorithm 2 simply copies each value of channel 132 to the FIFO every trigger cycle. With Algorithm 1 only executing at some multiple of the trigger rate, there will be repeated FIFO readings of the same value indicating a slower frequency.

The "Offset/Frequency" object has the SCPI commands used to control the scalar variable updates, and the object "Download Waveform" controls writing to the array "waveform" in Algorithm 1.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Algorithm Modification

swap1419.vee: This program operates stand-alone. It shows you how to modify algorithms while the E1419 is running. It includes further examples on custom function generation.

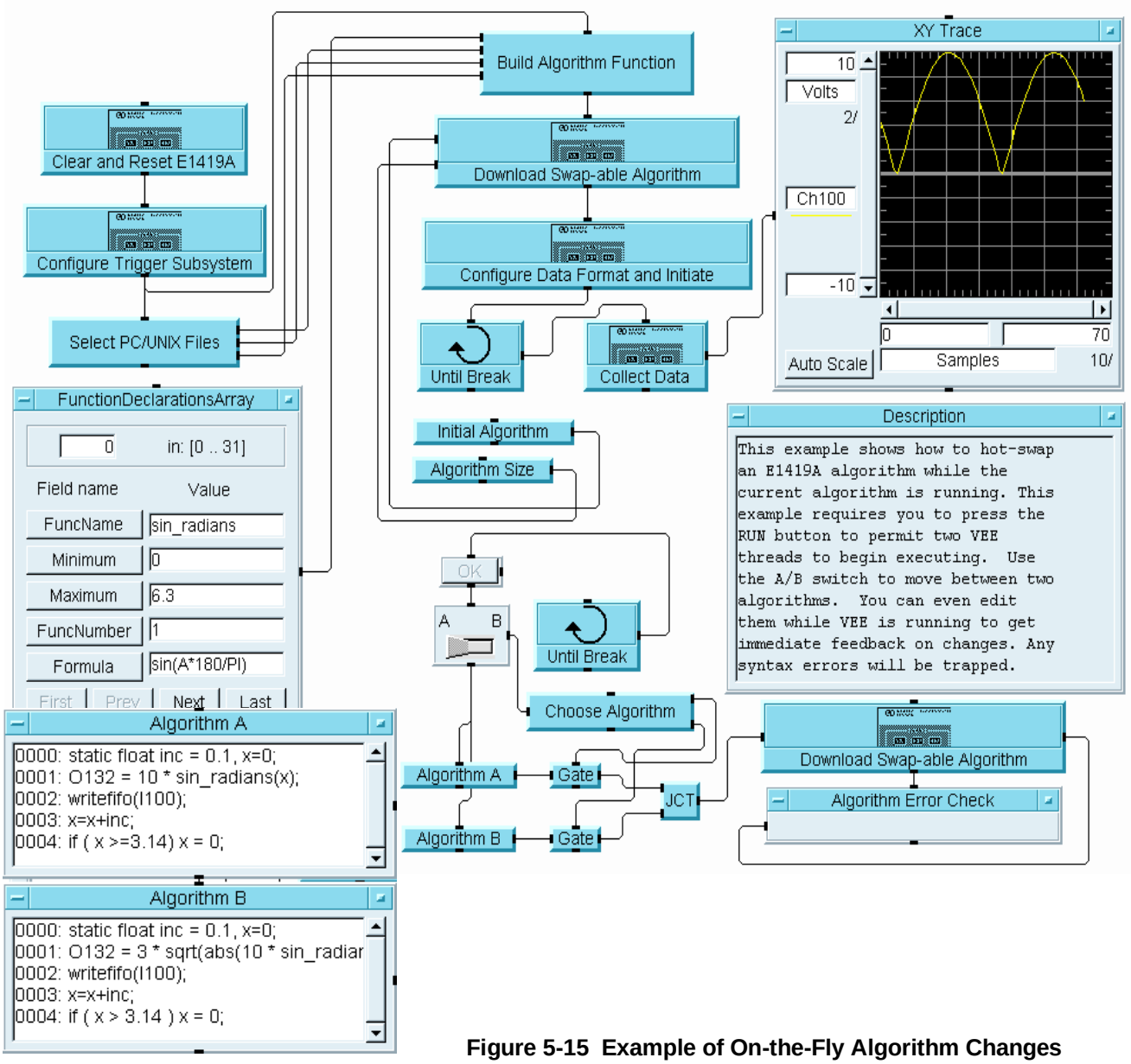


Figure 5-15 Example of On-the-Fly Algorithm Changes

Analog output channel 132 is assumed connected to analog input channel 100 for this example. Rather than use a custom function to generate the sine wave, HP VEE's function generator objects are used to generate a sine wave, triangle wave, and square wave. There are three 100-element arrays created that will be downloaded into the E1419's memory, dependent upon

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

which waveform you want to generate. Algorithm 1 is expanded in the picture above and shows how it sequences through the array "waveform" to send values to the analog output. With each trigger cycle, Algorithm 1 executes and picks a value from the array dependent upon a counter variable(i). The variable "inc" is used to increment the counter so elements in the array can be skipped to generate a higher frequency waveform. Also note in Algorithm 1 that the output value to O132 consists of both the "waveform" array plus the variable "offset".

The vertical slider controls the value of "offset" and the horizontal slider controls the variable "inc". When the toggle switch is in the DDS(direct digital synthesis) mode, the horizontal slider modifies "inc" to generate lower resolution/higher frequency waveforms. When in the PPC(point per cycle) mode, the slider modifies the ALG:SCAN:RATIO command of Algorithm 1 to vary how many trigger cycles to wait before executing the algorithm and writing different data to the output channel . This has the effect of slowing down the waveform and lowering its frequency. Algorithm 2 simply copies each value of channel 132 to the FIFO every trigger cycle. With Algorithm 1 only executing at some multiple of the trigger rate, there will be repeated FIFO readings of the same value indicating a slower frequency.

The "Offset/Frequency" object has the SCPI commands used to control the scalar variable updates, and the object "Download Waveform" controls writing to the array "waveform" in Algorithm 1.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Driver Download

drvr1419.vee: This program allows you to download the E1419 driver and any other drivers you might need into the HP E1405/6 Command Module. You specify the directory of where to find the driver files and the actual driver files(.DU) you want downloaded into the E1405/6 Driver RAM. The program will first list the drivers found in the E1405/6's memory, and you must press the CONTINUE button to proceed with the download.

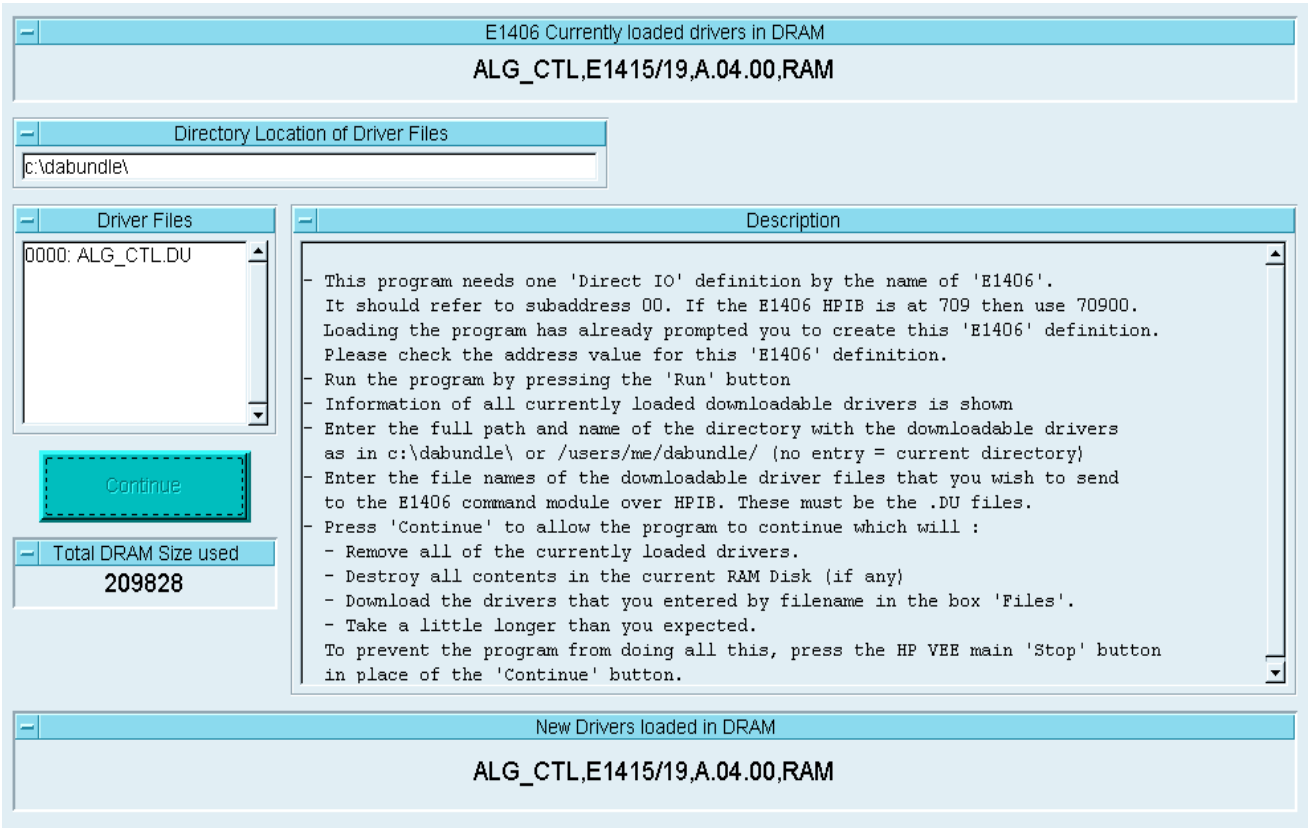


Figure 5-16 Example of Command Module Driver Download

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Firmware-Update Download

flsh1419.vee: This program allows you to save and reprogram the FLASH memory of the E1419. Updating the flash memory for the E1419 is usually a rare occurrence, but should a new revision become available, you can download that new firmware into the E1419's flash memory. To safe-guard against the remote chance that the new flash causes you problems, the program also allows you to save your old flash memory.

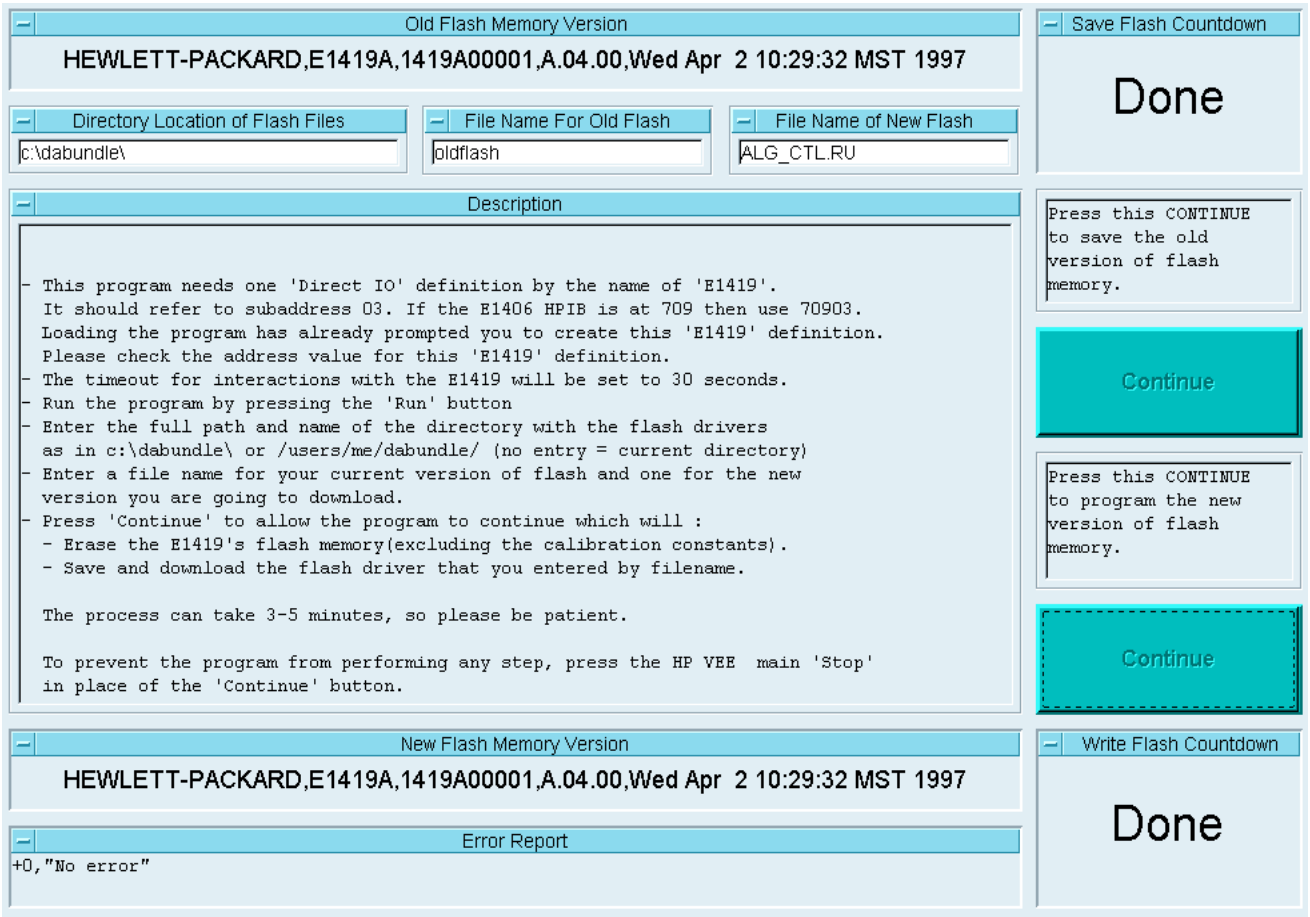


Figure 5-17 Example of Firmware (Flash) Download

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 6
HP E1419 Command Reference

Using This Chapter

This chapter describes the **Standard Commands for Programmable Instruments** (SCPI) command set and the **IEEE-488.2 Common Commands** for the HP E1419.

- Overall Command Index 177
- Command Fundamentals 182
- SCPI Command Reference 188
- Common Command Reference 316
- Command Quick Reference 326

Overall Command Index

SCPI Commands

ABORt 189

ALGorithm[:EXPLicit]:ARRay *<alg_name>,<array_name>,<block_data>* 191
ALGorithm[:EXPLicit]:ARRay? *<alg_name>,<array_name>* 192
ALGorithm[:EXPLicit]:DEFine *<alg_name>*,[*<swap_enable>,<size>*,]*<source_code>* 192
ALGorithm[:EXPLicit]:SCALar *<alg_name>,<var_name>,<value>* 196
ALGorithm[:EXPLicit]:SCALar? *<alg_name>,<var_name>* 197
ALGorithm[:EXPLicit]:SCAN:RATio *<alg_name>,<value>* 197
ALGorithm[:EXPLicit]:SCAN:RATio? *<alg_name>* 198
ALGorithm[:EXPLicit]:SIZe? *<alg_name>* 199
ALGorithm[:EXPLicit][:STATe] *<alg_name>*,1 | 0 | ON | OFF 199
ALGorithm[:EXPLicit][:STATe]? *<alg_name>* 200
ALGorithm[:EXPLicit]:TIME? *<alg_name>* 201
ALGorithm:FUNCtion:DEFine *<func_name>,<range>,<offset>,<func_data>* 201
ALGorithm:OUTPut:DELay *<delay>* | AUTO 203
ALGorithm:OUTPut:DELay? 204
ALGorithm:UPDate[:IMMediate] 204
ALGorithm:UPDate:CHANnel (@*<channel>*) 205
ALGorithm:UPDate:WINDow *<num_updates>* 206
ALGorithm:UPDate:WINDow? 207

ARM[:IMMediate] 209
ARM:SOURce BUS | EXT | HOLD | IMM | SCP | TTLTrg<n> 209
ARM:SOURce? 210

CALibration:CONFigure:RESistance 212

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CALibration:CONFigure:VOLTage <*range*>
CALibration:SETup 214
CALibration:SETup? 214
CALibration:STORe ADC | TARE 215
CALibration:TARE (@<*ch_list*>) 216
CALibration:TARE:RESet 218
CALibration:TARE? 218
CALibration:VALue:RESistance <*ref_ohms*> 219
CALibration:VALue:VOLTage <*ref_volts*> 219
CALibration:ZERO? 220

DIAGnostic:CALibration:SETup[:MODE] 0 | 1 223
DIAGnostic:CALibration:SETup[:MODE]? 223
DIAGnostic:CALibration:TARE[:OTDetect][:MODE] 0 | 1 224
DIAGnostic:CALibration:TARE[:OTDetect][:MODE]? 224
DIAGnostic:CHECksum? 225
DIAGnostic:CUStom:LINear <*table_range*>,<*table_block*>,(@<*ch_list*>) 225
DIAGnostic:CUStom:PIECewise <*table_range*>,<*table_block*>,(@<*ch_list*>) 226
DIAGnostic:CUSTom:REFerence:TEMPerature 226
DIAGnostic:IEEE 0 | 1 227
DIAGnostic:IEEE? 227
DIAGnostic:INTerrupt[:LINe] <*int_line*> 228
DIAGnostic:INTerrupt[:LINe]? 228
DIAGnostic:OTDetect[:STATe] 1 | 0 | ON | OFF,(@<*ch_list*>) 228
DIAGnostic:OTDetect[:STATe]? (@<*channel*>) 229
DIAGnostic:QUERy:SCPREAD? <*reg_addr*> 230
DIAGnostic:VERSion? 230

FETCh? 231

FORMat[:DATA] <*format*>[,<*size*>] 233
FORMat[:DATA]? 234

INITiate[:IMMediate] 236

INPut:DEBounce:TIME <*time*>,(@<*ch_list*>) 237
INPut:FILTer[:LPASs]:FREQuency <*cutoff_freq*>,(@<*ch_list*>) 238
INPut:FILTer[:LPASs]:FREQuency? (@<*channel*>) 239
INPut:FILTer[:LPASs][:STATe] 1 | 0 | ON | OFF,(@<*ch_list*>) 240
INPut:FILTer[:LPASs][:STATe]? (@<*channel*>) 240
INPut:GAIN <*chan_gain*>,(@<*ch_list*>) 241
INPut:GAIN? (@<*channel*>) 242
INPut:LOW <*wvolt_type*>,(@<*ch_list*>) 242
INPut:LOW? (@<*channel*>) 243
INPut:POLarity NORMal | INverted,(@<*ch_list*>) 243
INPut:POLarity? (@<*channel*>) 243
INPut:THReshold:LEVel? (@<*channel*>) 244

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MEMory:VME:ADDRess *<A24_address>* . . . 246
MEMory:VME:ADDRess? . . . 246
MEMory:VME:SIZE *<mem_size>* . . . 246
MEMory:VME:SIZE? . . . 247
MEMory:VME:STATe 1 | 0 | ON | OFF . . . 247
MEMory:VME:STATe? . . . 248

OUTPut:CURRent:AMPLitude *<amplitude>*,(@*<ch_list>*) . . . 249
OUTPut:CURRent:AMPLitude? (@*<channel>*) . . . 250
OUTPut:CURRent[:STATe] 1 | 0 | ON | OFF,(@*<ch_list>*) . . . 251
OUTPut:CURRent[:STATe]? (@*<channel>*) . . . 251
OUTPut:POLarity NORMal | INVerted,(@*<ch_list>*) . . . 252
OUTPut:POLarity? (@*<channel>*) . . . 252
OUTPut:SHUNt 1 | 0,(@*<ch_list>*) . . . 253
OUTPut:SHUNt? (@*<channel>*) . . . 253
OUTPut:TTLTrg:SOURce ALGorithm | FTRigger | SCPlugon | TRIGger . . . 254
OUTPut:TTLTrg:SOURce? . . . 254
OUTPut:TTLTrg<n>[:STATe] 1 | 0 | ON | OFF . . . 255
OUTPut:TTLTrg<n>[:STATe]? . . . 255
OUTPut:TYPE PASSive | ACTive,(@*<ch_list>*) . . . 255
OUTPut:TYPE? (@*<channel>*) . . . 256
OUTPut:VOLTage:AMPLitude *<amplitude>*,(@*<ch_list>*) . . . 256
OUTPut:VOLTage:AMPLitude? (@*<channel>*) . . . 257

ROUTe:SEQuence:DEFine? AIN | AOUT | DIN | DOUT . . . 258
ROUTe:SEQuence:POINts? AIN | AOUT | DIN | DOUT . . . 259

SAMPle:TIMer *<interval>* . . . 260
SAMPle:TIMer? . . . 260

[SENSe:]CHANnel:SETTling <settle_time>,(@*<ch_list>*) . . . 263
[SENSe:]CHANnel:SETTling? (@*<channel>*) . . . 264
[SENSe:]DATA:CVTable? (@*<element_list>*) . . . 264
[SENSe:]DATA:CVTable:RESet . . . 265
[SENSe:]DATA:FIFO[:ALL]? . . . 266
[SENSe:]DATA:FIFO:COUNT? . . . 266
[SENSe:]DATA:FIFO:COUNT:HALF? . . . 267
[SENSe:]DATA:FIFO:HALF? . . . 267
[SENSe:]DATA:FIFO:MODE BLOCk | OVERwrite . . . 268
[SENSe:]DATA:FIFO:MODE? . . . 269
[SENSe:]DATA:FIFO:PART? *<n_readings>* . . . 269
[SENSe:]DATA:FIFO:RESet . . . 270
[SENSe:]FREQuency:APERture *<gate_time>*,(@*<ch_list>*) . . . 270
[SENSe:]FREQuency:APERture? (@*<channel>*) . . . 271
[SENSe:]FUNCtion:CONDition (@*<ch_list>*) . . . 271
[SENSe:]FUNCtion:CUSTom [*<range>*,](@*<ch_list>*) . . . 272

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]FUNCtion:CUSTom:REFerence [<*range*>,](@<*ch_list*>) . . . 273
[SENSe:]FUNCtion:CUSTom:TCouple <*type*>,[<*range*>,](@<*ch_list*>) . . . 274
[SENSe:]FUNCtion:FREQuency (@<*ch_list*>) . . . 275
[SENSe:]FUNCtion:RESistance <*excite_current*>,[<*range*>,] @<*ch_list*>) . . . 275
[SENSe:]FUNCtion:STRain:FBENding [<*range*>,](@<*ch_list*>) . . . 277
[SENSe:]FUNCtion:STRain:FBPoisson [<*range*>,](@<*ch_list*>) . . . 277
[SENSe:]FUNCtion:STRain:FPOisson [<*range*>,](@<*ch_list*>) . . . 277
[SENSe:]FUNCtion:STRain:HBENding [<*range*>,](@<*ch_list*>) . . . 277
[SENSe:]FUNCtion:STRain:HPOisson [<*range*>,](@<*ch_list*>) . . . 277
[SENSe:]FUNCtion:STRain[:QUARter] [<*range*>,](@<*ch_list*>) . . . 277
[SENSe:]FUNCtion:TEMPerature <*sensor_type*>,<*sub_type*>,[<*range*>,] (@<*ch_list*>) . . . 278
[SENSe:]FUNCtion:TOTalize (@<*ch_list*>) . . . 280
[SENSe:]FUNCtion:VOLTage[:DC] [<*range*>,](@<*ch_list*>) . . . 280
[SENSe:]REFerence <*sensor_type*>,[<*sub_type*>,][<range>,] (@<*ch_list*>) . . . 281
[SENSe:]REFerence:CHANnels (@<*ref_channel*>),(@<*tc_channels*>) . . . 283
[SENSe:]REFerence:TEMPerature <*degrees_c*> . . . 284
[SENSe:]STRain:EXCitation <*excite_v*>,(@<*ch_list*>) . . . 284
[SENSe:]STRain:EXCitation? (@<*channel*>) . . . 285
[SENSe:]STRain:GFACtor <*gage_factor*>,(@<*ch_list*>) . . . 285
[SENSe:]STRain:GFACtor? (@<*channel*>) . . . 286
[SENSe:]STRain:POISson <*poisson_ratio*>,(@<*ch_list*>) . . . 286
[SENSe:]STRain:POISson? (@<*channel*>) . . . 287
[SENSe:]STRain:UNSTrained <*unstrained_v*>,(@<*ch_list*>) . . . 287
[SENSe:]STRain:UNSTrained? (@<*channel*>) . . . 288
[SENSe:]TOTalize:RESet:MODE INIT | TRIGger,(@<*ch_list*>) . . . 288
[SENSe:]TOTalize:RESet:MODE? (@<*channel*>) . . . 289

SOURce:FM[:STATe] 1 | 0 | O | OFF,(@<*ch_list*>) . . . 290
SOURce:FM[:STATe]? (@<*channel*>) . . . 291
SOURce:FUNCtion:[SHAPe:]CONDition (@<*ch_list*>) . . . 291
SOURce:FUNCtion:[SHAPe:]PULSe (@<*ch_list*>) . . . 292
SOURce:FUNCtion:[SHAPe:]SQUare (@<*ch_list*>) . . . 292
SOURce:PULM[:STATe] 1 | 0 | ON | OFF,(@<*ch_list*>) . . . 292
SOURce:PULM[:STATe]? (@<*channel*>) . . . 293
SOURce:PULSe:PERiod <*period*>,(@<*ch_list*>) . . . 293
SOURce:PULSe:PERiod? (@<*channel*>) . . . 294
SOURce:PULSe:WIDth <*width*>,(@<*ch_list*>) . . . 294
SOURce:PULSe:WIDth? (@<*channel*>) . . . 294

STATus:OPERation:CONDition? . . . 298
STATus:OPERation:ENABle <*enable_mask*> . . . 299
STATus:OPERation:ENABle? . . . 300
STATus:OPERation[:EVENt]? . . . 300
STATus:OPERation:NTRansition <*transition_mask*> . . . 300
STATus:OPERation:NTRansition? . . . 301
STATus:OPERation:PTRansition <*transition_mask*> . . . 302

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STATus:OPERation:PTRansition? . . . 302
STATus:PRESet . . . 303
STATus:QUEStionable:CONDition? . . . 304
STATus:QUEStionable:ENABle <*enable_mask*> . . . 304
STATus:QUEStionable:ENABle? . . . 305
STATus:QUEStionable[:EVENt]? . . . 305
STATus:QUEStionable:NTRansition <*transition_mask*> . . . 305
STATus:QUEStionable:NTRansition? . . . 306
STATus:QUEStionable:PTRansition <*transition_mask*> . . . 307
STATus:QUEStionable:PTRansition? . . . 307

SYSTem:CTYPe? (@<*channel*>) . . . 308
SYSTem:ERRor? . . . 308
SYSTem:VERSion? . . . 309

TRIGger:COUNt <*trig_count*> . . . 312
TRIGger:COUNt? . . . 312
TRIGger[:IMMediate] . . . 313
TRIGger:SOURce BUS | EXT | HOLD | IMM | SCP | TIMer | TTLTrg<n> . . . 313
TRIGger:SOURce? . . . 314
TRIGger:TIMer[:PERiod] <*trig_interval*> . . . 314
TRIGger:TIMer[:PERiod]? . . . 315

Common Commands

*CAL? . . . 316
*CLS . . . 317
*DMC <*name*>,<*cmd_data*> . . . 317
*EMC <*enable*> . . . 317
*EMC? . . . 317
*ESE . . . 317
*ESE? . . . 318
*ESR? . . . 318
*GMC? <*name*> . . . 318
*IDN? . . . 318
*LMC? . . . 319
*OPC . . . 319
*OPC? . . . 319
*PMC . . . 320
*RMC <*name*> . . . 320
*RST . . . 320
*SRE . . . 321
*SRE? . . . 321
*STB? . . . 321
*TRG . . . 321
*TST? . . . 322
*WAI . . . 325

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command Fundamentals

Commands are separated into two types: IEEE-488.2 Common Commands and SCPI Commands. The SCPI command set for the HP E1419 is 1990 compatible

Common Command Format

The IEEE-488.2 standard defines the Common commands that perform functions like reset, self-test, status byte query, etc. Common commands are four or five characters in length, always begin with the asterisk character (*), and may include one or more parameters. The command keyword is separated from the first parameter by a space character. Some examples of Common commands are:

```
*RST
*ESR 32
*STB?
```

SCPI Command Format

The SCPI commands perform functions like configuring channels, setting up the trigger system, and querying instrument states or retrieving data. A subsystem command structure is a hierarchical structure that usually consists of a top level (or root) command, one or more lower level commands, and their parameters. The following example shows part of a typical subsystem:

```
MEMory
    :VME
        :ADDRess  <A24_address>
        :ADDRess?
        :SIZE  <mem_size>
        :SIZE?
```

MEMory is the root command, :VME is the second level command, and :ADDRess, and SIZE are third level commands.

Command Separator

A colon (**:**) always separates one command from the next lower level command as shown below:

```
ROUTE:SEQUENCE:DEFINE?
```

Colons separate the root command from the second level command (ROUTE:SEQUENCE) and the second level from the third level (SEQUENCE:DEFINE?). If parameters are present, the first is separated from the command by a space character. Additional parameters are separated from each other by a commas.

Abbreviated Commands

The command syntax shows most commands as a mixture of upper and lower case letters. The upper case letters indicate the abbreviated spelling for the command. For shorter program lines, send the abbreviated form. For better program readability, send the entire command. The instrument will accept either the abbreviated form or the entire command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For example, if the command syntax shows SEQuence, then SEQ and SEQUENCE are both acceptable forms. Other forms of SEQuence, such as SEQUEN or SEQU will generate an error. You may use upper or lower case letters. Therefore, SEQUENCE, sequence, and SeQuEnCe are all acceptable.

Implied Commands

Implied commands are those which appear in square brackets (**[ ]**) in the command syntax. (Note that the brackets are not part of the command, and are not sent to the instrument.) Suppose you send a second level command but do not send the preceding implied command. In this case, the instrument assumes you intend to use the implied command and it responds as if you had sent it. Examine the INITiate subsystem shown below:

INITiate
    [:IMMediate]

The second level command :IMMediate is an implied command. To set the instrument's trigger system to INIT:IMM, you can send either of the following command statements:

INIT:IMM *or* INIT

Variable Command Syntax

Some commands will have what appears to be a variable syntax. As an example:
OUTPut:TTLTrg<n>:STATe ON

In these commands, the "<n>" is replaced by a number. No space is left between the command and the number because the number is not a parameter. The number is part of the command syntax. The purpose of this notation is to save a great deal of space in the Command Reference. In the case of …TTLTrg<n>…, n can be from 0 through 7. An example command statement:

OUTPUT:TTLTRG2:STATE ON

Parameters

Parameter Types. The following section contains explanations and examples of parameter types you will see later in this chapter.

Parameter Types	Explanations and Examples
Numeric	Accepts all commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation: 123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E-01. Special cases include MIN, MAX, and INFinity. A parameter that represents units may also include a units suffix. These are: Volts; V, mv=10^{-3}, uv=10^{-6}

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Ohms; ohm, kohm=10^3, mohm=10^6
Seconds; s, msec=10^{-3}, usec=10^{-6}
Hertz; hz, khz=10^3, mhz=10^6, ghz=10^9

The Comments section within the Command Reference will state whether a numeric parameter can also be specified in hex, octal, and/or binary.

#H7B, #Q173, #B1111011

Boolean

Represents a single binary condition that is either true or false.

ON, OFF, 1, 0.

Discrete

Selects from a finite number of values. These parameters use mnemonics to represent each valid setting.

An example is the TRIGger:SOURce <*source*> command where <*source*> can be BUS, EXT, HOLD, IMM, SCP, TIMer, or TTLTrg<n>.

Channel List

The general form of a single channel specification is:

ccnn

where cc represents the card number and nn represents the channel number.

Since the HP E1419 has an on-board 64 channel multiplexer, the card number will be 1 and the channel number can range from 00 to 63. Some example channel specifications:

channel 0=100, channel 5=105, channel 54=154

The General form of a channel range specification is:

ccnn:ccnn (colon separator)

(the second channel must be greater than the first)
Example:

channels 0 through 15=100:115

By using commas to separate them, individual and range specifications can be combined into a single channel list:

0, 5, 6 through 32, and 45=(@100,105,106:132,145)

Note that a channel list is always contained within "(@" and ")". The Command Reference always shows the "(@" and ")" punctuation:

(@<*ch_list*>)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Arbitrary Block Program and Response Data — This parameter or data type is used to transfer a block of data in the form of bytes. The block of data bytes is preceded by a preamble which indicates either 1) the number of data bytes which follow (definite length), or 2) that the following data block will be terminated upon receipt of a New Line message, and for HP-IB operation, with the EOI signal true (indefinite length). The syntax for this parameter is:

Definite Length — #<non-zero digit><digit(s)><data byte(s)>

Where the value of <non-zero digit> is 1-9 and represents the number of <digit(s)>. The value of <digit(s)> taken as a decimal integer indicates the number of <data byte(s)> in the block.

Example of sending or receiving 1024 data bytes:
#41024<byte><byte1><byte2><byte3><byte4>…
…<byte1021><byte1022><byte1023><byte1024>

OR

Indefinite Length — #0<data byte(s)><NL^END>

Examples of sending or receiving 4 data bytes:
#0<byte><byte><byte><byte><NL^END>

Optional Parameters. Parameters shown within square brackets (**[ ]**) are optional parameters. (Note that the brackets are not part of the command, and should not be sent to the instrument.) If you do not specify a value for an optional parameter, the instrument chooses a default value. For example, consider the FORMAT:DATA <*type*>[,<*length*>] command. If you send the command without specifying <*length*>, a default value for <*length*> will be selected depending on the <*type*> of format you specify. For example:

FORMAT:DATA ASC will set [,<*length*>] to the default for ASC of 7
FORMAT:DATA REAL will set [,<*length*>] to the default for REAL of 32
FORMAT:DATA REAL, 64 will set [,<*length*>] to 64

Be sure to place a space between the command and the first parameter.

Linking Commands

Linking commands is used when you want to send more than one complete command in a single command statement.

Linking IEEE-488.2 Common Commands with SCPI Commands. Use a semicolon between the commands. For example:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*RST;OUTP:TTLT3 ON *or* TRIG:SOUR IMM;*TRG

Linking Multiple complete SCPI Commands. Use both a semicolon and a colon between the commands. For example:

OUTP:TTLT2 ON;:TRIG:SOUR EXT

The semicolon as well as separating commands tells the SCPI parser to expect the command keyword following the semicolon to be at the same hierarchical level (and part of the same command branch) as the keyword preceding the semicolon. The colon immediately following the semicolon tells the SCPI parser to reset the expected hierarchical level to Root.

Linking a complete SCPI Command with other keywords from the same branch and level. Separate the first complete SCPI command from next partial command with the semicolon only. For example take the following portion of the [SENSE] subsystem command tree (the FUNCtion branch):

```
[SENSe:]
    FUNCtion
        :RESistance <range>,(@<ch_list>)
        :TEMPerature <sensor>[,<range>,](@<ch_list>)
        :VOLTage[:DC] [<range>,](@<ch_list>)
```

Rather than send a complete SCPI command to set each function, you could send:

FUNC:RES 10000,(@100:107);TEMP RTD, 92,(@108:115);VOLT (@116,123)

This sets the first 8 channels to measure resistance, the next 8 channels to measure temperature, and the next 8 channels to measure voltage.

NOTE The command keywords following the semicolon must be from the same command branch and level as the complete command preceding the semicolon or a -113,"Undefined header" error will be generated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C-SCPI Data Types

The following table shows the allowable type and sizes of the C-SCPI parameter data sent to the module and query data returned by the module. The parameter and returned value type is necessary for programming and is documented in each command in this chapter.

Data Types	Description
int16	Signed 16-bit integer number.
int32	Signed 32-bit integer number.
uint16	Unsigned 16-bit integer number.
uint32	Unsigned 32-bit integer number.
float32	32-bit floating point number.
float64	64-bit floating point number.
string	String of characters (null terminated)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SCPI Command Reference

The following section describes the SCPI commands for the HP E1419. Commands are listed alphabetically by subsystem and also within each subsystem. A command guide is printed in the top margin of each page. The guide indicates the current subsystem on that page.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The ABORt subsystem is a part of the HP E1419's trigger system. ABORt resets the trigger system from its Wait For Trigger state to its Trigger Idle state.

Subsystem Syntax

ABORt

CAUTION! ABORT stops execution of a running algorithm. The control output is left at the last value set by the algorithm. Depending on the process, this uncontrolled situation could even be dangerous. Make certain that you have put your process into a safe state before you halt execution of a controlling algorithm.

Comments

- ABORt does not affect any other settings of the trigger system. When the INITiate command is sent, the trigger system will respond just as it did before the ABORt command was sent.
- **Related Commands:** INITiate[:IMMediate], TRIGger…
- ***RST Condition:** TRIG:SOUR HOLD

Usage

ABORT

If INITed, goes to Trigger Idle state. If running algorithms, stops and goes to Trigger Idle State.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The ALGorithm command subsystem provides:

- Definition of measurement and control algorithms
- Communication with algorithm array and scalar variables
- Controls to enable or disable individual algorithms
- Control of ratio of number of scan triggers per algorithm execution
- Control of algorithm execution speed
- Easy definition of algorithm data conversion functions

Subsystem Syntax

```
ALGorithm
    [:EXPLicit]
        :ARRay <alg_name>,<array_name>,<block_data>
        :ARRay? <alg_name>,<array_name>
        :DEFine <alg_name>[,<swap_size>],<program_block>
        :SCALar <alg_name>,<var_name>,<value>
        :SCALar? <alg_name>,<var_name>
        :SCAN:RATio <alg_name>,<value>
        :SCAN:RATio? <alg_name>
        :SIZe? <alg_name>
        [:STATe] <alg_name>,ON | OFF
        [:STATe]? <alg_name>
        :TIME? <alg_name>
    :FUNCtion:DEFine <func_name>,<range>,<offset>,<block_data>
    :OUTPut:DELay <usec> | AUTO
    :OUTPut:DELay?
    :UPDate
        [:IMMediate]
        :CHANnel <channel_item>
        :WINDow <num_updates>
        :WINDow?
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ALGorithm[:EXPLicit]:ARRay

ALGorithm[:EXPLicit]:ARRay <*alg_name*>,<*array_name*>,<*array_block*> places values of <*array_name*> for algorithm <*alg_name*> into the Update Queue. This update is then pending until ALG:UPD is sent or an update event (as set by ALG:UPD:CHANNEL) occurs.

NOTE ALG:ARRAY places a variable update request in the Update Queue. You can not place more update requests in the Update Queue than are allowed by the current setting of ALG:UPD:WINDOW or a "Too many updates -- send ALG:UPDATE command' error message will be generated.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32 | GLOBALS	none
array_name	string	valid 'C' variable name	none
array_block	block data	block of IEEE-754 64-bit floating point numbers	none

Comments

- To send values to a Global array, set the <*alg_name*> parameter to "GLOBALS". To define a global array see the ALGorithm:DEFine command.
- An error is generated if <*alg_name*> or <*array_name*> is not defined.
- When an array is defined (in an algorithm or in 'GLOBALS'), the HP E1419 allocates twice the memory required to store the array. When you send the ALG:ARRAY command, the new values for the array are loaded into the second space for this array. When you send the ALG:UPDATE, or ALG:UPDATE:CHANNEL commands, the HP E1419 switches a pointer to the space containing the new array values. This is how even large arrays can be "updated" as if they were a single update request. If the array is again updated, the new values are loaded into the original space and the pointer is again switched.
- <*progname*> is not case sensitive. However, <*array_name*> is case sensitive.
- **Related Commands:** ALG:DEFINE, ALG:ARRAY?
- ***RST Condition:** No algorithms or variables are defined.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Usage

```
                    send array values to my_array in ALG4
ALG:ARR 'ALG4','my_array',<block_array_data>
                    send array values to the global array glob_array
ALG:ARR 'GLOBALS','glob_array',<block_array_data>
ALG:UPD                                          force update of variables
```

ALGorithm[:EXPLicit]:ARRay?

ALGorithm[:EXPLicit]:ARRay? <*alg_name*>,<*array_name*> returns the contents of <*array_name*> from algorithm <*alg_name*>. ALG:ARR? can return contents of global arrays when <*alg_name*> specifies 'GLOBALS'.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32 | GLOBALS	none
array_name	string	valid 'C' variable name	none

Comments

- An error is generated if <*alg_name*> or <*array_name*> is not defined.
- **Returned Value:** Definite length block data of IEEE-754 64-bit float

ALGorithm[:EXPLicit]:DEFine

ALGorithm[:EXPLicit]:DEFine '<*alg_name*>',[<*swap_size*>,] '<*source_code*>' is used to define control algorithms, and global variables.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32 | GLOBALS	none
swap_size	numeric (uint16)	0 - Max Available Algorithm Memory	words
source_code	string or block data see Comments	algorithm source	none

Comments

- The <*alg_name*> must be one of ALG1, ALG2, ALG3 etc. through ALG32 or GLOBALS. The parameter is not case sensitive. 'ALG1' and 'alg1' are equivalent as are 'GLOBALS' and 'globals'.
- The <*swap_size*> parameter is optional. Include this parameter with the first definition of <*alg_name*> when you will want to change <*alg_name*> later while it is running. The value can range up to about 23Kwords (ALG:DEF will then allocate 46K words as it creates two spaces for this algorithm).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- If included, <*swap_size*> specifies the number of words of memory to allocate for the algorithm specified by <*alg_name*>. The HP E1419 will then allocate this much memory again, as an update buffer for this algorithm. Note that this doubles the amount of memory space requested. Think of this as "space1" and "space2" for algorithm <*alg_name*>. When you later send a replacement algorithm (must be sent without the <*swap_size*> parameter), it will be placed in "space2". You must send an ALG:UPDATE command for execution to switch from the original, to the replacement algorithm. If you again change the algorithm for <*alg_name*>, it will be executed from "space1" and so on. Note that <*swap_size*> must be large enough to contain the original executable code derived from <*source_code*> and any subsequent replacement for it or an error 3085 "Algorithm too big" will be generated.

- If <*swap_size*> is not included, the HP E1419 will allocated just enough memory for algorithm <*alg_name*>. Since there is no swapping buffer allocated, this algorithm cannot be changed until a *RST command is sent to clear all algorithms. See "When Accepted and Usage".

- The <*source_code*> parameter contents can be:

  - When <*alg_name*> is 'ALG1' through 'ALG32', Algorithm Language source code representing a user's algorithm.

    ```
    ALG:DEF  'ALG5','if( First_loop ) O136=0; O136=O136+0.01;'
    ```

  - When <*alg_name*> is 'GLOBALS', Algorithm Language variable declarations. A variable name must not be the same as an already define user function.

    ```
    ALG:DEF  'GLOBALS','static float my_glob_scalar, my_glob_array[24];'
    ```

  The Algorithm Language source code is translated by the HP E1419's driver into an executable form and sent to the module.

- The <*source_code*> parameter can be one of three different SCPI types:

  Quoted String: For short segments (single lines) of code, enclose the code string within single (apostrophes), or double quotes. Because of string length limitations within SCPI and some programming platforms, we recommend that the quoted string length not exceed a single program line. Example:

  ```
  ALG:DEF 'ALG1','static float outval = 0 ; O132 = outval; outval = outval + 1;'
  ```

  Definite Length Block Program Data: For longer code segments (like complete custom algorithms) this parameter works well because it specifies the exact length of the data block that will be transferred. The syntax for this parameter type is: #<non-zero digit><digit(s)><data byte(s)>

  Where the value of <non-zero digit> is 1-9 and represents the number of <digit(s)>. The value of <digit(s)> taken as a decimal integer indicates the number of <data byte(s)> in the block. Example from "Quoted String" above:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
ALG:DEF  'ALG1',#211O132=I100;∅     (where "∅" is a null byte, required for C-SCPI only)
```

NOTE for HP C-SCPI

For Block Program Data, the Algorithm Parser requires that the *source_code* data end with a null (0) byte. You must append the null byte to the end of the block's <data byte(s)>, and account for it in the byte count <digit(s)> from above. If the null byte is not included, or <digit(s)> doesn't include it, the error "Algorithm Block must contain termination '\0'" will be generated.

Indefinite Length Block Program Data: This form terminates the data transfer when it received an End Identifier with the last data byte. Use this form only when you are sure your controller platform will include the End Identifier. If it is not included, the ALG:DEF command will "swallow" whatever data follows the algorithm code. The syntax for this parameter type is:

#0<data byte(s)><null byte with End Identifier>

Example from "Quoted String" above:

```
ALG:DEF  'ALG1',#0O132=I100;∅     (where "∅" is a null byte, required for C-SCPI only)
```

NOTE for HP C-SCPI

For Block Program Data, the Algorithm Parser requires that the *source_code* data end with a null (0) byte. You must append the null byte to the end of the block's <data byte(s)>. The null byte is sent with the End Identifier. If the null byte is not included, the error "Algorithm Block must contain termination '\0'" will be generated.

When accepted and Usage

1. If <*alg_name*> is not enabled to swap (not originally defined with the <*swap_size*> parameter included) then both of the following conditions must be true:

   a. Module is in Trigger Idle State (after *RST, or ABORT, and before INIT).

OK

```
*RST
ALG:DEF 'GLOBALS','static float My_global;'
ALG:DEF 'ALG3','My_global = My_global + 1;'
```

Error

```
INIT
ALG:DEF 'ALG5','static float a_out; O136=a_out;'
    "Can't define new algorithm while running"
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

b. The <*alg_name*> has not already been defined since a *RST command. Here <*alg_name*> specifines either an algorithm name or 'GLOBALS'.

OK

```
*RST
ALG:DEF 'GLOBALS','static float My_global;'
```

Error

```
*RST
ALG:DEF 'GLOBALS','static float My_global;'
    "No error"
ALG:DEF 'GLOBALS','static float A_different_global'
    "Algorithm already defined"      Because 'GLOBALS' already defined
```

Error

```
*RST
ALG:DEF 'ALG3','static float z;if(First_loop) z = 0; z = z + 1;'
    "No error"
ALG:DEF 'ALG3','static float Cntr, Inc; O132 = Cntr; Cntr = Cntr + Inc;'
    "Algorithm already defined"      Because 'ALG3' already defined
```

2. If <*alg_name*> has been enabled to swap (originally defined with the <*swap_size*> parameter included) then the <*alg_name*> can be re-defined (do not include <*swap_size*> now) either while the module is in the Trigger Idle State, or while in Waiting For Trigger State (INITed). Here <*alg_name*> is an algorithm name only, not 'GLOBALS'.

OK

```
*RST
ALG:DEF 'ALG3',200,'if(O132<15.0) O132=O132 + 0.1; else O132 = -15.0;'
INIT                                starts algorithm
ALG:DEF 'ALG3','if(O132<12.0) O132=O132 + 0.2; else O132 = -12.0;'
ALG:UPDATE                          Required to cause new code to run
    "No error"
```

Error

```
*RST
ALG:DEF 'ALG3',200,'if(O132<15.0) O132=O132 + 0.1; else O132 = -15.0;'
INIT                                starts algorithm
ALG:DEF 'ALG3',200,'if(O132<12.0) O132=O132 + 0.2; else O132 = -12.0;'
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

"Algorithm swapping already enabled; Can't change size"
Because <swap_size> included at re-definition

NOTES

1. Channels referenced by algorithms when they are defined, are only placed in the channel list before INIT. The list cannot be changed after INIT. If you re-define an algorithm (by swapping) after INIT, and it references channels not already in the channel list, it will not be able to access the newly referenced channels. No error message will be generated. To make sure all required channels will be included in the channel list, define <*alg_name*> and re-define all algorithms that will replace <*alg_name*> by swapping them before you send INIT. This insures that all channels referenced in these algorithms will be available after INIT.

2. If you re-define an algorithm (by swapping) after INIT, and it declares an existing variable, the declaration-initialization statement (e.g. `static float my_var = 3.5`) will not change the current value of that variable.

3. The driver only calculates overall execution time for algorithms defined before INIT. This calculation is used to set the default output delay (same as executing ALG:OUTP:DELAY AUTO). If an algorithm is swapped after INIT that take longer to execute than the original, the output delay will behave as if set by ALG:OUTP:DEL 0, rather than AUTO (see ALG:OUTP:DEL command). Use the same procedure from note 1 to make sure the longest algorithm execution time is used to set ALG:OUTP:DEL AUTO before INIT.

ALGorithm[:EXPLicit]:SCALar

ALGorithm[:EXPLicit]:SCALar <*alg_name*>,<*var_name*>,<*value*> sets the value of the scalar variable <*var_name*> for algorithm <*alg_name*> into the Update Queue. This update is then pending until ALG:UPD is sent or an update event (as set by ALG:UPD:CHANNEL) occurs.

NOTE ALG:SCALAR places a variable update request in the Update Queue. You can not place more update requests in the Update Queue than are allowed by the current setting of ALG:UPD:WINDOW or a "Too many updates -- send ALG:UPDATE command" error message will be generated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32 or GLOBALS	none
var_name	string	valid 'C' variable name	none
value	numeric (float32)	IEEE-754 32-bit floating point number	none

Comments

- To send values to a global scalar variable, set the <*alg_name*> parameter to 'GLOBALS'. To define a scalar global variable see the ALGorithm:DEFine command.
- An error is generated if <*alg_name*> or <*var_name*> is not defined.
- **Related Commands:** ALG:DEFINE, ALG:SCAL?
- ***RST Condition:** No algorithms or variables are defined.

Usage

```
ALG:SCAL  'ALG1','my_var',1.2345          1.2345 to variable my_var in ALG1
ALG:SCAL  'ALG1','another',5.4321         5.4321 to variable another also in ALG1
ALG:SCAL  'ALG3','my_global_var',1.001    1.001 to global variable
ALG:UPD                                   update variables from update queue
```

ALGorithm[:EXPLicit]:SCALar?

ALGorithm[:EXPLicit]:SCALar? <*alg_name*>,<*var_name*> returns the value of the scalar variable <*var_name*> in algorithm <*alg_name*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32	none
var_name	string	valid 'C' variable name	none

Comments

- An error is generated if <*alg_name*> or <*var_name*> is not defined.
- **Returned Value:** numeric value. The type is **float32**.

ALGorithm[:EXPLicit]:SCAN:RATio

ALGorithm[:EXPLicit]:SCAN:RATio <*alg_name*>,<*num_trigs*> specifies the number of scan triggers that must occur for each execution of algorithm <*alg_name*>. This allows you to execute the specified algorithm less often than other algorithms. This can be useful for algorithm tuning.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTES

1. The command ALG:SCAN:RATio <*alg_name*>,<*num_trigs*> does not take effect until an ALG:UPDATE, or ALG:UPD:CHAN command is received. This allows you to send multiple ALG:SCAN:RATIO commands and then synchronize their effect with ALG:UPDATE.
2. ALG:SCAN:RATio places a variable update request in the Update Queue. You can not place more update requests in the Update Queue than are allowed by the current setting of ALG:UPD:WINDOW or a "Too many updates -- send ALG:UPDATE command" error message will be generated.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32	none
num_trigs	numeric (int16)	1 to 32,767	none

Comments Specifying a value of 1 (the default) causes the named algorithm to be executed each time a trigger is received. Specifying a value of n will cause the algorithm to be executed once every n triggers. All enabled algorithms execute on the first trigger after INIT.

- The algorithm specified by <*alg_name*> may or may not be currently defined. The specified setting will be used when the algorithm is defined.
- **Related Commands:** ALG:UPDATE, ALG:SCAN:RATIO?
- **When Accepted: Both before and after INIT**. Also accepted before and after the algorithm referenced is defined.
- ***RST Condition:** ALG:SCAN:RATIO = 1 for all algorithms

Usage

```
ALG:SCAN:RATIO 'ALG4',16          ALG4 executes once every 16 triggers.
```

ALGorithm[:EXPLicit]:SCAN:RATio?

ALGorithm[:EXPLicit]:SCAN:RATio? <*alg_name*> returns the number of triggers that must occur for each execution of <*alg_name*>.

Comments

- Since ALG:SCAN:RATIO is valid for an undefined algorithm, ALG:SCAN:RATIO? will return the current ratio setting for <*alg_name*> even if it is not currently defined.
- **Returned Value:** numeric, 1 to 32,768. The type is **int16**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ALGorithm[:EXPLicit]:SIZe?

ALGorithm[:EXPLicit]:SIZe? ***<alg_name>*** returns the number of words of memory allocated for algorithm <*alg_name*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32	none

Comments

- Since the returned value is the memory allocated to the algorithm, it will only equal the actual size of the algorithm if it was defined by ALG:DEF without its [<*swap_size*>] parameter. If enabled for swapping (if <*swap_size*> included at original definition), the returned value will be equal to (<*swap_size*>)*2.

NOTE If <*alg_name*> specifies an undefined algorithm, ALG:SIZ? returns 0. This can be used to determine whether algorithm <*alg_name*> is defined.

- **Returned Value:** numeric value up to the maximum available algorithm memory (this approximately 46K words). The type is **int32**.
- ***RST Condition:** returned value is 0.

ALGorithm[:EXPLicit][:STATe]

ALGorithm[:EXPLicit][:STATe] ***<alg_name>,<enable>*** specifies that algorithm <*alg_name*> , when defined, should be executed (ON), or not executed (OFF) during run-time.

NOTES

1. The command ALG:STATE <*alg_name*>, ON | OFF does <u>not</u> take effect until an ALG:UPDATE, or ALG:UPD:CHAN command is received. This allows you to send multiple ALG:STATE commands and then synchronize their effect.
2. ALG:STATE places a variable update request in the Update Queue. You can not place more update requests in the Update Queue than are allowed by the current setting of ALG:UPD:WINDOW or a "Too many updates -- send ALG:UPDATE command" error message will be generated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CAUTION! When ALG:STATE OFF disables an algorithm, any outputs are left at the last value set by the algorithm. Depending on the process, this uncontrolled situation could even be dangerous. Make certain that you have put your process into a safe state before you halt execution of a controlling algorithm.

The HP E1535 Watchdog Timer SCP was specifically developed to automatically signal that an algorithm has stopped controlling a process. Use of the Watchdog Timer is recommended for critical processes.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32	none
enable	boolean (uint16)	0 | 1 | ON | OFF	none

Comments

- The algorithm specified by <*alg_name*> may or may not be currently defined. The setting specified will be used when the algorithm is defined.
- ***RST Condition:** ALG:STATE ON
- **When Accepted: Both before and after INIT**. Also accepted before and after the algorithm referenced is defined.
- **Related Commands:** ALG:UPDATE, ALG:STATE?, ALG:DEFINE

Usage

```
ALG:STATE 'ALG2',OFF                    disable ALG2
```

ALGorithm[:EXPLicit][:STATe]?

ALGorithm[:EXPLicit][:STATe]? <*alg_name*> returns the state (enabled or disabled) of algorithm <*alg_name*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32	none

Comments

- Since ALG:STATE is valid for an undefined algorithm, ALG:STATE? will return the current state for <*alg_name*> even if it is not currently defined.
- **Returned Value:** Numeric, 0 or 1. Type is **uint16**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- ***RST Condition:** ALG:STATE 1

ALGorithm[:EXPLicit]:TIME?

ALGorithm[:EXPLicit]:TIME? ***<alg_name>*** computes and returns a worst-case execution time estimate in seconds.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
alg_name	string	ALG1 - ALG32 or MAIN	none

Comments

- When <*alg_name*> is ALG1 through ALG32, ALG:TIME? returns only the time required to execute the algorithm's code.
- When <*alg_name*> is 'MAIN', ALG:TIME? returns the worst-case execution time for an entire measurement & control cycle (sum of MAIN, all enabled algorithms, analog and digital inputs, and control outputs).
- If triggered more rapidly than the value returned by ALG:TIME? 'MAIN', the HP E1419 will generate a "Trigger too fast" error.

NOTE If <*alg_name*> specifies an undefined algorithm, ALG:TIME? returns 0. This can be used to determine whether algorithm <*alg_name*> is defined.

- **When Accepted: Before INIT only**.
- **Returned Value:** numeric value. The type is **float32**

ALGorithm:FUNCtion:DEFine

ALGorithm:FUNCtion:DEFine ***<function_name>,<range>,<offset>, <func_data>*** defines a custom function that can be called from within a custom algorithm. See Appendix E "Generating User Defined Functions" for full information. Also see the HP VEE example program "fn_1419.vee" on page 5-160

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
function_name	string	valid 'C' identifier (if not already defined in 'GLOBALS')	none
range	numeric (float32)	see comments	none
offset	numeric (float32)	see comments	none
func_data	512 element array of uint16	see comments	none

Comments

- By providing this custom function capability, the HP E1419's algorithm language can be kept simple in terms of mathematical capability. This increases speed. Rather than having to calculate high-order polynomial approximations of non-linear functions, this custom function scheme loads a pre-computed look-up table of values into memory. This method allows computing virtually any transcendental or non-linear function in about 18µseconds. Resolution is 16 bits.

- <*function_name*> is a global identifier and cannot be the same as a previously define global variable. A user function is globally available to all defined algorithms.

- You generate values for <*range*>, <*offset*>, and <*func_data*> with the HP VEE program "fn_1419.vee" supplied with your HP E1419. See Appendix E "Generating User Defined Functions" for background information.

- <*range*>, and <*offset*> define the allowable input values to the function (domain). If values input to the function are equal to or outside of (±<*range*>+<*offset*>), the function may return ±INF in IEEE-754 format. For example; <*range*> = 8 (-8 to 8), <*offset*> = 12. The allowable input values must be greater than 4 and less than 20.

- <*func_data*> is a 512 element array of type uint16.

- The algorithm syntax for calling is: <*function_name*> **(** <*expression*> **)**. for example:

  ```
  O136 = squareroot( 2 * Input_val );
  ```

- Functions must be defined before defining algorithms that reference them.

- **When Accepted: Before INIT only**.

Usage

```
ALG:FUNC:DEF 'F1',8,12,<block_data>
```

send range, offset and table values for function F1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ALGorithm:OUTPut:DELay

ALGorithm:OUTPut:DELay <*delay*> sets the delay from Scan Trigger to start of output phase.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
delay	numeric (float32)	0 - .081 | AUTO (2.5µs resolution)	seconds

Comments

- The algorithm output statements (e.g. O136 = Out_val) DO NOT program outputs when they are executed. Instead, these statements write to an intermediate Output Channel Buffer which is read and used for output AFTER all algorithms have executed AND the algorithm output delay has expired (see Figure 6-1). Also note that not all outputs will occur at the same time but will take approximately 10usec per channel to write.

- When <*delay*> is 0, the Output phase begins immediately after the Execute Algorithms phase. This provides the fastest possible execution speed while potentially introducing variations in the time between trigger and beginning of the Output phase. The variation can be caused by conditional execution constructs in algorithms, or other execution time variations.

- If you set <*delay*> to less time than is required for the Input + Update + Execute Algorithms phases, ALG:OUTP:DELAY? will report the time you set, but the effect will revert to the same that is set by ALG:OUTP:DELAY 0 (Output begins immediately after Execute phase).

- When <*delay*> is AUTO, the delay is set to the worst-case time required to execute phases 1 through 3. This provides the fastest execution speed while maintaining a fixed time between trigger and the OUTPUT phase.

- When you want to set the time from trigger to the beginning of OUTPUT, use the following procedure. After defining all of your algorithms, execute:

ALG:OUTP:DEL AUTO	*sets minimum stable delay*
ALG:OUTP:DEL?	*returns this minimum delay*
ALG:OUTP:DEL <*minimum+additional*>	*additional* = **desired** - *minimum*

  Note that the delay value returned by ALG:OUTP:DEL? is valid only until another algorithm is loaded. After that, you would have to re-issue the ALG:OUTP:DEL AUTO and ALG:OUTP:DEL? commands to determine the new delay that includes the added algorithm.

- **When Accepted:** Before INIT only.

- ***RST Condition:** ALG:OUTP:DELAY AUTO

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ALGorithm:OUTPut:DELay?

ALGorithm:OUTPut:DELay? returns the delay setting from ALG:OUTP:DEL.

Comments

- The value returned will be either the value set by ALG:OUTP:DEL <*delay*>, or the value determined by ALG:OUTP:DEL AUTO.
- **When Accepted:** Before INIT only.
- ***RST Condition:** ALG:OUTP:DEL AUTO, returns delay setting determined by AUTO mode.
- **Returned Value:** number of seconds of delay. The type is **float32**.

ALGorithm:UPDate[:IMMediate]

ALGorithm:UPDate[:IMMediate] requests an immediate update of any scalar, array, algorithm code, ALG:STATE, or ALG:SCAN:RATIO changes that are pending.

Comments

- Variables and algorithms can be accepted during Phase 1-INPUT or Phase 2-UPDATE in Figure 6-1 when INIT is active. All writes to variables and algorithms occur to their buffered elements upon receipt. However, these changes do not take effect until the ALG:UPD:IMM command is processed at the beginning of the UPDATE phase. The update command can be received at any time prior to the UPDATE phase and will be the last command accepted. Note that the ALG:UPD:WINDow command specifies the maximum number of updates to do. If no update command is pending when entering the UPDATE phase, then this time is dedicated to receiving more changes from the system.
- As soon as the ALG:UPD:IMM command is received, no further changes are accepted until all updates are complete. A query of an algorithm value following an UPDate command will not be executed until the UPDate completes; this may be a useful synchronizing method.

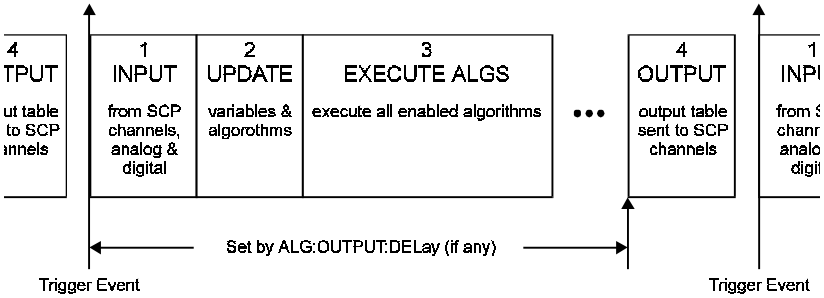


Figure 6-1 Updating Variables and Algorithms

- **When Accepted: Before or after INIT.**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Related Commands: ALG:UPDATE:WINDOW, ALG:SCALAR, ALG:ARRAY, ALG:STATE, and ALG:SCAN:RATIO, ALG:DEF (with swapping enabled)

Command Sequence

The following example shows three scalars being written with the associated update command following. See ALG:UPD:WINDOW.

```
ALG:SCAL ALG1','Setpoint',25
ALG:SCAL 'ALG1','P_factor',1.3
ALG:SCAL 'ALG2','P_factor',1.7
ALG:UPD
ALG:SCAL? 'ALG2','Setpoint'
```

ALGorithm:UPDate:CHANnel

ALGorithm:UPDate:CHANnel <*dig_chan*> This command is used to update variables, algorithms, ALG:SCAN:RATIO, and ALG:STATE changes when the specified digital input level changes state. When the ALG:UPD:CHAN command is executed, the current state of the digital input specified is saved. The update will be performed at the next update phase (UPDATE in Figure 6-1), following the channel's change of digital state. This command is useful to synchronize multiple HP E1419s when you want all variable updates to be processed at the same time.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
dig_chan	Algorithm Language channel specifier (string)	Input channel for HP E1533: Iccc.Bb for HP E1534: Iccc where ccc=normal channel number and b=bit number (include ".B")	none

Comments

- The duration of the level change to the designated bit or channel MUST be at least the length of time between scan triggers. Variable and algorithm changes can be accepted during the INPUT or UPDATE phases (Figure 6-1) when INIT is active. All writes to variables and algorithms occur to their buffered elements upon receipt. However, these changes do not take effect until the ALG:UPD:CHAN command is processed at the beginning of the UPDATE phase. Note that the ALG:UPD:WINDow command specifies the maximum number of updates to do. If no update command is pending when entering the UPDATE phase, then this time is dedicated to receiving more changes from the system.

NOTE As soon as the ALG:UPD:CHAN command is received, the HP E1419 begins to closely monitor the state of the update channel and can not execute other commands until the update channel changes state to complete the update

---

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Note that an update command issued after the start of the UPDATE phase will be buffered but not executed until the beginning of the next INPUT phase. At that time, the current stored state of the specified digital channel is saved and used as the basis for comparison for state change. If at the beginning of the scan trigger the digital input state had changed, then at the beginning of the UPDATE phase the update command would detect a change from the previous scan trigger and the update process would begin.

- **When Accepted: Before and After INIT**.

Command Sequence

The following example shows three scalars being written with the associated update command following. When the ALG:UPD:CHAN command is received, it will read the current state of channel 136, bit 0. At the beginning of the UPDATE phase, a check will be made to determine if the stored state of channel 136 bit 0, is different from the current state. If so, the update of all three scalars take effect next Phase 2.

```
INIT
ALG:SCAL 'ALG1','Outplimit',25
ALG:SCAL 'ALG1','Alarmtrip',1.3
ALG:SCAL 'ALG2','Alarmtrip',1.7
ALG:UPD:CHAN 'I136.B0'          update on state change at bit zero of 8-bit channel 36
```

ALGorithm:UPDate:WINDow

ALGorithm:UPDate:WINDow <*num_updates*> specifies how many updates you may need to perform during phase 2 (UPDATE). The DSP will process this command and assign a constant window of time for UPDATE.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
num_updates	numeric (int16)	1 - 512	none

Comments

- The default value for num_updates is 20. If you know you will need fewer updates, specifying a smaller number will result in slightly greater loop execution speeds.

- This command creates a time interval in which to perform all pending algorithm and variable updates. To keep the loop times predictable and stable, the time interval for UPDATE is constant. That is, it exists for all active algorithms, each time they are executed whether or not an update is pending.

- ***RST Condition:** ALG:UPD:WIND 20

- **When Accepted: Before INIT only**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Usage You decide you will need to update a maximum of 8 variables per execution of ALG:UPDATE.

```
ALG:UPD:WIND 8
```

NOTES

1. When the number of update requests exceeds the Update Queue size set with ALG:UPD:WINDOW by one, the module will refuse the request and will issue the error message "Too many updates in queue. Must send UPDATE command". Send ALG:UPDATE, then re-send the update request that caused the error.

2. The "Too many updates in queue..." error can occur before the module is INITialized. It's not uncommon with several algorithms defined, to have more variables that need to be pre-set before INIT than you will change in one update after the algorithms are running. You may send INIT with updates pending. The INIT command automatically performs the updates before starting the algorithms.

ALGOrithm:UPDate:WINDow?

ALGOrithm:UPDate:WINDow? returns the number of variable, and algorithm updates allowed within the UPDATE window.

- **Returned Value:** number of updates in the UPDATEwindow. The type is **int16**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

With the HP E1419, when the TRIG:SOURCE is set to TIMer, an ARM event must occur to start the timer. This can be something as simple as executing the ARM[:IMMediate] command, or it could be another event selected by ARM:SOURCE.

NOTE The ARM subsystem may only be used then the TRIGger:SOURce is TIMer. If the TRIGger:SOURce is not TIMer and ARM:SOURce is set to anything other than IMMediate, an Error -221,"Settings conflict" will be generated.

The ARM command subsystem provides:

- An immediate software ARM (ARM:IMM).
- Selection of the ARM source (ARM:SOUR BUS | EXT | HOLD | IMM | SCP | TTLTRG<n>) when TRIG:SOUR is TIMer.

Figure 6-7 shows the overall logical model of the Trigger System.

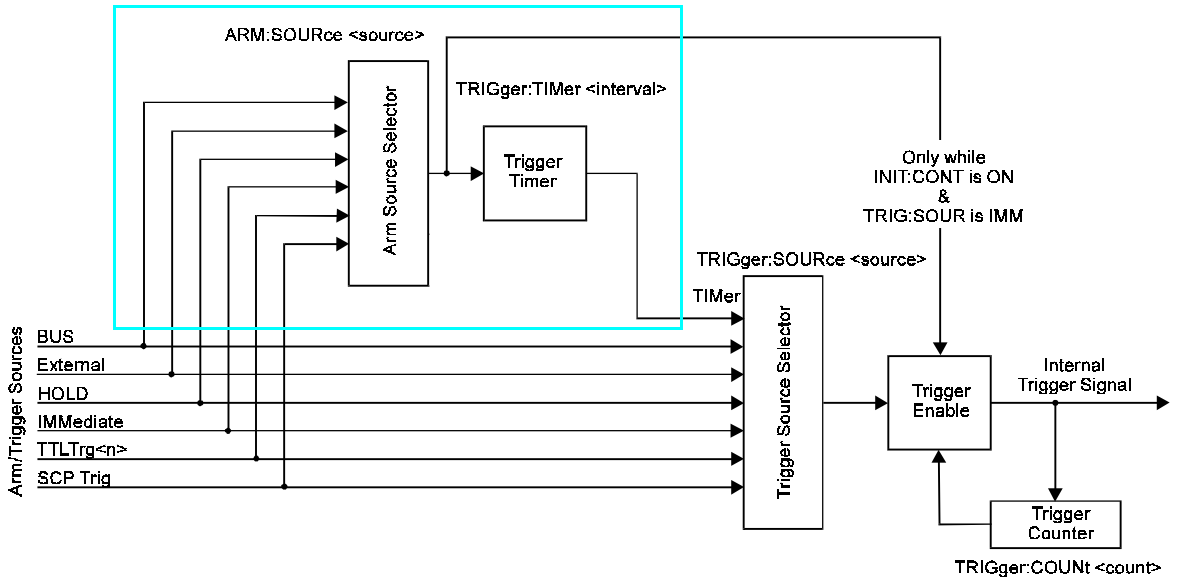


Figure 6-2 Logical Trigger Model

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Subsystem Syntax

```
ARM
    [:IMMediate]
    :SOURce BUS | EXTernal | HOLD | IMMediate | SCP | TTLTrg<n>
    :SOURce?
```

ARM[:IMMediate]

ARM[:IMMediate] arms the trigger system when the module is set to the ARM:SOUR BUS or ARM:SOUR HOLD mode.

Comments

- **Related Commands:** ARM:SOURCE, TRIG:SOUR
- ***RST Condition:** ARM:SOUR IMM

Usage

ARM:IMM	*After INIT, system is ready for trigger event*
ARM	*Same as above (:IMM is optional)*

ARM:SOURce

ARM:SOURce ***<arm_source>*** configures the ARM system to respond to the specified source.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
arm_source	discrete (string)	BUS | EXT | HOLD | IMM | SCP | TTLTrg<n>	none

Comments

- The following table explains the possible choices.

Parameter Value	Source of Arm
BUS	ARM[:IMMediate]
EXTernal	"TRG" signal on terminal module
HOLD	ARM[:IMMediate]
IMMediate	The arm signal is always true (continuous arming).
SCP	SCP Trigger Bus (future HP or SCP Breadboard)
TTLTrg<n>	The VXIbus TTLTRG lines (n=0 through 7)

- See note about ARM subsystem on page 6-208.
- When TRIG:SOURCE is TIMER, an ARM event is required only to trigger the first scan. After that the timer continues to run and the module goes to the Waiting For Trigger State ready for the next Timer trigger. An ABORT command will return the module to the Trigger Idle State after the current scan is completed. See TRIG:SOURce for more detail.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

While ARM:SOUR is IMM, you need only INITiate the trigger system to start a measurement scan.

- **When Accepted: Before INIT only.**
- **Related Commands:** ARM:IMM, ARM:SOURCE?, INIT[:IMM], TRIG:SOUR
- ***RST Condition:** ARM:SOUR IMM

Usage

```
ARM:SOUR   BUS          Arm with ARM command
ARM:SOUR TTLTRG3        Arm with VXIbus TTLTRG3 line
```

ARM:SOURce?

ARM:SOURce? returns the current arm source configuration. See the ARM:SOUR command for more response data information.

- **Returned Value:** Discrete, one of BUS, HOLD, IMM, SCP, or TTLT0 through TTLT7. The C-SCPI type is **string**.

Usage

```
ARM:SOUR?               An enter statement return arm source
                        configuration
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Calibration subsystem provides for two major categories of calibration.

1. "A/D Calibration"; In these procedures, an external multimeter is used to calibrate the A/D gain on all 5 of its ranges. The multimeter also determines the value of the HP E1419's internal calibration resistor. The values generated from this calibration are then stored in nonvolatile memory and become the basis for "Working Calibrations. These procedures each require a sequence of several commands from the CALibration subsystem (**CAL:CONFIG**…, **CAL:VALUE**…, and **CAL:STORE ADC)**. Always execute *CAL? or a CAL:TARE operation after A/D Calibration.

2. "Working Calibration", of which there are three levels (see Figure 6-3):

   – "A/D Zero"; This function quickly compensates for any short term A/D converter offset drift. This would be called the auto-zero function in a conventional voltmeter. In the HP E1419 where channel scanning speed is of primary importance, this function is performed only when the **CAL:ZERO?** command is executed. Execute CAL:ZERO? as often as your control setup will allow.

   – "Channel Calibration"; This function corrects for offset and gain errors for each module channel. The internal current sources are also calibrated. This calibration function corrects for thermal offsets and component drift for each channel out to the input side of the Signal Conditioning Plug-On (SCP). All calibration sources are on-board and this function is invoked using either the ***CAL?** or **CAL:SETup** command.

   – "Channel Tare"; This function (**CAL:TARE**) corrects for voltage offsets in external system wiring. Here, the user places a short across transducer wiring and the voltage that the module measures is now considered the new "zero" value for that channel. The new offset value can be stored in non-volatile calibration memory (**CAL:STORE TARE**) but is in effect whether stored or not. System offset constants which are considered long-term should be stored. Offset constants which are measured relatively often would not require non-volatile storage. CAL:TARE automatically executes a *CAL?.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

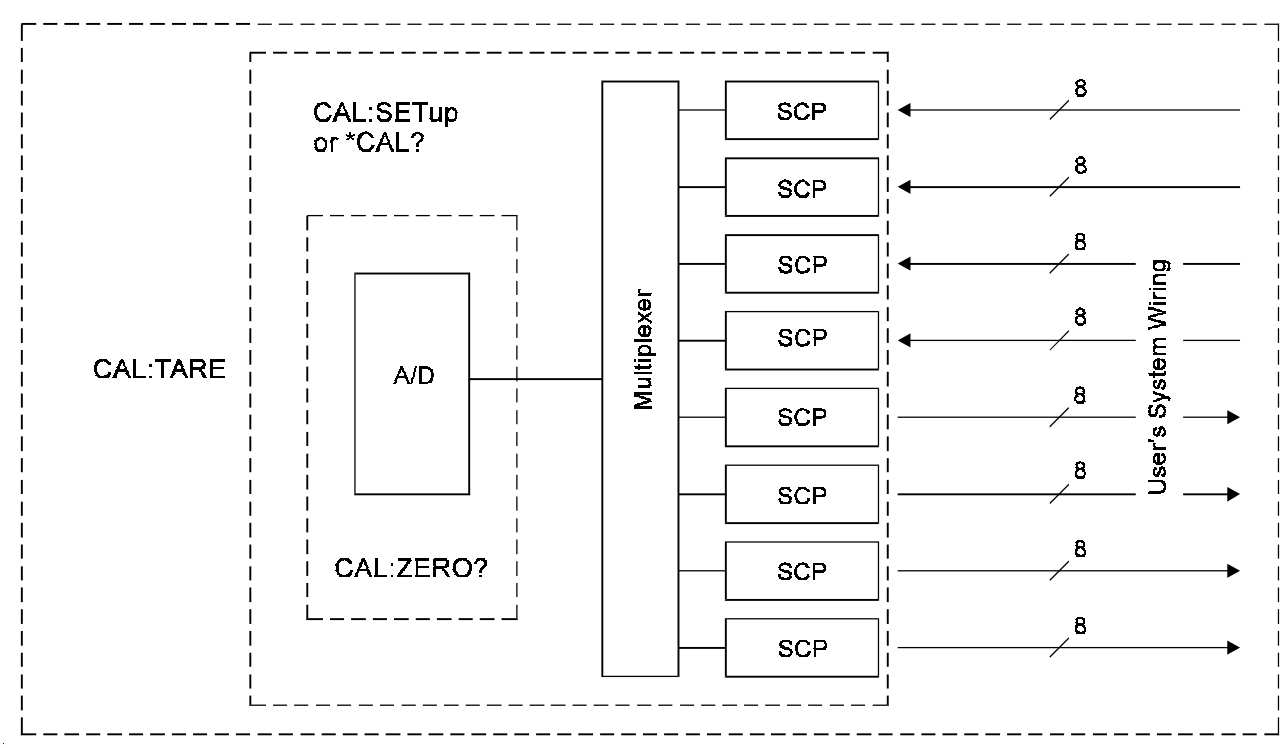


Figure 6-3 Levels of Working Calibration

Subsystem Syntax

```
CALibration
    :CONFigure
        :RESistance
        :VOLTage  <range>, ZERO | FS
    :SETup
    :SETup?
    :STORe ADC | TARE
    :TARE (@<ch_list>)
        :RESet
    :TARE?
    :VALue
        :RESistance  <ref_ohms>
        :VOLTage  <ref_volts>
    :ZERO?
```

CALibration:CONFigure:RESistance

CALibration:CONFigure:RESistance connects the on-board calibration reference resistor to the Calibration Bus. A four-wire measurement of the resistor can be made with an external "calibration DVM" connected to the **H Cal**, **L Cal**, **H ohm**, and **L**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ohm terminals on the Terminal Module, or the **V H**, **V L**, **Ω H**, and **Ω L** terminals on the Cal Bus connector.

Comments

- **Related Commands:** CAL:VAL:RES, CAL:STOR ADC
- **When Accepted:** Not while INITiated

Command Sequence

CAL:CONF:RES	*connect reference resistor to Calibration Bus*
*OPC? or SYST:ERR?	*must wait for CAL:CONF:RES to complete*
(*now measure ref resistor with external DMM*)	
CAL:VAL:RES <measured value>	*Send measured value to module*
CAL:STORE ADC	*Store cal constants in non-volatile memory (used only at end of complete cal sequence)*

CALibration:CONFigure:VOLTage

CALibration:CONFigure:VOLTage <***range***>,<***zero_fs***> connects the on-board calibration source to the Calibration Bus. A measurement of the calibration source voltage can be made with an external "calibration DVM" connected to the **H Cal** and **L Cal** terminals on the Terminal Module, or the **V H** and **V L** terminals on the Cal Bus connector. The *range* parameter controls the source voltage level available when the *zero_fs* parameter is set to FSCale (full scale).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
range	numeric (float32)	see comments	volts
zero_fs	discrete (string)	ZERO | FSCale	none

Comments

- The *range* parameter must be within ±5% of one of the 5 following values: .0625VDC, .25VDC, 1VDC, 4VDC, 16VDC
  range may be specified in millivolts (mv).
- The FSCALE output voltage of the calibration source will be approximately 90% of the nominal value for each range, except the 16V range where the output is approximately 10V.
- **When Accepted:** Not while INITiated
- **Related Commands:** CAL:VAL:VOLT, STOR ADC

Command Sequence

CAL:CONF:VOLTAGE .0625, ZERO	*connect voltage reference to Calibration Bus*
*OPC? or SYST:ERR?	*must wait for CAL:CONF:VOLT to complete*
(*now measure voltage with external DMM*)	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
CAL:VAL:VOLT  <measured value>      Send measured value to module
repeat above sequence for full-scale
repeat zero and full-scale for remaining ranges (.25, 1, 4, 16)
CAL:STORE ADC                       Store cal constants in non-volatile memory
                                    (used only at end of complete cal sequence)
```

CALibration:SETup

CALibration:SETup causes the Channel Calibration function to be performed for every module channelwith an analog SCP installed (input or output). The Channel Calibration function calibrates the A/D Offset, and the Gain/Offset for these analog channels. This calibration is accomplished using internal calibration references. For more information see *CAL? on page 6-316.

Comments

- CAL:SET performs the same operation as the *CAL? command except that since it is not a query command it doesn't tie-up the C-SCPI driver waiting for response data from the instrument. If you have multiple HP E1419s in your system you can start a CAL:SET operation on each and then execute a CAL:SET? command to complete the operation on each instrument.
- **Related Commands:** CAL:SETup?, *CAL?
- **When Accepted:** Not while INITiated

Usage

```
CAL:SET             start SCP Calibration on 1st HP E1419
:                   start SCP Calibration on more HP E1419s
CAL:SET             start SCP Calibration on last HP E1419
CAL:SET?            query for results from 1st HP E1419
:                   query for results from more HP E1419s
CAL:SET?            query for results from last HP E1419
```

CALibration:SETup?

CALibration:SETup? Returns a value to indicate the success of the last CAL:SETup or *CAL? operation. CAL:SETup? returns the value only after the CAL:SETup operation is complete.

Comments

- **Returned Value:**

Value	Meaning	Further Action
0	Cal OK	None
-1	Cal Error	Query the Error Queue (SYST:ERR?) See Error Messages in Appendix B.
-2	No results available	No *CAL? or CAL:SETUP done

The C-SCPI type for this returned value is **int16**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Related Commands:** CAL:SETup, *CAL?

Usage see CAL:SETup

CALibration:STORe

CALibration:STORe **<*type*>** stores the most recently measured calibration constants into Flash Memory (Electrically Erasable Programmable Read Only Memory). When *type*=ADC, the module stores its A/D calibration constants as well as constants generated from *CAL?/CAL:SETup into Flash Memory. When *type*=TARE, the module stores the most recently measured CAL:TARE channel offsets into Flash Memory.

NOTE The HP E1419's Flash Memory has a finite lifetime of approximately ten thousand write cycles (unlimited read cycles). While executing CAL:STOR once every day would not exceed the lifetime of the Flash Memory for approximately 27 years, an application that stored constants many times each day would unnecessarily shorten the Flash Memory's lifetime. See Comments below.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
type	discrete (string)	ADC | TARE	none

Comments

- The Flash Memory Protect jumper (JM2201) must be set to the enable position before executing this command (See Chapter 1).
- Channel offsets are compensated by the CAL:TARE command even when not stored in the Flash Memory. There is no need to use the CAL:STORE TARE command for channels which are re-calibrated frequently.
- **When Accepted:** Not while INITiated
- **Related Commands:** CAL:VAL:RES, CAL:VAL:VOLT
- ***RST Condition:** Stored calibration constants are unchanged

Usage

CAL:STORE ADC	*Store cal constants in non-volatile memory after A/D calibration*
CAL:STORE TARE	*Store channel offsets in non-volatile memory after channel tare*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command Sequence

Storing A/D cal constants

perform complete A/D calibration, then...

CAL:STORE ADC

Storing channel tare (offset) values

CAL:TARE <*ch_list*>	*to correct channel offsets*
CAL:STORE TARE	*Optional depending on necessity of long term storage*

CALibration:TARE

CALibration:TARE (@<*ch_list*>) measures offset (or tare) voltage present on the channels specified and stores the value in on-board RAM as a calibration constant for those channels. Future measurements made with these channels will be compensated by the amount of the tare value. Use CAL:TARE to compensate for voltage offsets in system wiring and residual sensor offsets. Where tare values need to be retained for long periods, they can be stored in the module's Flash Memory (Electrically Erasable Programmable Read Only Memory) by executing the CAL:STORe TARE command.
For more information see Compensating for System Offsets on page 3-100.

Note for Thermocouples

- You must not use CAL:TARE on field wiring that is made up of thermocouple wire. The voltage a thermocouple wire pair generates can not be removed by introducing a short anywhere between its junction and its connection to an isothermal panel (either the HP E1419's Terminal Module or a remote isothermal reference block). Thermal voltage is generated along the entire length of a thermocouple pair where there is any temperature gradient along that length. To CAL:TARE thermocouple wire this way would introduce an unwanted offset in the voltage/temperature relationship for that channel. If you inadvertently CAL:TARE a thermocouple wire pair, use CAL:TARE:RESET to reset all tare constants to zero.
- You should use CAL:TARE to compensate wiring offsets (copper wire, not thermocouple wire) between the HP E1419 and a remote thermocouple reference block. Disconnect the thermocouples and introduce copper shorting wires between each channel's HI and LO, then execute CAL:TARE for these channels.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ch_list	channel list (string)	100 - 163	none

Comments

- CAL:TARE also performs the equivalent of a *CAL? operation. This operation uses the Tare constants to set a DAC which will remove each channel offset as

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

"seen" by the module's A/D converter. As an example assume that the system wiring to channel 0 generates a +0.1Volt offset with 0Volts (a short) applied at the Unit Under Test (UUT). Before CAL:TARE the module would return a reading of 0.1Volts for channel 0. After CAL:TARE (@100), the module will return a reading of 0Volts with a short applied at the UUT and the system wiring offset will be removed from all measurements of the signal to channel 0.

- Set Amplifier/Filter SCP gain before CAL:TARE. For best accuracy, choose the gain that will be used during measurements. If you decide to change the range or gain setup later, be sure to perform another *CAL?.

- Output SCP channels referenced in <*ch_list*> will not be affected by CAL:TARE. Some output have input channels associated with them in order to approximately verify their output values. These input channels will be not be affected by CAL:TARE even if they are referenced in <*ch_list*>.

- If Open TransducerDetect (OTD) is enabled when CAL:TARE is executed, the module will disable OTD, wait 1 minute to allow channels to settle, perform the calibration, and then re-enable OTD. If you wish the OTD current to remain on while CAL:TARE executes, you must use the DIAG:CAL:TARE:OTD:MODE:STATE command to set this configuration.

- The maximum voltage that CAL:TARE can compensate for is dependent on the range chosen and SCP gain setting. The following table lists these values.

Maximum CAL:TARE Offsets

A/D range ±V F.Scale	Offset V Gain x1	Offset V Gain x8	Offset V Gain x16	Offset V Gain x64
16	3.2213	.40104	.20009	.04970
4	.82101	.10101	.05007	.01220
1	.23061	.02721	.01317	.00297
.25	.07581	.00786	.00349	.00055
.0625	.03792	.00312	.00112	n/a

- Channel offsets are compensated by the CAL:TARE command even when not stored in the Flash Memory. There is no need to use the CAL:STORE TARE command for channels which are re-calibrated frequently.

- The HP E1419's Flash Memory has a finite lifetime of approximately ten thousand write cycles (unlimited read cycles). While executing CAL:STOR once every day would not exceed the lifetime of the Flash Memory for approximately 27 years, an application that stored constants many times each day would unnecessarily shorten the Flash Memory's lifetime. See Comments below.

- Executing CAL:TARE sets the Calibrating bit (bit 0) in Operation Status Group. Executing CAL:TARE? resets the bit.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **When Accepted:** Not while INITiated
- **Related Commands:** CAL:TARE?, CAL:STOR TARE
- ***RST Condition:** Channel offsets are not affected by *RST.

Command Sequence

CAL:TARE *<ch_list>*	*to correct channel offsets*
CAL:TARE?	*to return the success flag from the CAL:TARE operation*
CAL:STORE TARE	*Optional depending on necessity of long term storage*

CALibration:TARE:RESet

CALibration:TARE:RESet resets the tare calibration constants to zero for all 64 channels. Executing CAL:TARE:RES affects the tare cal constants in RAM only. To reset the tare cal constants in Flash Memory, execute CAL:TARE:RES and then execute CAL:STORE TARE.

Command Sequence

CAL:TARE:RESET	*to reset channel offsets*
CAL:STORE TARE	*Optional if necessary to reset tare cal constants in Flash Memory.*

CALibration:TARE?

CALibration:TARE? Returns a value to indicate the success of the last CAL:TARE operation. CAL:TARE? returns the value only after the CAL:TARE operation is complete.

- **Returned Value:**

Value	Meaning	Further Action
0	Cal OK	None
-1	Cal Error	Query the Error Queue (SYST:ERR?) See Error Messages in Appendix B. Also run *TST?
-2	No results available	Perform CAL:TARE before CAL:TARE?

The C-SCPI type for this returned value is **int16**.

- Executing CAL:TARE sets the Calibrating bit (bit 0) in Operation Status Group. Executing CAL:TARE? resets the bit.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Related Commands:** CAL:STOR TARE

Command Sequence

CAL:TARE <*ch_list*>	*to correct channel offsets*
CAL:TARE?	*to return the success flag from the CAL:TARE operation*
CAL:STORE TARE	*Optional depending on necessity of long term storage*

CALibration:VALue:RESistance

CALibration:VALue:RESistance <***ref_ohms***> sends the value of the on-board reference resistor -as measured with the external "calibration DVM"- to the module. This value will be used to calibrate current sources.

Parameters

Parameter Name	Parameter Type	Range of Value	Default Units
ref_ohms	numeric (float32)	7,500 ± 4%	ohms

Comments

- Use the CAL:CONF:RES command to configure the reference resistor for measurement at the Calibration Bus connector.
- A four-wire measurement of the resistor is made with an external multimeter connected to the **H Cal**, **L Cal**, **H ohm**, and **L ohm** terminals on the Terminal Module, or the **V H**, **V L**, **Ω H**, and **Ω L** terminals on the Cal Bus connector.
- <*ref_ohms*> must be within 4% of the 7500 Ohm nominal resistor value or a -222 'Data out of range' error will be generated.
- <*ref_ohms*> may be specified in Kohm (kohm).
- **When Accepted:** Not while INITiated
- **Related Commands:** CAL:CONF:RES, CAL:STORE ADC

Command Sequence

CAL:CONF:RES

(*now measure ref resistor with external DMM*)

CAL:VAL:RES <measured value> *Send measured value to module*

CALibration:VALue:VOLTage

CALibration:VALue:VOLTage <***ref_volts***> sends the value of the calibration reference source voltage -as measured by an external "calibration DVM"- to the module for A/D calibration.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ref_volts	numeric (float32)	must be within 4% of range nominal	volts

Comments

- Use the CAL:CONF:VOLT command to configure the on-board voltage source for measurement at the Calibration Bus connector.
- A measurement of the source voltage is made with an external multimeter connected to the **H Cal**, and **L Cal** terminals on the Terminal Module, or the **V H**, and **V L** terminals on the Cal Bus connector.
- The value sent must be for the currently configured range and output (zero or full scale) as set by the previous **CAL:CONF:VOLT <*range*>, ZERO | FSCale** command.
- <*ref_volts*> must be within 4% of the actual reference voltage value as read after CAL:CONF:VOLT, or an error 3042 '0x400: DSP-DAC adjustment went to limit' will be generated.
- <*ref_volts*> may be specified in millivolts (mv).
- **When Accepted:** Not while INITiated
- **Related Commands:** CAL:CONF:VOLT, CAL:STORE ADC

Command Sequence

```
CAL:CONF:VOLTAGE  4,FSCALE
*OPC?                                   Wait for operation to complete
enter statement

(now measure voltage with external DMM)
CAL:VAL:VOLT  <measured value>          Send measured value to module
```

CALibration:ZERO?

CALibration:ZERO? corrects Analog to Digital converter offset for any drift since the last *CAL? or CAL:ZERO? command was executed. The offset calibration takes about 5 seconds and should be done as often as you control set up allows.

Comments

- The CAL:ZERO? command only corrects for A/D offset drift (zero). Use the *CAL? common command to perform on-line calibration of channels as well as A/D offset. *CAL? performs gain and offset correction of the A/D and each channel with an analog SCP installed (both input and output).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Returned Value:**

Value	Meaning	Further Action
0	Cal OK	None
-1	Cal Error	Query the Error Queue (SYST:ERR?) See Error Messages in Appendix B

The C-SCPI type for this returned value is **int16**.

- Executing this command **does not** alter the module's programmed state (function, range etc.).
- **Related Commands:** *CAL?
- ***RST Condition:** A/D offset performed

Usage

```
CAL:ZERO?
enter statement here                    returns 0 or -1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The DIAGnostic subsystem allows you to perform special operations that are not standard in the SCPI language. This includes checking the current revision of the Control Processor's firmware, and that it has been properly loaded into Flash Memory.

Subsystem Syntax

```
DIAGnostic
    :CALibration
        :SETup
            :MODE  0 | 1
            :MODE?
        :TARe
            [:OTD]
                :MODE  0 | 1
                :MODE?
    :CHECksum?
    :CUSTom
        :LINear <table_range>,<table_block>,(@<ch_list>)
        :PIECewise <table_range>,<table_block>,(@<ch_list>)
        :REFerence
            :TEMPerature
    :IEEE  1 | 0
    :IEEE?
    :INTerrupt
        [:LINe]  <intr_line>
        [:LINe]?
    :OTDetect
        [:STATe]  1 | 0 | ON | OFF,(@<ch_list>)
        [:STATe]?  (@<channel>)
    :QUERy
        :SCPREAD? <reg_addr>
    :VERSion?
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DIAGnostic:CALibration:SETup[:MODE]

DIAGnostic:CALibration:SETup[:MODE] <***mode***> sets the type of calibration to use for analog output SCPs like the HP E1531 and HP E1532 when *CAL? or CAL:SET are executed.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
mode	boolean (uint 16)	0 | 1	volts

Comments

- When <*mode*> is set to 1 (the *RST Default) channels are calibrated using the Least Squares Fit method to provide the minimum error overall (over the entire output range). When <*mode*> is 0, channels are calibrated to provide the minimum error at their zero point. See your SCPs User's Manual for its accuracy specifications using each mode.
- **Related Commands:** *CAL?, CAL:SET, DIAG:CAL:SET:MODE?
- ***RST Condition:** DIAG:CAL:SET:MODE 1

Usage

set analog DAC SCP cal mode for best zero accuracy

```
DIAG:CAL:SET:MODE 0        set mode for best zero cal
*CAL?                      start channel calibration
```

DIAGnostic:CALibration:SETup[:MODE]?

DIAGnostic:CALibration:SETup[:MODE]? returns the currently set calibration mode for analog output DAC SCPs.

Comments

- Returns a 1 when channels are calibrated using the Least Squares Fit method to provide the minimum error overall (over the entire output range). Returns a 0 when channels are calibrated to provide the minimum error at their zero point. See your SCPs User's Manual for its accuracy specifications using each mode. The C-SCPI type is **int16**.
- **Related Commands:** DIAG:CAL:SET:MOD, *CAL?, CAL:SET
- ***RST Condition:** DIAG:CAL:SET:MODE 1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DIAGnostic:CALibration:TARE[:OTDetect]:MODE

DIAGnostic:CALibration:TARE[:OTDetect]:MODE ***<mode>*** sets whether Open Transducer Detect current will be turned off or left on (the default mode) during the CAL:TARE operation.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
mode	boolean (uint 16)	0 | 1	volts

Comments

- When <*mode*> is set to 0 (the *RST Default), channels are tare calibrated with their OTD current off. When <*mode*> is 1, channels that have their OTD current on (DIAGnostic:OTDetect ON,(@<*ch_list*>)) are tare calibrated with their OTD current left on.
- By default (*RST) the CALibration:TARE? command will calibrate all channels with the OTD circuitry disabled. This is done for two reasons: first, most users do not leave OTD enabled while taking readings, and second, the CALibration:TARE? operation takes much longer with OTD enabled. However, for users who intend to take readings with OTD enabled, setting DIAG:CAL:TARE:OTD:MODE to 1, will force the CAL:TARE? command to perform calibration with OTD enabled on channels so specified by the user with the DIAG:OTD command.
- **Related Commands:** *CAL?, CAL:SET, DIAG:CAL:SET:MODE?
- ***RST Condition:** DIAG:CAL:TARE:MODE 0

Usage

configure OTD on during CAL:TARE

```
DIAG:CAL:TARE:MODE 1        set mode for OTD to stay on
CAL:TARE?                   start channel tare cal.
```

DIAGnostic:CALibration:TARE[:OTDetect]:MODE?

DIAGnostic:CALibration:TARE[:OTDetect]:MODE? returns the currently set mode for controlling Open Transducer Detect current while performing CAL:TARE? operation.

Comments

- Returns a 0 when OTD current will be turned off during CAL:TARE?. Returns 1 when OTD current will be left on during CAL:TARE? operation. The C-SCPI type is **int16**.
- **Related Commands:** DIAG:CAL:TARE:MOD, DIAG:OTD, CAL:TARE?
- ***RST Condition:** DIAG:CAL:TARE:MODE 0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DIAGnostic:CHECksum?

DIAGnostic:CHECksum? performs a checksum operation on Flash Memory. A returned value of 1 indicates that Flash memory contents are correct. A returned value of 0 indicates that the Flash Memory is corrupted, or has been erased.

Comments

- **Returned Value:** Returns 1 or 0. The C-SCPI type is **int16**.

Usage

```
DIAG:CHEC?                    Checksum Flash Memory, return 1 for OK, 0 for corrupted
```

DIAGnostic:CUSTom:LINear

DIAGnostic:CUSTom:LINear <*table_range*>,<*table_block*>, (@<*ch_list*>) downloads a custom linear Engineering Unit Conversion table (in <*table_block*>) to the HP E1419. Contact your Hewlett-Packard System Engineer for more information on Custom Engineering Unit Conversion for your application.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
table_range	numeric (float32)	.015625 | .03125 | .0625 | .125 | .25 | .5 | 1 | 2 | 4 | 8 | 16 | 32 | 64	volts
table_block	definite length block data	see comments	none
ch_list	channel list (string)	100 - 163	none

Comments

- <*table_block*> is a block of 8 bytes that define 4, 16-bit values. SCPI requires that <*table_block*> include the definite length block data header. C-SCPI adds the header for you.
- <*table_range*> specifies the range of voltage that the table covers (from -<*table_range*> to +<*table_range*>). The value you specify must be within 5% of one of the nominal values from the table above.
- <*ch_list*> specifies which channels may use this custom EU table
- **Related Commands:** [SENSe:]FUNCtion:CUSTom
- ***RST Condition:** All custom EU tables erased

Usage

```
program puts table constants into array table_block
DIAG:CUST:LIN table_block,(@116:123)    send table to HP E1419 for chs 16-23
SENS:FUNC:CUST:LIN 1,1,(@116:123)       link custom EU with chs 16-23
INITiate then TRIGger module
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DIAGnostic:CUSTom:PIECewise

DIAGnostic:CUSTom:PIECewise <*table_range*>,<*table_block*>, (@<*ch_list*>) downloads a custom piece wise Engineering Unit Conversion table (in <*table_block*>) to the HP E1419. Contact your Hewlett-Packard System Engineer for more information on Custom Engineering Unit Conversion for your application.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
table_range	numeric (float32)	.015625 | .03125 | .0625 | .125 | .25 | .5 | 1 | 2 | 4 | 8 | 16 | 32 | 64	volts
table_block	definite length block data	see comments	none
ch_list	channel list (string)	100 - 163	none

Comments

- <*table_block*> is a block of 1,024 bytes that define 512 16-bit values. SCPI requires that <*table_block*> include the definite length block data header. C-SCPI adds the header for you.
- <*table_range*> specifies the range of voltage that the table covers (from -<*table_range*> to +<*table_range*>).
- <*ch_list*> specifies which channels may use this custom EU table.
- **Related Commands:** [SENSe:]FUNCtion:CUSTom
- ***RST Condition:** All custom EU tables erased.

Usage

program puts table constants into array table_block
DIAG:CUST:PIEC table_block,(@124:131) *send table for chs 24-31 to HP E1419*
SENS:FUNC:CUST:PIEC 1,1,(@124:131) *link custom EU with chs 24-31*
INITiate then TRIGger module

DIAGnostic:CUSTom:REFerence:TEMPerature

DIAGnostic:CUSTom:REFerence:TEMPerature extracts the current Reference Temperature Register Contents, converts it to 32-bit floating point format and sends it to the FIFO. This command is used to verify that the reference temperature is as expected after measuring it using a custom reference temperature EU conversion table.

Usage

your program must have EU table values stored in *table_block*
download the new reference EU table
DIAG:CUST:PIECEWISE <*table_range*>,<*table_block*>,(@<*ch_list*>)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

designate channel as reference

SENS:FUNC:CUST:REF <*range*>,(@<*ch_list*>)

set up scan list sequence (ch 0 in this case)

Now run the algorithm that uses the custom reference conversion table

dump reference temp register to FIFO

DIAG:CUST:REF:TEMP

read the diagnostic reference temperature value

SENS:DATA:FIFO?

DIAGnostic:IEEE

DIAGnostic:IEEE <***mode***> enables (1) or disables (0) IEEE-754 NAN (Not A Number) and ±INF value outputs. This command was created for the HP VEE platform.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
mode	boolean (uint 16)	0 | 1	volts

Comments

- When <*mode*> is set to 1, the module can return ±INF and NAN values according to the IEEE-754 standard. When <*mode*> is set to 0, the module returns values as ±9.9E37 for INF and 9.91E37 for NAN.
- **Related Commands:** DIAG:IEEE?
- ***RST Condition:** DIAG:IEEE 1

Usage

Set IEEE mode

DIAG:IEEE 1 *INF values returned in IEEE standard*

DIAGnostic:IEEE?

DIAGnostic:IEEE? returns the currently set IEEE mode.

Comments

- The C-SCPI type is **int16**.
- **Related Commands:** DIAG:IEEE
- ***RST Condition:** DIAG:IEEE 1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DIAGnostic:INTerrupt[:LINe]

DIAGnostic:INTerrupt[:LINe] ***<intr_line>*** sets the VXIbus interrupt line the module will use.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
intr_line	numeric (int16)	0 through 7	none

Comments

- **Related Commands:** DIAG:INT:LINE?
- **Power-on and *RST Condition:** DIAG:INT:LINE 1

Usage

```
DIAG:INT:LINE 5
```

Module will interrupt on VXIbus interrupt line 5

DIAGnostic:INTerrupt[:LINe]?

DIAGnostic:INTerrupt[:LINe]? returns the VXIbus interrupt line that the module is set to use.

Comments

- **Returned Value:** Numeric 0 through 7. The C-SCPI type is **int16**.
- **Related Commands:** DIAG:INT:LINE

Usage

```
DIAG:INT?
```

Enter statement will return 0 through 7

DIAGnostic:OTDetect[:STATe]

DIAGnostic:OTDetect[:STATe] ***<enable>*****,(@<*****ch_list*****>)** enables and disables the HP E1419's "Open Transducer Detection" capability (OTD). When Open Transducer Detection is enabled, a very high impedance path connects all SCP channels to a voltage source greater than 16 volts. If an enabled channel has an open transducer, the input signal becomes the source voltage and the channel returns an input over-range value. The value returned is +9.91E+37 (ASCII).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	boolean (uint16)	1 | 0 | ON | OFF	none
ch_list	channel list (string)	100 - 163	none

Comments

- Open Transducer Detection is enabled/disabled on a whole Signal Conditioning Plug-on basis. Selecting any channel on an SCP selects all channels on that SCP (8 channels per SCP).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- The DIAG:CAL:TARE:MODE <*mode*> command affects how OTD is controlled during the CAL:TARE? operation. When <*mode*> is set to 0 (the *RST Default), channels are tare calibrated with their OTD current off. When <*mode*> is 1, channels that have their OTD current on (DIAGnostic:OTDetect ON,(@<*ch_list*>)) are tare calibrated with their OTD current left on.
- **Related Commands:** DIAG:OTDETECT:STATE?, DIAG:CAL:TARE:MODE
- ***RST Condition:** DIAG:OTDETECT OFF

NOTE If OTD is enabled when *CAL?, or CAL:TARE is executed, the module will disable OTD, wait 1 minute to allow channels to settle, perform the calibration, and then re-enable OTD.

Usage

DIAG:OTD ON,(@100:107,115:123)	*select OTD for the first and third SCP (complete channel lists for readability only)*
DIAG:OTD:STATE ON,(@100,115)	*same function as example above (only first channel of each SCP specified)*
DIAG:OTDETECT:STATE OFF,(@108)	*disable OTD for the 8 channels on the second SCP (only first channel of SCP specified)*

DIAGnostic:OTDetect[:STATe]?

DIAGnostic:OTDetect[:STATe]? (@<*channel*>) returns the current state of "Open Transducer Detection" for the SCP containing the specified *channel*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- *channel* must specify a single channel only.
- **Returned Value:** Returns 1 (enabled) or 0 (disabled). The C-SCPI type is **int16**.
- **Related Commands:** DIAG:OTDETECT:STATE ON | OFF

Usage

DIAG:OTD:STATE? (@108)	*enter statement returns either a 1 or a 0*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DIAGnostic:QUERy:SCPREAD?

DIAGnostic:QUERy:SCPREAD? <*reg_addr*> returns data word from a Signal Conditioning Plug-on register.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
reg_addr	numeric (int32)	0-65,535	none

Comments

- **Returned Value:** returns numeric register value. C-SCPI type is **int32**.

Usage

```
DIAG:QUERY:SCPREAD? 258          read Watchdog SCP's config/status register
enter statement here             return SCP ID value
```

DIAGnostic:VERSion?

DIAGnostic:VERSion? returns the version of the firmware currently loaded into Flash Memory. The version information includes manufacturer, model, serial number, firmware version and date.

Comments

- **Returned Value:** Examples of the response string format:
  HEWLETT-PACKARD,E1419,US34000478,A.04.00,Thu Aug 5 9:38:07 MDT 1994

- The C-SCPI type is **string**.

- **Related Commands:** *IDN?

Usage

```
DIAG:VERS?                       Returns version string as shown above
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Subsystem Syntax FETCh?

The FETCh? command returns readings stored in VME memory.

Comments

- This command is only available in systems using an HP E1405B or HP E1406A command module.
- FETCH? does not alter the readings stored in VME memory. Only the *RST or INIT… commands will clear the readings in VME memory.
- The format of readings returned is set using the FORMat[:DATA] command.
- **Returned Value:** REAL,32, REAL,64, and PACK,64, readings are returned in the IEEE-488.2-1987 Definite Length Arbitrary Block Data format. This data return format is explained in "Arbitrary Block Program Data" on page 6-185 of this chapter. For REAL,32, readings are 4 bytes in length. For REAL 64, and PACK, 64, readings are 8 bytes in length.
- PACKed,64 returns the same values as REAL,64 except for Not-a-Number (NaN), IEEE +INF and IEEE -INF. The NaN, IEEE +INF and IEEE -INF values returned by PACKed,64 are in a form compatible with HP Workstation BASIC and HP BASIC/UX. Refer to the FORMat command for the actual values for NaN, +INF, and -INF.
- ASCii is the default format.
- ASCII readings are returned in the form ±1.234567E±123. For example 13.325 volts would be +1.3325000E+001. Each reading is followed by a comma (,). A line feed (LF) and End-Or-Identify (EOI) follow the last reading.
- **Related Commands:** MEMory Subsystem, FORMat[:DATA]
- ***RST Condition:** MEMORY:VME:ADDRESS 240000; MEMORY:VME:STATE OFF; MEMORY:VME:SIZE 0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Use Sequence

```
MEM:VME:ADDR #H300000
MEM:VME:SIZE #H100000          1M byte or 262144 readings
MEM:VME:STAT ON
        *
        *       (set up E1419 for scanning)
        *
TRIG:SOUR IMM                  let unit trigger on INIT
INIT                           program execution remains here until VME
                               memory is full or the HP E1419 has stopped
                               taking readings
FORM REAL,64                   affects only the return of data
FETCH?
```

NOTE When using the MEM subsystem, the module must be triggered before executing the INIT command (as shown above) unless you are using an external trigger (EXT trigger). When using EXT trigger, the trigger can occur at any time.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The FORMat subsystem provides commands to set and query the response data format of readings returned using the [SENSe:]DATA:FIFO:…? commands.

Subsystem Syntax

FORMat
[:DATA] <*format*>[,<*size*>]
[:DATA]?

FORMat[:DATA]

FORMat[:DATA] <*format*>[,<*size*>] sets the format for data returned using the [SENSe:]DATA:FIFO:…?, [SENSe:]DATA:CVTable, and FETCh? commands.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
format	discrete (string)	REAL | ASCii | PACKed	none
size	numeric	for ASCii, 7 for REAL, 32 | 64 for PACKed, 64	none

Comments

- The REAL format is IEEE-754 Floating Point representation.
- REAL, 32 provides the highest data transfer performance since no format conversion step is placed between reading and returning the data. The default *size* for the REAL format is 32 bits. Also see DIAG:IEEE command.
- PACKed, 64 returns the same values as REAL, 64 except for Not-a-Number (NaN), IEEE +INF and IEEE -INF. The NaN, IEEE +INF and IEEE -INF values returned by PACKed,64 are in a form compatible with HP Workstation BASIC and HP BASIC/UX (see table on following page).
- REAL 32, REAL 64, and PACK 64, readings are returned in the IEEE-488.2-1987 Arbitrary Block Data format. The Block Data may be either Definite Length or Indefinite Length depending on the data query command executed. These data return formats are explained in "Arbitrary Block Program Data" on page 6-185 of this chapter. For REAL 32, readings are 4 bytes in length (C-SCPI type is **float32 array**). For REAL 64, and PACK, 64, readings are 8 bytes in length (C-SCPI type is **float64 array**).
- ASCii is the default format. ASCII readings are returned in the form ±1.234567E±123. For example 13.325 volts would be +1.3325000E+001. Each reading is followed by a comma (,). A line feed (LF) and End-Or-Identify (EOI) follow the last reading (C-SCPI type is **string array**).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE *TST? leaves the instrument in its power-on reset state. This means that the ASC,7 data format is set even if you had it set to something else before executing *TST?. If you need to read the FIFO for test information, set the format after *TST? and before reading the FIFO.

- **Related Commands:** [SENSe:]DATA:FIFO:…?, [SENSe:]DATA:CVTable?, MEMory subsystem, and FETCh?, Also see how DIAG:IEEE can modify REAL,32 returned values.
- ***RST Condition:** ASCII, 7
- After *RST/Power-on, each channel location in the CVT contains the IEEE-754 value "Not-a-number" (NaN). Channel readings which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF. The NaN, +INF, and -INF values for each format are shown in the following table.

Format	IEEE Term	Value	Meaning
ASCii	+INF	+9.9E37	Positive Overload
	-INF	-9.9E37	Negative Overload
	NaN	+9.91E37	No Reading
REAL,32	+INF	$7F800000_{16}$	Positive Overload
	-INF	$FF800000_{16}$	Negative Overload
	NaN	$7FFFFFFF_{16}$	No Reading
REAL,64	+INF	$7FF000...00_{16}$	Positive Overload
	-INF	$FFF000...00_{16}$	Negative Overload
	NaN	$7FFF...FF_{16}$	No Reading
PACKed,64	+INF	47D2 9EAD 3677 $AF6F_{16}$ ($+9.0E37_{10}$)	Positive Overload
	-INF	C7D2 9EAD 3677 $AF6F_{16}$ ($-9.0E37_{10}$)	Negative Overload
	NaN	47D2 A37D CED4 6143_{16} ($+9.91E37_{10}$)	No Reading

Usage

```
FORMAT  REAL          Set format to IEEE 32-bit Floating Point
FORM  REAL, 64        Set format to IEEE 64-bit Floating Point
FORMAT  ASCII, 7      Set format to 7-bit ASCII
```

FORMat[:DATA]?

FORMat[:DATA]? returns the currently set response data format for readings.

Comments

- **Returned Value:** Returns REAL, +32 | REAL, +64 | PACK, +64 | ASC, +7. The C-SCPI type is **string, int16**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Related Commands:** FORMAT
- ***RST Condition:** ASCII, 7

Usage FORMAT? *Returns REAL, +32 | REAL, +64 | PACK, +64 | ASC, +7*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The INITiate command subsystem moves the HP E1419 from the Trigger Idle State to the Waiting For Trigger State. When initiated, the instrument is ready to receive one (:IMMediate) or more (depending on TRIG:COUNT) trigger events. On each trigger, the module will perform one control cycle which includes reading analog and digital input channels (Input Phase), executing all defined algorithms (Calculate Phase), and updating output channels (Output Phase). See the TRIGger subsystem to specify the trigger source and count.

Subsystem Syntax

```
INITiate
    [:IMMediate]
```

INITiate[:IMMediate]

INITiate[:IMMediate] changes the trigger system from the Idle state to the Wait For Trigger state. When triggered, one or more (depending on TRIGger:COUNt) trigger cycles occur and the instrument returns to the Trigger Idle state.

Comments

- INIT:IMM clears the FIFO and Current Value Table.
- If a trigger event is received before the instrument is Initiated, a -211 "Trigger ignored" error is generated.
- If another trigger event is received before the instrument has completed the current trigger cycle (measurement scan), the Questionable Data Status bit 9 is set and a +3012 "Trigger too fast" error is generated.
- Sending INIT while the system is still in the Wait for Trigger state (already INITiated) will cause an error -213,"Init ignored".
- Sending the ABORt command send the trigger system to the Trigger Idle state when the current input-calculate-output cycle is completed.
- If updates are pending, they are made prior to beginning the Input phase.
- **When Accepted:** Not while INITiated
- **Related Commands:** ABORt, CONFigure, TRIGger
- ***RST Condition:** Trigger system is in the Idle state.

Usage

```
INIT                                    Both versions same function
INITIATE:IMMEDIATE
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The INPut subsystem controls configuration of programmable *input* Signal Conditioning Plug-Ons (SCPs).

Subsystem Syntax

```
INPut
    :DEBounce
        :TIME  <time>,(@<ch_list>)
    :FILTer
        [:LPASs]
            :FREQuency  <cutoff_freq>,(@<ch_list>)
            :FREQuency?  (@<channel>)
            [:STATe]  1 | 0 | ON | OFF,(@<channel>)
            [:STATe]?  (@<channel>)
    :GAIN  <chan_gain>,(@<ch_list>)
    :GAIN?  (@<channel>)
    :LOW <wvolt_type>,(@<ch_list>)
    :LOW?  (@<channel>)
    :POLarity  NORMal | INVerted,(@<ch_list>)
    :POLarity?  (@<channel>)
    :THReshold
        LEVel?  (@<channel>)
```

INPut:DEBounce:TIME

INPut:DEBounce:TIME <*time*>,(@<*ch_list*>) sets the debounce time on the specified digital input channels.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
time	numeric (float32) (string)	see comment | MIN | MAX	Hz
ch_list	channel list (string)	132 - 163	none

Comments

- For a description of the debounce function see "Debounce Function" in the HP E1536 SCP manual. The HP E1536A has two debounce timers. One for the lower four channels, and one for the upper four channels. To set the debounce timers use the command:
  INPut:DEBounce:TIME <*time*>,(@<*ch_list*>)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- <*time*> can be one of 16 possible numeric values or MIN and MAX:

0	150 µS	300 µS	600 µS
1.2 mS	2.4 mS	4.8 mS	9.6 mS
19.2 mS	38.4 mS	76.6 mS	153.6 mS
307.2 mS	614.4 mS	1.2288 S	2.4576 S

  - Sending 0 or MIN turns debounce off. Sending MAX selects 2.458 seconds.
  - If a value is sent that is slightly greater than one of these values, the next higher value (or MAX) is selected. Values outside of the range 0 - 2.4576 will generate the error -222, "Data out of range".

- Since the HP E1536 has two debounce timers (one for each bank of 4 channels) <*ch_list*> must contain all 4 of the upper-bank channels, or all 4 of the lower-bank channels, or all 8 channels for a given SCP. This is because the HP E1536 has two debounce timers, one for its lower 4 channels and one for its upper 4 channels.

Note The INP:DEB:TIME generate the error 3108, "E1536 debounce - each referenced 4 Ch bank must contain at least one input". This error indicates that <*ch_list*> referenced a bank of channels that contains no input configured channel.

- **Usage:** To set the debounce period to 153.6 msec for the lower 4 channels on an HP E1536A in SCP position 0 send:

```
INP:DEB  0.1536,(@100:103)
```

To set the debounce period to 1.229 seconds for the upper 4 channels on an HP E1536A in SCP position 3 send:

```
INP:DEB  1.229,(@128:131)
```

- There is also the query form; **INPut:DEBounce:TIME? (@<*channel*>)** where <*channel*> must specify a single channel. INP:DEB:TIME? returns the currently set debounce period.

INPut:FILTer[:LPASs]:FREQuency

INPut:FILTer[:LPASs]:FREQuency <*cutoff_freq*>,(@<*ch_list*>) sets the cutoff frequency of the filter on the specified channels.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
cutoff_freq	numeric (float32) (string)	see comment | MIN | MAX	Hz
ch_list	channel list (string)	132 - 163	none

Comments

- *cutoff_freq* may be specified in killoHertz (khz). A programmable Filter SCP has a choice of several discrete cutoff frequencies. The cutoff frequency set will be the one closest to the value specified by *cutoff_freq*. Refer to Chapter 6 for specific information on the SCP you are programming.
- Sending MAX for the *cutoff_freq* selects the SCP's highest cutoff frequency. Sending MIN for the *cutoff_freq* selects the SCP's lowest cutoff frequency. To disable filtering (the "pass through" mode), execute the INP:FILT:STATE OFF command.
- Sending a value greater than the SCP's highest cutoff frequency or less than the SCP's lowest cutoff frequency generates a -222 "Data out of range" error.
- **When Accepted:** Not while INITiated
- **Related Commands:** INP:FILT:FREQ?, INP:FILT:STAT ON | OFF
- ***RST Condition:** set to MIN

Usage

```
INP:FILT:FREQ  100,(@140:143)        Set cutoff frequency of 100 Hz for channels 40 through 43
INPUT:FILTER:FREQ  2,(@155)          Set cutoff frequency of 2 Hz for channel 55
```

INPut:FILTer[:LPASs]:FREQuency?

INPut:FILTer[:LPASs]:FREQuency? (@<*channel*>) returns the cutoff frequency currently set for *channel*. Non-programmable SCP channels may be queried to determine their fixed cutoff frequency. If the channel is not on an input SCP, the query will return zero.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- *channel* must specify a single channel only.
- This command is for programmable filter SCPs only.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Returned Value:** Numeric value of Hz as set by the INP:FILT:FREQ command. The C-SCPI type is **float32**.
- **When Accepted:** Not while INITiated
- **Related Commands:** INP:FILT:LPAS:FREQ, INP:FILT:STATE
- ***RST Condition:** MIN

Usage

```
INPUT:FILTER:LPASS:FREQUENCY?  (@155)   Check cutoff freq on channel 55
INP:FILT:FREQ?  (@100)                  Check cutoff freq on channel 0
```

INPut:FILTer[:LPASs][:STATe]

INPut:FILTer[:LPASs][:STATe] <*enable*>,(@<*ch_list*>) enables or disables a programmable filter SCP channel. When disabled (*enable*=OFF), these channels are in their "pass through" mode and provide no filtering. When re-enabled (*enable*=ON), the SCP channel reverts to its previously programmed setting.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	boolean (uint16)	1 | 0 | ON | OFF	none
ch_list	channel list (string)	132 - 163	none

Comments

- If the SCP has not yet been programmed, ON enables the SCP's default cutoff frequency.
- **When Accepted:** Not while INITiated
- ***RST Condition:** ON

Usage

```
INPUT:FILTER:STATE  ON,(@132,134)   Channels 32 and 34 return to previously set
                                    (or default) cutoff frequency
INP:FILT  OFF,(@132:139)            Set channels 32-39 to "pass-through" state
```

INPut:FILTer[:LPASs][:STATe]?

INPut:FILTer[LPASs][:STATe]? (@<*channel*>) returns the currently set state of filtering for the specified channel. If the channel is not on an input SCP, the query will return zero.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- **Returned Value:** Numeric value either 0 (off or "pass-through") or 1 (on). The C-SCPI type is **int16**.
- *channel* must specify a single channel only.

Usage

```
INPUT:FILTER:LPASS:STATE? (@115)    Enter statement returns either 0 or 1
INP:FILT? (@115)                    Same as above
```

INPut:GAIN

INPut:GAIN ***<gain>,(@<ch_list>)*** sets the channel gain on programmable amplifier Signal Conditioning Plug-Ons.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
gain	numeric (float32) discrete (string)	see comment | MIN | MAX	none
ch_list	channel list (string)	132 - 163	none

Comments

- A programmable amplifier SCP has a choice of several discrete gain settings. The gain set will be the one closest to the value specified by *gain*. Refer to your SCP manual for specific information on the SCP you are programming. Sending MAX will program the highest gain available with the SCP installed. Sending MIN will program the lowest gain.
- Sending a value for *gain* that is greater than the highest or less than the lowest setting allowable for the SCP will generate a -222 "Data out of range" error.
- **When Accepted:** Not while INITiated
- **Related Commands:** INP:GAIN?
- ***RST Condition:** gain set to MIN

Usage

```
INP:GAIN 8,(@140:147)      Set gain of 8 for channels 40 through 47
INPUT:GAIN 64,(@155)       Set gain of 64 for channel 55
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

INPut:GAIN?

INPut:GAIN? (@<*channel*>) returns the gain currently set for *channel*. If the channel is not on an input SCP, the query will return zero.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- *channel* must specify a single channel only.
- If the channel specified does not have a programmable amplifier, INP:GAIN? will return the nominal as-designed gain for that channel.
- **Returned Value:** Numeric value as set by the INP:GAIN command. The C-SCPI type is **float32.**
- **When Accepted:** Not while INITiated
- **Related Commands:** INP:GAIN
- ***RST Condition:** gain set to 1

Usage

```
INPUT:GAIN? (@105)        Check gain on channel 5
INP:GAIN? (@100)          Check gain on channel 0
```

INPut:LOW

INPut:LOW <*wvolt_type*>,(@<*ch_list*>) controls the connection of input LO at a Strain Bridge SCP channel specified by <*ch_list*>. LO can be connected to the Wagner Voltage ground or left floating.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
wvolt_type	discrete (string)	FLOat | WVOLtage	none
ch_list	channel list (string)	132 - 163	none

Comments

- **Related Commands:** INP:LOW?
- ***RST Condition:** INP:LOW FLOAT (all HP E1511 channels)

Usage

```
INP:LOW WVOL (@132:139,148:155)   connect LO of channels 32 through 39 and
                                  48 through 55 to Wagner Ground.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

INPut:LOW?

INPut:LOW? (@<*channel*>) returns the LO input configuration for the channel specified by <*channel*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	132 - 163	none

Comments

- *channel* must specify a single channel only.
- **Returned Value:** Returns FLO or WV. The C-SCPI type is **string**.
- **Related Commands:** INP:LOW

Usage

```
INP:LOW? (@148)          enter statement will return either FLO or
                         WV for channel 48
```

INPut:POLarity

INPut:POLarity <*mode*>,<*ch_list*> sets logical input polarity on a digital SCP channel.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
mode	discrete (string)	NORMal | INVerted	none
ch_list	string	132 - 163	none

Comments

- If the channels specified are on an SCP that doesn't support this function, an error will be generated. See your SCP's User's Manual to determine its capabilities.
- **Related Commands:** for output sense; SOURce:PULSe:POLarity
- ***RST Condition:** INP:POL NORM for all digital SCP channels.

Usage

```
INP:POL INV,(@140:143)   invert first 4 channels on SCP at SCP
                         position 5. Channels 40 through 43
```

INPut:POLarity?

INPut:POLarity? <*channel*> returns the logical input polarity on a digital SCP channel.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- <*channel*> must specify a single channel.
- If the channel specified is on an SCP that doesn't support this function, an error will be generated. See your SCP's User's Manual to determine its capabilities.
- **Returned Value:** returns "NORM" or "INV". The type is **string**.

INPut:THReshold:LEVel?

INPut:THReshold:LEVel (@<*channel*>) returns the currently set input threshold level.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- Channel must specify a single channel
- For the HP E1536 Isolated Digital I/O SCP, INP:THR:LEV? returns a numeric value which is one of 5, 12, 24, 48, or 0 (zero) where zero means that the channel is configured as an output and non-zero values indicate the input threshold in volts.

Note

If an invalid switch combination is set on an HP E1536, INP:THR:LEV? will NOT return a value and will generate the error 3105 "Invalid SCP switch setting". This error will also be generated when *RST is executed. Channels associated with this error will behave as input channels with unknown threshold levels.

- **Usage:** To query the threshold level on the second channel at SCP position 2 send:

```
INP:THR:LEV?  (@117)          query 2nd chan on SCP pos. 2
enter statement here          returns 0 | 5 | 12 | 24 | 48
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The MEMory subsystem allows using VME memory as an additional reading storage buffer.

Subsystem Syntax

```
MEMory
    :VME
        :ADDRess  <A24_address>
        :ADDRess?
        :SIZE  <mem_size>
        :SIZE?
        :STATe  1 | 0 | ON | OFF
        :STATe?
```

NOTE This subsystem is only available in systems using an HP E1405B or HP E1406A command module.

Use Sequence

```
*RST
MEM:VME:ADDR #H300000
MEM:VME:SIZE #H100000          1M byte or 262144 readings
MEM:VME:STAT ON
        *
        *       (set up E1419 for scanning)
        *
TRIG:SOUR IMM                  let unit trigger on INIT
INIT
*OPC?                          program execution remains here until VME
                               memory is full or the HP E1419 has stopped
                               taking readings
FORM REAL,64                   affects only the return of data
FETCH?                         return data from VME memory
```

NOTE When using the MEM subsystem, the module must be triggered before executing the INIT command (as shown above) unless you are using an external trigger (EXT trigger). When using EXT trigger, the trigger can occur at any time.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MEMory:VME:ADDRess

MEMory:VME:ADDRess <***A24_address***> sets the A24 address of the VME memory card to be used as additional reading storage.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
A24_address	numeric	valid A24 address	none

Comments

- This command is only available in systems using an HP E1405B or HP E1406A command module.
- The default (if MEM:VME:ADDR not executed) is 240000_{16}.
- *A24_address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B).
- **Related Commands:** MEMory subsystem, FORMat, and FETCH?
- ***RST Condition:** VME memory address starts at 200000_{16}. When using an HP E1405/6 command module, the first HP E1419 occupies 200000_{16} - $23FFFF_{16}$.

Usage

```
MEM:VME:ADDR  #H400000
```

Set the address for the VME memory card to be used as reading storage

MEMory:VME:ADDRess?

MEMory:VME:ADDRess? returns the address specified for the VME memory card used for reading storage.

Comments

- **Returned Value:** numeric.
- This command is only available in systems using an HP E1405B or HP E1406A command module.
- **Related Commands:** MEMory subsystem, , FORMat, and FETCH?

Usage

```
MEM:VME:ADDR?
```

Returns the address of the VME memory card.

MEMory:VME:SIZE

MEMory:VME:SIZE <***mem_size***> Specifies the number of bytes of VME memory to allocate for additional reading storage.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
mem_size	numeric	to limit of available VME memory	none

Comments

- This command is only available in systems using an HP E1405B or HP E1406A command module.
- *mem_size* may be specified in decimal, hex (#H), octal (#Q), or binary(#B).
- *mem_size* should be a multiple of four (4) to accommodate 32 bit readings.
- **Related Commands:** MEMory subsystem, FORMAT, and FETCH?
- ***RST Condition:** MEM:VME:SIZE 0

Usage

```
MEM:VME:SIZE  32768
```

Allocate 32 Kbytes of VME memory to reading storage (8192 readings)

MEMory:VME:SIZE?

MEMory:VME:SIZE? returns the amount (in bytes) of VME memory allocated to reading storage.

Comments

- This command is only available in systems using an HP E1405B or HP E1406A command module.
- **Returned Value:** Numeric.
- **Related Commands:** MEMory subsystem, and FETCH?

Usage

```
MEM:VME:SIZE?
```

Returns the number of bytes allocated to reading storage.

MEMory:VME:STATe

MEMory:VME:STATe ***<enable>*** enables or disables use of the VME memory card as additional reading storage.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	boolean (uint16)	1 | 0 | ON | OFF	none

Comments

- This command is only available in systems using an HP E1405B or HP E1406A command module.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- When the VME memory card is enabled, the INIT command does not terminate until data acquisition stops or VME memory is full.
- **Related Commands:** Memory subsystem, and FETCH?
- ***RST Condition:** MEM:VME:STAT OFF

Usage

```
MEMORY:VME:STATE  ON        enable VME card as reading storage
MEM:VME:STAT  0             Disable VME card as reading storage
```

MEMory:VME:STATe?

MEMory:VME:STATe? returned value of 0 indicates that VME reading storage is disabled. Returned value of 1 indicates VME memory is enabled.

Comments

- This command is only available in systems using an HP E1405B or HP E1406A command module.
- **Returned Value:** Numeric 1 or 0. C-SCPI type **uint16**.
- **Related Commands:** MEMory subsystem, and FETCH?

Usage

```
MEM:VME:STAT?               Returns 1 for enabled, 0 for disabled
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The OUTPut subsystem is involved in programming source SCPs as well as controlling the state of VXIbus TTLTRG lines 0 through 7.

Subsystem Syntax

```
OUTPut
    :CURRent
        :AMPLitude  <amplitude>,(@<ch_list>)
        :AMPLitude?  (@<channel>)
        [:STATe]  1 | 0 | ON | OFF,(@<ch_list>)
        [:STATe]?  (@<channel>)
    :POLarity  NORMal | INVerted,(@<ch_list>)
    :POLarity?  (@<channel>)
    :SHUNt  1 | 0 | ON | OFF,(@<ch_list>)
    :SHUNt?  (@<channel>)
    :TTLTrg
        :SOURce  TRIGger | FTRigger | SCPlugon | LIMit
        :SOURce?
    :TTLTrg<n>
        [:STATe]  1 | 0 | ON | OFF
        [:STATe]?
    :TYPE  PASSive | ACTive,(@<ch_list>)
    :TYPE?  (@<channel>)
    :VOLTage
        :AMPLitude <amplitude>,(@<ch_list>)
        :AMPLitude? (@<channel>)
```

OUTPut:CURRent:AMPLitude

OUTPut:CURRent:AMPLitude *<amplitude>,(@<ch_list>)* sets the HP E1505 Current Source SCP channels specified by *ch_list* to either 488 µA, or 30 µA. This current is typically used for four-wire resistance and resistance temperature measurements.

NOTE This command does not set current amplitude on SCPs like the HP E1532 Current Output SCP.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
amplitude	numeric (float32)	MIN | 30E-6 | MAX | 488E-6	ADC
ch_list	channel list (string)	132 - 163	none

Comments

- Select 488E-6 (or MAX) for measuring resistances of less than 8000 Ohms. Select 30E-6 (or MIN) for resistances of 8000 Ohms and above. ***amplitude*** may be specified in µA (ua).
- For resistance temperature measurements ([SENSe:]FUNCtion:TEMPerature) the Current Source SCP must be set as follows:

Required Current Amplitude	Temperature Sensor Types and Subtypes
MAX (488µA) MIN (30µA)	RTD,85 | 92 and THER,2250 THER,5000 | 10000

- When *CAL? is executed, the current sources are calibrated on the range selected at that time.
- **When Accepted:** Not while INITiated
- **Related Commands:** *CAL?, OUTP:CURR:AMPL?
- ***RST Condition:** MIN

Usage

```
OUTP:CURR:AMPL  488ua,(@140:147)
```
Set Current Source SCP at channels 40 through 47 to 488 µA

```
OUTP:CURR:AMPL  30E-6,(@148)
```
Set Current Source SCP at ch 48 to 30 µA

OUTPut:CURRent:AMPLitude?

OUTPut:CURRent:AMPLitude? (@<*channel*>) returns the range setting of the Current Source SCP channel specified by *channel*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	132 - 163	none

Comments

- *channel* must specify a single channel only.
- If *channel* specifies an SCP which is not a Current Source, a +3007, "Invalid signal conditioning plug-on" error is generated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Returned Value:** Numeric value of amplitude set. The C-SCPI type is **float32**.
- **Related Commands:** OUTP:CURR:AMPL

Usage

```
OUTP:CURR:AMPLITUDE?  (@140)
```

Check SCP current set for channel 40 (returns +3.0E-5 or +4.88E-4)

OUTPut:CURRent[:STATe]

OUTPut:CURRent[:STATe] <*enable*>,(@<*ch_list*>) enables or disables current source on channels specified in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	boolean (uint16)	1 | 0 | ON | OFF	none
ch_list	channel list (string)	132 - 163	none

Comments

- OUTP:CURR:STAT does not affect a channel's amplitude setting. A channel that has been disabled, when re-enabled sources the same current set by the previous OUTP:CURR:AMPL command.
- OUTP:CURR:STAT is most commonly used to turn off excitation current to four-wire resistance (and resistance temperature device) circuits during execution of CAL:TARE for those channels.
- **When Accepted:** Not while INITiated
- **Related Commands:** OUTP:CURR:AMPL, CAL:TARE
- ***RST Condition:** OUTP:CURR OFF (all channels)

Usage

```
OUTP:CURR OFF,(@140,147)
```

turn off current source channels 40 and 47

OUTPut:CURRent[:STATe]?

OUTPut:CURRent[:STATe]? (@<*channel*>) returns the state of the Current Source SCP channel specified by <*channel*>. If the channel is not on an HP E1505 Current Source SCP, the query will return zero.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	132 - 163	none

Comments

- *channel* must specify a single channel only.
- **Returned Value:** returns 1 for enabled, 0 for disabled. C-SCPI type is **uint16**.
- **Related Commands:** OUTP:CURR:STATE, OUTP:CURR:AMPL

Usage

```
OUTP:CURR?  (@147)                  query for state of Current SCP channel 47
execute enter statement here        enter query value, either 1 or 0
```

OUTPut:POLarity

OUTPut:POLarity <*select*>,(@<ch_list>) sets the polarity on digital output channels in <ch_list>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
select	discrete (string)	NORMal | INVerted	none
ch_list	string	132 - 163	none

Comments

- If the channels specified do not support this function, an error will be generated.
- Related Commands: INPut:POLarity, OUTPut:POLarity?
- ***RST Condition:** OUTP:POL NORM for all digital channels

Usage

```
OUTP:POL  INV,(@156)                invert output logic sense on channel 56
```

OUTPut:POLarity?

OUTPut:POLarity? (@<*channel*>) returns the polarity on the digital output channel in <*channel*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- *Channel* must specify a single channel

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Returned Value:** returns one of NORM or INV. The type is **string**.

OUTPut:SHUNt

OUTPut:SHUNt <*enable*>,(@<*ch_list*>) adds shunt resistance to one leg of bridge on Strain Bridge Completion SCPs. This can be used for diagnostic purposes and characterization of bridge response.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	boolean (uint16)	0 | 1 | ON | OFF	none
ch_list	channel list (string)	132 - 163	none

Comments

- If *ch_list* specifies a non strain SCP, a 3007 "Invalid signal conditioning plug-on" error is generated.
- **When Accepted:** Not while INITiated
- **Related Commands:** [SENSe:]FUNCtion:STRain…, [SENSe:]STRain…
- ***RST Condition:** OUTP:SHUNT 0 on all Strain SCP channels

Usage

```
OUTP:SHUNT 1,(@148:151)        add shunt resistance at channels 48 - 51
```

OUTPut:SHUNt?

OUTPut:SHUNt? (@<*channel*>) returns the status of the shunt resistance on the specified Strain SCP channel.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	132 - 163	none

Comments

- channel must specify a single channel only.
- If *channel* specifies a non strain SCP, a 3007 "Invalid signal conditioning plug-on" error is generated.
- **Returned Value:** Returns 1 or 0. The C-SCPI type is **uint16**.
- **Related Commands:** OUTP:SHUNT

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Usage OUTPUT:SHUNt? (@151) *Check status of shunt resistance on channel 51*

OUTPut:TTLTrg:SOURce

OUTPut:TTLTrg:SOURce <*trig_source*> selects the internal source of the trigger event that will operate the VXIbus TTLTRG lines.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
trig_source	discrete (string)	ALGorithm | TRIGger | FTRigger | SCPlugon	none

Comments

- The following table explains the possible choices.

Parameter Value	Source of Trigger
ALGorithm	Generated by the Algorithm Language function "interrupt()"
FTRigger	Generated on the First Trigger of a multiple "counted scan" (set by TRIG:COUNT <*trig_count*>)
SCPlugon	Generated by a Signal Conditioning Plug-on (SCP). Do not use this when Sample-and-Hold SCPs are installed.
TRIGger	Generated every time a scan is triggered (see TRIG:SOUR <*trig_source*>)

- FTRigger (First TRigger) is used to generate a single TTLTRG output when repeated triggers are being used to make multiple executions of the enabled algorithms. The TTLTRG line will go low (asserted) at the first trigger event and stay low through subsequent triggers until the trigger count (as set by TRIG:COUNT) is exhausted. At this point the TTLTRG line will return to its high state (de-asserted). This feature can be used to signal when the HP E1419 has started running its control algorithms.

- **Related Commands:** OUTP:TTLT<n>[:STATE], OUTP:TTLT:SOUR?, TRIG:SOUR, TRIG:COUNT

- ***RST Condition:** OUTP:TTLT:SOUR TRIG

Usage OUTP:TTLT:SOUR TRIG *toggle TTLTRG line every time module is triggered (use to trigger other HP E1419s)*

OUTPut:TTLTrg:SOURce?

OUTPut:TTLTrg:SOURce? returns the current setting for the TTLTRG line source.

Comments

- **Returned Value:** Discrete, one of; TRIG, FTR, or SCP. C-SCPI type is **string**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Related Commands:** OUTP:TTLT:SOUR

Usage

```
OUTP:TTLT:SOUR?
```
enter statement will return on of FTR, SCP, or TRIG

OUTPut:TTLTrg<n>[:STATe]

OUTPut:TTLTrg<n>:STATe <*ttltrg_cntrl*> specifies which VXIbus TTLTRG line is enabled to source a trigger signal when the module is triggered. TTLTrg<n> can specify line 0 through 7. For example, …:TTLTRG4, or TTLT4 for VXIbus TTLTRG line 4.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ttltrg_cntrl	boolean (uint16)	1 | 0 | ON | OFF	none

Comments

- Only one VXIbus TTLTRG line can be enabled simultaneously.
- **When Accepted:** Not while INITiated
- **Related Commands:** ABORT, INIT…, TRIG…
- ***RST Condition:** OUTPut:TTLTrg<0 through 7> OFF

Usage

```
OUTP:TTLT2  ON
```
Enable TTLTRG2 line to source a trigger

```
OUTPUT:TTLTRG7:STATE  ON
```
Enable TTLTRG7 line to source a trigger

OUTPut:TTLTrg<n>[:STATe]?

OUTPut:TTLTrg<n>[:STATe]? returns the current state for TTLTRG line <n>.

Comments

- **Returned Value:** Returns 1 or 0. The C-SCPI type is **int16**.
- **Related Commands:** OUTP:TTLT<n>

Usage

```
OUTP:TTLT2?
```
See if TTLTRG2 line is enabled (returns 1 or 0)

```
OUTPUT:TTLTRG7:STATE?
```
See if TTLTRG7 line is enabled

OUTPut:TYPE

OUTPut:TYPE <*select*>,(@<*ch_list*>) sets the output drive characteristic for digital SCP channels.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
select	discrete (string)	PASSive | ACTive	seconds
ch_list	string	132 - 163	none

Comments

- If the channels specified are on an SCP that doesn't support this function an error will be generated. See your SCP's User's Manual to determine its capabilities.
- PASSive configures the digital channel/bit to be passive (resistor) pull-up to allow you to wire-or more than one output together.
- ACTive configures the digital channel/bit to both source and sink current.
- **Related Commands:** SOURce:PULSe:POLarity, OUTPut:TYPE?
- ***RST Condition:** OUTP:TYPE ACTIVE (for TTL compatibility)

Usage

```
OUTP:TYPE PASS,(@156:159)        make channels 56 to 59 passive pull-up
```

OUTPut:TYPE?

OUTPut:TYPE? <*channel*> returns the output drive characteristic for a digital channel.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- *Channel* must specify a single channel.
- If the channel specified is not on a digital SCP, an error will be generated.
- **Returned Value:** returns PASS or ACT. The type is **string**.
- ***RST Condition:** returns ACT

OUTPut:VOLTage:AMPLitude

OUTPut:VOLTage:AMPLitude <*amplitude*>,(@<*ch_list*>) sets the excitation voltage on programmable Strain Bridge Completion SCPs pointed to by <*ch_list*> (the HP E1511 for example). This command is not used to set output voltage on SCPs like the HP E1531 Voltage Output SCP.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
amplitude	numeric (float32)	MIN | 0 |1 | 2 | 5 | 10 | MAX	none
ch_list	channel list (string)	132 - 163	none

Comments

- To turn off excitation voltage (when using external voltage source) program *amplitude* to 0.
- **Related Commands:** OUTP:VOLT:AMPL?
- ***RST Condition:** MIN (0)

Usage

```
OUTP:VOLT:AMPL  5,(@132:135)
```

set excitation voltage for channels 32 - 35

OUTPut:VOLTage:AMPLitude?

OUTPut:VOLTage:AMPLitude? (@<*channel*>) returns the current setting of excitation voltage for the channel specified by <*channel*>. If the channel is not on an HP E1511 SCP, the query will return zero.

Comments

- *channel* must specify a single channel only.
- **Returned Value:** Numeric, one of 0, 1, 2 ,5, or 10. C-SCPI type is **float32**.
- **Related Commands:** OUTP:VOLT:AMPL

Usage

```
OUTP:VOLT:AMPL?  (@135)
```

returns current setting of excitation voltage for channel 3

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The ROUTe subsystem provides a method to query the overall channel list definition for its sequence of channels.

Subsystem Syntax

```
ROUTe
    :SEQuence
        :DEFine?
        :POINts?
```

ROUTe:SEQuence:DEFine?

ROUTe:SEQuence:DEFine? <*type*> returns the sequence of channels defined in the scan list.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
type	(string)	AIN | AOUT | DIN | DOUT	none

Comments

- The channel list contents and sequence are determined primarily by channel references in the algorithms currently defined. The SENS:REF:CHANNELS, and SENS:CHAN:SETTLING commands also effect the scan list contents.
- The <*type*> parameter selects which channel list will be queried:

  "AIN" selects the Analog Input channel list (this is the Scan List).
  "AOUT" selects the Analog Output channel list.
  "DIN" selects the Digital Input channel list.
  "DOUT" selects the Digital Output channel list.

- **Returned Value:** Definite Length Arbitrary Block Data format. This data return format is explained in "Arbitrary Block Program Data" on page 6-185 of this chapter. Each value is 2 bytes in length (the C-SCPI data type is an **int16 array**).
- ***RST Condition:** To supply the necessary time delay before Digital inputs are read, the analog input (AIN) scan list contains eight entries for channel 0 (100).This minimum delay is maintained by replacing these default channels as others are defined in algorithms. After algorithm definition, if some delay is still required, there will be repeat entries of the last channel referenced by an algorithm. The three other lists contain no channels.

Usage

```
ROUT:SEQ:DEF? AIN        query for analog input (Scan List) sequence
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ROUTe:SEQuence:POINts?

ROUTe:SEQuence:POINts? <*type*> returns the number of channels defined in each of the four channel list types.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
type	(string)	AIN | AOUT | DIN | DOUT	none

Comments

- The channel list contents and sequence are determined by channel references in the algorithms currently defined.
- The <*type*> parameter selects which channel list will be queried:

  "AIN" selects the Analog Input list.
  "AOUT" selects the Analog Output list.
  "DIN" selects the Digital Input list.
  "DOUT" selects the Digital Output list.

- **Returned Value:** Numeric. The C_SCPI type is **int16**.
- ***RST Condition:** The Analog Input list returns +8, the others return +0.

Usage

```
ROUT:SEQ:POINTS? AIN          query for analog input channel count
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The SAMPle subsystem provides commands to set and query the interval between channel measurements (pacing).

Subsystem Syntax

```
SAMPle
    :TIMer  <interval>
    :TIMer?
```

SAMPle:TIMer

SAMPle:TIMer ***<interval>*** sets the time interval between channel measurements. It is used to provide additional channel settling time. See "Settling Characteristics" discussion on page 3-104.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
interval	numeric (float32) (string)	1.0E-5 to 16.3825E-3 | MIN | MAX	seconds

Comments

- The minimum *interval* is 10 µ seconds. The resolution for *interval* is 2.5 µsecond.
- If the Sample Timer interval multiplied by the number of channels in the specified Scan List is longer than the Trigger Timer interval, at run time a "Trigger too fast" error will be generated.
- the SAMP:TIMER interval can change the effect of the SENS:CHAN:SETTLING command. ALG:CHAN:SETT specifies the number of times a channel measurement should be repeated. The total settling time per channel then is (SAMP:TIMER <*interval*>) X (<*chan_repeats*> from SENS:CHAN:SETT)
- **When Accepted:** Not while INITiated
- **Related Commands:** SENSE:CHAN:SETTLING, SAMP:TIMER?
- ***RST Condition:** Sample Timer for all Channel Lists set to 1.0E-5 seconds.

Usage

```
SAMPLE:TIMER  50E-6
```

Pace measurements at 50µsecond intervals

SAMPle:TIMer?

SAMPle:TIMer? returns the sample timer interval.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

- **Returned Value:** Numeric. The C-SCPI type is **float32**.
- **Related Commands:** SAMP:TIMER
- ***RST Condition:** Sample Timer set to 1.0E-5 seconds.

Usage

```
SAMPLE:TIMER?
```

Check the interval between channel measurements

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Subsystem Syntax [SENSe:]

```
:CHANnel
    :SETTling <settle_time>,(@<ch_list>)
    :SETTling? (@<channel>)
DATA
    :CVTable? (@<element_list>)
        :RESet
    :FIFO
        [:ALL]?
        :COUNt?
            :HALF?
        :HALF?
        :MODE BLOCk | OVERwrite
        :MODE?
        :PART? <n_values>
        :RESet
FREQuency:APERture <gate time>,<ch_list>
FREQuency:APERture? <channel>
FUNCtion
    :CONDition (@<ch_list>)
    :CUSTom [<range>,](@<ch_list>)
        :REFerence [<range>,](@<ch_list>)
        :TC <type>,[<range>,](@<ch_list>)
    :FREQuency (@<ch_list>)
    :RESistance <excite_current>,[<range>,](@<ch_list>)
    :STRain
        :FBENding [<range>,](@<ch_list>)
        :FBPoisson [<range>,](@<ch_list>)
        :FPOisson [<range>,](@<ch_list>)
        :HBENding [<range>,](@<ch_list>)
        :HPOisson [<range>,](@<ch_list>)
        [:QUARter] [<range>,](@<ch_list>)
    :TEMPerature <sensor_type>,<sub_type>,[<range>,](@<ch_list>)
    :TOTalize (@<ch_list>)
    :VOLTage[:DC] [<range>,](@<ch_list>)
REFerence <sensor_type>, [<sub_type>,](@<ch_list>)
    :CHANnels (@<ref_channel>),(@<ch_list>)
    :TEMPerature <degrees_celsius>
STRain
    :EXCitation <excite_v>,(@<ch_list>)
    :EXCitation? (@<channel>)
    :GFACtor <gage_factor>,(@<ch_list>)
    :GFACtor? (@<channel>)
    :POISson <poisson_ratio>,(@<ch_list>)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

:POISson? (@<*channel*>)
:UNSTrained <*unstrained_v*>,(@<*ch_list*>)
:UNSTrained? (@<*channel*>)
TOTalize:RESet:MODE INIT | TRIGger,(@<*ch_list*>)
TOTalize:RESet:MODE? (@<*channel*>)

[SENSe:]CHANnel:SETTling

[SENSe:]CHANnel:SETTling <*num_samples*>,<*ch_list*> specifies the number of measurement samples to make on channels in <*ch_list*>. SENS:CHAN:SETTLING is used to provide additional settling time only to selected channels that might need it. See the "Settling Characteristics" discussion on page 3-104.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
settle_time	numeric (int16)	1 to 64	none
ch_list	string	100 - 163	none

Comments

- SENS:CHAN:SETTLING causes each channel specified in <*ch_list*> that is also referenced in an algorithm to appear <*num_samples*> times in the analog input Scan List. Channels that do not appear in any SENS:CHAN:SETT command will be entered into the scan list only once when referenced in an algorithm.
- Since the scan list is limited to 64 entries, an error will be generated if the number of channels referenced in algorithms plus the additional entries from any SENS:CHAN:SETTLING commands that coincide with algorithm referenced channels exceeds 64.
- The SAMPLE:TIMER command can change the effect of the SENS:CHAN:SETTLING command since SAMPLE:TIMER changes the amount of time for each measurement sample.
- **When Accepted:** Not while INITiated
- **Related Commands:** [SENSe:]CHANnel:SETTling?, SAMPLE:TIMER
- ***RST Condition:** SENS:CHAN:SETTLING 1,(@100:163)

Usage

```
SENS:CHAN:SETT 4,(@144,156)
```

settle channels 44 and 56 for 4 measurement periods

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]CHANnel:SETTling?

[SENSe:]CHANnel:SETTling? <*channel*> returns the current number of samples to make on <*channel*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	100 - 163	none

Comments

- <*channel*> must specify a single channel.
- Related Commands: SENS:CHAN:SETT, SAMP:TIMER?
- ***RST Condition:** will return 1 for all channels.
- **Returned Value:** returns numeric number of samples, The type is **int16**.

[SENSe:]DATA:CVTable?

[SENSe:]DATA:CVTable? (@<*element_list*>) returns from the Current Value Table the most recent values stored by algorithms.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
element_list	channel list	10 - 511	none

Comments

- [SENSe:]DATA:CVTable? (@<*element_list*>) allows you to "view" the latest values of internal algorithm variables while algorithms are executing.
- The Current Value Table is an area in memory that can contain as many as 502 32-bit floating point values. Algorithms can copy any of their variable values into these CVT elements while they execute. The algorithm statements to put data into the CVT are:

  writecvt(<expr>, <element_number>), and
  writeboth(<expr>, <element_number>). See Chapters 3 and 4 for useage.
- Elements 0 through 9 are not accessible. They are used internally by the DSP.
- The format of values returned is set using the FORMat[:DATA] command
- **Returned Value:** ASCII values are returned in the form ±1.234567E±123. For example 13.325 volts would be +1.3325000E+001. Each value is followed by a comma (,). A line feed (LF) and End-Or-Identify (EOI) follow the last value. The C-SCPI data type is a **string array**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

REAL 32, REAL 64, and PACK 64, values are returned in the IEEE-488.2-1987 Definite Length Arbitrary Block Data format. This data return format is explained in "Arbitrary Block Program Data" on page 6-185 of this chapter. For REAL 32, each value is 4 bytes in length (the C-SCPI data type is a **float32 array**). For REAL 64 and PACK 64, each value is 8 bytes in length (the C-SCPI data type is a **float64 array**).

NOTE After *RST/Power-on, each element in the CVT contains the IEEE-754 value "Not-a-number" (NaN). Elements specified in the DATA:CVT? command that have not been written to be an algorithm will return the value 9.91E37.

- ***RST Condition:** All elements of CVT contains IEEE-754 "Not a Number".
- **Related Commands:** SENS:DATA:CVT:RES, FORMAT:DATA

Usage

SENS:DATA:CVT? (@10:13)	*Return algorithm values stored in CVT elements 10 through 13*
DATA:CVT? (@10,13)	*Return only element 10 and element 13*
DATA:CVT? (@330:337,350,360)	*Return algorithm values from elements 330-337, 350, and 360*

[SENSe:]DATA:CVTable:RESet

[SENSe:]DATA:CVTable:RESet sets all 502 Current Value Table entries to the IEEE-754 "Not-a-number".

Comments

- The value of NaN is +9.910000E+037 (ASCII).
- Executing DATA:CVT:RES while the module is INITiated will generate an error 3000, "Illegal while initiated".
- **When Accepted:** Not while INITiated
- **Related Commands:** SENSE:DATA:CVT?
- ***RST Condition:** SENSE:DATA:CVT:RESET

Usage

SENSE:DATA:CVT:RESET	*Clear the Current Value Table*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]DATA:FIFO[:ALL]?

[SENSe:]DATA:FIFO[:ALL]? returns all values remaining in the FIFO buffer until all measurements are complete or until the number of values returned exceeds FIFO buffer size (65,024).

Comments

- DATA:FIFO? may be used to acquire all values (even while they are being made) into a single large buffer, or can be used after one or more DATA:FIFO:HALF? commands to return the remaining values from the FIFO.
- The format of values returned is set using the FORMat[:DATA] command.
- **Returned Value:** ASCII values are returned in the form ±1.234567E±123. For example 13.325 volts would be +1.3325000E+001. Each value is followed by a comma (,). A line feed (LF) and End-Or-Identify (EOI) follow the last value. The C-SCPI data type is a **string array**.

  REAL 32, REAL 64, and PACK 64, values are returned in the IEEE-488.2-1987 Indefinite Length Arbitrary Block Data format. This data return format is explained in "Arbitrary Block Program Data" on page 6-185 of this chapter. For REAL 32, each value is 4 bytes in length (the C-SCPI data type is a **float32 array**). For REAL 64 and PACK 64, each value is 8 bytes in length (the C-SCPI data type is a **float64 array**).

NOTE Algorithm values which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see table on page 6-234 for actual values for each data format).

- **Related Commands:** SENSE:DATA:FIFO:HALF?, FORMAT:DATA
- ***RST Condition:** FIFO is empty

Usage

```
DATA:FIFO?                    return all FIFO values until measurements
                              complete and FIFO empty
```

Command Sequence

```
set up scan lists and trigger

SENSE:DATA:FIFO:ALL?
now execute read statement    read statement does not complete until
                              triggered measurements are complete and
                              FIFO is empty
```

[SENSe:]DATA:FIFO:COUNt?

[SENSe:]DATA:FIFO:COUNt? returns the number of values currently in the FIFO buffer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

- DATA:FIFO:COUNT? is used to determine the number of values to acquire with the DATA:FIFO:PART? command.
- **Returned Value:** Numeric 0 through 65,024. The C-SCPI type is **int32**.
- **Related Commands:** DATA:FIFO:PART?
- ***RST Condition:** FIFO empty

Usage

```
DATA:FIFO:COUNT?          Check the number of values in the FIFO buffer
```

[SENSe:]DATA:FIFO:COUNt:HALF?

[SENSe:]DATA:FIFO:COUNt:HALF? returns a 1 if the FIFO is at least half full (contains at least 32,768 values), or 0 if FIFO is less than half-full.

Comments

- DATA:FIFO:COUNT:HALF? is used as a fast method to poll the FIFO for the half-full condition.
- **Returned Value:** Numeric 1 or 0. The C-SCPI type is **int16**.
- **Related Commands:** DATA:FIFO:HALF?
- ***RST Condition:** FIFO empty

Command Sequence

```
DATA:FIFO:COUNT:HALF?     poll FIFO for half-full status
DATA:FIFO:HALF?           returns 32768 values
```

[SENSe:]DATA:FIFO:HALF?

[SENSe:]DATA:FIFO:HALF? returns 32,768 values if the FIFO buffer is at least half-full. This command provides a fast means of acquiring blocks of values from the buffer.

Comments

- For acquiring data from continuous algorithm executions, your application needs to execute a DATA:FIFO:HALF? command and a read statement often enough to keep up with the rate that values are being sent to the FIFO.
- Use the DATA:FIFO:ALL? command to acquire the values remaining in the FIFO buffer after the ABORT command has stopped execution.
- The format of values returned is set using the FORMat[:DATA] command.
- **Returned Value:** ASCII values are returned in the form ±1.234567E±123. For example 13.325 volts would be +1.3325000E+001. Each value is followed by a

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

comma (,). A line feed (LF) and End-Or-Identify (EOI) follow the last value. The C-SCPI data type is a **string array**.

REAL 32, REAL 64, and PACK 64, values are returned in the IEEE-488.2-1987 Definite Length Arbitrary Block Data format. This data return format is explained in "Arbitrary Block Program Data" on page 6-185 of this chapter. For REAL 32, each value is 4 bytes in length (the C-SCPI data type is a **float32 array**). For REAL 64 and PACK 64, each value is 8 bytes in length (the C-SCPI data type is a **float64 array**).

NOTE Algorithm values which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see table on page 6-234 for actual values for each data format).

- **Related Commands:** DATA:FIFO:COUNT:HALF?
- ***RST Condition:** FIFO buffer is empty

Command Sequence

```
DATA:FIFO:COUNT:HALF?        poll FIFO for half-full status
DATA:FIFO:HALF?              returns 32768 values
```

[SENSe:]DATA:FIFO:MODE

[SENSe:]DATA:FIFO:MODE ***<mode>*** sets the mode of operation for the FIFO buffer.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
mode	discrete (string)	BLOCk | OVERwrite	none

Comments

- In BLOCk(ing) mode, if the FIFO becomes full and measurements are still being made, the new values are discarded.
- OVERwrite mode is used to record the latest 65,024 values. The module must be halted (ABORT sent) before attempting to read the FIFO. In OVERwrite Mode, if the FIFO becomes full and measurements are still being made, new values overwrite the oldest values.
- In both modes Error 3021, "FIFO Overflow" is generated to let you know that measurements have been lost.
- **When Accepted:** Not while INITiated

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Related Commands:** SENSE:DATA:FIFO:MODE?, SENSE:DATA:FIFO:ALL?, SENSE:DATA:FIFO:HALF?, SENSE:DATA:FIFO:PART?, SENSE:DATA:FIFO:COUNT?

- ***RST Condition:** SENSE:DATA:FIFO:MODE BLOCk

Usage

```
SENSE:DATA:FIFO:MODE  OVERWRITE   Set FIFO to overwrite mode
DATA:FIFO:MODE  BLOCK             Set FIFO to block mode
```

[SENSe:]DATA:FIFO:MODE?

[SENSe:]DATA:FIFO:MODE? returns the currently set FIFO mode.

Comments

- **Returned Value:** String value either BLOCK or OVERWRITE. The C-SCPI type is **string**.

- **Related Commands:** SENSE:DATA:FIFO:MODE

Usage

```
DATA:FIFO:MODE?     Enter statement returns either BLOCK or
                    OVERWRITE
```

[SENSe:]DATA:FIFO:PART?

[SENSe:]DATA:FIFO:PART? <***n_values***> returns *n_values* from the FIFO buffer.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
n_values	numeric (int32)	1 - 2,147,483,647	none

Comments

- Use the DATA:FIFO:COUNT? command to determine the number of values in the FIFO buffer.

- The format of values returned is set using the FORMat[:DATA] command.

- **Returned Value:** ASCII values are returned in the form ±1.234567E±123. For example 13.325 volts would be +1.3325000E+001. Each value is followed by a comma (,). A line feed (LF) and End-Or-Identify (EOI) follow the last value. The C-SCPI data type is a **string array**.

  REAL 32, REAL 64, and PACK 64, values are returned in the IEEE-488.2-1987 Definite Length Arbitrary Block Data format. This data return format is explained in "Arbitrary Block Program Data" on page 6-185 of this chapter. For REAL 32, each value is 4 bytes in length (the C-SCPI data type is a **float32 array**). For

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

REAL 64 and PACK 64, each value is 8 bytes in length (the C-SCPI data type is a **float64 array**).

NOTE Algorithm values which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see table on page 6-234 for actual values for each data format).

- **Related Commands:** DATA:FIFO:COUNT?
- ***RST Condition:** FIFO buffer empty

Usage

```
DATA:FIFO:PART? 256          return 256 values from FIFO
```

[SENSe:]DATA:FIFO:RESet

[SENSe:]DATA:FIFO:RESet clears the FIFO of values. The FIFO counter is reset to 0.

Comments

- **When Accepted:** Not while INITiated
- **Related Commands:** SENSE:DATA:FIFO…
- ***RST Condition:** SENSE:DATA:FIFO:RESET

Usage

```
SENSE:DATA:FIFO:RESET        Clear the FIFO
```

[SENSe:]FREQuency:APERture

[SENSe:]FREQuency:APERture <*gate_time*>,<*ch_list*> sets the gate time for frequency measurement. The gate time is the time period that the SCP will allow for counting signal transitions in order to calculate frequency.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
gate_time	numeric (float32)	.001 to 1 (.001 resolution)	seconds
ch_list	string	132 - 163	none

Comments

- If the channels specified are on an SCP that doesn't support this function, an error will be generated. See your SCP's User's Manual for its capabilities.
- Related Commands: SENSe:FUNCtion:FREQuency

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- ***RST Condition:** .001 sec

Usage

```
SENS:FREQ:APER .01,(@148)          set channel 48 aperture to 10msec
```

[SENSe:]FREQuency:APERture?

[SENSe:]FREQuency:APERture? <*ch_list*> returns the frequency counting gate time.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- If the channels specified are on an SCP that doesn't support this function, an error will be generated. See your SCP's User's Manual for its capabilities.
- Related Commands: SENSe:FREQuency:APERture
- **Returned Value:** returns numeric gate time in seconds, The type is **float32**.

[SENSe:]FUNCtion:CONDition

[SENSe:]FUNCtion:CONDition <*ch_list*> sets the SENSe function to input the digital state for channels in <*ch_list*>. Also configures digital SCP channels as inputs (this is the *RST condition for all digital I/O channels).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ch_list	string	132 - 163	none

Comments

- The HP E1533 SCP senses 8 digital bits on each channel specified by this command. The HP E1534 SCP senses 1 digital bit on each channel specified by this command.
- If the channels specified are not on a digital SCP, an error will be generated.
- Use the INPut:POLarity command to set input logical sense.
- **Related Commands:** INPut:POLarity
- ***RST Condition:** SENS:FUNC:COND and INP:POL NORM for all digital SCP channels.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Usage To set first four channels of a HP E1534 in SCP position 6, and second 8-bits of HP E1533 at SCP position 7 to digital inputs send:

```
SENS:FUNC:COND  (@148:151,156)
```

[SENSe:]FUNCtion:CUSTom

[SENSe:]FUNCtion:CUSTom [<*range*>,](@<*ch_list*>) links channels with the custom Engineering Unit Conversion table loaded with the DIAG:CUST:LINEAR or DIAG:CUST:PIECE commands. Contact your Hewlett-Packard System Engineer for more information on Custom Engineering Unit Conversion for your application.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
range	numeric (float32)	see first comment	VDC
ch_list	channel list (string)	100 - 163	none

Comments

- <*range*> parameter: The HP E1419 has five ranges: .0625VDC, .25VDC, 1VDC, 4VDC, and 16VDC. To select a range, simply specify the range value (for example, 4 selects the 4VDC range). If you specify a value larger than one of the first four ranges, the HP E1419 selects the next higher range (for example, 4.1 selects the 16VDC range). Specifying a value larger than 16 causes an error -222 "Data out of range". Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range.

- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1VDC or an input out-of-range condition will exist.

- If an A/D reading is greater than the <*table_range*> specified with DIAG:CUSTOM:PIEC, an overrange condition will occur.

- If no custom table has been loaded for the channels specified with SENS:FUNC:CUST, an error will be generated when an INIT command is given.

- **When Accepted:** Not while INITiated

- **Related Commands:** DIAG:CUST:…

- ***RST Condition:** all custom EU tables erased

Usage

```
program must put table constants into array table_block
DIAG:CUST:LIN 1,table_block,(@116:123)  send table to HP E1419 for chs 16-23
SENS:FUNC:CUST 1,(@116:123)             link custom EU with chs 16-23
INITiate then TRIGger module
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]FUNCtion:CUSTom:REFerence

[SENSe:]FUNCtion:CUSTom:REFerence [<*range*>,](@<*ch_list*>) links channels with the custom Engineering Unit Conversion table loaded with the DIAG:CUST:PIECE command. Measurements from a channel linked with SENS:FUNC:CUST:REF will result in a temperature that is sent to the Reference Temperature Register. This command is used to measure the temperature of an isothermal reference panel using custom characterized RTDs or thermistors. Contact your Hewlett-Packard System Engineer for more information on Custom Engineering Unit Conversion for your application.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
range	numeric (float32)	see comments	VDC
ch_list	channel list (string)	100 - 163	none

Comments

- See "Linking Input Channels to EU Conversion" on page 3-59 for more information.
- The <*range*> parameter: The HP E1419 has five ranges: .0625VDC, .25VDC, 1VDC, 4VDC, and 16VDC. To select a range, simply specify the range value (for example, 4 selects the 4VDC range). If you specify a value larger than one of the first four ranges, the HP E1419 selects the next higher range (for example, 4.1 selects the 16VDC range). Specifying a value larger than 16 generates an error. Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range.
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1VDC or an input out-of-range condition will exist.
- The *CAL? command calibrates temperature channels based on Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.
- **Related Commands:** DIAG:CUST:PIEC, SENS:FUNC:TEMP, SENS:FUNC:CUST:TC, *CAL?
- ***RST Condition:** all custom EU tables erased

Usage

```
program must put table constants into array table_block
DIAG:CUST:PIEC 1,table_block,(@108)     send characterized reference transducer
                                        table for  use by channel 8
SENS:FUNC:CUST:REF .25,(@108)           link custom ref temp EU with ch 8
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

include this channel in a scan list with thermocouple channels (REF channel first)
INITiate then TRIGger module

[SENSe:]FUNCtion:CUSTom:TCouple

[SENSe:]FUNCtion:CUSTom:TCouple <*type*>,[<*range*>,](@<*ch_list*>) links channels with the custom Engineering Unit Conversion table loaded with the DIAG:CUST:PIECE command. The table is assumed to be for a thermocouple and the <*type*> parameter will specify the built-in compensation voltage table to be used for reference junction temperature compensation. SENS:FUNC:CUST:TC allows you to use an EU table that is custom matched to thermocouple wire you have characterized. Contact your Hewlett-Packard System Engineer for more information on Custom Engineering Unit Conversion for your application.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
type	discrete (string)	E | EEXT | J | K | N | R | S | T	none
range	numeric (float32)	see comments	VDC
ch_list	channel list (string)	100 - 163	none

Comments

- See "Linking Input Channels to EU Conversion" on page 3-59 for more information.
- The <*range*> parameter: The HP E1419 has five ranges: .0625VDC, .25VDC, 1VDC, 4VDC, and 16VDC. To select a range, simply specify the range value (for example, 4 selects the 4VDC range). If you specify a value larger than one of the first four ranges, the HP E1419 selects the next higher range (for example, 4.1 selects the 16VDC range). Specifying a value larger than 16 generates an error. Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range.
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1VDC or an input out-of-range condition will exist.
- The *sub_type* EEXTended applies to E type thermocouples at 800°C and above.
- The *CAL? command calibrates temperature channels based on Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.
- **Related Commands:** DIAG:CUST:PIEC, *CAL?,SENS:REF, and SENS:REF:TEMP

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- ***RST Condition:** all custom EU tables erased

Usage

program must put table constants into array table_block

```
DIAG:CUST:PIEC 1,table_block,(@100:107)   send characterized thermocouple table for use by channels 0-7
SENS:FUNC:CUST:TC N,.25,(@100:107)        link custom thermocouple EU with chs 0-7, use reference temperature compensation for N type wire.
SENSE:REF  RTD,92,(@120)                  designate a channel to measure the reference junction temperature
```

include these channels in a scan list (REF channel first)

INITiate then TRIGger module

[SENSe:]FUNCtion:FREQuency

[SENSe:]FUNCtion:FREQuency <*ch_list*> sets the SENSe function to frequency for channels in <*ch_list*>. Also configures the channels specified as digital inputs.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ch_list	string	132 - 163	none

Comments

- If the channels specified are on an SCP that doesn't support this function, an error will be generated. See your SCP's User's Manual for its capabilities.
- Use the SENSe:FREQuency:APERture command to set the gate time for the frequency measurement.
- **Related commands:** SENS:FREQ:APER
- ***RST Condition:** SENS:FUNC:COND and INP:POL NORM for all digital SCP channels

Usage

```
SENS:FUNC:FREQ  (@144)   set channel 44's sense function to frequency
```

[SENSe:]FUNCtion:RESistance

[SENSe:]FUNCtion:RESistance <*excite_current*>,[<*range*>,](@<*ch_list*>) links the EU conversion type for resistance and range with the channels specified by *ch_list*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
excite_current	discrete(string)	30E-6 | 488E-6 | MIN | MAX	Amps
range	numeric (float32)	see first comment	VDC
ch_list	channel list (string)	100 - 163	none

Comments

- The <*range*> parameter: The HP E1419 has five ranges: .0625VDC, .25VDC, 1VDC, 4VDC, and 16VDC. To select a range, simply specify the range value (for example, 4 selects the 4VDC range). If you specify a value larger than one of the first four ranges, the HP E1419 selects the next higher range (for example, 4.1 selects the 16VDC range). Specifying a value larger than 16 causes an error. Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range.
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1VDC or an input out-of-range condition will exist.
- Resistance measurements require the use of Current Source Signal Conditioning Plug-Ons.
- The *excite_current* parameter (excitation current) does not control the current applied to the channel to be measured. The *excite_current* parameter only passes the setting of the SCP supplying current to channel to be measured. The current must have already been set using the OUTPUT:CURRENT:AMPL command. The choices for *excite_current* are 30E-6 (or MIN) and 488E-6 (or MAX). *excite_current* may be specified in milliamps (ma) and microamps (ua).
- The *CAL? command calibrates resistance channels based on Current Source SCP and Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.
- See "Linking Input Channels to EU Conversion" on page 3-59 for more information.
- **When Accepted:** Not while INITiated
- **Related Commands:** OUTP:CURR, *CAL?
- ***RST Condition:** SENSE:FUNC:VOLT (@100:163)

Usage

```
FUNC:RES  30ua,(@100,105,107)
```

Set channels 0, 5, and 7 to convert voltage to resistance assuming current source set to 30 μA use auto-range (default)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]FUNCtion:STRain
:FBENding
:FBPoisson
:FPOisson
:HBENding
:HPOisson
[:QUARter]

Note on Syntax: Although the strain function is comprised of six separate SCPI commands, the only difference between them is the bridge type they specify to the strain EU conversion algorithm.

- **[SENSe:]FUNCtion:STRain:<*bridge_type*> [<*range*>,](@<*ch_list*>)** links the strain EU conversion with the channels specified by *ch_list* to measure the bridge voltage. See "Linking Input Channels to EU Conversion" on page 3-59 for more information.

bridge_type is not a parameter but is part of the command syntax. The following table relates the command syntax to bridge type. See the user's manual for the optional Strain SCP for bridge schematics and field wiring information.

Command	Bridge Type
:FBENding	Full Bending Bridge
:FBPoisson	Full Bending Poisson Bridge
:FPOisson	Full Poisson Bridge
:HBENding	Half Bending Bridge
:HPOisson	Half Poisson Bridge
[:QUARter]	Quarter Bridge (default)

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
range	numeric (flt32)	see comments	VDC
ch_list	channel list (string)	1320 - 163	none

Comments

- Strain measurements require the use of Bridge Completion Signal Conditioning Plug-Ons.

- Bridge Completion SCPs provide the strain measurement bridges and their excitation voltage sources. *ch_list* specifies the voltage sensing channels that are to measure the bridge outputs. Measuring channels on a Bridge Completion SCP only returns that SCP's excitation source voltage.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- The <*range*> parameter: The HP E1419 has five ranges: .0625VDC, .25VDC, 1VDC, 4VDC, and 16VDC. To select a range, simply specify the range value (for example, 4 selects the 4VDC range). If you specify a value larger than one of the first four ranges, the HP E1419 selects the next higher range (for example, 4.1 selects the 16VDC range). Specifying a value larger than 16 generates an error. Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range.

- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1VDC or an input out-of-range condition will exist.

- The channel calibration command (*CAL?) calibrates the excitation voltage source on each Bridge Completion SCP.

- **When Accepted:** Not while INITiated

- **Related Commands:** *CAL?, [SENSE:]STRAIN…

- ***RST Condition:** SENSE:FUNC:VOLT 0,(@100:163)

Usage

```
FUNC:STRAIN  1,(@100:,105,107)        quarter bridge sensed at channels 0, 5, and 7
```

[SENSe:]FUNCtion:TEMPerature

[SENSe:]FUNCtion:TEMPerature <*type*>,<*sub_type*>,[<*range*>,](@<*ch_list*>) links channels to an EU conversion for temperature based on the sensor specified in *type* and *sub_type*. **Not for sensing thermocouple reference temperature (for that, use the SENS:REF <*type*>,<*sub_type*>,(@<*channel*>) command)**.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
type	discrete (string)	RTD | THERmistor | TCouple	none
sub_type	numeric (float32) numeric (float32) discrete (string)	for RTD use 85 | 92 for THER use 2250 | 5000 | 10000 for TC use CUSTom | E | EEXT | J | K | N | R | S | T	none Ohms none
range	numeric (float32)	see comments	VDC
ch_list	channel list (string)	100 - 163	none

Comments

- Resistance temperature measurements (RTDs and THERmistors) require the use of Current Source Signal Conditioning Plug-Ons. The following table shows the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Current Source setting that must be used for the following RTDs and Thermistors:

Required Current Amplitude	Temperature Sensor Types and Subtypes
MAX (488µA) MIN (30µA)	for RTD and THER,2250 for THER,5000 and THER,10000

- The <*range*> parameter: The HP E1419 has five ranges: .0625VDC, .25VDC, 1VDC, 4VDC, and 16VDC. To select a range, simply specify the range value (for example, 4 selects the 4VDC range). If you specify a value larger than one of the first four ranges, the HP E1419 selects the next higher range (for example, 4.1 selects the 16VDC range). Specifying a value larger than 16 generates an error. Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range.

- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1VDC or an input out-of-range condition will exist.

- The ***sub_type* parameter:** values of 85 and 92 differentiate between 100 Ohm (@ 0°C) RTDs with temperature coefficients of 0.00385 and and 0.00392 Ohm/Ohm/°C respectively. The *sub_type* values of 2250, 5000, and 10000 refer to thermistors that match the Omega 44000 series temperature response curve. These 44000 series thermistors are selected to match the curve within 0.1 or 0.2°C. For thermistors sub_type may be specified in Kohms (kohm).

  The *sub_type* EEXTended applies to E type thermocouples at 800°C and above.

  CUSTom is pre-defined as Type K, with no reference junction compensation (reference junction assumed to be at 0 °C).

- The *CAL? command calibrates temperature channels based on Current Source SCP and Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.

- See "Linking Input Channels to EU Conversion" on page 3-59 for more information.

- **When Accepted:** Not while INITiated

- **Related Commands:** *CAL?, OUTP:CURR (for RTDs and Thermistors), SENS:REF, and SENS:REF:TEMP (for Thermocouples)

- ***RST Condition:** SENSE:FUNC:VOLT AUTO,(@100:163)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Usage

Link two channels to the K type thermocouple temperature conversion

```
SENS:FUNC:TEMP  TCOUPLE,K,(@101,102)
```

Link channel 0 to measure reference temperature using 5K thermistor

```
SENS:REF  THER,5000,(@100)
```

[SENSe:]FUNCtion:TOTalize

[SENSe:]FUNCtion:TOTalize <*ch_list*> sets the SENSe function to TOTalize for channels in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ch_list	string	132 - 163	none

Comments

- The totalize function counts rising edges of digital transitions at Frequency/Totalize SCP channels. The counter is 24 bits wide and can count up to 16,777,215.
- The SENS:TOT:RESET:MODE command controls which events will reset the counter.
- If the channels specified are not on a Frequency/Totalize SCP, an error will be generated.
- **Related Commands:** SENS:TOT:RESET:MODE, INPUT:POLARITY
- ***RST Condition:** SENS:FUNC:COND and INP:POL NORM for all digital SCP channels.

Usage

```
SENS:FUNC:TOT  (@148)          channel 48 is a totalizer
```

[SENSe:]FUNCtion:VOLTage[:DC]

[SENSe:]FUNCtion:VOLTage[:DC] [<*range*>,](@<*ch_list*>) links the specified channels to return DC voltage.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
range	numeric (float32)	see comments	VDC
ch_list	channel list (string)	100 - 163	none

Comments

- The <range> parameter: The HP E1419 has five ranges: .0625VDC, .25VDC, 1VDC, 4VDC, and 16VDC. To select a range, simply specify the range value (for example, 4 selects the 4VDC range). If you specify a value larger than one of the first four ranges, the HP E1419 selects the next higher range (for example, 4.1 selects the 16VDC range). Specifying a value larger than 16 causes an error. Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range.
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1VDC or an input out-of-range condition will exist.
- The *CAL? command calibrates channels based on Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.
- See "Linking Input Channels to EU Conversion" on page 3-59 for more information.
- **When Accepted:** Not while INITiated
- **Related Commands:** *CAL?, INPUT:GAIN…
- ***RST Condition:** SENSE:FUNC:VOLT AUTO,(@100:163)

Usage

```
FUNC:VOLT (@140:163)
```

Channels 40 - 63 measure voltage in auto-range (defaulted)

[SENSe:]REFerence

[SENSe:]REFerence <*type*>,<*sub_type*>,[<*range*>,](@<*ch_list*>) links channel in <*ch_list*> to the reference junction temperature EU conversion based on *type* and *sub_type*. When scanned, the resultant value is stored in the Reference Temperature Register, and by default the FIFO and CVT. This is a resistance temperature measurement and uses the on-board 122 µA current source.

NOTE The reference junction temperature value generated by scanning the reference channel is stored in the Reference Temperature Register. This reference temperature

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

is used to compensate all subsequent thermocouple measurements until the register is overwritten by another reference measurement or by specifying a constant reference temperature with the SENSE:REF:TEMP command. If used, the reference junction channel must be scanned before any thermocouple channels. Use the SENSE:REF:CHANNELS command to place the reference measuring channel into the scan list ahead of the thermocouple measuring channels.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
type	discrete (string)	THERmistor | RTD | CUSTom	none
sub_type	numeric (float32) numeric (float32)	for THER use 5000 for RTD use 85 | 92 for CUSTom use 1	Ohm none none
range	numeric (float32)	see comments	VDC
ch_list	channel list (string)	100 - 163	none

Comments

- See "Linking Input Channels to EU Conversion" on page 3-59 for more information.
- The <*range*> parameter: The HP E1419 has five ranges: .0625VDC, .25VDC, 1VDC, 4VDC, and 16VDC. To select a range, simply specify the range value (for example, 4 selects the 4VDC range). If you specify a value larger than one of the first four ranges, the HP E1419 selects the next higher range (for example, 4.1 selects the 16VDC range). Specifying a value larger than 16 causes an error. Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range.
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1VDC or an input out-of-range condition will exist.
- The <*type*> parameter specifies the sensor type that will be used to determine the temperature of the isothermal reference panel. <*type*> CUSTom is pre-defined as Type E with 0°C reference junction temp and is not re-defineable.
- For <*type*> THERmistor, the <*sub_type*> parameter may be specified in ohms or kohm.
- The *CAL? command calibrates resistance channels based on Current Source SCP and Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Related Commands:** SENSE:FUNC:TEMP
- ***RST Condition:** Reference temperature is 0 °C

Usage

sense the reference temperature on channel 20 using an RTD

```
SENSE:REF  RTD,92,(@120)
```

[SENSe:]REFerence:CHANnels

[SENSe:]REFerence:CHANnels (@<*ref_channel*>),(@<*ch_list*>) causes channel specified by <*ref_channel*> to appear in the scan list just before the channel(s) specified by <*ch_list*>. This command is used to include the thermocouple reference temperature channel in the scan list before other thermocouple channels are measured.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ref_channel	channel list (string)	100 - 163	none
ch_list	channel list (string)	100 - 163	none

Comments

- Use SENS:FUNC:TEMP to configure channels to measure thermocouples. Then use SENS:REF to configure one or more channels to measure an isothermal reference temperature. Now use SENS:REF:CHAN to group the reference channel with its thermocouple measurement channels in the scan list.
- If thermocouple measurements are made through more than one isothermal reference panel, you will set up a reference channel for each. Execute the SENS:REF:CHAN command for each reference/measurement channel group.
- **Related commands:** SENS:FUNC:TEMP, SENS:REF
- ***RST Condition:** Scan List contains no channel references.

Usage

```
SENS:FUNC:TEMP  TC,E,.0625,(@108:115)   E type TCs on channels 8 through 15
SENS:REF  THER,5000,1,(@106)            Reference ch is thermistor at channel 6
SENS:REF  RTD,85,.25,(@107)             Reference ch is RTD at channel 7
SENS:REF:CHAN (@106),(@108:111)         Thermistor measured before chs 8 - 11
SENS:REF:CHAN (@107),(@112:115)         RTD measured before chs 12 - 15
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]REFerence:TEMPerature

[SENSe:]REFerence:TEMPerature ***<degrees_c>*** stores a fixed reference junction temperature in the Reference Temperature Register. Use when the thermocouple reference junction is kept at a controlled temperature.

NOTE This reference temperature is used to compensate all subsequent thermocouple measurements until the register is overwritten by another SENSE:REF:TEMP value or by scanning a channel linked with the SENSE:REFERENCE command. If used, SENS:REF:TEMP must be executed before scanning any thermocouple channels.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
degrees_c	numeric (float32)	-126 to +126	none

Comments

- This command is used to specify to the HP E1419 the temperature of a controlled temperature thermocouple reference junction.
- **When Accepted:** Not while INITiated
- **Related Commands:** FUNC:TEMP TC…
- ***RST Condition:** Reference temperature is 0 °C

Usage

```
SENSE:REF:TEMP  40
```

subsequent thermocouple conversion will assume compensation junction at 40 degrees C

[SENSe:]STRain:EXCitation

[SENSe:]STRain:EXCitation <*excite_v*>,(@<*ch_list*>) specifies the excitation voltage value to be used to convert strain bridge readings for the channels specified by <*ch_list*>. This command does not control the output voltage of any source.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
excite_v	numeric (flt32)	.01 - 99	volts
ch_list	channel list (string)	100 - 163	none

Comments

- <*ch_list*> must specify the channel used to sense the bridge voltage, **not** the channel position on a Bridge Completion SCP.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Related Commands:** SENSE:STRAIN:…, SENSE:FUNC:STRAIN…
- ***RST Condition:** 3.9V

Usage

```
STRAIN:EXC 4,(@100:107)          set excitation voltage for channels 0 through 7
```

[SENSe:]STRain:EXCitation?

[SENSe:]STRain:EXCitation? (@<*channel*>) returns the excitation voltage value currently set for the sense channel specified by <*channel*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- **Returned Value:** Numeric value of excitation voltage. The C-SCPI type is **flt32**.
- <*channel*> must specify a single channel only.
- **Related Commands:** STRAIN:EXCitation

Usage

```
STRAIN:EXC? (@107)               query excitation voltage for channel 7
enter statement here             returns the excitation voltage set by STR:EXC
```

[SENSe:]STRain:GFACtor

[SENSe:]STRain:GFACtor <*gage_factor*>,(@<*ch_list*>) specifies the gage factor to be used to convert strain bridge readings for the channels specified by <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
gage_factor	numeric (flt32)	1 - 5	none
ch_list	channel list (string)	100 - 163	none

Comments

- <*ch_list*> must specify the channel used to sense the bridge voltage, **not** the channel position on a Bridge Completion SCP.
- **Related Commands:** SENSE:STRAIN:GFAC?, SENSE:FUNC:STRAIN…
- ***RST Condition:** Gage factor is 2

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Usage STRAIN:GFAC 3,(@100:107) *set gage factor for channels 0 through 7*

[SENSe:]STRain:GFACtor?

[SENSe:]STRain:GFACtor? (@<*channel*>) returns the gage factor currently set for the sense channel specified by <*channel*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- **Returned Value:** Numeric value of gage factor. The C-SCPI type is **flt32**.
- <*channel*> must specify a single channel only.
- **Related Commands:** STRAIN:GFACTOR

Usage STRAIN:GFAC? (@107) *query gage factor for channel 7*
enter statement here *returns the gage factor set by STR:GFAC*

[SENSe:]STRain:POISson

[SENSe:]STRain:POISson <*poisson_ratio*>,(@<*ch_list*>) sets the Poisson ratio to be used for EU conversion of values measured on sense channels specified by <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
poisson_ratio	numeric (flt32)	.1 - .5	none
ch_list	channel list (string)	100 - 163	none

Comments

- <*ch_list*> must specify channels used to sense strain bridge output, **not** channel positions on a Bridge Completion SCP.
- **Related Commands:** FUNC:STRAIN…, STRAIN:POISson?
- ***RST Condition:** Poisson ratio is .3

Usage STRAIN:POISSON .5,(@124:131) *set Poisson ratio for sense channels 24 through 31*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]STRain:POISson?

[SENSe:]STRain:POISson? (@<*channel*>) returns the Poisson ratio currently set for the sense channel specified by <*channel*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- **Returned Value:** numeric value of the Poisson ratio. C-SCPI type is **flt32**.
- <*channel*> must specify a single channel only.
- **Related Commands:** FUNC:STRAIN…, STRAIN:POISSON

Usage

```
STRAIN:POISSON?  (@131)          query for the Poisson ratio specified for
                                 sense channel 31
enter statement here             enter the Poisson ratio value
```

[SENSe:]STRain:UNSTrained

[SENSe:]STRain:UNSTrained <*unstrained_v*>,(@<*ch_list*>) specifies the unstrained voltage value to be used to convert strain bridge readings for the channels specified by <*ch_list*>. This command does not control the output voltage of any source.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
unstrained_v	numeric (flt32)	-16 through +16	volts
ch_list	channel list (string)	100 - 163	none

Comments

- Use a voltage measurement of the unstrained bridge sense channel to determine the correct value for *unstrained_v*.
- <*ch_list*> must specify the channel used to sense the bridge voltage, **not** the channel position on a Bridge Completion SCP.
- **Related Commands:** SENSE:STRAIN:UNST?, SENSE:FUNC:STRAIN…
- ***RST Condition:** Unstrained voltage is zero

Usage

```
STRAIN:UNST .024,(@100)          set unstrained voltage for channel 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]STRain:UNSTrained?

[SENSe:]STRain:UNSTrained? (@<*channel*>) returns the unstrained voltage value currently set for the sense channel specified by <*channel*>. This command does not make a measurement.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- **Returned Value:** Numeric value of unstrained voltage. The C-SCPI type is **flt32**.
- <*channel*> must specify a single channel only.
- **Related Commands:** STRAIN:UNST

Usage

```
STRAIN:UNST? (@107)          query unstrained voltage for channel 7
enter statement here         returns the unstrained voltage set by STR:UNST
```

[SENSe:]TOTalize:RESet:MODE

[SENSe:]TOTalize:RESet:MODE <*select*>,<*ch_list*> sets the mode for resetting totalizer channels in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
select	discrete (string)	INIT | TRIGger	seconds
ch_list	string	132 - 163	none

Comments

- In the INIT mode the total is reset only when the INITiate command is executed. In the TRIGger mode the total is reset every time a new scan is triggered.
- If the channels specified are not on a Frequency/Totalize SCP, an error will be generated.
- **Related Commands:** SENS:FUNC:TOT, INPUT:POLARITY
- ***RST Condition:** SENS:TOT:RESET:MODE INIT

Usage

```
SENS:TOT:RESET:MODE TRIG,(@148)   totalizer at channel 48 resets at each trigger event
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[SENSe:]TOTalize:RESet:MODE?

[SENSe:]TOTalize:RESet:MODE? <*channel*> returns the reset mode for the totalizer channel in <channel>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- *Channel* must specify a single channel.
- If the channel specified is not on a frequency/totalize SCP, an error will be generated.
- **Returned Value:** returns INIT or TRIG. The type is **string**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The SOURce command subsystem allows configuring output SCPs as well as linking channels to output functions.

Subsystem Syntax

```
SOURce
    :FM
        :STATe  1 | 0 | ON | OFF,(@<ch_list>)
        :STATe?  (@<channel>)
    :FUNCtion
        [:SHAPe]
            :CONDition  (@<ch_list>)
            :PULSe  (@<ch_list>)
            :SQUare  (@<ch_list>)
    :PULM
        :STATe  1 | 0 | ON | OFF,(@<ch_list>)
        :STATe?  (@<channel>)
    :PULSe
        :PERiod  <period>,(@<ch_list>)
        :PERiod?  (@<channel>)
        :WIDTh  <pulse_width>,(@<ch_list>)
        :WIDTh?  (@<channel>)
```

SOURce:FM[:STATe]

SOURce:FM[:STATe] <*enable*>,(@<*ch_list*>) enables the Frequency Modulated mode for a PULSe channel.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	boolean (uint16)	1 | 0 | ON | OFF	none
ch_list	string	132 - 163	none

Comments

- This command is coupled with the SOURce:PULM:STATE command. If the FM state is ON then the PULM state is OFF. If the PULM state is ON then the FM state is OFF. If both the FM and the PULM states are OFF then the PULSe channel is in the single pulse mode.
- If the channels specified are not on a Frequency/Totalize SCP, an error will be generated.
- Use SOURce:FUNCtion[:SHAPe]:SQUare to set FM pulse train to 50% duty cycle. Use SOURce:PULSe:PERiod to set the period

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- ***RST Condition:** SOUR:FM:STATE OFF, SOUR:PULM:STATE OFF, SENS:FUNC:COND and INP:POL for all digital SCP channels
- **Related Commands:** SOUR:PULM[:STATe], SOUR:PULS:POLarity, SOUR:PULS:PERiod, SOUR:FUNC[:SHAPe]:SQUare
- The variable frequency control for this channel is provided by the algorithm language. When the algorithm executes an assignment statement to this channel, the value assigned will be the frequency setting. For example:

```
O148 = 2000 /* set channel 48 to 2KHz */
```

SOURce:FM:STATe?

SOURce:FM:STATe? (@<*channel*>) returns the frequency modulated mode state for a PULSe channel.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- *Channel* must specify a single channel.
- If the channel specified is not on a Frequency/Totalize SCP, an error will be generated.
- **Returned Value:** returns 1 (ON) or 0 (OFF). The type is **uint16**.

SOURce:FUNCtion[:SHAPe]:CONDition

SOURce:FUNCtion[:SHAPe]:CONDition (@<*ch_list*>) sets the SOURce function to output digital patterns to bits in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ch_list	string	132 - 163	none

Comments

- The HP E1533 SCP sources 8 digital bits on the channel specified by this command. The HP E1534 SCP can source 1 digital bit on each of the the channels specified by this command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SOURce:FUNCtion[:SHAPe]:PULSe

SOURce:FUNCtion[:SHAPe]:PULSe (@<*ch_list*>) sets the SOURce function to PULSe for the channels in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ch_list	string	132 - 163	none

Comments

- This PULSe channel function is further defined by the SOURce:FM:STATe and SOURce:PULM:STATe commands. If the FM state is enabled then the frequency modulated mode is active. If the PULM state is enabled then the pulse width modulated mode is active. If both the FM and the PULM states are disabled then the PULSe channel is in the single pulse mode.

SOURce:FUNCtion[:SHAPe]:SQUare

SOURce:FUNCtion[:SHAPe]:SQUare (@<*ch_list*>) sets the SOURce function to output a square wave (50% duty cycle) on the channels in <ch_list>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ch_list	string	132 - 163	none

Comments

- The frequency control for these channels is provided by the algorithm language function:.

```
O149 = 2000 /* set channel 49 to 2KHz */
```

SOURce:PULM[:STATe]

SOURce:PULM[:STATe] <*enable*>,(@<*ch_list*>) enable the pulse width modulated mode for the PULSe channels in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	boolean (uint16)	1 | 0 | ON | OFF	none
ch_list	string	132 - 163	none

Comments

- This command is coupled with the SOURce:FM command. If the FM state is enabled then the PULM state is disabled. If the PULM state is enabled then the FM state is disabled. If both the FM and the PULM states are disabled then the PULSe channel is in the single pulse mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- If the channels specified are not on a Frequency/Totalize SCP, an error will be generated.
- ***RST Condition:** SOUR:PULM:STATE OFF

SOURce:PULM:STATe?

SOURce:PULM[:STATe]? (@<*channel*>) returns the pulse width modulated mode state for the PULSe channel in <*channel*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

Channel must specify a single channel.

- **Returned Value:** returns ON or OFF. The type is **string**.

SOURce:PULSe:PERiod

SOURce:PULSe:PERiod <*period*>,(@<*ch_list*>) sets the fixed pulse period value on a pulse width modulated pulse channel. This sets the frequency (1/period) of the pulse-width-modulated pulse train.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
period	numeric (float32)	25E-6 to 7.8125E-3 (resolution 0.238µsec)	seconds
ch_list	string	132 - 163	none

Comments

- If the channels specified are not on a Frequency/Totalize SCP, an error will be generated.
- ***RST Condition:** SOUR:FM:STATE OFF and SOUR:PULM:STATE OFF
- **Related Commands:** SOUR:PULM:STATE, SOUR:PULS:POLarity
- The variable pulse-width control for this channel is provided by the algorithm language. When the algorithm executes an assignment statement to this channel, the value assigned will be the pulse-width setting. For example:

```
O150 = .0025 /* set channel 50 pulse-width to 2.5 msec */
```

Usage

```
SOUR:PULS:PER .005,(@148)        set PWM pulse train to 200 Hz on chn 48
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SOURce:PULSe:PERiod?

SOURce:PULSe:PERiod? (@<*channel*>) returns the fixed pulse period value on the pulse width modulated pulse channel in <*channel*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- If the channels specified are not on a Frequency/Totalize SCP, an error will be generated.
- **Returned Value:** numeric period. The type is **float32**.

SOURce:PULSe:WIDTh

SOURce:PULSe:WIDTh <*pulse_width*>,(@<*ch_list*>) sets the fixed pulse width value on the frequency modulated pulse channels in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
pulse_width	numeric (float32)	7.87E-6 to 7.8125E-3 (238.4E-9 resolution)	seconds
ch_list	string	132 - 163	none

Comments

- If the channels specified are not on a Frequency/Totalize SCP, an error will be generated.
- ***RST Condition:** SOUR:FM:STATE OFF and SOUR:PULM:STATE OFF
- **Related Commands:** SOUR:PULM:STATE, SOUR:PULS:POLarity
- The variable frequency control for this channel is provided by the algorithm language. When the algorithm executes an assignment statement to this channel, the value assigned will be the frequency setting. For example:

```
O149 = 2000 /* set channel 49 to 2KHz */
```

Usage

```
SOUR:PULS:WIDTH  2.50E-3,(@149)
```

set fixed pulse width of 2.5 msec on channel 49

SOURce:PULSe:WIDTh?

SOURce:PULSe:WIDTh? (@<*ch_list*>) returns the fixed pulse width value on a frequency modulated pulse channel.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	string	132 - 163	none

Comments

- *Channel* must specify a single channel.
- If the channels specified are not on a Frequency/Totalize SCP, an error will be generated.
- **Returned Value:** returns the numeric pulse width. The type is **float32**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The STATus subsystem communicates with the SCPI defined Operation and Questionable Data status register sets. Each is comprised of a Condition register, a set of Positive and Negative Transition Filter registers, an Event register, and an Enable register. Condition registers allow you to view the current real-time states of their status signal inputs (signal states are not latched). The Positive and Negative Transition Filter registers allow you to control the polarity of change from the Condition registers that will set Event register bits. Event registers contain latched representations of signal transition events from their Condition register. Querying an Event register reads and then clears its contents, making it ready to record further event transitions from its Condition register. Enable registers are used to select which signals from an Event register will be logically ORed together to form a summary bit in the Status Byte Summary register. Setting a bit to one in an Enable register enables the corresponding bit from its Event register.

NOTE For a complete discussion and more detailed status illustration see "Using the Status System" on page 3-90.

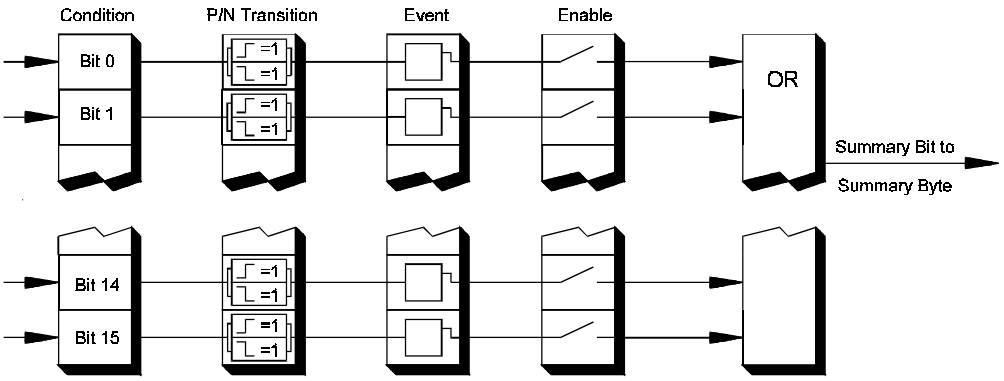


Figure 6-4 General Status Register Organization

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Initializing the Status System

The following table shows the effect of Power-on, *RST, *CLS and STATus:PRESet on the status system register settings.

	SCPI Transition Filters	SCPI Enable Registers	SCPI Event Registers	IEEE 488.2 Registers ESE and SRE	IEEE 488.2 Registers SESR and STB
Power-on	preset	preset	clear	clear	clear
***RST**	none	none	none	none	none
***CLS**	none	none	clear	none	clear
STAT:PRESET	preset	preset	none	none	none

Subsystem Syntax

```
STATus
    :OPERation
        :CONDition?
        :ENABle  <enable_mask>
        :ENABle?
        [:EVENt]?
        :NTRansition <transition_mask>
        :NTRansition?
        :PTRansition <transition_mask>
        :PTRansition?
    :PRESet
    :QUEStionable
        :CONDition?
        :ENABle  <enable_mask>
        :ENABle?
        [:EVENt]?
        :NTRansition <transition_mask>
        :NTRansition?
        :PTRansition <transition_mask>
        :PTRansition?
```

The Status system contains four status groups

- Operation Status Group
- Questionable Data Group
- Standard Event Group
- Status Byte Group

This SCPI STATus subsystem communicates with the first two groups while IEEE-488.2 Common Commands (documented later in this chapter) communicate with Standard Event and Status Byte Groups.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Weighted Bit Values Register queries are returned using decimal weighted bit values. Enable registers can be set using decimal, hex, octal, or binary. The following table can be used to help set Enable registers using decimal, and decode register queries.

Status System Decimal Weighted Bit Values

bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value	always 0	16,384	8,192	4,096	2,048	1,024	512	256	128	64	32	16	8	4	2	1

The Operation Status Group

The Operation Status Group indicates the current operating state of the HP E1419. The bit assignments are:

Bit #	dec value	hex value	Bit Name	Description
0	1	0001_{16}	Calibrating	Set by CAL:TARE, and CAL:SETup. Cleared by CAL:TARE?, and CAL:SETup?. Set while *CAL? executes and reset when *CAL? completes. Set by CAL:CONFIG:VOLT or CAL:CONFIG:RES, cleared by CAL:VAL:VOLT or CAL:VAL:RES.
1-3				Not used
4	16	0010_{16}	Measuring	Set when instrument INITiated. Cleared when instrument returns to Trigger Idle State.
5-7				Not used
8	256	0100_{16}	Scan Complete	Set when each pass through a Scan List completed (may not indicate all measurements have been taken when TRIG:COUNT >1).
9	512	0200_{16}	SCP Trigger	An SCP has sourced a trigger event (future HP 1419 SCPs)
10	1024	0400_{16}	FIFO Half Full	The FIFO contains at least 32,768 readings
11	2048	0800_{16}	Algorithm Interrupted	The *interrupt()* function was called in an algorithm
12-15				Not used

STATus:OPERation:CONDition?

STATus:OPERation:CONDition? returns the decimal weighted value of the bits set in the Condition register.

Comments

- The Condition register reflects the real-time state of the status signals. The signals are not latched; therfore past events are not retained in this register (see STAT:OPER:EVENT?).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** *CAL?, CAL:ZERO, INITiate[:IMMediate], STAT:OPER:EVENT?, STAT:OPER:ENABLE, STAT:OPER:ENABLE?
- ***RST Condition:** No Change

Usage STATUS:OPERATION:CONDITION? *Enter statement will return value from condition register*

STATus:OPERation:ENABle

STATus:OPERation:ENABle ***<enable_mask>*** sets bits in the Enable register that will enable corresponding bits from the Event register to set the Operation summary bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable_mask	numeric (uint16)	0-32767	none

Comments

- Enable_mask may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- VXI Interrupts: When Operation Status Group bits 4, 8, 9, 10, or 11 are enabled, VXI card interrupts will occur as follows:

  When the event corresponding to bit 4 occurs and then is cleared, the card will generate a VXI interrupt. When the event corresponding to bit 8, 9, 10, or 11 occurs, the card will generate a VXI interrupt.

  NOTE: In C-SCPI, the C-SCPI overlap mode must be on for VXIbus interrupts to occur.
- **Related Commands:** *STB?, SPOLL, STAT:OPER:COND?, STAT:OPER:EVENT?, STAT:OPER:ENABLE?
- Cleared By: STAT:PRESet and power-on.
- ***RST Condition:** No change

Usage STAT:OPER:ENABLE 1 *Set bit 0 in the Operation Enable register*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STATus:OPERation:ENABle?

STATus:OPERation:ENABle? returns the value of bits set in the Operation Enable register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** *STB?, SPOLL, STAT:OPER:COND?, STAT:OPER:EVENT?, STAT:OPER:ENABLE
- ***RST Condition:** No change

Usage

STAT:OPER:ENABLE?	*Enter statement returns current value of bits set in the Operation Enable register*

STATus:OPERation[:EVENt]?

STATus:OPERation[:EVENt]? returns the decimal weighted value of the bits set in the Event register.

Comments

- When using the Operation Event register to cause SRQ interrupts, STAT:OPER:EVENT? must be executed after an SRQ to clear the event register and re-enable future interrupts.
- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** *STB?, SPOLL, STAT:OPER:COND?, STAT:OPER:ENABLE, STAT:OPER:ENABLE?
- Cleared By: *CLS, power-on, and by reading the register.
- ***RST Condition:** No change

Usage

STAT:OPER:EVENT?	*Enter statement will return the value of bits set in the Operation Event register*
STAT:OPER?	*Same as above*

STATus:OPERation:NTRansition

STATus:OPERation:NTRansition ***<transition_mask>*** sets bits in the Negative Transition Filter (NTF) register. When a bit in the NTF register is set to one, the corresponding bit in the Condition register must change from a one to a zero in order to set the corresponding bit in the Event register. When a bit in the NTF register is

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

zero, a negative transition of the Condition register bit will not change the Event register bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
transition_mask	numeric (uint16)	0-32767	none

Comments

- *transition_mask* may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- If both the STAT:OPER:PTR and STAT:OPER:NTR registers have a corresponding bit set to one, any transition, positive or negative will set the corresponding bit in the Event register.
- If neither the STAT:OPER:PTR or STAT:OPER:NTR registers have a corresponding bit set to one, transitions from the Condition register will have no effect on the Event register.
- **Related Commands:** STAT:OPER:NTR?, STAT:OPER:PTR
- Cleared By: STAT:PRESet and power-on.
- ***RST Condition:** No change

Usage

```
STAT:OPER:NTR  16
```

When "Measuring" bit goes false, set bit 4 in Status Operation Event register.

STATus:OPERation:NTRansition?

STATus:OPERation:NTRansition? returns the value of bits set in the Negative Transition Filter (NTF) register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** STAT:OPER:NTR
- ***RST Condition:** No change

Usage

```
STAT:OPER:NTR?
```

Enter statement returns current value of bits set in the NTF register

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STATus:OPERation:PTRansition

STATus:OPERation:PTRansition <***transition_mask***> sets bits in the Positive Transition Filter (PTF) register. When a bit in the PTF register is set to one, the corresponding bit in the Condition register must change from a zero to a one in order to set the corresponding bit in the Event register. When a bit in the PTF register is zero, a positive transition of the Condition register bit will not change the Event register bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
transition_mask	numeric (uint16)	0-32767	none

Comments

- *transition_mask* may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- If both the STAT:OPER:PTR and STAT:OPER:NTR registers have a corresponding bit set to one, any transition, positive or negative will set the corresponding bit in the Event register.
- If neither the STAT:OPER:PTR or STAT:OPER:NTR registers have a corresponding bit set to one, transitions from the Condition register will have no effect on the Event register.
- **Related Commands:** STAT:OPER:PTR?, STAT:OPER:NTR
- Set to all ones by: STAT:PRESet and power-on.
- ***RST Condition:** No change

Usage

```
STAT:OPER:PTR  16
```

When "Measuring" bit goes true, set bit 4 in Status Operation Event register.

STATus:OPERation:PTRansition?

STATus:OPERation:PTRansition? returns the value of bits set in the Positive Transition Filter (PTF) register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** STAT:OPER:PTR
- ***RST Condition:** No change

Usage

```
STAT:OPER:PTR?
```

Enter statement returns current value of bits set in the PTF register

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STATus:PRESet

STATus:PRESet sets the Operation Status Enable and Questionable Data Enable registers to 0. After executing this command, none of the events in the Operation Event or Questionable Event registers will be reported as a summary bit in either the Status Byte Group or Standard Event Status Group. STATus:PRESet does not clear either of the Event registers.

Comments

- **Related Commands:** *STB?, SPOLL, STAT:OPER:ENABLE, STAT:OPER:ENABLE?, STAT:QUES:ENABLE, STAT:QUES:ENABLE?
- ***RST Condition:** No change

Usage

```
STAT:PRESET                    Clear both of the Enable registers
```

The Questionable Data Group

The Questionable Data Group indicates when errors are causing lost or questionable data. The bit assignments are:

Bit #	dec value	hex value	Bit Name	Description
0-7				Not used
8	256	0100_{16}	Calibration Lost	At *RST or Power-on Control Processor has found a checksum error in the Calibration Constants. Read error(s) with SYST:ERR? and re-calibrate area(s) that lost constants.
9	512	0200_{16}	Trigger Too Fast	Scan not complete when another trigger event received.
10	1024	0400_{16}	FIFO Overflowed	Attempt to store more than 65,024 readings in FIFO.
11	2048	0800_{16}	Over voltage Detected on Input	If the input protection jumper has not been cut, the input relays have been opened and *RST is required to reset the module. Overvoltage will also generate an error.
12	4096	1000_{16}	VME Memory Overflow	The number of readings taken exceeds VME memory space.
13	8192	2000_{16}	Setup Changed	Channel Calibration in doubt because SCP setup may have changed since last *CAL? or CAL:SETup command. (*RST always sets this bit).
14-15				Not used

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STATus:QUEStionable:CONDition?

STATus:QUEStionable:CONDition? returns the decimal weighted value of the bits set in the Condition register.

Comments

- The Condition register reflects the real-time state of the status signals. The signals are not latched; therfore past events are not retained in this register (see STAT:QUES:EVENT?).
- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** CAL:VALUE:RESISTANCE, CAL:VALUE:VOLTAGE, STAT:QUES:EVENT?, STAT:QUES:ENABLE, STAT:QUES:ENABLE?
- ***RST Condition:** No change

Usage

```
STATUS:QUESTIONABLE:CONDITION?    Enter statement will return value from condition register
```

STATus:QUEStionable:ENABle

STATus:QUEStionable:ENABle **<*enable_mask*>** sets bits in the Enable register that will enable corresponding bits from the Event register to set the Questionable summary bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable_mask	numeric (uint16)	0-32767	none

Comments

- *Enable_mask* may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- VXI Interrupts: When bits 9, 10, or 11 are enabled and C-SCPI overlap mode is on (or if you are using non-compiled SCPI), VXI card interrupts will be enabled. When the event corresponding to bit 9, 10, or 11 occurs, the card will generate a VXI interrupt.
- **Related Commands:** *STB?, SPOLL, STAT:QUES:COND?, STAT:QUES:EVENT?, STAT:QUES:ENABLE?
- Cleared By: STAT:PRESet and power-on.
- ***RST Condition:** No change

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Usage STAT:QUES:ENABLE 128 *Set bit 7 in the Questionable Enable register*

STATus:QUEStionable:ENABle?

STATus:QUEStionable:ENABle? returns the value of bits set in the Questionable Enable register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** *STB?, SPOLL, STAT:QUES:COND?, STAT:QUES:EVENT?, STAT:QUES:ENABLE
- ***RST Condition:** No change

Usage STAT:QUES:ENABLE? *Enter statement returns current value of bits set in the Questionable Enable register*

STATus:QUEStionable[:EVENt]?

STATus:QUEStionable[:EVENt]? returns the decimal weighted value of the bits set in the Event register.

Comments

- When using the Questionable Event register to cause SRQ interrupts, STAT:QUES:EVENT? must be executed after an SRQ to clear the register and re-enable future interrupts.
- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- Cleared By: *CLS, power-on, and by reading the register.
- **Related Commands:** *STB?, SPOLL, STAT:QUES:COND?, STAT:QUES:ENABLE, STAT:QUES:ENABLE?

Usage STAT:QUES:EVENT? *Enter statement will return the value of bits set in the Questionable Event register*

STAT:QUES? *Same as above*

STATus:QUEStionable:NTRansition

STATus:QUEStionable:NTRansition <***transition_mask***> sets bits in the Negative Transition Filter (NTF) register. When a bit in the NTF register is set to one, the corresponding bit in the Condition register must change from a one to a zero in order to set the corresponding bit in the Event register. When a bit in the NTF

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

register is zero, a negative transition of the Condition register bit will not change the Event register bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
transition_mask	numeric (uint16)	0-32767	none

Comments

- *transition_mask* may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- If both the STAT:QUES:PTR and STAT:QUES:NTR registers have a corresponding bit set to one, any transition, positive or negative will set the corresponding bit in the Event register.
- If neither the STAT:QUES:PTR or STAT:QUES:NTR registers have a corresponding bit set to one, transitions from the Condition register will have no effect on the Event register.
- **Related Commands:** STAT:QUES:NTR?, STAT:QUES:PTR
- Cleared By: STAT:PRESet and power-on.
- ***RST Condition:** No change

Usage

```
STAT:QUES:NTR  1024
```

When "FIFO Overflowed" bit goes false, set bit 10 in Status Questionable Event register.

STATus:QUEStionable:NTRansition?

STATus:QUEStionable:NTRansition? returns the value of bits set in the Negative Transition Filter (NTF) register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** STAT:QUES:NTR
- ***RST Condition:** No change

Usage

```
STAT:QUES:NTR?
```

Enter statement returns current value of bits set in the NTF register

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STATus:QUEStionable:PTRansition

STATus:QUEStionable:PTRansition <***transition_mask***> sets bits in the Positive Transition Filter (PTF) register. When a bit in the PTF register is set to one, the corresponding bit in the Condition register must change from a zero to a one in order to set the corresponding bit in the Event register. When a bit in the PTF register is zero, a positive transition of the Condition register bit will not change the Event register bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
transition_mask	numeric (uint16)	0-32767	none

Comments

- *transition_mask* may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- If both the STAT:QUES:PTR and STAT:QUES:NTR registers have a corresponding bit set to one, any transition, positive or negative will set the corresponding bit in the Event register.
- If neither the STAT:QUES:PTR or STAT:QUES:NTR registers have a corresponding bit set to one, transitions from the Condition register will have no effect on the Event register.
- **Related Commands:** STAT:QUES:PTR?, STAT:QUES:NTR
- Set to all ones by: STAT:PRESet and power-on.
- ***RST Condition:** No change

Usage

```
STAT:QUES:PTR  1024
```

When "FIFO Overflowed" bit goes true, set bit 10 in Status Operation Event register.

STATus:QUEStionable:PTRansition?

STATus:QUEStionable:PTRansition? returns the value of bits set in the Positive Transition Filter (PTF) register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type is **uint16**.
- **Related Commands:** STAT:QUES:PTR
- ***RST Condition:** No change

Usage

```
STAT:OPER:PTR?
```

Enter statement returns current value of bits set in the PTF register

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The SYSTem subsystem is used to query for error messages, types of Signal Conditioning Plug-ons (SCPs), and the SCPI version currently implemented.

Subsystem Syntax

```
SYSTem
    :CTYPe?  (@<channel>)
    :ERRor?
    :VERSion?
```

SYSTem:CTYPe?

SYSTem:CTYPe? (@<*channel*>) returns the identification of the Signal Conditioning Plug-On installed at the specified channel.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- *channel* must specify a single channel only.
- **Returned Value:** An example of the response string format is:
  HEWLETT-PACKARD,E1419 Option <*option number and description*> SCP,0,0

  The C-SCPI type is **string**. For specific response string, refer to the appropriate SCP manual. If <*channel*> specifies a position where no SCP is installed, the module returns the response string:
  0,No SCP at this Address,0,0

Usage

```
SYST:CTYPE?  (@100)          return SCP type install at channel 0
```

SYSTem:ERRor?

SYSTem:ERRor? returns the latest error entered into the Error Queue.

Comments

- SYST:ERR? returns one error message from the Error Queue (returned error is removed from queue). To return all errors in the queue, repeatedly execute SYST:ERR? until the error message string = +0, "No error"
- **Returned Value:** Errors are returned in the form:
  ±<error number>, "<*error message string*>"
- RST Condition: Error Queue is empty.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Usage SYST:ERR? *returns the next error message from the Error Queue*

SYSTem:VERSion?

SYSTem:VERSion? returns the version of SCPI this instrument complies with.

Comments

- **Returned Value:** String "1990". The C-SCPI type is **string**.

Usage SYST:VER? *Returns "1990"*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The TRIGger command subsystem controls the behavior of the trigger system once it is initiated (see INITiate command subsystem).

Figure 6-5 shows the overall Trigger System model. The shaded area shows the ARM subsystem portion.

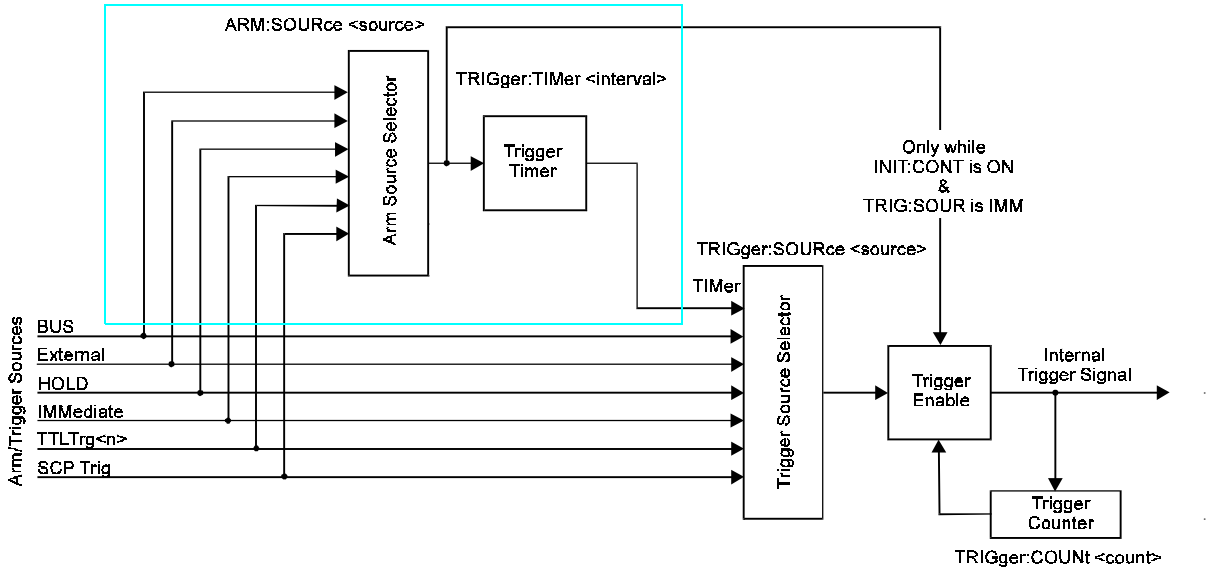


Figure 6-5 Logical Trigger Model

CAUTION!

- Algorithms execute at most once per trigger event. Should trigger events cease (external trigger source stops) or are ignored (TRIGger:COUNt reached), algorithms execution will stop. In this case control outputs are left at the last value set by the algorithms. Depending on the process, this uncontrolled situation could even be dangerous. Make certain that you have put your process into a safe state before you halt (stop triggering) execution of a controlling algorithm.

- The HP E1535 Watchdog Timer SCP was specifically developed to automatically signal that an algorithm has stopped controlling a process. Use of the Watchdog Timer is recommended for critical processes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Event Sequence Figure 6-6 shows how the module responds to various trigger/arm configurations.

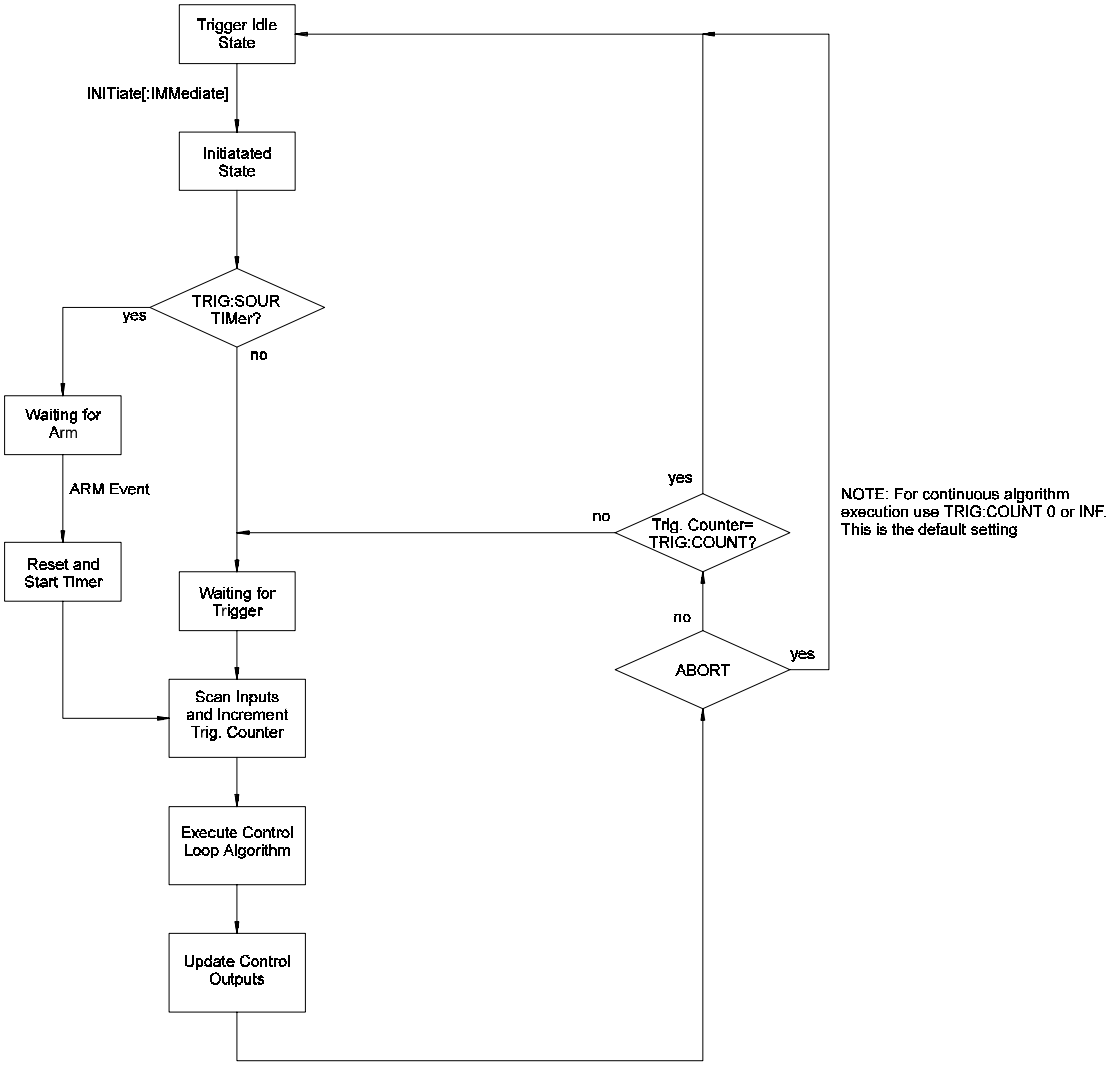


Figure 6-6 Trigger/Scan Sequence Diagram

Subsystem Syntax

```
TRIGger
    :COUNt <trig_count>
    :COUNt?
    [:IMMediate]
    :SOURce BUS | EXTernal | HOLD | SCP | IMMediate | TIMer | TTLTrg<n>
    :SOURce?
    :TIMer
        [:PERiod] <trig_interval>
        [:PERiod]?
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TRIGger:COUNt

TRIGger:COUNt <***trig_count***> sets the number of times the module can be triggered before it returns to the Trigger Idle State. The default count is 0 (same as INF) so accepts continuous triggers. See Figure 6-6

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
Trig_count	numeric (uint16) (string)	0 to 65535 | INF	none

Comments

- When *trig_count* is set to 0 or INF, the trigger counter is disabled. Once INITiated the module will return to the Waiting For Trigger State after each trigger event. The ABORT (preferred) and *RST commands will return the module to the Trigger Idle State. ABORT is preferred since *RST also returns other module configurations to their default settings.
- The default count is 0
- **Related Commands:** TRIG:COUNT?
- ***RST Condition:** TRIG:COUNT 0

Usage

```
TRIG:COUNT  10          Set the module to make 10 passes all enabled algorithms.
TRIG:COUNT  0           Set the module to accept unlimited triggers (the default)
```

TRIGger:COUNt?

TRIGger:COUNt? returns the currently set trigger count.

Comments

- If TRIG:COUNT? returns 0, the trigger counter is disabled and the module will accept an unlimited number of trigger events.
- **Returned Value:** Numeric 0 through 65,535. The C-SCPI type is **int32**.
- **Related Commands:** TRIG:COUNT
- ***RST Condition:** TRIG:COUNT? returns 0

Usage

```
TRIG:COUNT?             Query for trigger count setting
enter statement         Returns the TRIG:COUNT setting
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TRIGger[:IMMediate]

TRIGger[:IMMediate] causes one trigger when the module is set to the TRIG:SOUR BUS or TRIG:SOUR HOLD mode.

Comments

- This command is equivalent to the *TRG common command or the IEEE-488.2 "GET" bus command.
- **Related Commands:** TRIG:SOURCE

Usage TRIG:IMM *Use TRIGGER to start a measurement scan*

TRIGger:SOURce

TRIGger:SOURce ***<trig_source>*** configures the trigger system to respond to the trigger event.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
trig_source	discrete (string)	BUS | EXT | HOLD | IMM | SCP | TIM | TTLTrg<n>	none

Comments

- The following table explains the possible choices.

Parameter Value	Source of Trigger
BUS	TRIGger[:IMMediate], *TRG, GET (for HP-IB)
EXTernal	"TRG" signal on terminal module
HOLD	TRIGger[:IMMediate]
IMMediate	The trigger event is always satisfied.
SCP	SCP Trigger Bus (future HP or SCP Breadboard)
TIMer	The internal trigger timer
TTLTrg<n>	The VXIbus TTLTRG lines (n=0 through 7)

NOTE The ARM system only exists while TRIG:SOUR is TIMer. When TRIG:SOUR is not TIMer, SCPI compatibility requires that ARM:SOUR be IMM or an Error -221,"Settings conflict" will be generated.

- While TRIG:SOUR is IMM, you need only INITiate the trigger system to start a measurement scan.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- While TRIG:SOUR IMM provides the fastest trigger repetition rate, the trigger occurrence time is not deterministic. In addition, there is no means to synchronize the start of algorithm execution with an external input since when TRIG:SOUR is IMM, ARM:SOUR must also be set to IMM. The TRIG:SOUR TIMER provides both a deterministic occurrence of algorithm executions, and the ability to synchronize this with an external signal (ARM:SOUR EXT).

- **When Accepted: Before INIT only**.

- **Related Commands:** ABORt, INITiate, *TRG

- ***RST Condition:** TRIG:SOUR TIMER

Usage

```
TRIG:SOUR  EXT
```

Hardware trigger input at Connector Module

TRIGger:SOURce?

TRIGger:SOURce? returns the current trigger source configuration.

- **Returned Value:** Discrete; one of BUS, EXT, HOLD, IMM, SCP, TIM, or TTLT0 through TTLT7. The C-SCPI type is **string**. See the TRIG:SOUR command for more response data information.

Usage

```
TRIG:SOUR?
```

ask HP E1419 to return trigger source configuration

TRIGger:TIMer[:PERiod]

TRIGger:TIMer[:PERiod] <***trig_interval***> sets the interval between scan triggers. Used with the TRIG:SOUR TIMER trigger mode.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
trig_interval	numeric (float32) (string)	100E-6 to 6.5536 | MIN | MAX	seconds

Comments

- In order for the TRIG:TIMER to start it must be Armed. For information on timer arming see the ARM subsystem in this command reference.

- The default interval is 10E-3 seconds. ***interval*** may be specified in seconds, milliseconds (ms), or microseconds (us). For example; .0016, 1.6ms or 1600us. The resolution for *interval* is 100 µ second.

- **When Accepted: Before INIT only**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Related Commands:** TRIG:SOUR TIMER, ARM:SOUR, ARM:IMM, INIT, TRIG:SOUR?, ALG:EXPL:TIME?
- ***RST Condition:** TRIG:TIM 1.0E-3

Usage

TRIG:TIMER 1.0E-1	*Set the module to scan inputs and execute all algorithms every 100 mS*
TRIG:TIMER 1	*Set the module to scan inputs and execute all algorithms every second*

TRIGger:TIMer[:PERiod]?

TRIGger:TIMer[:PERiod]? returns the currently set Trigger Timer interval.

Comments

- **Returned Value:** Numeric 1 through 6.5536. The C-SCPI type is **float32**.
- **Related Commands:** TRIG:TIMER
- ***RST Condition:** 1.0E-4

Usage

TRIG:TIMER?	*Query trig timer*
enter statement	*Returns the timer setting*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following reference discusses the HP E1419 IEEE-488.2 Common commands.

*CAL?

Calibration command. The calibration command causes the Channel Calibration function to be performed for every module channel. The Channel Calibration function includes calibration of A/D Offset, and Gain and Offset for all 64 channels. This calibration is accomplished using internal calibration references. The *CAL? command causes the module to calibrate A/D offset and gain, and all channel offsets. This may take many minutes to complete. The actual time it will take your HP E1419 to complete *CAL? depends on the mix of SCPs installed. *CAL performs literally hundreds of measurements of the internal calibration sources for each channel and must allow 17 time constants of settling wait each time a filtered channel's calibrations source value is changed. The *CAL procedure is internally very sophisticated and results in an extremely well calibrated module.

To perform Channel Calibration on multiple HP E1419s, use CAL:SETup.

- **Returned Value:**

Value	Meaning	Further Action
0	Cal OK	None
-1	Cal Error	Query the Error Queue (SYST:ERR?) See Error Messages in Appendix B

The C-SCPI type for this returned value is **int16**.

- **When Accepted:** Not while INITiated
- **Related Commands:** CALibration:SETup, CALibration:SETup?, CALibration:STORe ADC
- CAL:STOR ADC stores the calibration constants for *CAL? and CAL:SETup into non-volatile memory.
- Executing this command **does not** alter the module's programmed state (function, range, etc.). It does however clear STAT:QUES:COND? register bit 13.

NOTE If Open Transducer Detect (OTD) is enabled when *CAL? is executed, the module will disable OTD, wait 1 minute to allow channels to settle, perform the calibration, and then re-enable OTD. If your program turns off OTD before executing *CAL?, it should also wait 1 minute for settling.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*CLS

Clear Status Command. The *CLS command clears all status event registers (Standard Event Status Event Register, Standard Operation Status Event Register, Questionable Data Event Register) and the instrument's error queue. This clears the corresponding summary bits (bits 3, 5, & 7) in the Status Byte Register. *CLS does not affect the enable bits in any of the status register groups. (The SCPI command STATus:PRESet *does* clear the Operation Status Enable and Questionable Data Enable registers.) *CLS disables the Operation Complete function (*OPC command) and the Operation Complete Query function (*OPC? command).

*DMC <*name*>,<*cmd_data*>

Define Macro Command. Assigns one, or a sequence of commands to a named macro.

The command sequence may be composed of SCPI and/or Common commands.

<*name*> may be the same as a SCPI command, but may not be the same as a Common command. When a SCPI named macro is executed, the macro rather than the SCPI command is executed. To regain the function of the SCPI command, execute *EMC 0 command.

<*cmd_data*> is sent as *arbitrary block program data* (see page 6-185).

*EMC

Enable Macro Command. When <enable> is non-zero, macros are enabled. When <*enable*> is zero, macros are disabled.

*EMC?

Enable Macro query. Returns either 1 (macros are enabled), or 0 (macros are disabled).

*ESE <*mask*>

Standard Event Status Enable Register Command. Enables one or more events in the Standard Event Status Register to be reported in bit 5 (the Standard Event Status Summary Bit) of the Status Byte Register. You enable an event by specifying its decimal weight for <*mask*>. To enable more than one event (bit), specify the sum of the decimal weights. The C-SCPI type for <*mask*> is **int16**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bit #	7	6	5	4	3	2	1	0
Weighted Value	128	64	32	16	8	4	2	1
Event	power-On	User Request	Command Error	Execution Error	Device Dependent Error	Query Error	Request Control	Operation Complete

*ESE?

Standard Event Status Enable Query. Returns the weighted sum of all enabled (unmasked) bits in the Standard Event Status Register. The C-SCPI type for this returned value is **int16**.

*ESR?

Standard Event Status Register Query. Returns the weighted sum of all set bits in the Standard Event Status Register. After reading the register, *ESR? clears the register. The events recorded in the Standard Event Status Register are independent of whether or not those events are enabled with the *ESE command to set the Standard Event Summary Bit in the Status Byte Register. The Standard Event bits are described in the *ESE command. The C-SCPI type for this returned value is **int16**.

*GMC? <*name*>

Get Macro query. Returns arbitrary block response data which contains the command or command sequence defined for <*name*>. For more information on arbitrary block response data see page 6-185.

*IDN?

Identity. Returns the device identity. The response consists of the following four fields (fields are separated by commas):

- Manufacturer
- Model Number
- Serial Number (returns 0 if not available)
- Driver Revision (returns 0 if not available)

*IDN? returns the following response strings depending on model and options:
HEWLETT-PACKARD,E1419B,<*serial number*>,<*revision number*>

- The C-SCPI type for this returned value is **string**.

NOTE The revision will vary with the revision of the driver software installed in your system. This is the only indication of which version of the driver is installed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*LMC?

Learn Macros query. Returns a quoted string name for each currently defined macro. If more than one macro is defined, the strings are separated by commas (,). If no macro is defined, *LMC? returns a null string.

*OPC

Operation Complete. Causes an instrument to set bit 0 (Operation Complete Message) in the Standard Event Status Register when all pending operations invoked by SCPI commands have been completed. By enabling this bit to be reflected in the Status Byte Register (*ESE 1 command), you can ensure synchronization between the instrument and an external computer or between multiple instruments.

NOTE Do not use *OPC to determine when the CAL:SETUP or CAL:TARE commands have completed. Instead, use their query forms CAL:SETUP? or CAL:TARE?.

*OPC?

Operation Complete Query. Causes an instrument to place a 1 into the instrument's output queue when all pending instrument operations invoked by SCPI commands are finished. By requiring your computer to read this response before continuing program execution, you can ensure synchronization between one or more instruments and the computer. The C-SCPI type for this returned value is **int16**.

NOTE Do not use *OPC? to determine when the CAL:SETUP or CAL:TARE commands have completed. Instead, use their query forms CAL:SETUP? or CAL:TARE?.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*PMC

Purge Macros Command. Purges all currently defined macros.

*RMC <*name*>

Remove individual Macro Command. Removes the named macro command.

*RST

Reset Command. Resets the HP E1419 as follows:

- Erases all algorithms
- All elements in the Input Channel Buffer (I100 - I163) set to zero.
- All elements in the Output Channel Buffer (O100-O163) set to zero
- Defines all Analog Input channels to measure voltage
- Configures all Digital I/O channels as inputs
- Resets HP E1531 and HP E1532 Analog Output SCP channels to zero

- **When Accepted:** Not while INITiated

WARNING Note the change in character of output channels when *RST is received. Digital outputs change to inputs (appearing now is 1kW to +3v, a TTL one), and analog control outputs change to zero (current or voltage). Keep these changes in mind when applying the HP E1419 to your system, or engineering a system for operation with the HP E1419. Also note that each analog output channels disconnects for 5-6 milliseconds to discharge to zero at each *RST.

It isn't difficult to have the HP E1419 signal your system when *RST is executed. A solution that can provide signals for several types of failures as well as signaling when *RST is executed is the HP E1535 Watchdog Timer SCP. The Watchdog SCP even has an input through which you can command all of the HP E1419's channels to disconnect from your system.

- Sets the trigger system as follows:
  - TRIGGER:SOURCE TIMER
  - TRIGGER:TIMER 10E-3
  - TRIGGER:COUNT 0 (infinite)
  - ARM:SOURCE IMMEDIATE
- SAMPLE:TIMER 10E-6
- Aborts all pending operations, returns to Trigger Idle state
- Disables the *OPC and *OPC? modes
- MEMORY:VME:ADDRESS 240000; MEMORY:VME:STATE OFF; MEMORY:VME:SIZE 0
- Sets STAT:QUES:COND? bit 13

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*RST does not affect:

- Calibration data
- The output queue
- The Service Request Enable (SRE) register
- The Event Status Enable (ESE) register

*SRE <*mask*>

Service Request Enable. When a service request event occurs, it sets a corresponding bit in the Status Byte Register (this happens whether or not the event has been enabled (unmasked) by *SRE). The *SRE command allows you to identify which of these events will assert an HP-IB service request (SRQ). When an event is enabled by *SRE and that event occurs, it sets a bit in the Status Byte Register and issues an SRQ to the computer (sets the HP-IB SRQ line true). You enable an event by specifying its decimal weight for <*mask*>. To enable more than one event, specify the sum of the decimal weights. Refer to "The Status Byte Register"r for a table showing the contents of the Status Byte Register. The C-SCPI type for <*mask*> is **int16**.

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Weighted Value	128	64	32	16	8	4	2	1
Event	Operation Status	Request Service	Standard Event	Message Available	Questionable Status	not used	not used	not used

*SRE?

Status Register Enable Query. Returns the weighted sum of all enabled (unmasked) events (those enabled to assert SRQ) in the Status Byte Register. The C-SCPI type for this returned value is **int16**.

*STB?

Status Byte Register Query. Returns the weighted sum of all set bits in the Status Byte Register. Refer to the *ESE command earlier in this chapter for a table showing the contents of the Status Byte Register. *STB? does not clear bit 6 (Service Request). The Message Available bit (bit 4) may be cleared as a result of reading the response to *STB?. The C-SCPI type for this returned value is **int16**.

*TRG

Trigger. Triggers an instrument when the trigger source is set to bus (TRIG:SOUR BUS command) and the instrument is in the Wait for Trigger state.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*TST?

Self-Test. Causes an instrument to execute extensive internal self-tests and returns a response showing the results of the self-test.

NOTES

1. During the first 5 minutes after power is applied, *TST? may fail. Allow the module to warm-up before executing *TST?.
2. Module must be screwed securely to mainframe.
3. The HP E1419 C-SCPI driver for MS-DOS® implements two versions of *TST. The default version is an abbreviated self test that executes only the Digital Tests. By loading an additional object file, you can execute the full self test as described below. See the documentation that comes with the HP E1419 C-SCPI driver for MS-DOS®.

Comments

- **Returned Value:**

Value	Meaning	Further Action
0	*TST? OK	None
-1	*TST? Error	Query the Error Queue (SYST:ERR?) for error 3052. See explanation below.

- IF error 3052 'Self test failed. Test info in FIFO' is returned. A FIFO value of 1 through 99 or >=300 is a failed test number. A value of 100 through 163 is a channel number for the failed test. A value of 200 through 204 is an A/D range number for the failed test where 200=.0625, 201=.25V, 202=1V, 203=4V, and 204=16V ranges. For example DATA:FIFO? returns the values 72 and 108. This indicates that test number 72 failed on channel 8.

  Test numbers 20, 30-37, 72, 74-76, and 80-93 may indicate a problem with a Signal Conditioning Plug-on.

  For tests 20, and 30-37, remove all SCPs and see if *TST? passes. If so, replace SCPs one at a time until you find the one causing the problem.

  For tests 72, 74-76, and 80-93, try to re-seat the SCP that the channel number(s) points to, or move the SCP and see if the failure(s) follow the SCP. If the problems move with the SCP, replace the SCP.

  These are the only tests where the user should troubleshoot a problem. Other tests which fail should be referred to qualified repair personnel.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE Executing *TST? returns the module to its *RST state. *RST causes the FIFO data format to return to its default of ASC,7. If you want to read the FIFO for *TST? diagnostic information and you want that data in other than the ASCII,7 format, be certain to set the data FIFO format to the desired format (FORMAT command) after completion of *TST? but before executing a SENSE:DATA:FIFO: query command.

- The C-SCPI type for this returned value is **int16**.
- Following *TST?, the module is placed in the *RST state. This returns many of the module's programmed states to their defaults. See page 3-53 for a list of the module's default states.
- *TST? performs the following tests on the HP E1419 and installed Signal Conditioning Plug-ons:

DIGITAL TESTS:

Test#	Description
1-3:	Writes and reads patterns to registers via A16 & A24
4-5:	Checks FIFO and CVT
6:	Checks measurement complete (Measuring) status bit
7:	Checks operation of FIFO half and FIFO full IRQ generation
8-9:	Checks trigger operation

ANALOG FRONT END DIGITAL TESTS:

Test#	Description
20:	Checks that SCP ID makes sense
30-32:	Checks relay driver and fet mux interface with EU CPU
33,71:	Checks opening of all relays on power down or input overvoltage
34-37:	Check fet mux interface with A/D digital

ANALOG TESTS:

Test#	Description
40-42:	Checks internal voltage reference

ANALOG TESTS: (continued)

Test#	Description
43-44:	Checks zero of A/D, internal cal source and relay drives
45-46:	Checks fine offset calibration DAC
47-48:	Checks coarse offset calibration DAC
49:	Checks internal + and -15V supplies

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

50-53: Checks internal calibration source
54-55: Checks gain calibration DAC
56-57: Checks that autorange works
58-59: Checks internal current source
60-63: Checks front end and A/D noise and A/D filter
64: Checks zeroing of coarse and fine offset calibration DACs
65-70: Checks current source and CAL BUS relay and relay drives and OHM relay drive
71: See 33
72-73: Checks continuity through SCPs, bank relays and relay drivers
74: Checks open transducer detect
75: Checks current leakage of the SCPs
76: Checks voltage offset of the SCPs
80: Checks mid-scale strain dac output. Only reports first channel of SCP.
81: Checks range of strain dac. Only reports first channel of SCP.
82: Checks noise of strain dac. Only reports first channel of SCP.
83: Checks bridge completion leg resistance each channel.
84: Checks combined leg resistance each channel.
86: Checks current source SCP's OFF current.
87: Checks current source SCP's current dac mid-scale.
88: Checks current source SCP's current dac range on HI and LO ranges.
89: Checks current source compliance
90: Checks strain SCP's Wagner Voltage control.
91: Checks autobalance dac range with input shorted.
92: Sample and Hold channel holds value even when input value changed.
93: Sample and Hold channel held value test for droop rate.

ANALOG OUTPUT AND DIGITAL I/O TESTS

301: Current and Voltage Output SCPs digital DAC control.
302: Current and Voltage Output SCPs DAC noise.

303: Current Output SCP offset
304: Current Output SCP gain shift
305: Current Output SCP offset
306: Current Output SCP linearity
307: Current Output SCP linearity
308: Current Output SCP turn over

313: Voltage Output SCP offset
315: Voltage Output SCP offset
316: Voltage Output SCP linearity
317: Voltage Output SCP linearity
318: Voltage Output SCP turn over

331: Digital I/O SCP internal digital interface
332: Digital I/O SCP user input
333: Digital I/O SCP user input

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

334:	Digital I/O SCP	user output
335:	Digital I/O SCP	user output
336:	Digital I/O SCP	output current
337:	Digital I/O SCP	output current
341:	Freq/PWM/FM SCP	internal data0 register
342:	Freq/PWM/FM SCP	internal data1 register
343:	Freq/PWM/FM SCP	internal parameter register
344:	Freq/PWM/FM SCP	on-board processor self-test
345:	Freq/PWM/FM SCP	on-board processor self-test
346:	Freq/PWM/FM SCP	user inputs
347:	Freq/PWM/FM SCP	user outputs
348:	Freq/PWM/FM SCP	outputs ACTive/PASSive
349:	Freq/PWM/FM SCP	output interrupts
350:	Watchdog SCP	enable/disable timer
351:	Watchdog SCP	relay drive and coil closed
352:	Watchdog SCP	relay drive and coil open
353:	Watchdog SCP	I/O Disconnect line
354:	Watchdog SCP	I/O Disconnect supply

*WAI

Wait-to-continue. Prevents an instrument from executing another command until the operation begun by the previous command is finished (sequential operation).

NOTE Do not use *WAI to determine when the CAL:SETUP or CAL:TARE commands have completed. Instead, use their query forms CAL:SETUP? or CAL:TARE?. CAL:SETUP? and CAL:TARE? return a value only after the CAL:SETUP or CAL:TARE operations are complete.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following tables summarize SCPI and IEEE-488.2 Common (*) commands for the HP E1419A Multifunction*Plus*.

SCPI Command Quick Reference

Command	Description
ABORt	Stops scanning immediately and sets trigger system to idle state (scan lists are unaffected)
ALGorithm	Subsystem to define, configure, and enable loop control algorithms
[:EXPLicit]	
:ARRay <*alg_name*>,<*array_name*>,<*block_data*>	Defines contents of array <*array_name*> in algorithm <*alg_name*> or if <*alg_name*> is "GLOBALS", defines values global to all algorithms.
:ARRay? <*alg_name*>,<*array_name*>	Returns block data from <*array_name*> in algorithm <*alg_name*> or if <*alg_name*> is "GLOBALS", returns values from a global array.
:DEFine <*alg_name*>[,<*swap_size*>],<*program_data*>	Defines algorithms or global variables. <program_data> is 'C' source of algorithm or global declaration.
:SCALar <*alg_name*>,<*var_name*>,<*value*>	Defines value of variable <*var_name*> in algorithm <*alg_name*> or if <*alg_name*> is "GLOBALS", defines a value global to all algorithms.
:SCALar? <*alg_name*>,<var_name>	Returns value from <*var_name*> in algorithm <*alg_name*> or if <*alg_name*> is "GLOBALS", returns a value from global variable.
:SCAN	
:RATio <*alg_name*>,<*ratio*>	Sets scan triggers per execution of <*alg_name*> (send also ALG:UPD)
:RATio? <*alg_name*>	Returns scan triggers per execution of <*alg_name*>
:SIZe? <*alg_name*>	Returns size in words of named algorithm
:STATe <*alg_name*>,ON | OFF	Enables/disables named algorithm after ALG:UPDATE sent
:STATe? <*alg_name*>	Returns state of named algorithm
:TIME? <*alg_name*> | MAIN	Returns worst case alg execution time. Use "MAIN" for overall time.
:FUNCtion	
:DEFine <*function_name*>,<*range*>,<*offset*>,<*func_data*>	Defines a custom conversion function
:OUTPut	
:DELay <*delay*> | AUTO	Sets the delay from scan trigger to start of outputs
:DELay?	Returns the delay from scan trigger to start of outputs
:UPDate	
[:IMMediate]	Requests immediate update of algorithm code, variable, or array
:CHANnel (@<*channel*>	Sets dig channel to synch algorithm updates
:WINDow <*num_updates*>	Sets a window for *num_updates* to occur. *RST default is 20
:WINDow?	Returns setting for allowable number variable and algorithm updates.
ARM	
[:IMMediate]	Arm if ARM:SOUR is BUS or HOLD (software ARM)
:SOURce BUS | EXT | HOLD | IMM | SCP | TTLTrg<n>	Specify the source of Trigger Timer ARM
:SOURce?	Return current ARM source
CALibration	
:CONFigure	Prepare to measure on-board references with an external multimeter
:RESistance	Configure to measure reference resistor
:VOLTage <*range*>, ZERO | FSCale	Configure to measure reference voltage range at zero or full scale
:SETup	Performs Channel Calibration procedure
:SETup?	Returns state of CAL:SETup operation (returns error codes or 0 for OK)
:STORe ADC | TARE	Store cal constants to Flash RAM for either A/D calibration or those generated by the CAL:TARE command

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SCPI Command Quick Reference

Command	Description
CALibration (cont.)	
:TARE (@<*ch_list*>)	Calibrate out system field wiring offsets
:RESet	Resets cal constants from CAL:TARE back to zero for all channels
:TARE?	Returns state of CAL:TARE operation (returns error codes or 0 for OK)
:VALue	
:RESistance <*ref_ohms*>	Send to instrument the value of just measured reference resistor
:VOLTage <*ref_volts*>	Send to instrument the value of just measured voltage reference
:ZERO?	Correct A/D for short term offset drift (returns error codes or 0 for OK)
DIAGnostic	
:CALibration	
:SETup	
[:MODE] 0 | 1	Set analog DAC output SCP calibration mode
[:MODE]?	Return current setting of DAC calibration mode
:TARe	
[:OTD]	
[:MODE] 0 | 1	Set mode to control OTD current during tare calibration
[:MODE]?	Return current setting of OTD control during tare calibration
:CHECksum?	Perform checksum on Flash RAM and return a '1' for OK, a '0' for corrupted or deleted memory contents
:COMMand	
:SCPWRITE <*reg_addr*>,<*reg_data*>	Writes values to SCP registers
:CUSTom	
:LINear <*table_ad_range*>,<*table_block*>,(@<*ch_list*>)	Loads linear custom EU table
:PIECewise <*table_ad_range*>,<*table_block*>,(@<*ch_list*>)	Loads piecewise custom EU table
:REFerence:TEMPerature	Puts the contents of the Reference Temperature Register into the FIFO
:FLOor <*range*>,(@<*ch_list*>)	Sets the lowest range that autorange can select for the specified channels
:DUMP	Places the autorange floor value for all 64 channels into the FIFO
:INTerrupt[:LINe] <*intr_line*>	Sets the VXIbus interrupt line the module will use
:INTerrupt[:LINe]?	Returns the VXIbus interrupt line the module is using
:OTDetect[:STATe] ON | OFF, (@<*ch_list*>)	Controls "Open Transducer Detect" on SCPs contained in <*ch_list*>
:OTDetect[:STATe]? (@<*channel*>)	Returns current state of OTD on SCP containing <*channel*>
:QUERy	
:SCPREAD? <*reg_addr*>	Returns value from an SCP register
:VERSion?	Returns manufacturer, model, serial#, flash revision #, and date e.g. HEWLETT-PACKARD,E1419B,US34000478,A.04.00, Wed Jul 08 11:06:22 MDT 1994
FETCh?	Return readings stored in VME Memory (format set by FORM cmd)
FORMat	
[:DATA] <*format*>[, <*size*>]	Set format for response data from [SENSe:]DATA?
ASCii[, 7]	Seven bit ASCII format (not as fast as 32-bit because of conversion)
PACKed[, 64]	Same as REAL, 64 except NaN, +INF, and -INF format compatible with HP BASIC
REAL[, 32]	IEEE 32-bit floating point (requires no conversion so is fastest)
REAL, 64	IEEE 64-bit floating point (not as fast as 32-bit because of conversion)
[:DATA]?	Returns format: REAL, +32 | REAL, +64 | PACK, +64 | ASC, +7
INITiate	
[:IMMediate]	Put module in Waiting for Trigger state (ready to make one scan)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SCPI Command Quick Reference

Command	Description
INPut	
:FILTer	Control filter Signal Conditioning Plug-ons
[:LPASs]	
:FREQuency <*cutoff_freq*>,(@<*ch_list*>)	Sets the cutoff frequency for active filter SCPs
:FREQuency? (@<*channel*>)	Returns the cutoff frequency for the channel specified
[:STATe] ON | OFF, (@<*channel*>)	Turn filtering OFF (pass through) or ON (filter)
[:STATe]? (@<*channel*>)	Return state of SCP filters
:GAIN <*chan_gain*>,(@<*ch_list*>)	Set gain for amplifier-per-channel SCP
:GAIN? (@<*channel*>)	Returns the channel's gain setting
:LOW <*wvolt_type*>,(@<*ch_list*>)	Controls the connection of input LO on a Strain Bridge (Opt. 21 SCP)
:LOW? (@<*channel*>)	Returns the LO connection for the Strain Bridge at *channel*
:POLarity NORmal | INVerted,(@<*ch_list*>)	Sets input polarity on a digital SCP channel
:POLarity? (@<*channel*>)	Returns digital polarity currently set for <*channel*
MEMory	
:VME	
:ADDRess <*mem_address*>	Specify address of VME memory card to be used as reading storage
:ADDRess?	Returns address of VME memory card
:SIZE <*mem_size*>	Specify number of bytes of VME memory to be used to store readings
:SIZE?	Returns number of VME memory bytes allocate to reading storage
:STATe 1 | 0 | ON | OFF	Enable or disable reading storage in VME memory at INIT
:STATe?	Returns state of VME memory, 1=enabled, 0=disabled
OUTPut	
:CURRent	
:AMPLitude <*amplitude*>,(@<*ch_list*>)	Set amplitude of Current Source SCP channels
:AMPLitude? (@<*channel*>)	Returns the setting of the Current Source SCP channel
:STATe ON | OFF,(@<*ch_list*>)	Enable or disable the Current Source SCP channels
:STATe? (@<*channel*>)	Returns the state of the Current Source SCP channel
:POLarity NORmal | INVerted,(@<*ch_list*>)	Sets output polarity on a digital SCP channel
:POLarity? (@<*channel*>)	Returns digital polarity currently set for <*channel*>
:SHUNt ON | OFF,(@<*ch_list*>)	Adds shunt resistance to leg of Bridge Completion SCP channels
:SHUNt? (@<*channel*>)	Returns the state of the shunt resistor on Bridge Completion SCP channel
:TTLTrg	
:SOURce FTRigger | LIMit | SCPlugon | TRIGger	Sets the internal trigger source that can drive the VXIbus TTLTrg lines
:SOURce?	Returns the source of TTLTrg drive.
:TTLTrg<n>	
[:STATe] ON | OFF	When module triggered, source a VXIbus trigger on TTLTrg<n>
[:STATe]?	Returns whether the TTL trigger line specified by n is enabled
:TYPE PASSive | ACTive,(@<*ch_list*>)	sets the output drive type for a digital channel
:TYPE? (@<*channel*>)	Returns the output drive type for <*channel*>
:VOLTage	
:AMPLitude <*amplitude*>,(@<*ch_list*>)	Sets the voltage amplitude on Voltage Output and Strain SCPs
:AMPLitude? (@<*channel*>)	Returns the voltage amplitude setting
ROUTe	
:SEQuence	
:DEFine? AIN | AOUT | DIN | DOUT	Returns comma separated list of channels in analog I, O, dig I, O ch lists
:POINts? AIN | AOUT | DIN | DOUT	Returns number of channels defined in above lists.
SAMPle	
:TIMer <*num_samples*>,(@<*ch_list*>)	Sets number of samples that will be made on channels in <*ch_list*>
:TIMer? (@<*channel*>)	Returns number of samples that will be made on channels in <*ch_list*>
[SENSe:]	
CHANnel	
:SETTling <settle_time>,(@<*ch_list*>)	Sets the channel settling time for channels in *ch_list*
:SETTling? (@<*channel*>)	Returns the channel settling time for *channel*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SCPI Command Quick Reference

Command	Description
DATA	
:CVTable? (@<*ch_list*>)	Returns elements of Current Value Table specified by *ch_list*
:RESet	Resets all entries in the Current Value Table to IEEE "Not-a-number"
:FIFO	
[:ALL]?	Fetch all readings until instrument returns to trigger idle state
:COUNt?	Returns the number of measurements in the FIFO buffer
:HALF?	Returns 1 if at least 32,768 readings are in FIFO, else returns 0
:HALF?	Fetch 32,768 readings (half the FIFO) when available
:MODE BLOCK | OVERwrite	Set FIFO mode.
:MODE?	Return the currently set FIFO mode
:PART? <*n_readings*>	Fetch *n_readings* from FIFO reading buffer when available
:RESet	Reset the FIFO counter to 0
FREQuency	
:APERture <*gate_time*>,(@<*ch_list*>)	Sets the gate time for frequency counting
:APERture? (@<*channel*>)	Returns the gate time set for frequency counting
FUNCtion	Equate a function and range with groups of channels
:CONDition (@<*ch_list*>)	Sets function to sense digital state
:CUSTom [<*range*>,](@<*ch_list*>)	Links channels to custom EU conversion table loaded by DIAG:CUST:LIN or DIAG:CUST:PIEC commands
:REFerence [<*range*>,](@<*ch_list*>)	Links channels to custom reference temperature EU conversion table loaded by DIAG:CUST:PIEC commands
:TC <*type*>,[<*range*>,](@<*ch_list*>)	Links channels to custom temperature EU conversion table loaded by DIAG:CUST:PIEC, and performs ref temp compensation for <*type*>
:FREQuency (@<*ch_list*>)	Configure channels to measure frequency
:RESistance <*excite_current*>,[<*range*>,](@<*ch_list*>)	Configure channels to sense resistance measurements
:STRain	Links measurement channels as having read bridge voltage from:
:FBENding [<*range*>,](@<*ch_list*>)	Full BENding
:FBPoisson [<*range*>,](@<*ch_list*>)	Full Bending Poisson
:FPOisson [<*range*>,](@<*ch_list*>)	Full POisson
:HBENding [<*range*>,](@<*ch_list*>)	Half BENding
:HPOisson [<*range*>,](@<*ch_list*>)	Half Poisson
[:QUARter] [<*range*>,](@<*ch_list*>)	QUARter
RTD, 85 | 92 TCouple, CUST | E | EEXT | J | K | N | S | T THERmistor, 2250 | 5000 | 10000	RTDs thermocouples thermistors
:TEMPerature <*sensor_type*>,<*sub_type*>, [<*range*>,](@<*ch_list*>)	Configure channels for temperature measurement types above: excitation current comes from Current Output SCP.
:TOTalize (@<*ch_list*>)	Configure channels to count digital state transitions
:VOLTage[:DC] [<*range*>,](@<*ch_list*>)	Configure channels for DC voltage measurement
RTD, 85 | 92 THERmistor,5000	RTDs thermistors
:REFerence <*sensor_type*>,<*sub_type*>,[<*range*>,](@<*ch_list*>)	Configure channel for reference temperature measurements above:
:CHANnels (@<*ref_channel*>),(@<*ch_list*>)	Groups reference temperature channel with TC measurement channels
:TEMPerature <*degrees_c*>	Specifies the temperature of a controlled temperature reference junction
:STRain	
:EXCitation <*excite_v*>,(@<*ch_list*>)	Specifies the Excitation Voltage by channel to the strain EU conversion
:STRain	
:EXCitation <*excite_v*>,(@<*ch_list*>)	Specifies the Excitation Voltage by channel to the strain EU conversion
:EXCitation? (@<*channel*>)	Returns the Excitation Voltage set for <*channel*>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SCPI Command Quick Reference

Command	Description
:GFACtor <*gage_factor*>,(@<*ch_list*>)	Specifies the Gage Factor by channel to the strain EU conversion
:GFACtor? (@<*channel*>)	Returns the Gage Factor set for <*channel*>
:POISson <*poisson_ratio*>,(@<*ch_list*>)	Specifies the Poisson Ratio by channel to the strain EU conversion
[SENSe:]STRAin (continued)	
:POISson? (@<*channel*>)	Returns the Poisson Ratio set for <*channel*>
:UNSTrained <*unstrained_v*>,(@<*ch_list*>)	Specifies the Unstrained Voltage by channel to the strain EU conversion
:UNSTrained? (@<*channel*>)	Returns the Unstrained Voltage set for <*channel*>
SOURce	
:FM	
[:STATe] 1 | 0 | ON | OFF,(@<*ch_list*>)	Configure digital channels to output frequency modulated signal
[:STATe]? (@<*channel*>)	Returns state of channels for FM output
:FUNCtion	
[:SHAPe]	
:CONDition (@<*ch_list*>)	Configures channels to output static digital levels
:PULSe (@<*ch_list*>)	Configures channels to output digital pulse(s)
:SQUare (@<*ch_list*>)	Configures channels to output 50/50 duty cycle digital pulse train
:PULM	
:STATe 1 | 0 | ON | OFF,(@<*ch_list*>)	Configure digital channels to output pulse width modulated signal
:STATe? (@<*channel*>)	Returns state of channels for PW modulated output
:PERiod <*period*>,(@<*ch_list*>)	Sets pulse period for PW modulated signals
:PERiod? ,(@<*channel*>)	Returns pulse period for PW modulated signals
:WIDTh <*width*>,(@<*ch_list*>)	Sets pulse width for FM modulated signals
:WIDTh? (@<*channel*>)	Returns pulse width setting for FM modulated signals
STATus	
:OPERation	Operation Status Group: Bit assignments; 0=Calibrating, 4=Measuring, 8=Scan Complete, 10=FIFO Half Full, 11=algorithm interrupt
:CONDition?	Returns state of Operation Status signals
:ENABle <*enable_mask*>	Bits set to 1 enable status events to be summarized into Status Byte
:ENABle?	Returns the decimal weighted sum of bits set in the Enable register
[:EVENt]?	Returns weighted sum of bits that represent Operation status events
:NTRansition <*transition_mask*>	Sets mask bits to enable pos. Condeition Reg. transitions to Event reg
:NTRansition?	Returns positive transition mask value
:PTRansition <*transition_mask*>	Sets mask bits to enable neg. Condeition Reg. transitions to Event reg
:PTRansition?	Returns negative transition mask value
:PRESet	Presets both the Operation and Questionable Enable registers to 0
:QUEStionable	Questionable Data Status Group: Bit assignments; 8=Calibration Lost, 9=Trigger Too Fast, 10=FIFO Overflowed, 11=Over voltage, 12=VME Memory Overflow, 13=Setup Changed.
:CONDition?	Returns state of Questionable Status signals
:ENABle <*enable_mask*>	Bits set to 1 enable status events to be summarized into Status Byte
:ENABle?	Returns the decimal weighted sum of bits set in the Enable register
[:EVENt]?	Returns weighted sum of bits that represent Questionable Data events
:NTRansition <*transition_mask*>	Sets mask bits to enable pos. Condeition Reg. transitions to Event reg
:NTRansition?	Returns positive transition mask value
:PTRansition <*transition_mask*>	Sets mask bits to enable neg. Condeition Reg. transitions to Event reg
:PTRansition?	Returns negative transition mask value
SYSTem	
:CTYPe? (@<*channel*>)	Returns the identification of the SCP at *channel*
:ERRor?	Returns one element of the error queue "0" if no errors
:VERSion?	Returns the version of SCPI this instrument complies with

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SCPI Command Quick Reference

Command	Description
TRIGger	
:COUNt <*trig_count*>	Specify the number of trigger events that will be accepted
:COUNt?	Returns the current trigger count setting
[:IMMediate]	Triggers instrument when TRIG:SOUR is TIMer or HOLD (same as *TRG and IEEE 488.1 GET commands.
:SOURce BUS|EXT|HOLD|IMM|SCP|TIMer|TTLTrg<n>	Specify the source of instrument triggers
:SOURce?	Returns the current trigger source
:TIMer	Sets the interval between scan triggers when TRIG:SOUR is TIMer
[:PERiod] <*trig_interval*>	Sets the interval between scan triggers when TRIG:SOUR is TIMer
[:PERiod]?	Returns setting of trigger timer

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IEEE-488.2 Common Command Quick Reference			
Category	Command	Title	Description
Calibration	*CAL?	Calibrate	Performs internal calibration on all 64 channels out to the terminal module connector. Returns error codes or 0 for OK
Internal Operation	*IDN?	Identification	Returns the response: HEWLETT-PACKARD,E1419B,<*serial#*>,<*driver rev#*>
	*RST	Reset	Resets all scan lists to zero length and stops scan triggering. Status registers and output queue are unchanged.
	*TST?	Self Test	Performs self test. Returns 0 to indicate test passed.
Status Reporting	*CLS	Clear Status	Clears all status event registers and so their status summary bits (except the MAV bit).
	*ESE <*mask*>	Event Status Enable	Set Standard Event Status Enable register bits mask.
	*ESE?	Event Status Enable query	Return current setting of Standard Event Status Enable register.
	*ESR?	Event Status Register query	Return Standard Event Status Register contents.
	*SRE <*mask*>	Service Request Enable	Set Service Request Enable register bit mask.
	*SRE?	Service Request Enable query	Return current setting of the Service Request Enable register.
	*STB?	Read Status Byte query	Return current Status Byte value.
Macros	*DMC <*name*>,<*cmd_data*>	Define Macro Command	Assigns one, or a sequence of commands to a macro.
	*EMC 1 | 0	Enable Macro Command	Enable/Disable defined macro commands.
	*EMC?	Enable Macros query	Returns 1 for macros enabled, 0 for disabled.
	*GMC? <*name*>	Get Macro query	Returns command sequence for named macro.
	*LMC?	Learn Macro query	Returns comma-separated list of defined macro names
	*PMC	Purge Macro Commands	Purges all macro commands
	*RMC <*name*>	Remove Individual Macro	Removes named macro command.
Synchronization	*OPC	Operation Complete	Standard Event register's Operation Complete bit will be 1 when all pending device operations have been finished.
	*OPC?	Operation Complete query	Places an ASCII 1 in the output queue when all pending operations have finished.
	*TRG	Trigger	Trigger s module when TRIG:SOUR is HOLD.
	*WAI	Wait to Complete	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix A
Specifications

Power Requirements (with no SCPs installed)

	+5V		+12V		-12V		+24V		-24V		-5.2V	
IPm=Peak Module Current	I_{Pm}	I_{Dm}	I_{Pm}	IDm	IPm	IDm	IPm	IDm	IPm	IDm	IPm	IDm
IDm=Dynamic Module Current	1.0	0.02	0.06	0.01	0.01	0.01	0.1	0.01	0.1	0.01	0.15	0.01

Cooling Requirements

Average Watts/Slot	ΔPressure (mmH_2O)	Air Flow (liters/s)
14	0.08	0.08

Power Available for SCPs (See VXI Catalog or SCP manuals for SCP current)

1.0A ±24V, 3.5A 5V

Measurement ranges

DC Volts	(HP E1501 or HP E1502) ±62.5mV to ±16V Full Scale
Temperature	Thermocouples - -200 to +1700 °C Thermistors - (Opt 15 required) -80 to +160 °C RTD's - (Opt 15 required) -200 to +850 °C
Resistance	(HP E1505 with HP E1501) 512 ohms to 131 Kohms FS
Strain	25,000 με or limit of linear range of strain gage

Measurement resolution

16 bits (including sign)

Trigger Timer and Sample Timer Accuracy

100ppm (.01%) from -10 °C to +70 °C

External Trigger Input

TTL compatible input. Negative true edge triggered except first trigger will occur if external trigger input is held low when module is INITiated. Minimum pulse width 100nS. Since each trigger starts a complete scan of 2 or more channel readings, maximum trigger rate depends on module configuration.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Maximum input voltage (Normal mode plus common mode)	With Direct Input, Passive Filter, or Amplifier SCPs: Operating: < ±16 V peak Damage level: > ±42 V peak With HP E1513 Divide by 16 Attenuator SCP: Operating: < ±60 VDC, < ±42 V peak
Maximum common mode voltage	With Direct Input, Passive Filter, or Amplifier SCPs: Operating: < ±16 V peak Damage level: > ±42 V peak With HP E1513 Divide by 16 Attenuator SCP: Operating: < ±60 VDC, < ±42 V peak
Common mode rejection	0 to 60Hz -105dB
Input impedance	greater than 90 MOhm differential (1 M Ohm with HP E1513 Attenuator)
On-board Current Source	122 µA ±0.02%, with ±17 Volts Compliance

Maximum tare cal offset

SCP Gain = 1 (Maximum tare offset depends on A/D range and SCP gain)

A/D range ±V F.Scale	16	4	1	0.25	0.0625
Max Offset	3.2213	.82101	.23061	.07581	.03792

Notice: Published specifications indicate accuracy degradation when subjected to radiated fields.

The following specifications reflect the performance of the HP E1419 with the HP E1501 Direct Input Signal Conditioning Plug-on. The performance of the HP E1419 with other SCPs is found in the Specifications section of that SCP's manual.

Measurement accuracy DC Volts

(90 days) 23°C ±1°C (with *CAL? done after 1 hr warm up and CAL:ZERO? within 5 min.).
NOTE: If autoranging is ON:
for readings <3.8V, add ±.02% to linearity specifications.
for readings ≥3.8V, add ±.05% to linearity specifications.

A/D range ±V F.Scale	Linearity % of reading	Offset Error	Noise 3 sigma	Noise* 3 sigma
.0625	0.01%	5.3 µV	18 µV	8 µV
.25	0.01%	10.3 µV	45 µV	24 µV
1	0.01%	31 µV	110 µV	90 µV
4	0.01%	122 µV	450 µV	366 µV
16	0.01%	488 µV	1.8 mV	1.5 mV

Temperature Coefficients: Gain - 10ppm/°C. Offset - (0 - 40°C) .14µV/°C, (40 - 55°C) .8µV+.38µV/°C

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Temperature Accuracy

The following pages have temperature accuracy graphs that include instrument and firmware linearization errors. The linearization algorithm used is based on the IPTS-68(78) standard transducer curves. Add your transducer accuracy to determine total measurement error.

The thermocouple graphs on the following pages include only the errors due to measuring the voltage output of the thermocouple, as well as the algorithm errors due to converting the thermocouple voltage to temperature. To this error must be added the error due to measuring the reference junction temperature with an RTD or a 5K thermistor. See the graphs for the RTD or the 5K thermistor to determine this additional error. Also, the errors due to gradients across the isothermal reference must be added. If an external isothermal reference panel is used, consult the manufacturer's specifications. If HP termination blocks are used as the isothermal reference, see the notes below.

NOTE

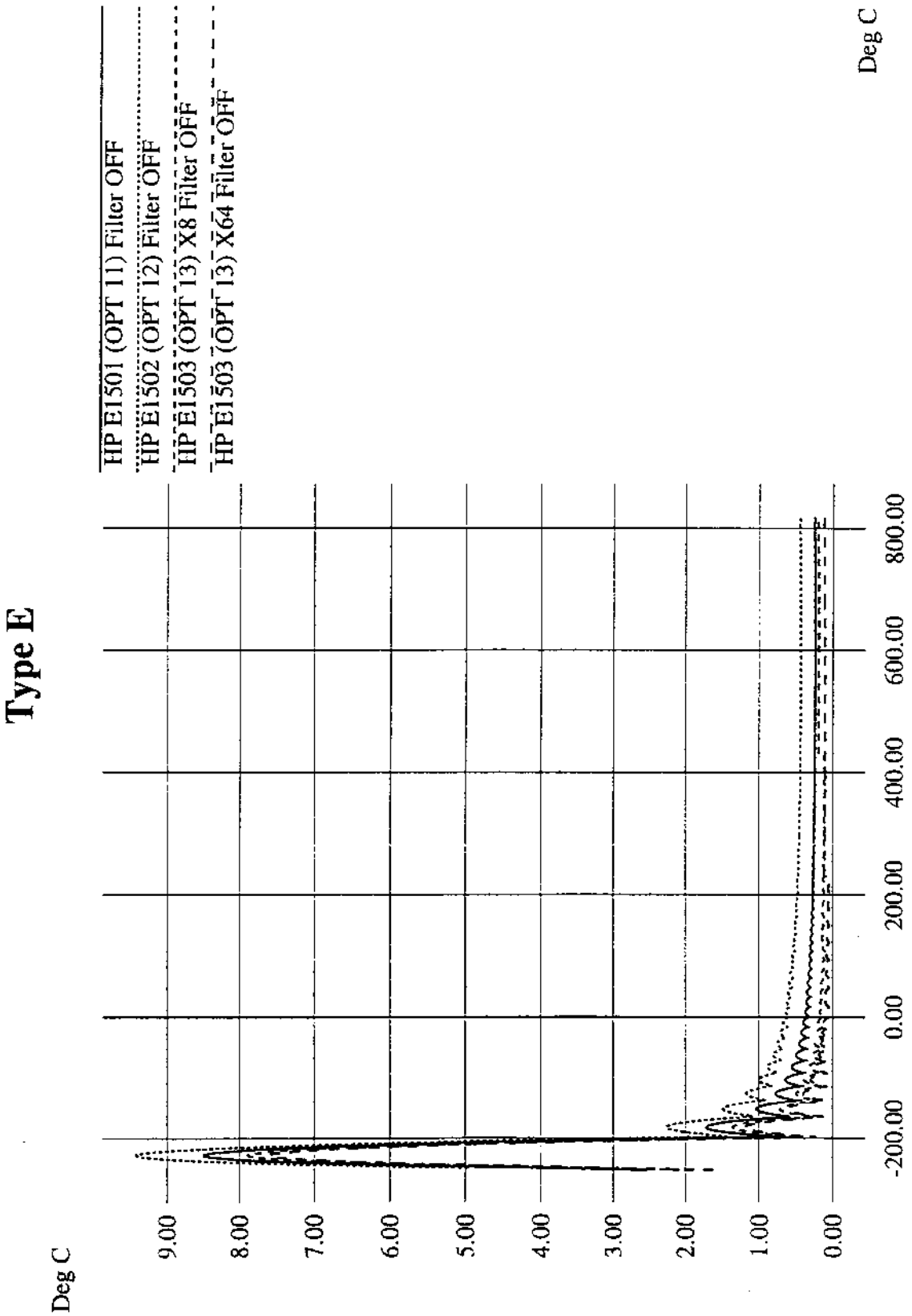
1) When using the Terminal Module as the isothermal reference, add ±0.6° C to the thermocouple accuracy specs to account for temperature gradients across the Terminal Module. The ambient temperature of the air surrounding the Terminal Module must be within ±2° C of the temperature of the inlet cooling air to the VXI mainframe.

2) When using the E1586 Rack-Mount Terminal Panel as the isothermal reference, add ±0.2° C to the thermocouple accuracy specs to account for temperature gradients across the E1586. The E1586A should be mounted in the bottom part of the rack, below and away from other heat sources for best performance.

The temperature graphs are found on the following pages:

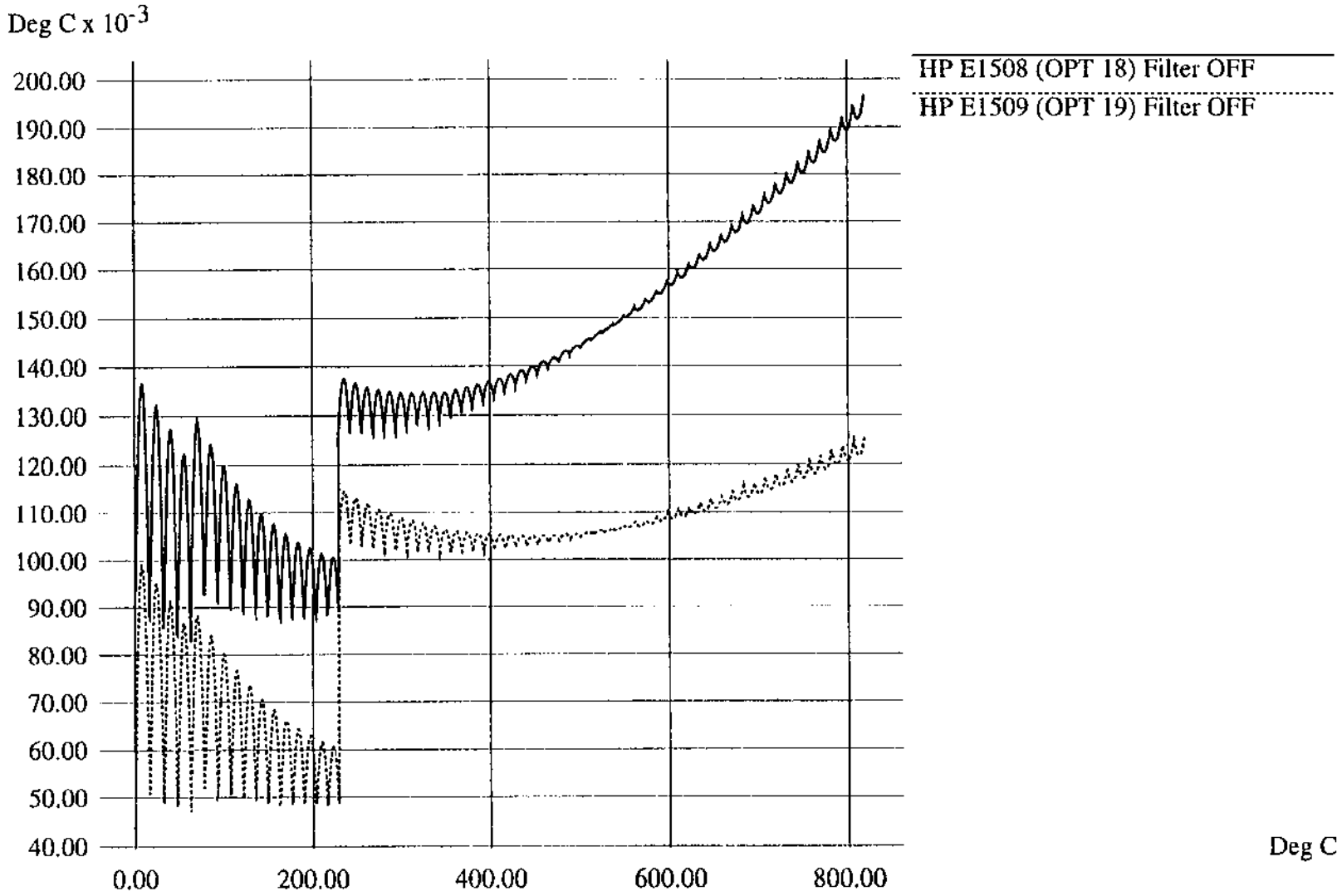
- Thermocouple Type E (-200 to 800°C) 336,337
- Thermocouple Type E (0 to 800°C) 338,339
- Thermocouple Type EEXtended 340,341
- Thermocouple Type J 342,343
- Thermocouple Type K 344
- Thermocouple Type R 345,346
- Thermocouple Type S 347,300
- Thermocouple Type T 349,350
- Reference Thermistor 5K 351,352
- Reference RTD 100Ω 353
- RTD 100Ω 354,355
- Thermistor 2250Ω 356,357
- Thermistor 5KΩ 358,359
- Thermistor 10KΩ 360,361

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

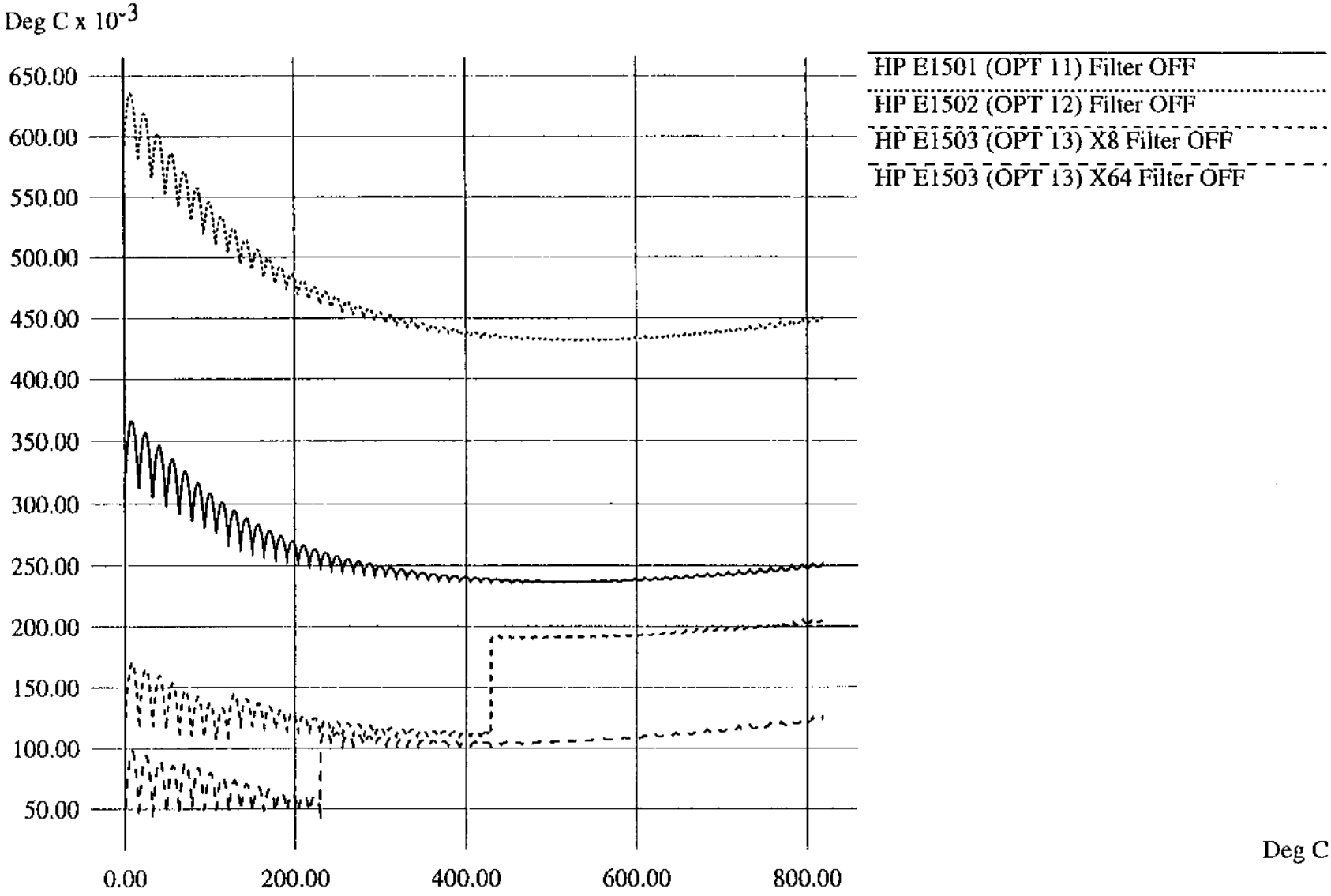
Type E

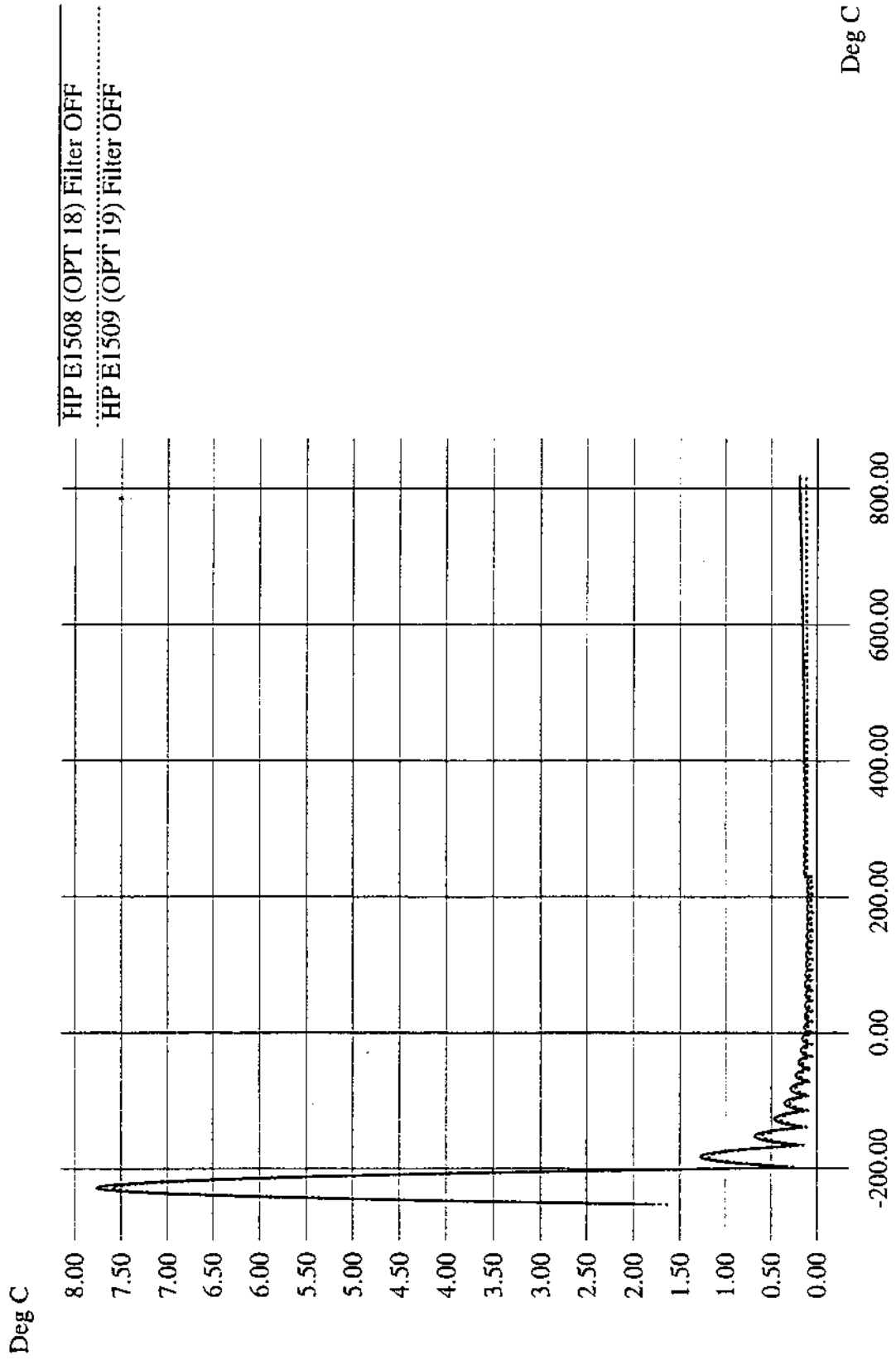


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

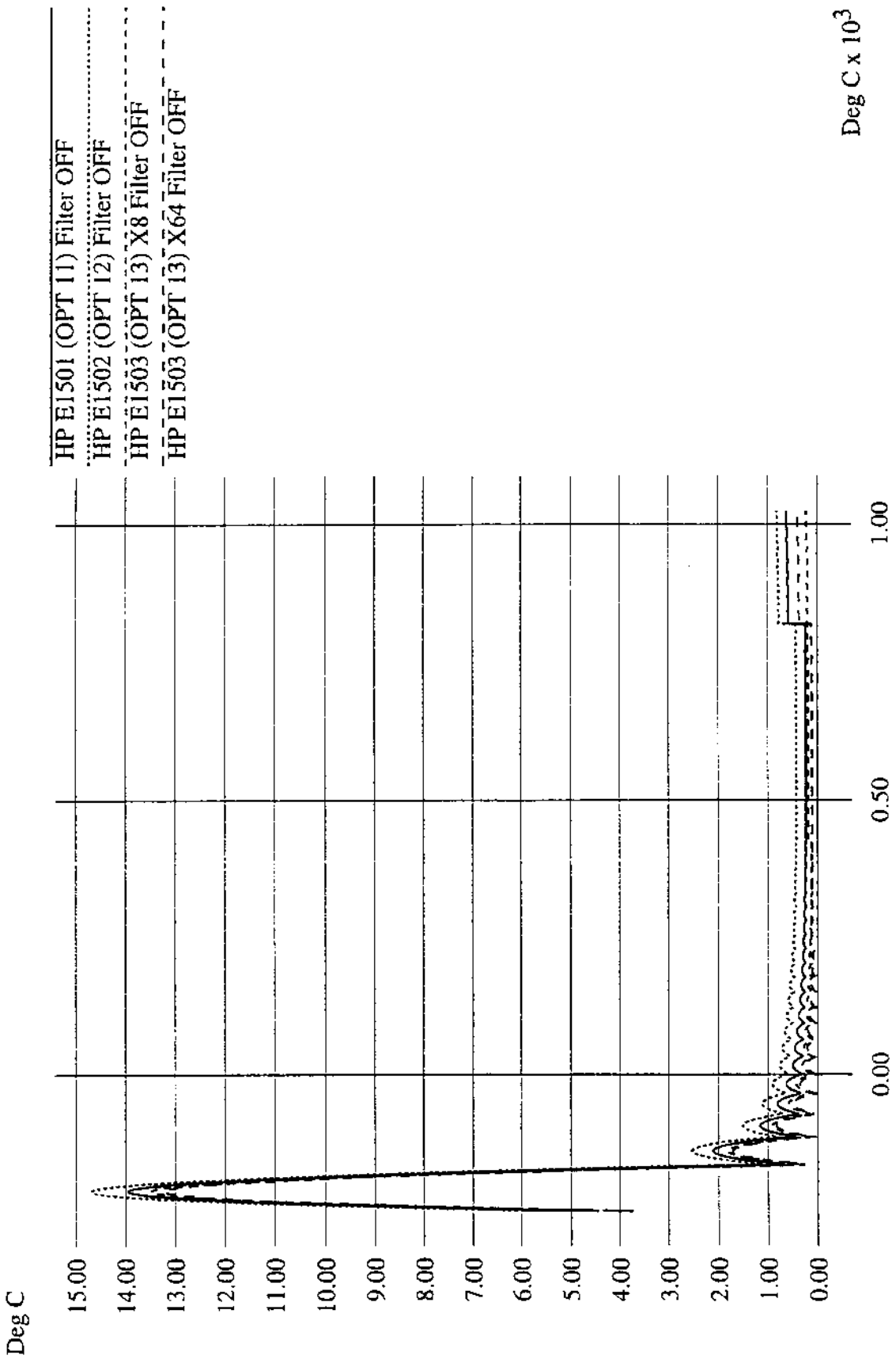
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Type E

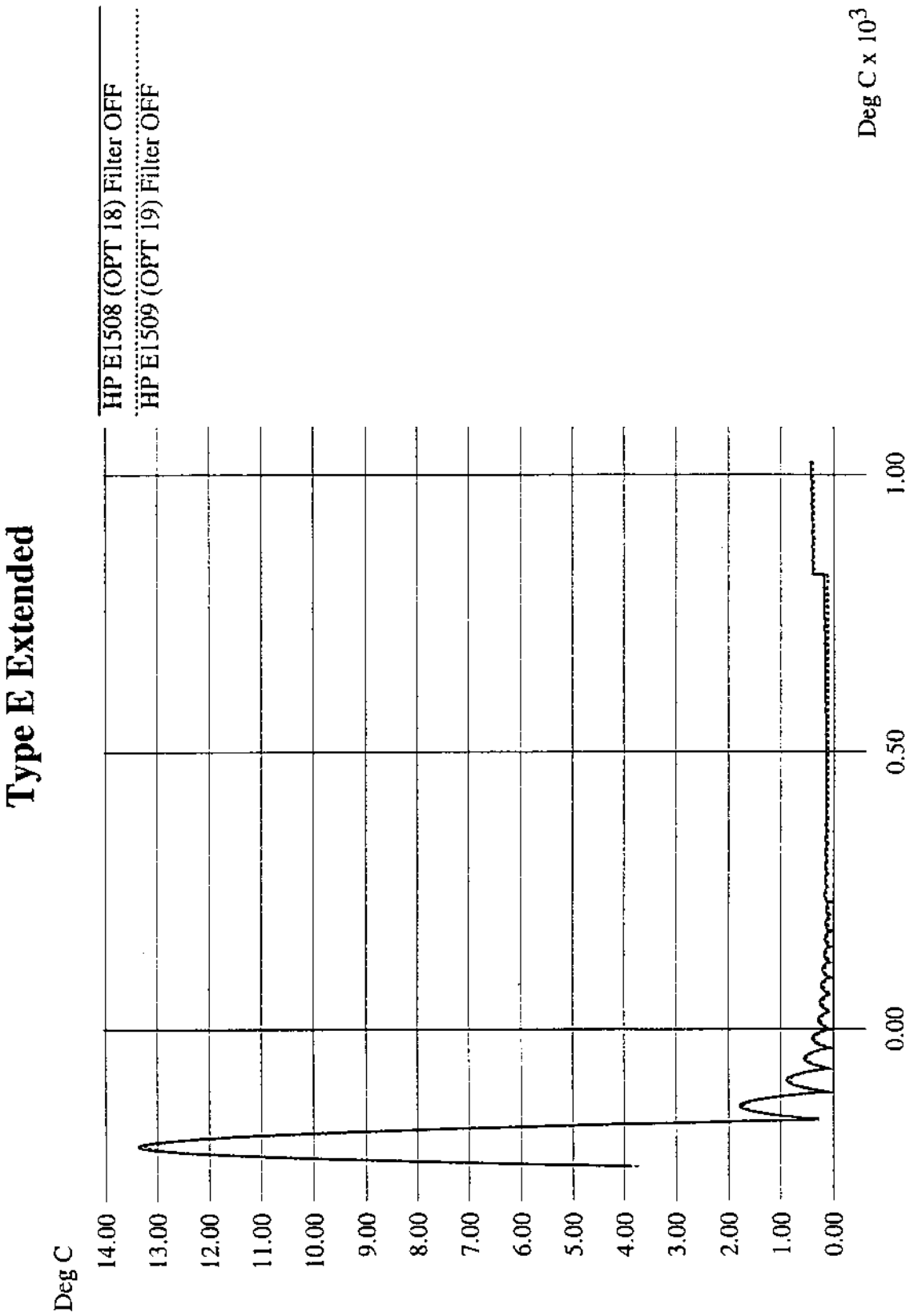




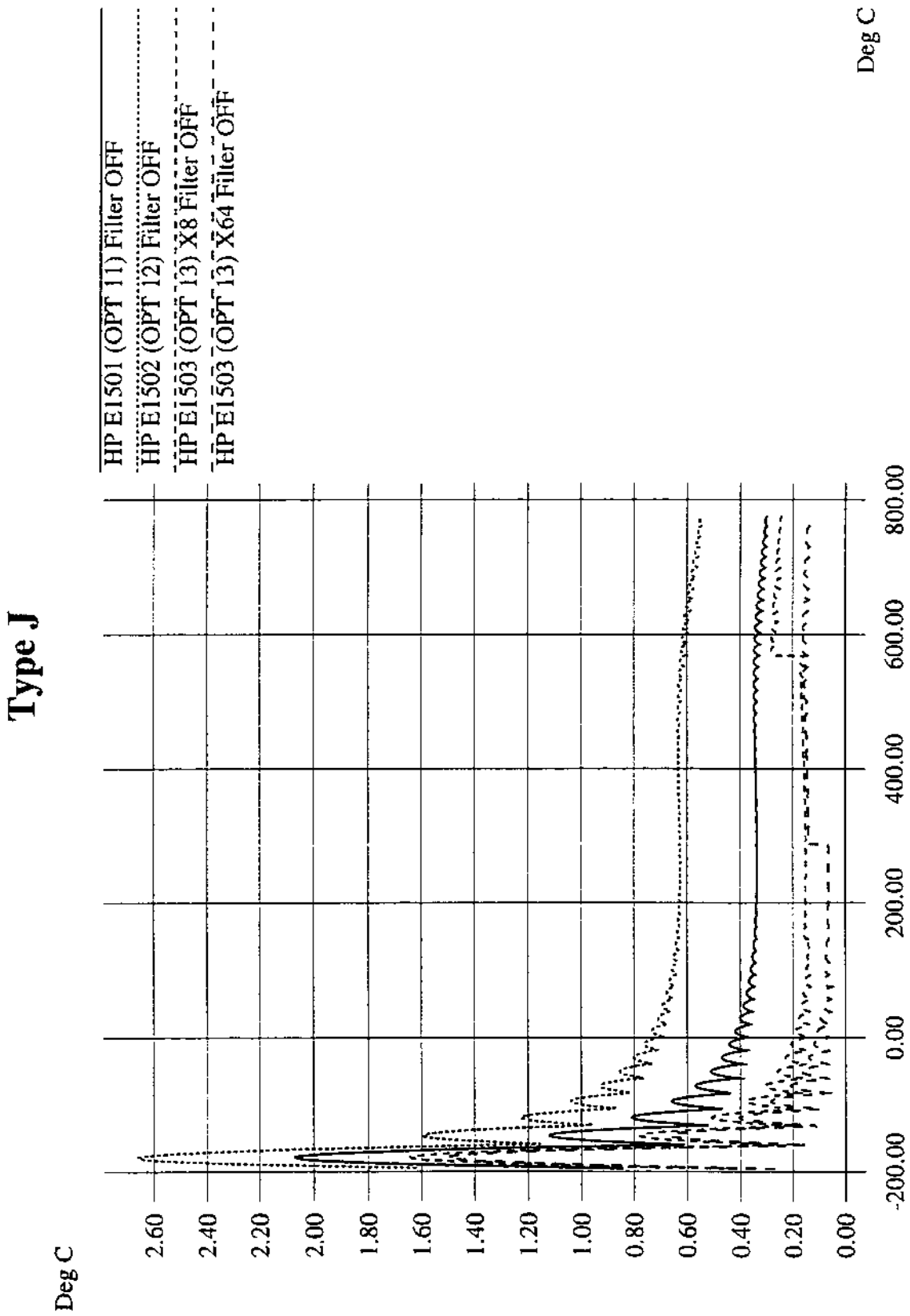
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



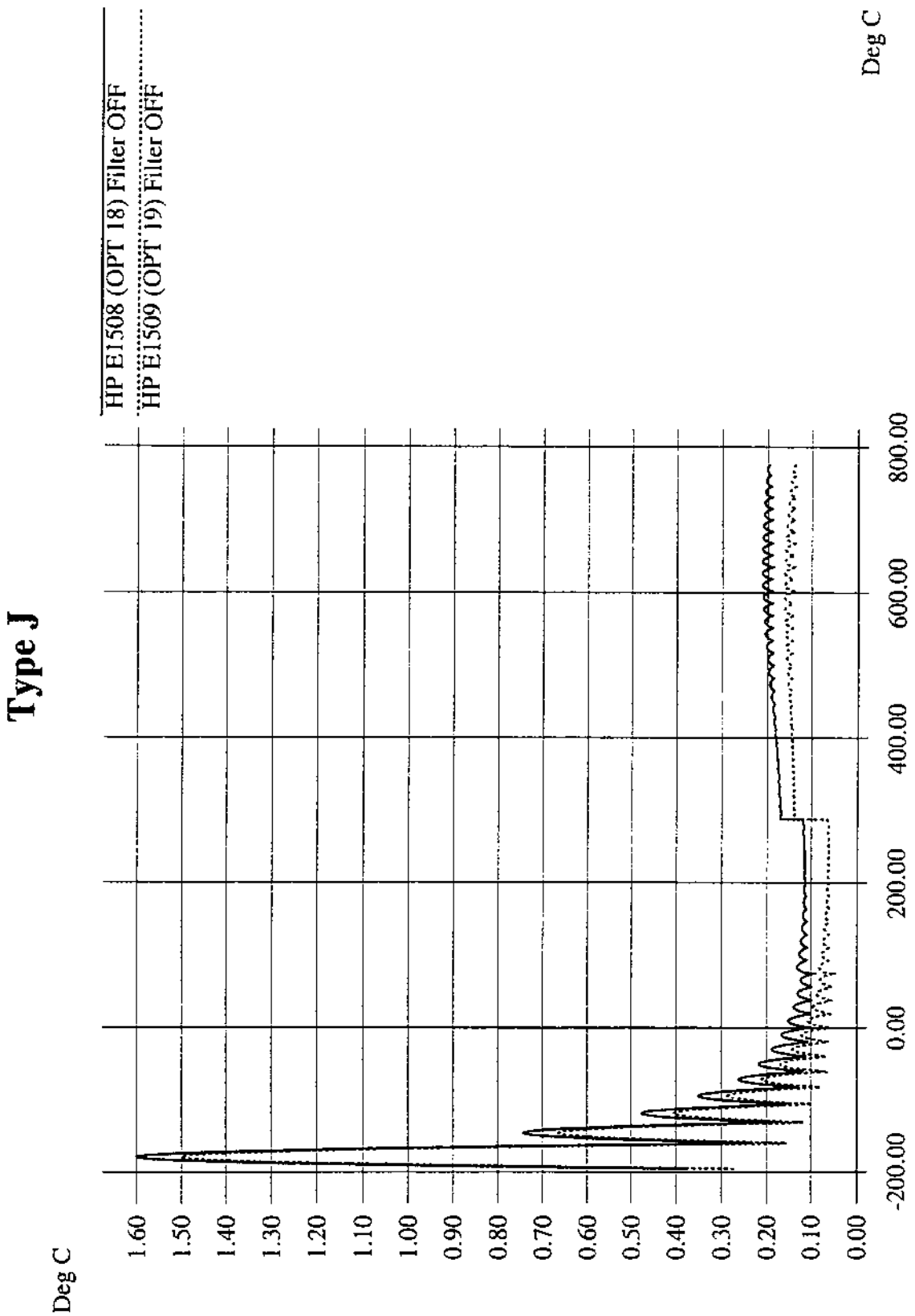
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



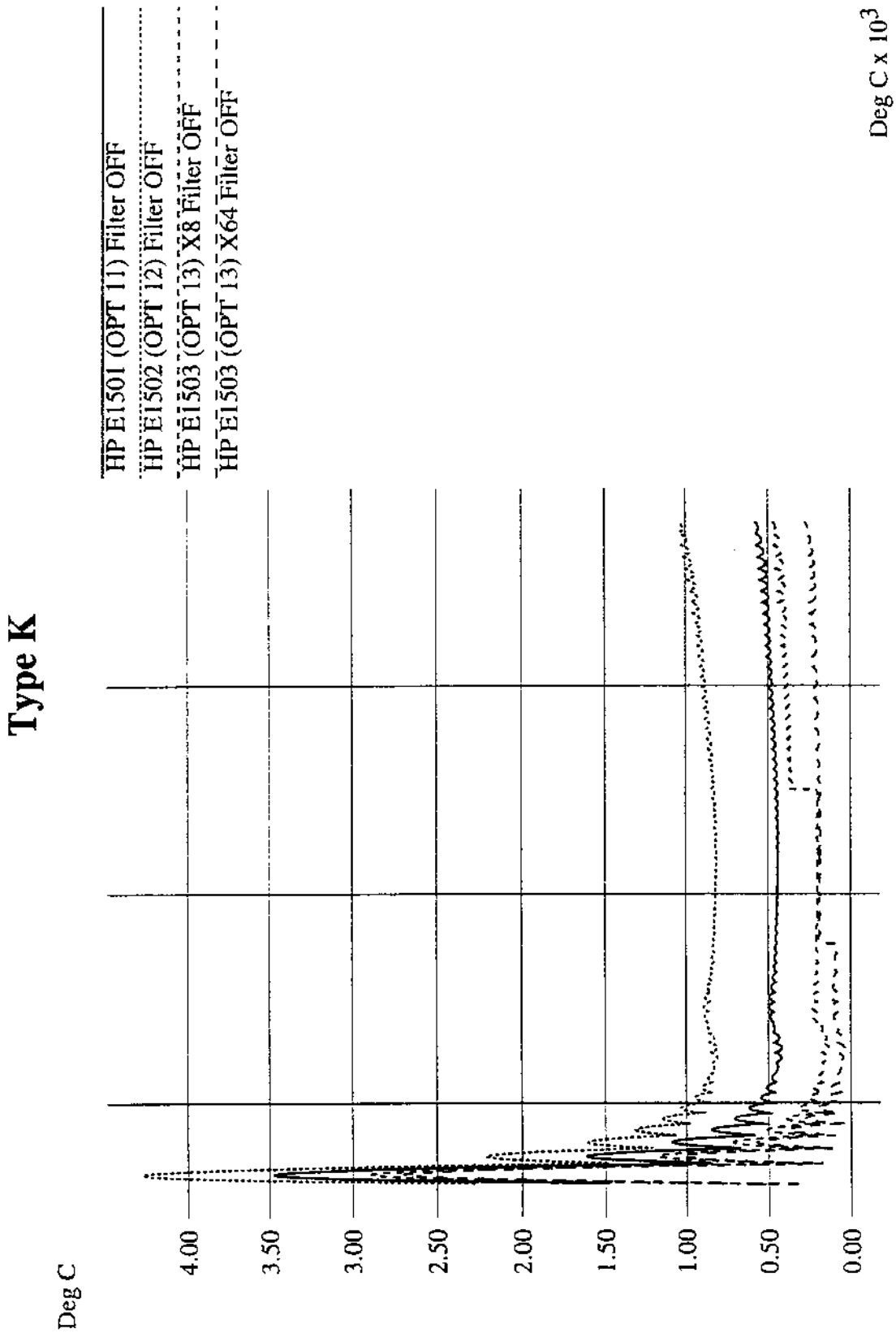
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

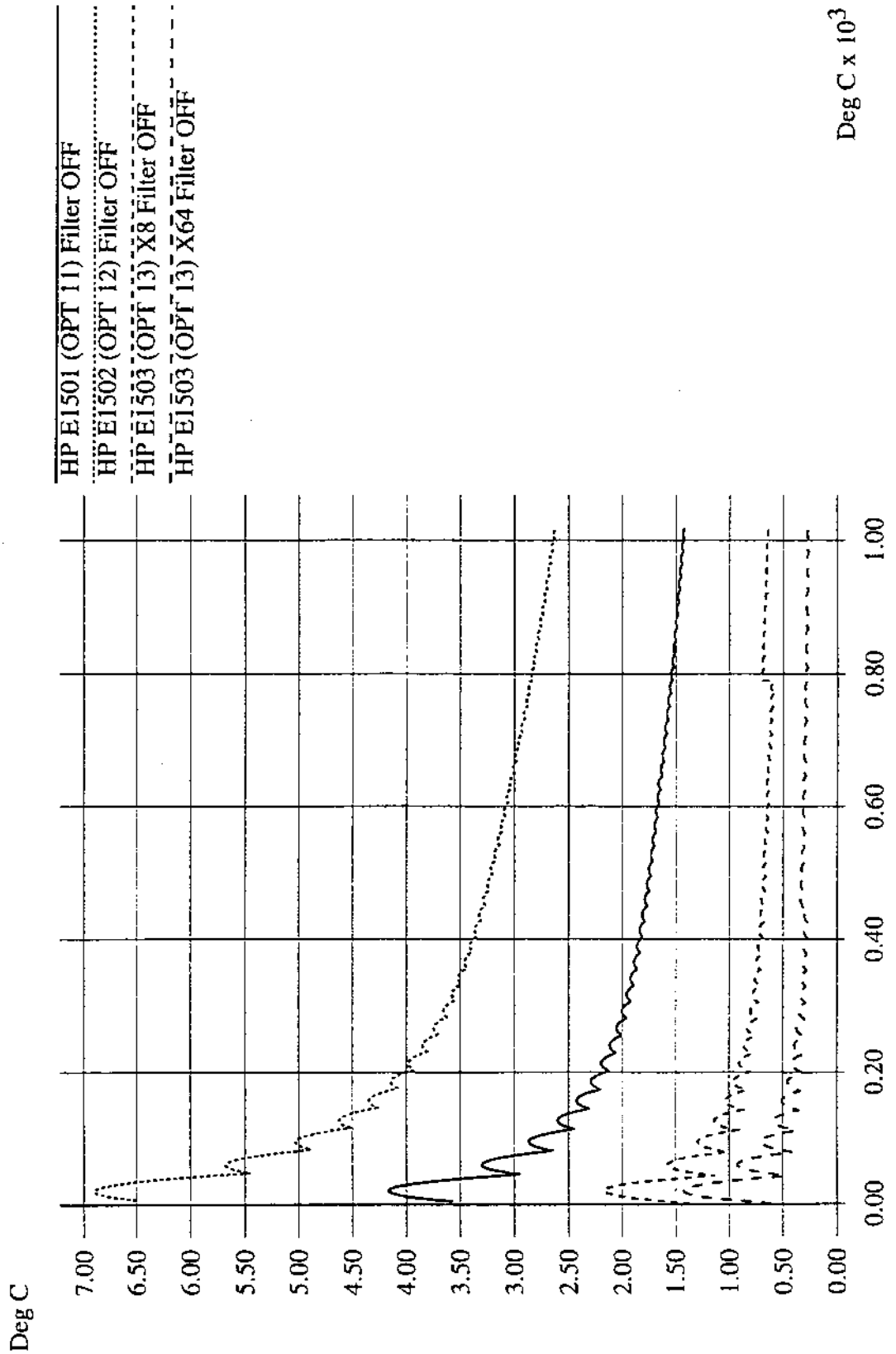


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

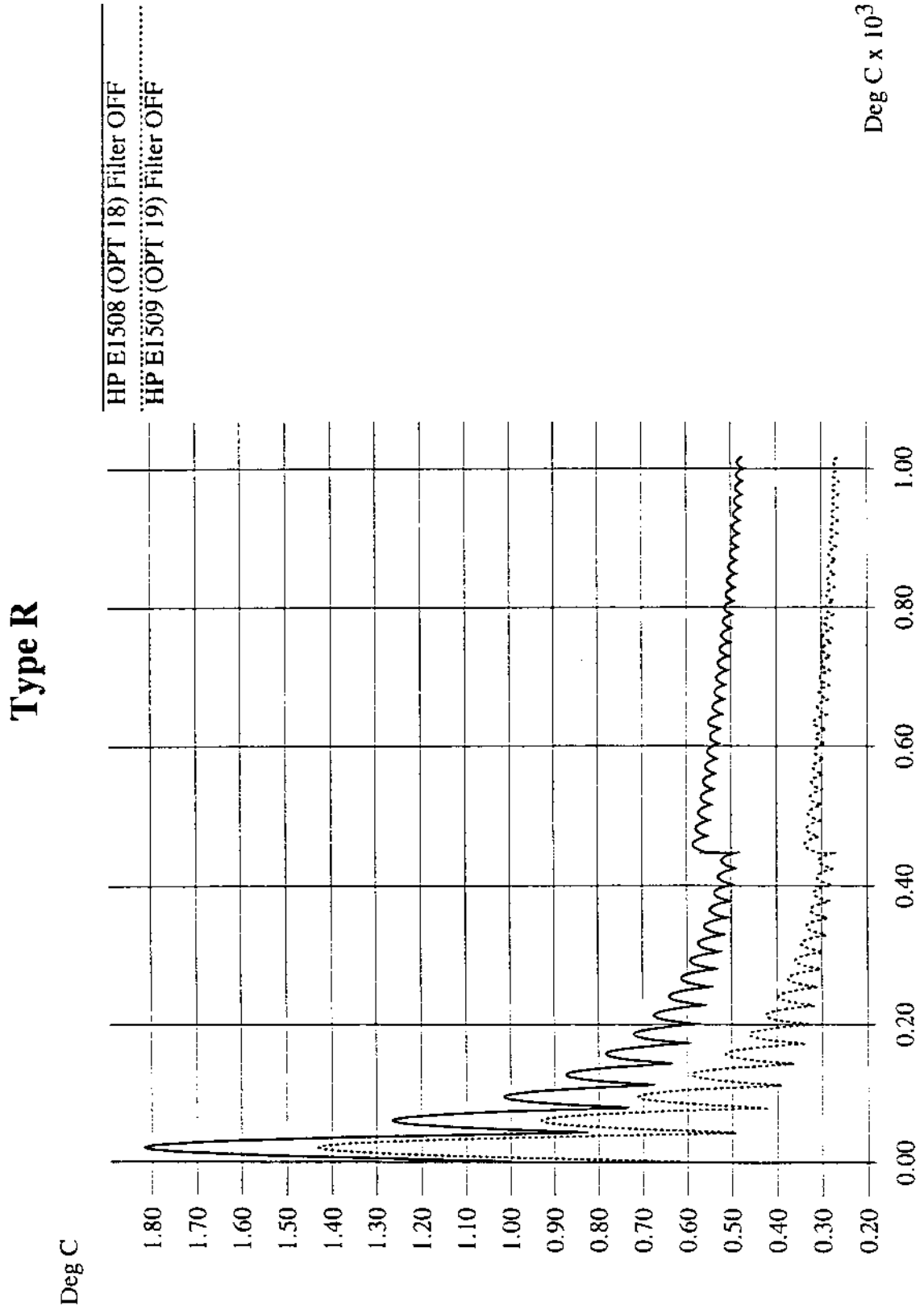


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

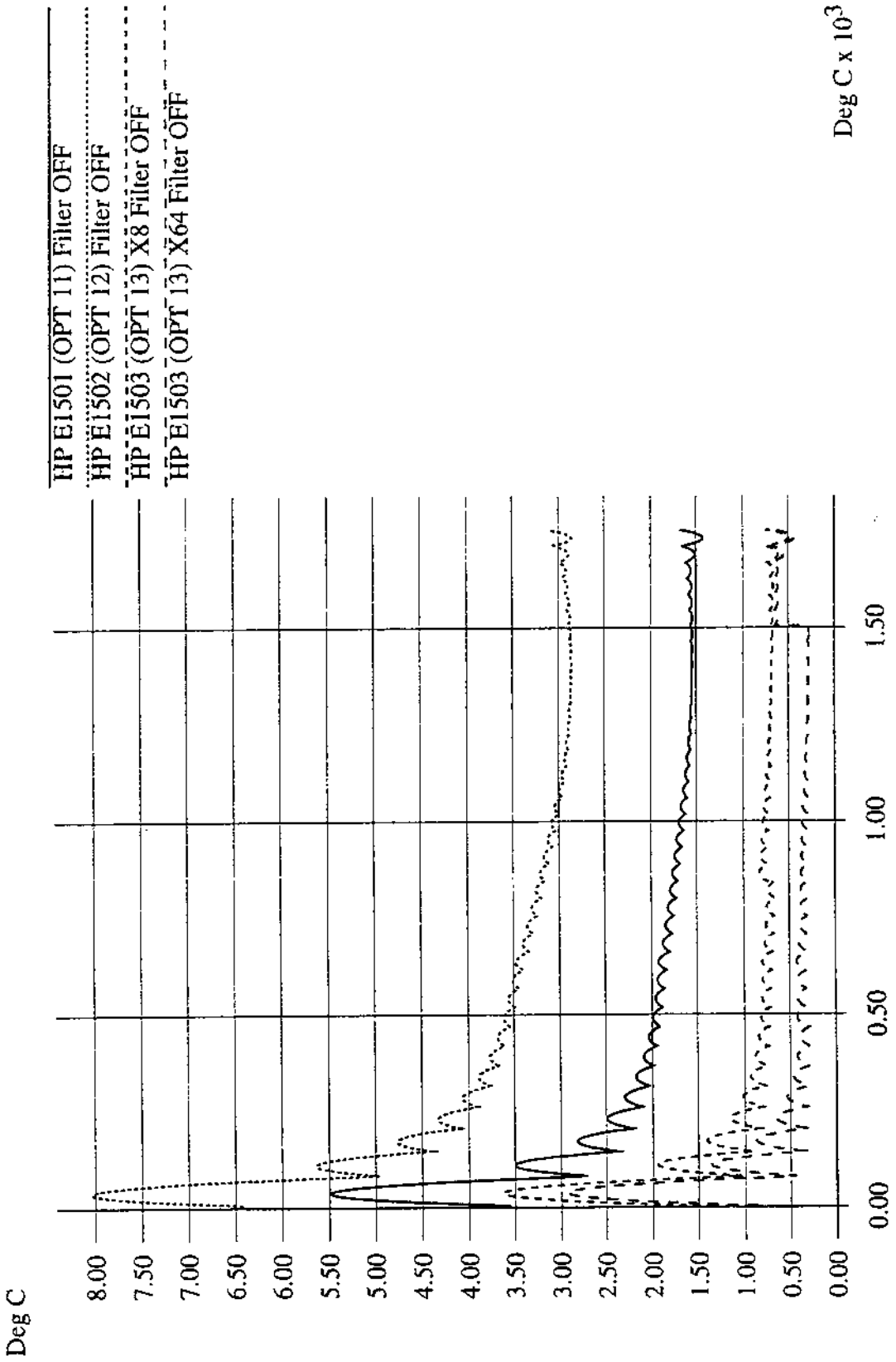
Type R



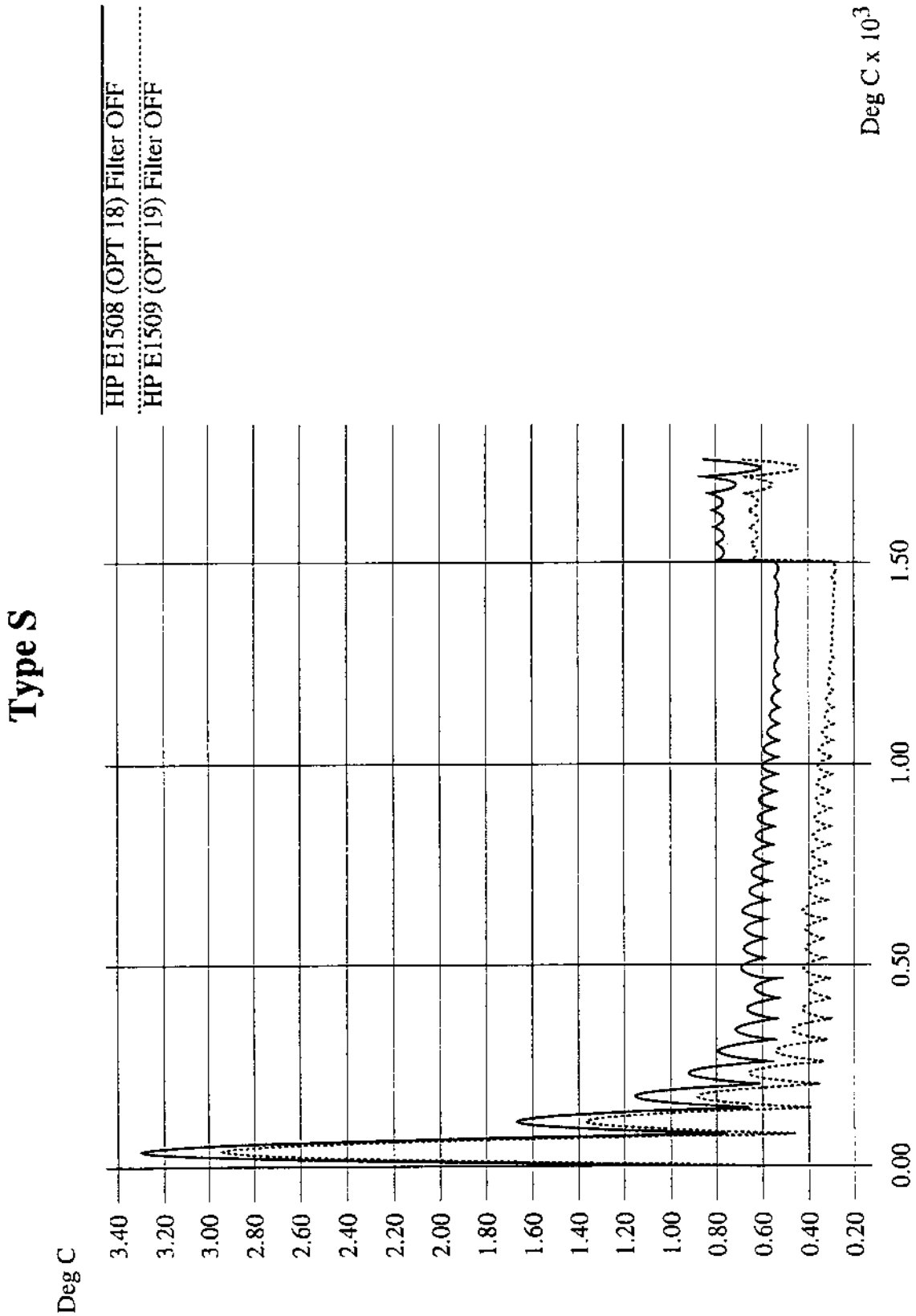
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



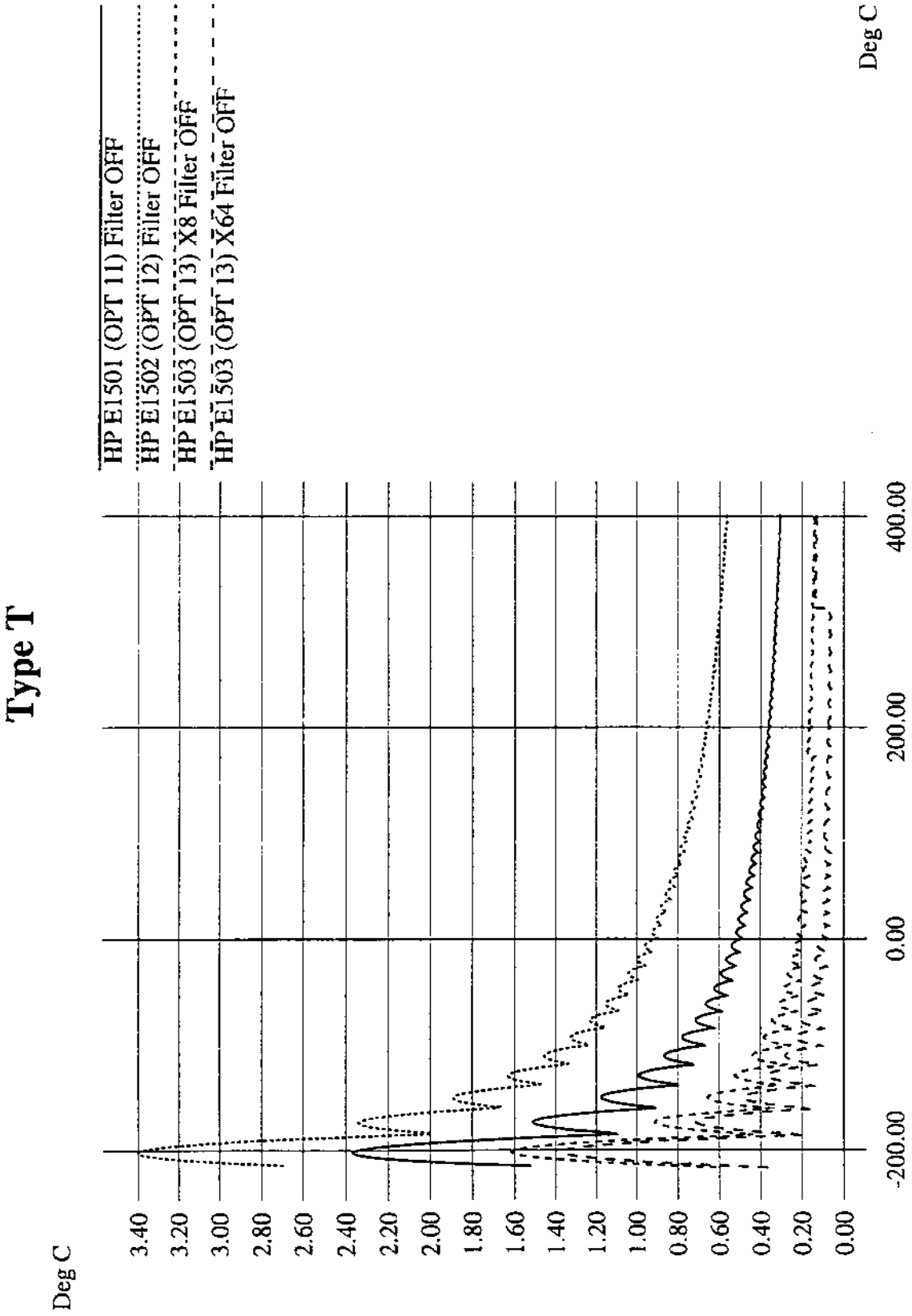
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



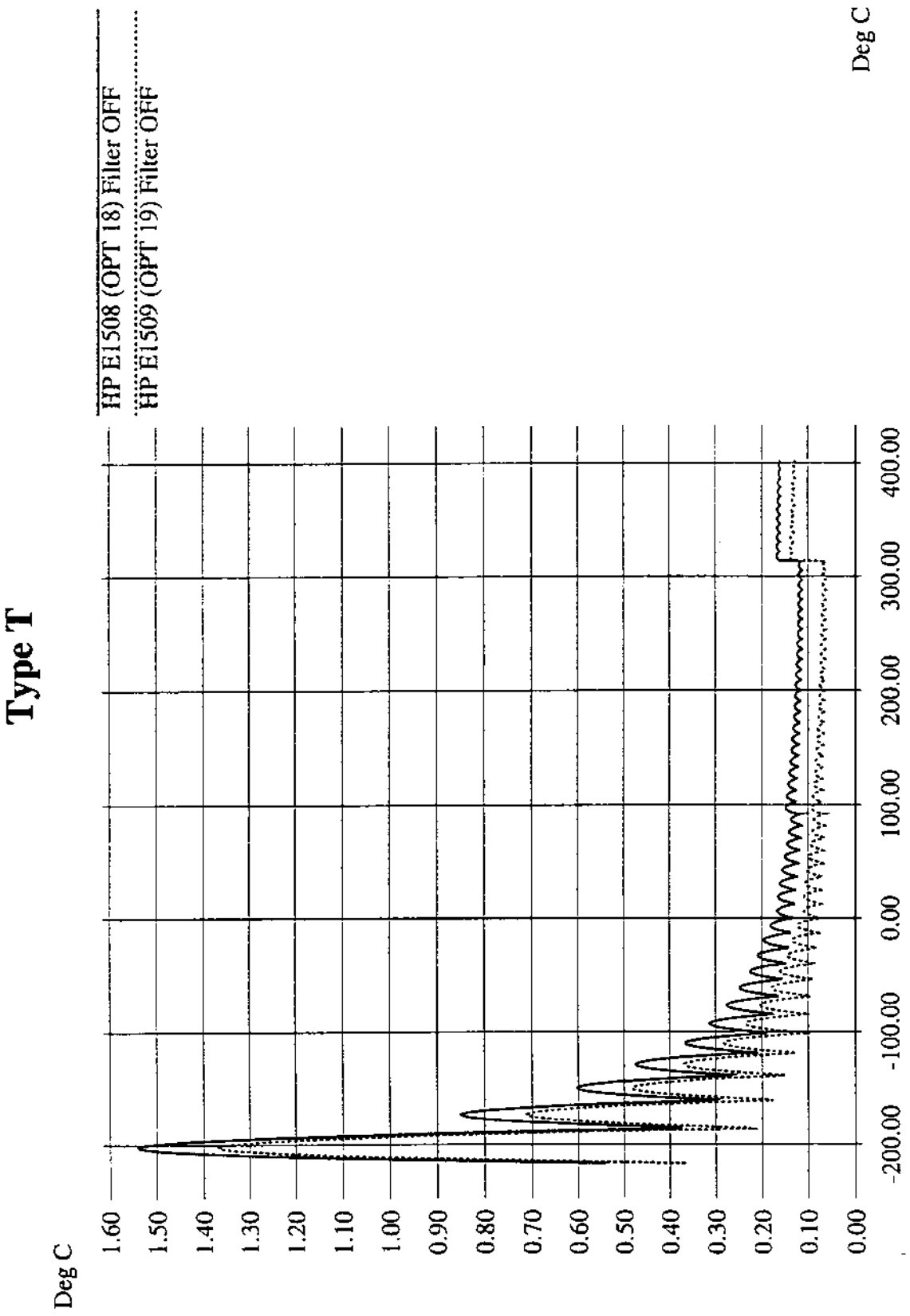
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



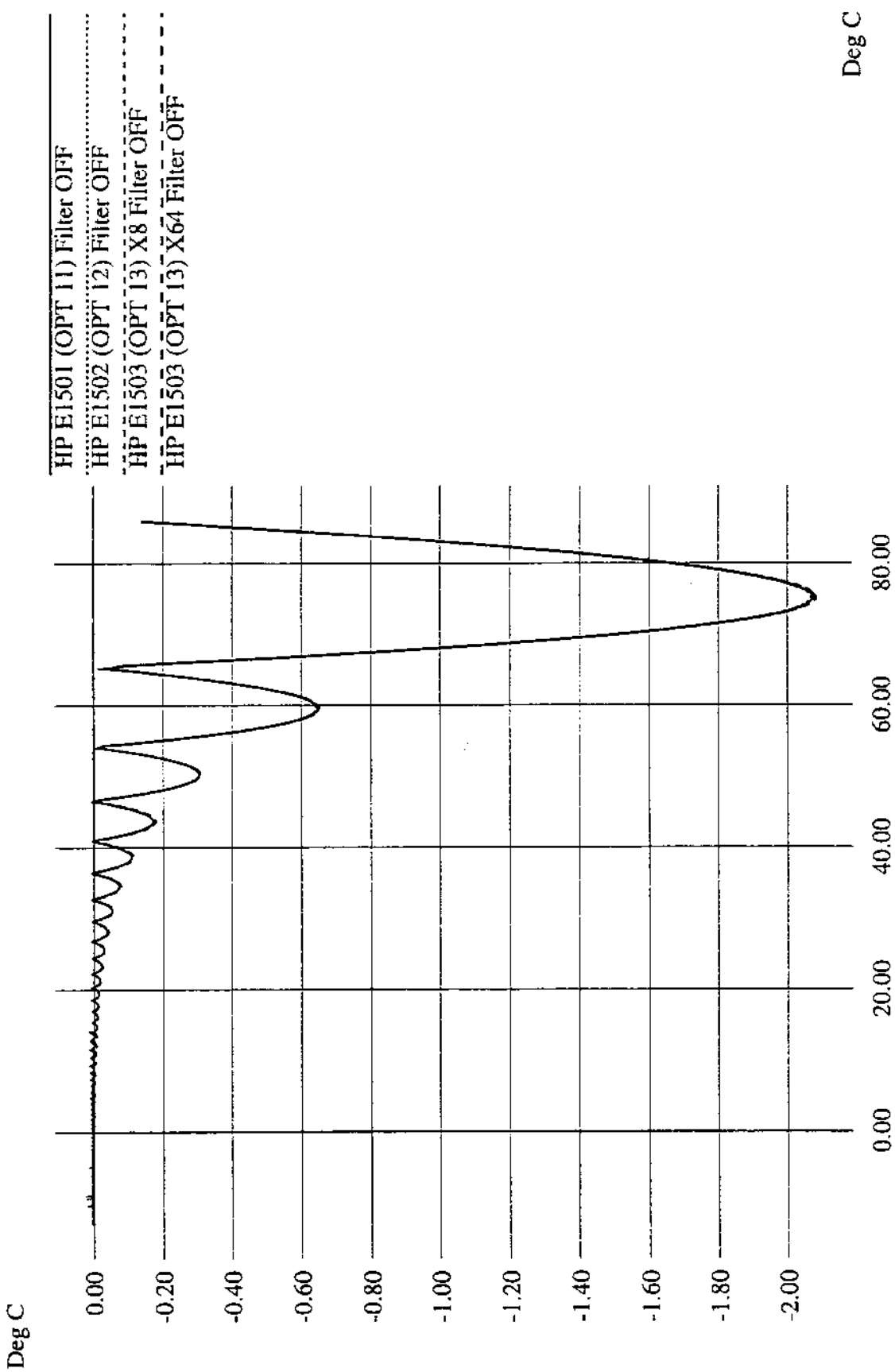
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



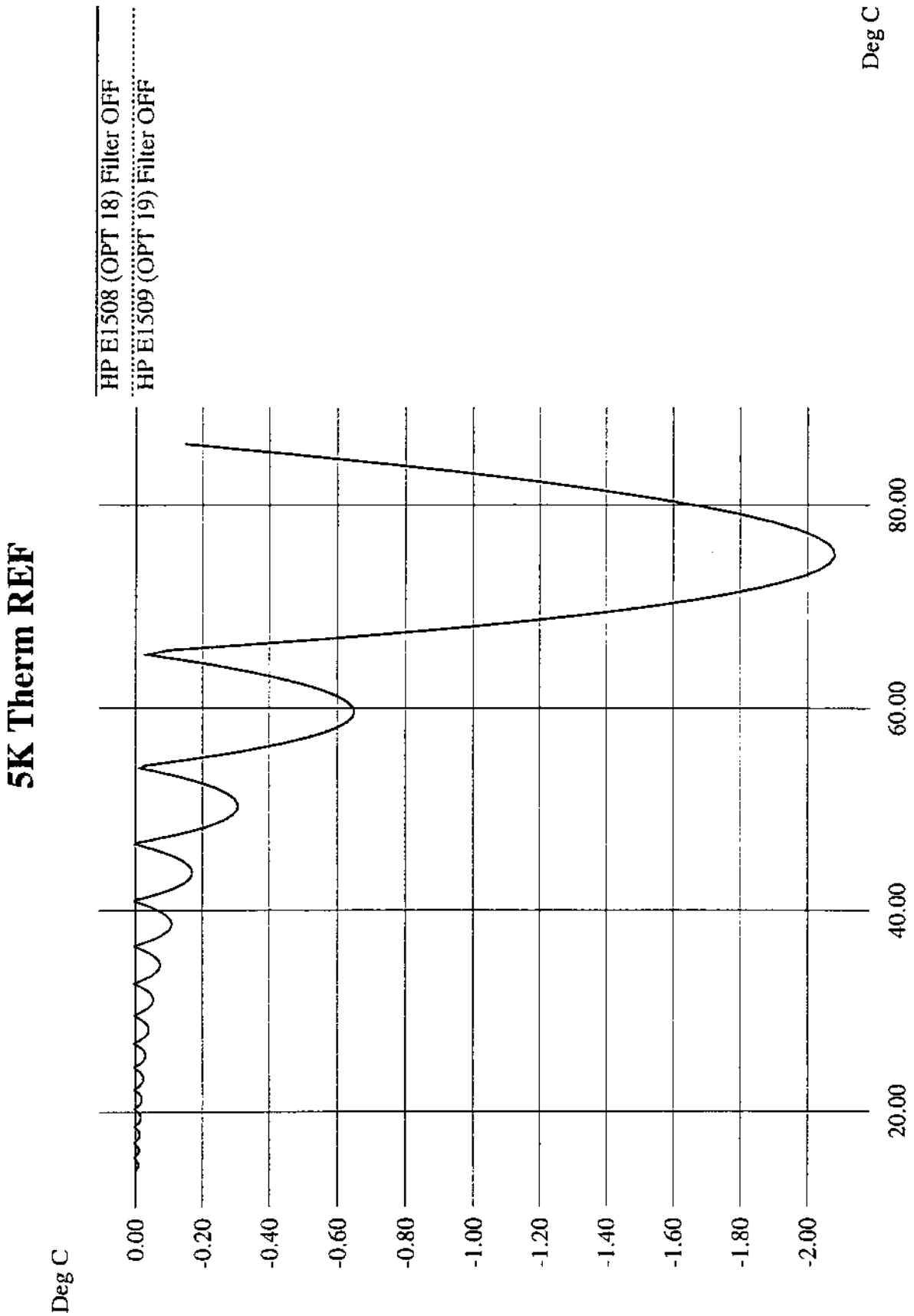
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



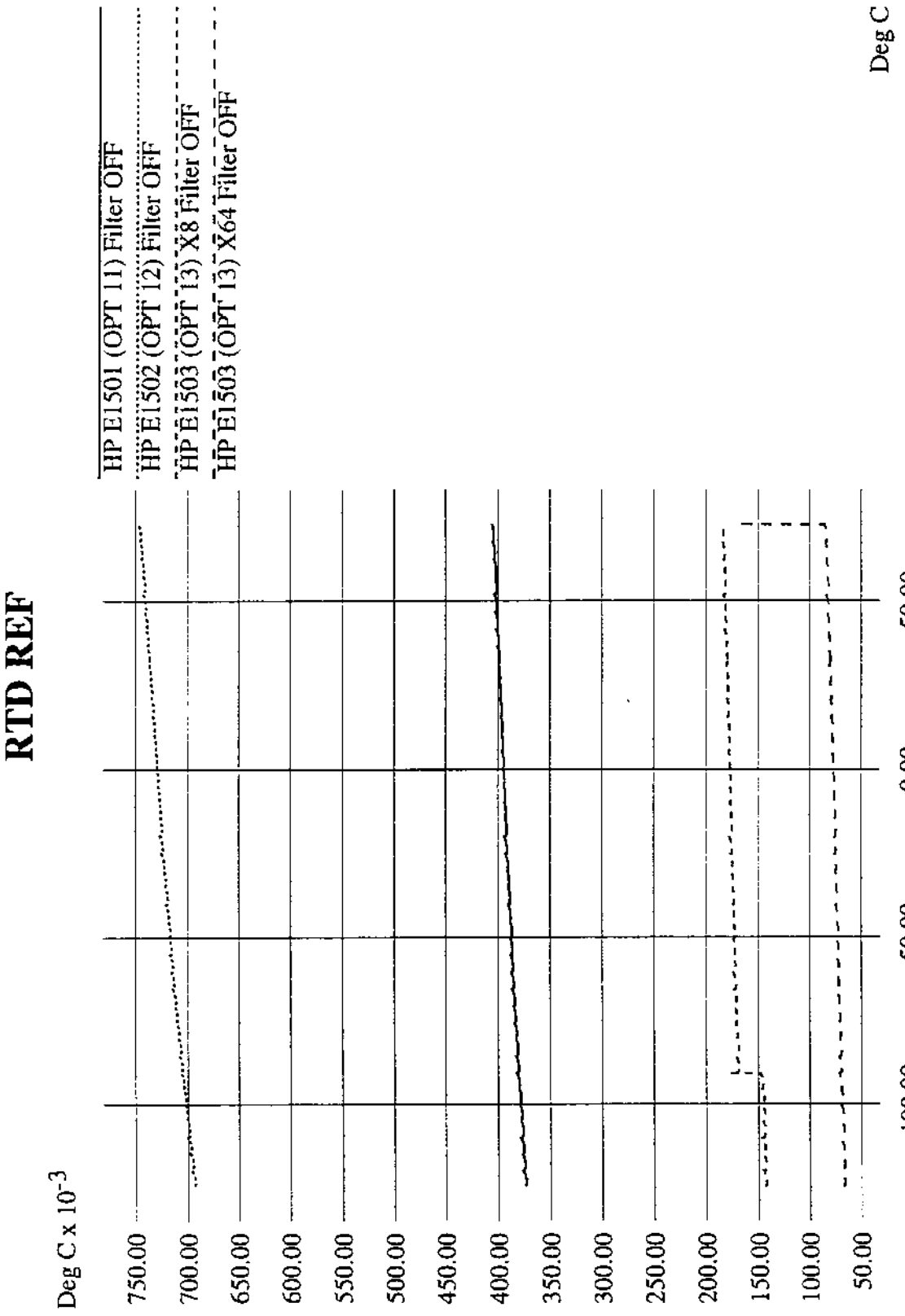
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



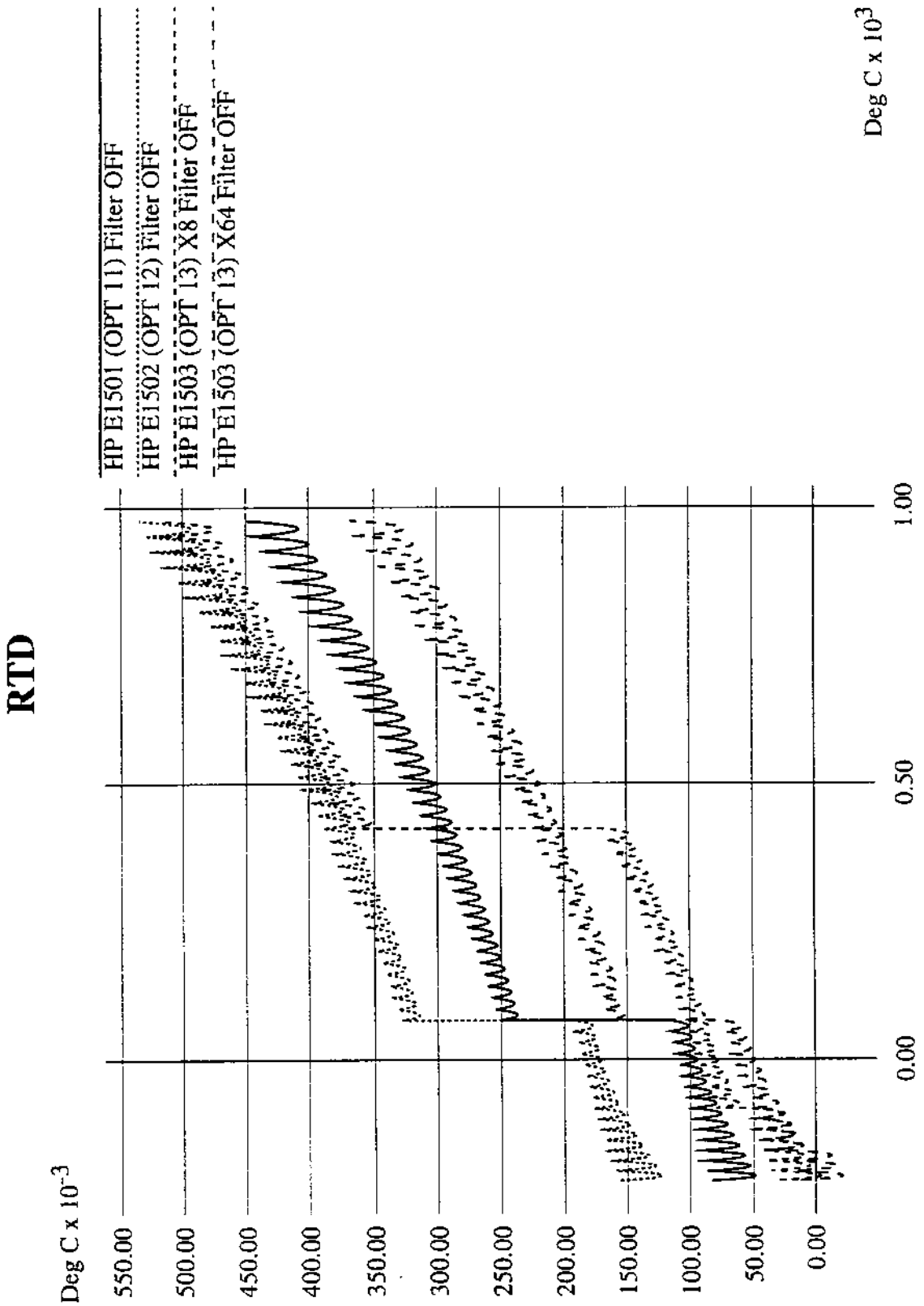
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



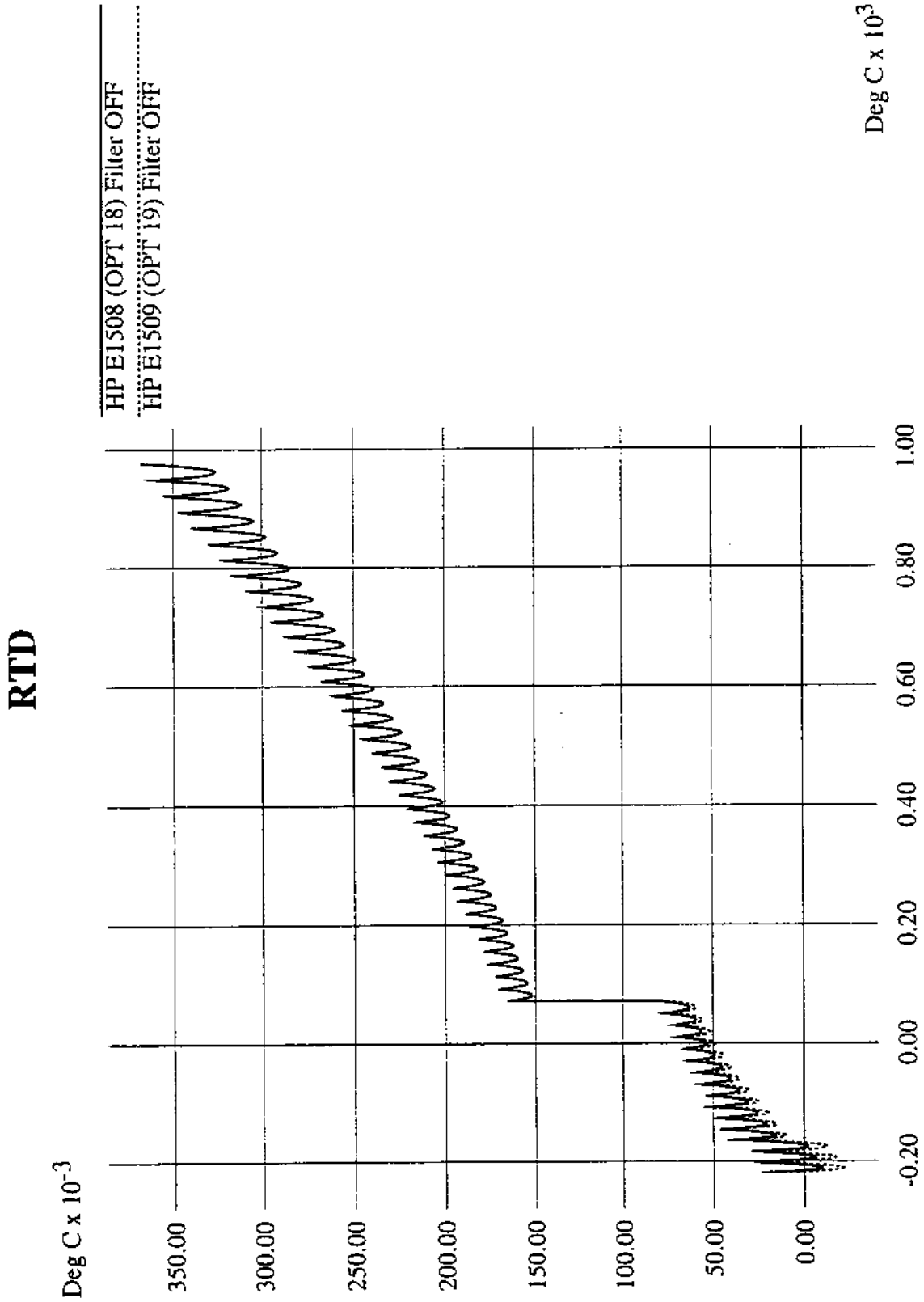
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



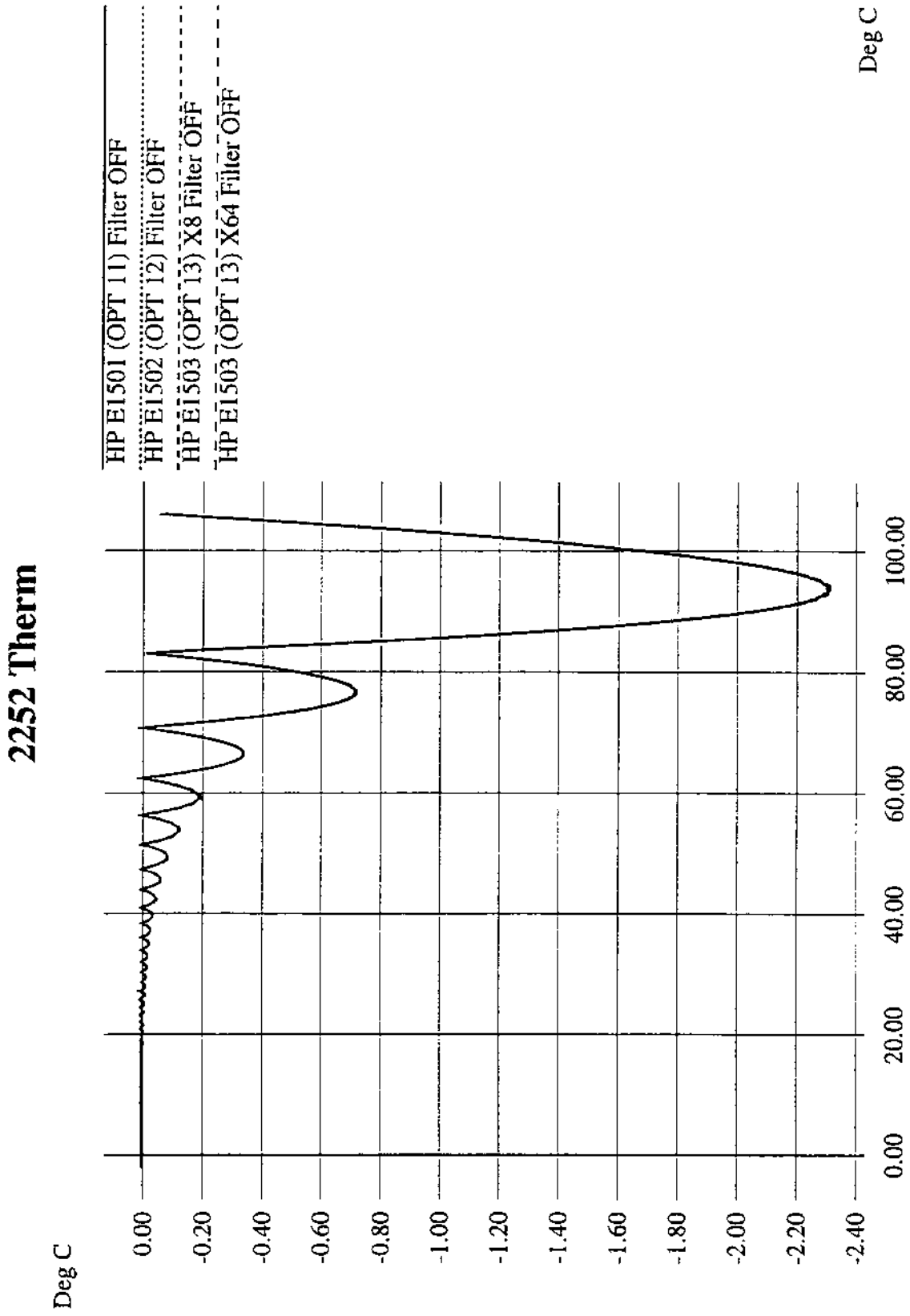
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



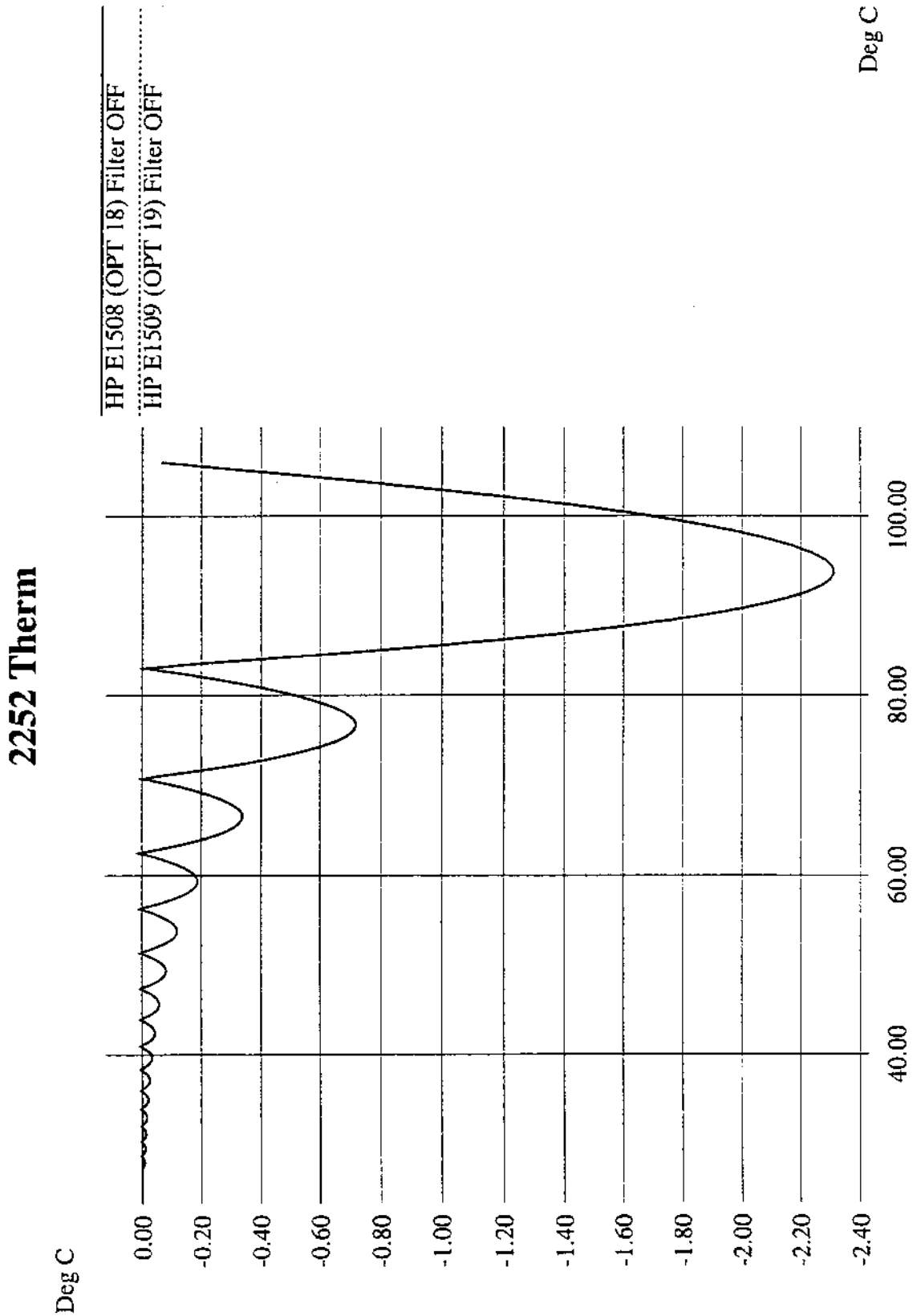
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



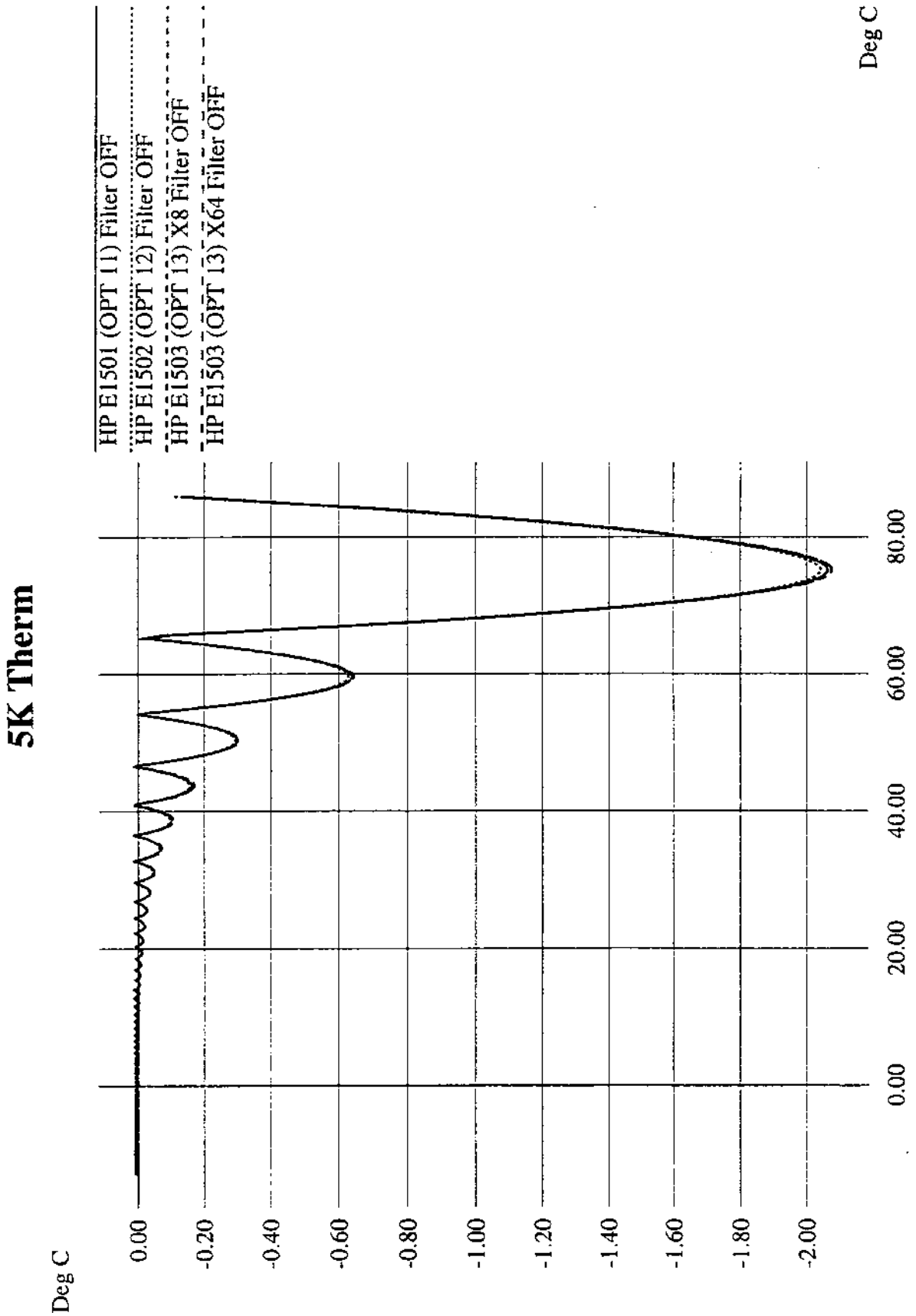
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



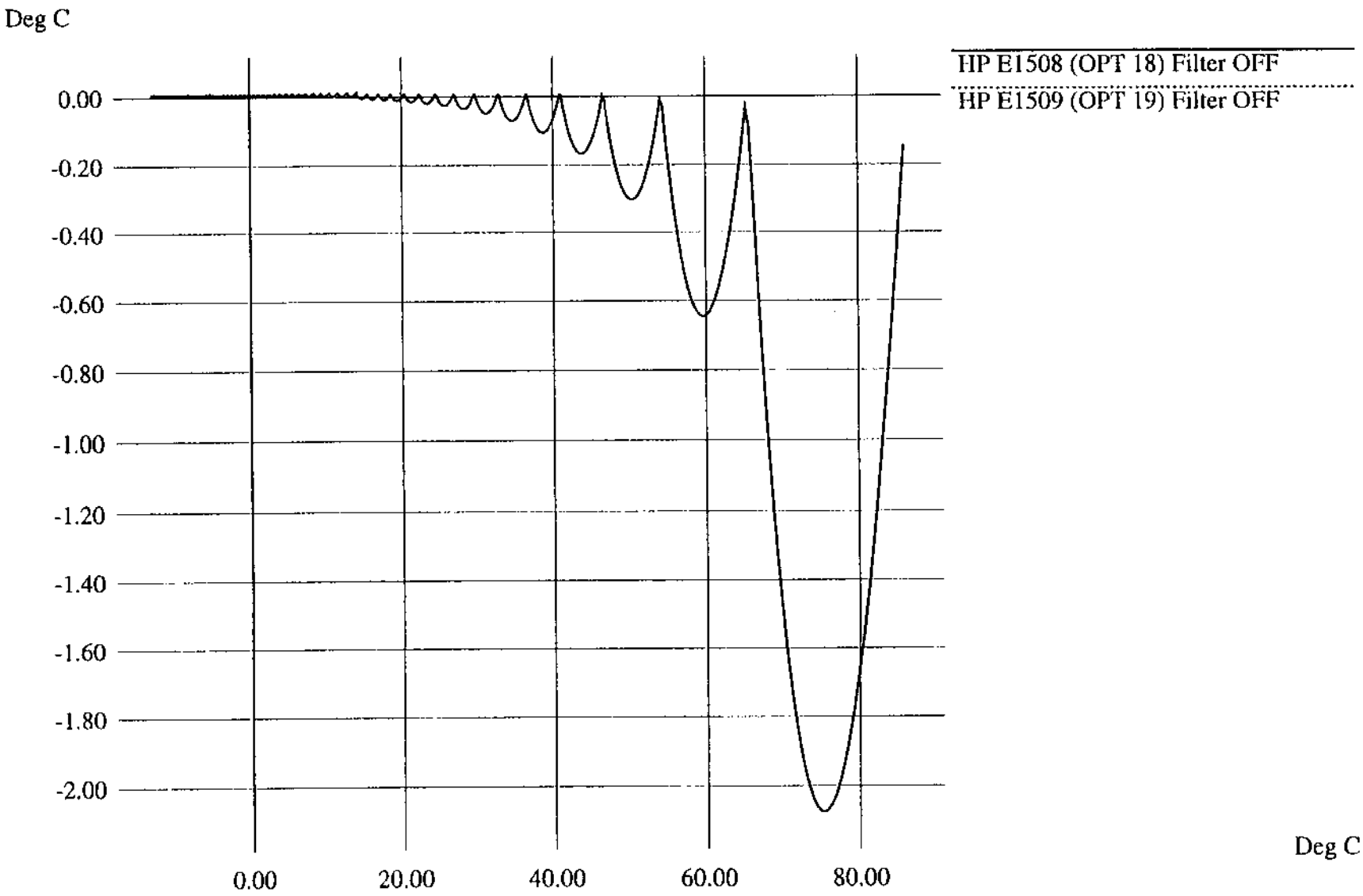
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

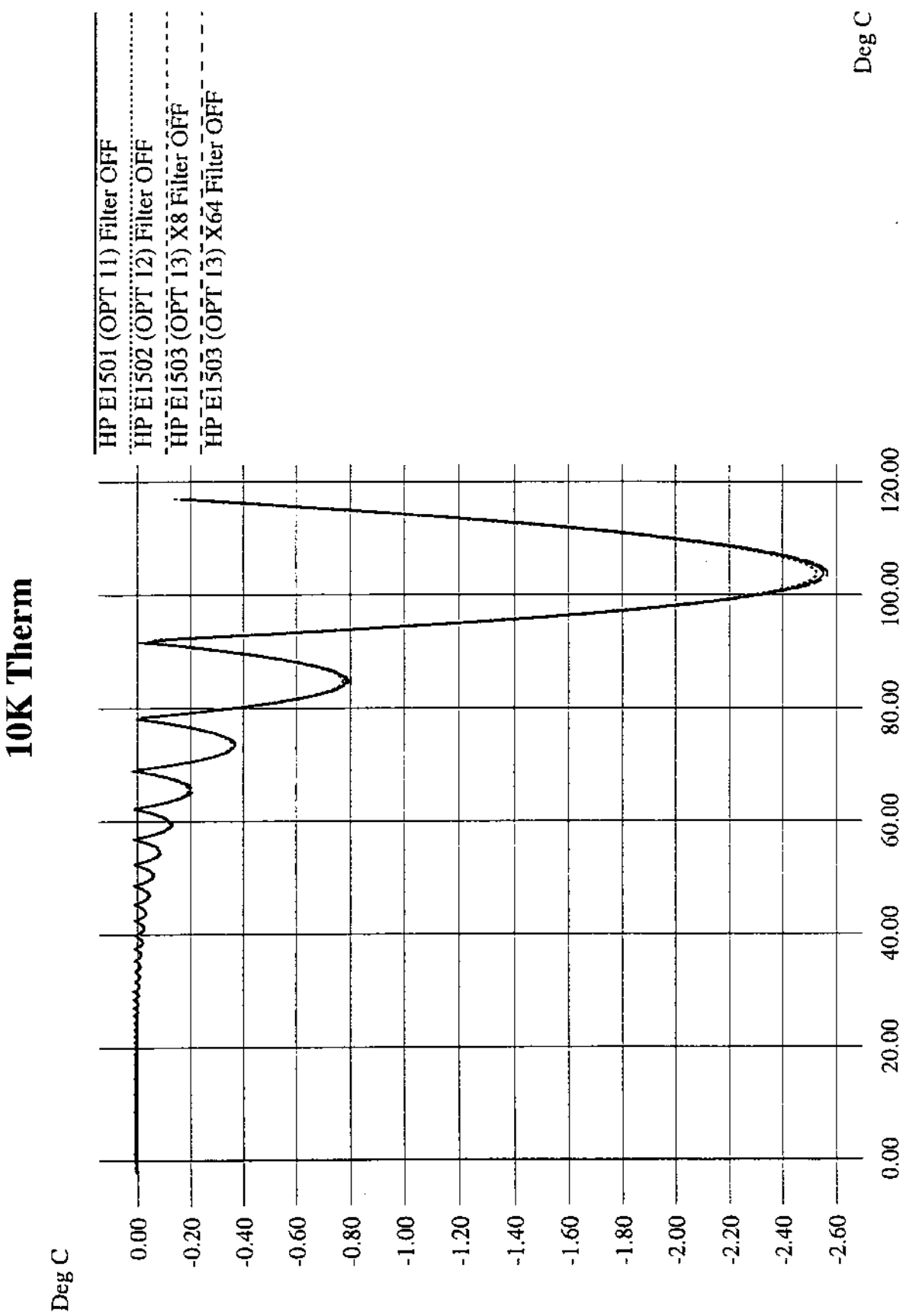


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5K Therm

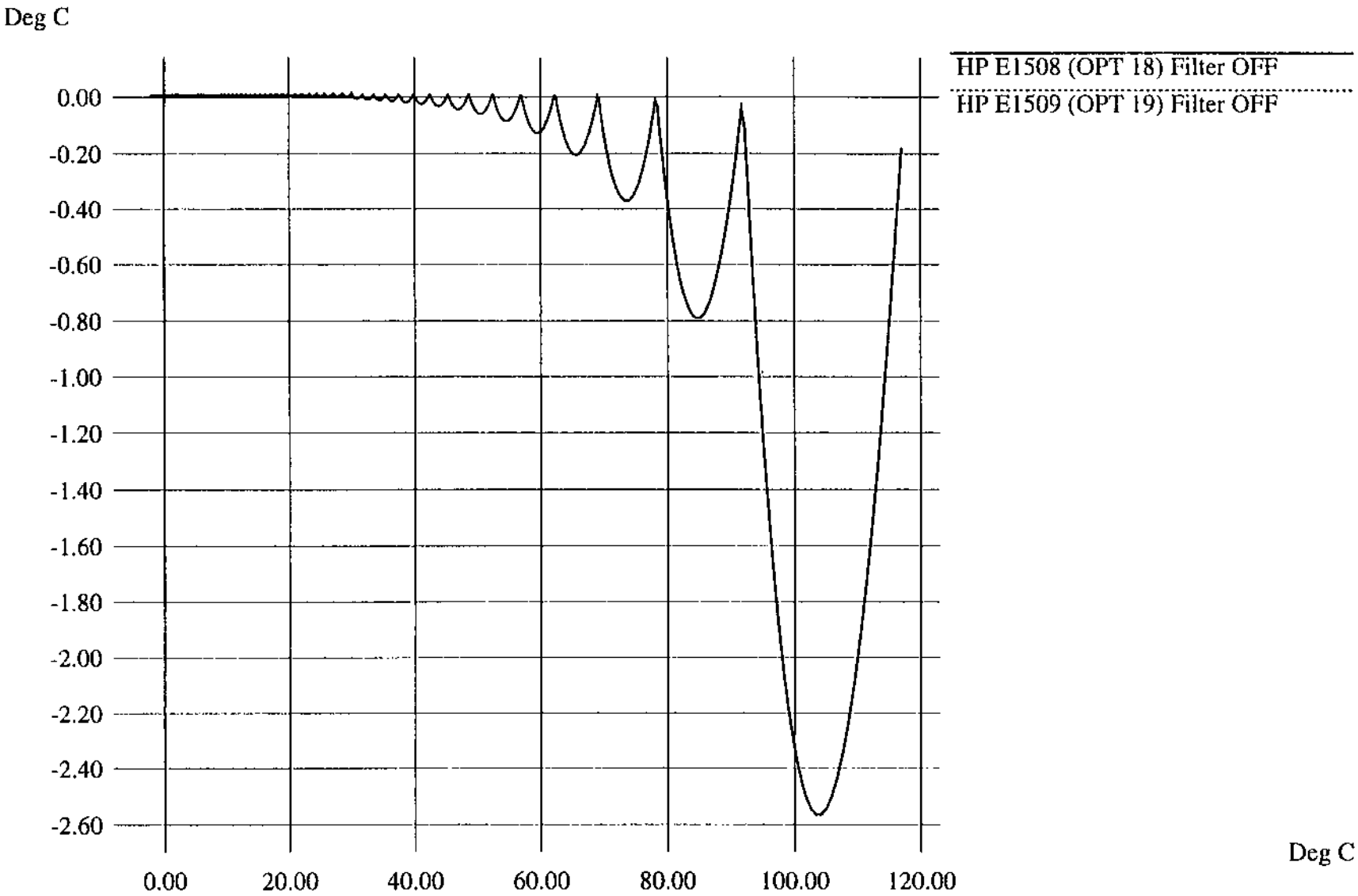




Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10K Therm



Notes

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix B
Error Messages

Possible Error Messages:

-108	’Parameter not allowed’.
-109	’Missing parameter’
-160	’Block data error’.
-211	’Trigger ignored’.
-212	’Arm ignored’.
-213	’Init ignored’.
-221	’Settings conflict’.
-222	’Data out of range’.
-224	’Illegal parameter value’.
-240	’Hardware error’. Execute *TST?.
-253	’Corrupt media’.
-281	’Cannot create program’.
-282	’Illegal program name’.
-310	’System error’.
-410	’Query INTERRUPTED’.
1000	’Out of memory’
2001	’Invalid channel number’.
2003	’Invalid word address’.
2007	’Bus error’.
2008	’Scan list not initialized’.
2009	’Too many channels in channel list’.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2016	’Byte count is not a multiple of two’.
3000	’Illegal while initiated’. Operation must be performed before INIT or INIT:CONT ON.
3004	’Illegal command. CAL:CONF not sent’. Incorrect sequence of calibration commands. Send CAL:CONF:VOLT command before CAL:VAL:VOLT and send CAL:CONF:RES command before CAL:VAL:RES
3005	’Illegal command. Send CAL:VAL:RES’. The only command accepted after a CAL:CONF:RES is a CAL:VAL:RES command.
3006	’Illegal command. Send CAL:VAL:VOLT’. The only command accepted after a CAL:CONF:VOLT is a CAL:VAL:VOLT command.
3007	’Invalid signal conditioning module’. The command sent to an SCP was illegal for its type.
3008	’Too few channels in scan list’. A Scan List must contain at least two channels.
3012	’Trigger too fast’. Scan list not completed before another trigger event occurs.
3015	’Channel modifier not permitted here’.
3019	’TRIG:TIM interval too small for SAMP:TIM interval and scan list size’. TRIG:TIM interval must allow for completion of entire scan list at currently set SAMP:TIM interval. See TRIG:TIM in Chapter 5, the Command Reference
3020	’Input overvoltage’. Calibration relays opened (if JM2202 not cut) to protect module inputs, and Questionable Data Status bit 11 set. Execute *RST to close relays and/or reset status bit.
3021	’FIFO overflow’. Lets you know that the FIFO buffer has filled and that one or more readings have been lost. Usually caused by algorithm values stored in FIFO faster than FIFO was read.
3026	’Calibration failed’.
3027	’Unable to map A24 VXI memory’.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3028	’Incorrect range value’. Range value sent is not supported by instrument.
3030	’Command not yet implemented!!’.
3032	’0x1: DSP-Unrecognized command code’.
3033	’0x2: DSP-Parameter out of range’.
3034	’0x4: DSP-Flash rom erase failure’.
3035	’0x8: DSP-Programming voltage not present’.
3036	’0x10: DSP-Invalid SCP gain value’. Check that SCP is seated or replace SCP. Channel numbers are in FIFO.
3037	’0x20: DSP-Invalid *CAL? constant or checksum. *CAL? required.’.
3038	’0x40: DSP-Couldn’t cal some channels’. Check that SCP is seated or replace SCP. Channel numbers are in FIFO.
3039	’0x80: DSP-Re-Zero of ADC failed’.
3040	’0x100: DSP-Invalid Tare CAL constant or checksum’. Perform CAL:TARE - CAL:TARE? procedure.
3041	’0x200: DSP-Invalid Factory CAL constant or checksum’. Perform A/D Cal procedure.
3042	’0x400: DSP-DAC adjustment went to limit’. Execute *TST?
3043	’0x800: DSP Status--Do *CAL?’.
3044	’0x1000: DSP-Overvoltage on input’.
3045	’0x2000: DSP-reserved error condition’.
3046	’0x4000: DSP-ADC hardware failure’.
3047	’0x8000: DSP-reserved error condition’.
3048	’Calibration or Test in Process’.
3049	’Calibration not in Process’.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3050 ’ZERO must be sent before FSCale’. Perform A/D Cal sequence as shown in Command Reference under CAL:CONF:VOLT

3051 ’Memory size must be multiple of 4’. From MEM:VME:SIZE. Each HP E1419 reading requires 4 bytes.

3052 ’Self test failed. Test info in FIFO’. Use SENS:DATA:FIFO:ALL? to retrieve data from FIFO.

NOTE: *TST? always sets the FIFO data FORMat to ASCII,7. Read FIFO data into string variables.

Meaning of *TST? FIFO data by Value	
FIIFO Value	Definition
1 - 99	ID number of failed test (see following table for possible corrective actions)
100 - 163	channel number(s) associated with test (ch 0-63)
164	special "channel" used for A/D tests only
200	A/D range 0.0625V associated with failed test
201	A/D range 0.25V associated with failed test
202	A/D range 1V associated with failed test
203	A/D range 4V associated with failed test
204	A/D range 16V associated with failed test

Possible Corrective Action by Failed Test ID Number	
Test ID	Corrective Actions
1 - 19, 21 - 29	(HP Service)*
20, 30 -37	Remove all SCPs and see if *TST? passes. If so, replace SCPs one at a time until you find the one causing the problem.
38 - 71	(HP Service)*
72,74 - 76, 80 - 93, 301 - 354	re-seat the SCP that the channel number(s) points to, or move the SCP and see if the failure(s) follow the SCP. If the problems move with the SCP, replace the SCP.
73, 77 - 79, 94 - 99	(HP Service)*

*Must send module to an HP Service Center for repair. Record information found in FIFO to assist the HP Service Center in repairing the problem.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Refer to the Command Reference under *TST? for a list of module functions tested.

NOTE During the first 5 minutes after power is applied, *TST? may fail. Allow the module to warm-up before executing *TST?

3053	'Corrupt on board Flash memory'.
3056	'Custom EU not loaded'. May have erased custom EU conversion table with *RST. May have linked channel with standard EU after loading custom EU, this erases the custom EU for this channel. Reload custom EU table using DIAG:CUST:LIN or DIAG:CUST:PIEC.
3057	'Invalid ARM or TRIG source when S/H SCP's enabled' Don't set TRIG:SOUR or ARM:SOUR to SCP with HP E1510 or HP E1511 installed.
3058	'Hardware does not have D32, S/H, or new trigger capabilities'. Module's serial number is earlier than 3313A00530.
3067	'Multiple attempts to erase Flash Memory failed'
3068	'Multiple attempts to program Flash Memory failed'
3069	'Programming voltage jumper not set properly'. See Disabling Flash Memory Access in Chapter 1 (JM2201)
3070	'Identification of Flash ROM incorrect'
3071	'Checksum error on Flash Memory'
3074	'WARNING! Old Opt 16 or Opt 17 card can damage SCP modules' must use HP E1506 or HP E1507.
3075	'Too many entries in CVT list'
3076	'Invalid entry in CVT list' Can only be 10 to 511
3077	'Too many updates in queue. Must send UPDATE command' To allow more updates per ALG:UPD, increase ALG:UPD:WINDOW

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3078	’Invalid Algorithm name’ Can only be ’ALG1’ through ’ALG32’, or ’GLOBALS’, or ’MAIN’
3079	’Algorithm is undefined’ In ALG:SCAL, ALG:SCAL?, ALG:ARR, or ALG:ARR?
3080	’Algorithm already defined’ Trying to repeat ALG:DEF with same <*alg_name*> (and is not enabled to swap), or trying to define ’GLOBALS’ again since last *RST
3081	’Variable is undefined’ Algorithm exists but has no local variable by that name.
3082	’Invalid Variable name’ Must be valid ’C’ identifier, see Chapter 5
3083	’Global symbol (variable or custom function) already defined’ Trying to define a global variable with same name as a user defined function, or vice versa. User functions are also global.
3084	’Algorithmic error queue full’ ALG:DEF has generated too many errors from your algorithm source code
3084	"Error 1: Number too big for a 32 bit float" "Error 2: Number too big for a 32 bit integer" "Error 3: ’8’ or ’9’ not allowed in an octal number" "Error 4: Syntax error" "Error 5: Expecting ’(’" "Error 6: Expecting ’)’" "Error 7: Expecting an expression" "Error 8: Out of driver memory" "Error 9: Expecting a bit number (Bn or Bnn)" "Error 10: Expecting ’]’" "Error 11: Expecting an identifier" "Error 12: Arrays can’t be initialized" "Error 13: Expecting ’static’" "Error 14: Expecting ’float’" "Error 15: Expecting ’;’" "Error 16: Expecting ’,’" "Error 17: Expecting ’=’" "Error 18: Expecting ’{’" "Error 19: Expecting ’}’" "Error 20: Expecting a statement" "Error 21: Expecting ’if’" "Error 22: Can’t write to input channels" "Error 23: Expecting a constant expression" "Error 24: Expecting an integer constant expression" "Error 25: Reference to an undefined variable" "Error 26: Array name used in a scalar context" "Error 27: Scalar name used in an array context"

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

"Error 28: Variable name used in a custom function context"
"Error 29: Reference to an undefined custom function"
"Error 30: Can’t have executable code in GLOBALS definition"
"Error 31: CVT address range is 10 - 511"
"Error 32: Numbered algorithms can only be called from MAIN"
"Error 33: Reference to an undefined algorithm"
"Error 34: Attempt to redefine an existing symbol (var or fn)"
"Error 35: Array size is 1 - 1024"
"Error 36: Expecting a default PID parameter"
"Error 37: Too many FIFO or CVT writes per scan trigger"
"Error 38: Statement is too complex"
"Error 39: Unterminated comment"

3085	’Algorithm too big’ Algorithm exceded 46K words (23K if enabled to swap), or exceded size specified in <swap_size>.
3086	’Not enough memory to compile Algorithm’ Your algorithm’s constructs are using too much translator memory. Need more memory in your HP E1406. Try breaking your algorithm into smaller algorithms.
3088	’Too many functions’ Limit is 32 user defined functions
3089	’Bad Algorithm array index’ Must be from 0 to (declared size)-1
3090	’Algorithm Compiler Internal Error’ Call HP with details of operation.
3091	’Illegal while not initiated’ Send INIT before this command
3092	’No updates in queue’
3093	’Illegal Variable Type’ Sent ALG:SCAL with identifier of array, ALG:ARR with scalar identifier, ALG:UPD:CHAN with identifier that is not a channel, etc.
3094	’Invalid Array Size’ Must be 1 to 1024
3095	’Invalid Algorithm Number’ Must be ’ALG1’ to ’ALG32’

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3096	’Algorithm Block must contain termination ’ Must append a null byte to end of algorithm string within the Block Data
3097	’Unknown SCP. Not Tested’ May receive if you are using a breadboard SCP
3099	’Invalid SCP for this product’
3100	’Analog Scan time to big. Too much settling time’ Count of channels referenced by algorithms combined with use of SENS:CHAN:SETTLING has attempted to build an analog Scan List greater than 64 channels.
3101	’Can’t define new algorithm while running’ Execute ABORT, then define algorithm
3102	’Need ALG:UPD before redefining this algorithm again’ Already have an algorithm swap pending for this algorithm.
3103	’Algorithm swapping already enabled; Can’t change size’ Only send <*swap_size*> parameter on initial definition.
3104	’GLOBALS can’t be enabled for swapping’ Don’t send <*swap_size*> parameter for ALG:DEF ’GLOBALS’

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix C
Glossary

The following terms have special meaning when related to the HP E1419.

Algorithm — In general, an algorithm is a tightly defined procedure that performs a task. This manual, uses the term to indicate a program executed within the HP E1419 that implements a data acquisition and control algorithm.

Algorithm Language — The algorithm programming language specific to the HP E1419. This programming language is a subset of the ANSI 'C' language.

Application Program — The program that runs in the VXIbus controller, either embedded within the VXIbus mainframe, or external and interfaced to the mainframe. The application program typically sends SCPI commands to configure the HP E1419, define its algorithms, then start the algorithms running. Typically, once the HP E1419 is running algorithms, the application need only "oversee" the control application by monitoring the algorithms' status. During algorithm writing, debugging, and tuning, the application program can retrieve comprehensive data from running algorithms.

Buffer — In this manual, a buffer is an area in RAM memory that is allocated to temporarily hold:

- Data input values that an algorithm will later access. This is the Input Channel Buffer.
- Data output values from an algorithm until these values are sent to hardware output channels. This is the Output Channel Buffer.
- Data output values from an algorithm until these values are read by your application program. This is the First-In-First-Out or FIFO buffer.
- A second copy of an array variable containing updated values until it is "activated" by an update. This is "double buffering".
- A second version of a running algorithm until it is "activated" by an update. This is only for algorithms that are enabled for swapping. This is also "double buffering".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Control Processor	The Digital Signal Processor (DSP) chip that performs all of the HP E1419's internal hardware control functions as well as performing the EU Conversion process.
DSP	Same as Control Processor
EU	Engineering Units
EU Conversion	Engineering Unit Conversion: Converting binary A/D readings (in units of A/D counts) into engineering units of voltage, resistance, temperature, strain. These are the "built in" conversions (see SENS:FUNC: ...). The HP E1419 also provides access to custom EU conversions (see SENS:FUNC:CUST in command reference and "Creating and Loading Custom EU Tables" in Chapter 3).
FIFO	The First-In-First-OUT buffer that provides output buffering for data sent from an algorithm to an application program.
Flash or Flash Memory	Non-volatile semiconductor memory used by the HP E1419 to store its control firmware and calibration constants
Scan List	A list of up to 64 channels that is built by the HP E1419. Channels referenced in algorithms are placed in the Scan List as the algorithm is defined. This list will be scanned each time the module is triggered.
SCP	Signal Conditioning Plug-on: Small circuit boards that plug onto the HP E1419's main circuit board. Available analog input SCPs can provide noise canceling filters, signal amplifiers, signal attenuators, and strain bridge completion. Analog output SCPs are available to provide measurement excitation current, controlling voltage, and controlling current. Digital SCPs are available to both read and write digital states, read frequency and counts, and output modulated pulse signals (FM and PWM).
Swapping	This term applies to algorithms that are enabled to swap. These algorithms can be exchanged with another of the same name while the original is running. The "new" algorithm becomes active after an update command is sent. This "new" algorithm may again be swapped with another, and so on. This capability allows changing algorithm operation without stopping and leaving this and perhaps other processes without control.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Terminal Blocks	The screw-terminal blocks you connect your system field wiring to. The terminal blocks are inside the Terminal Module
Terminal Module	The plastic encased module which contains the terminal blocks you connect your field wiring to. The Terminal Module then is plugged into the HP E1419's front panel.
Update	This is an intended change to an algorithm, algorithm variable, or global variable that is initiated by one of the commands ALG:SCALAR, ALG:ARRAY, ALG:DEFINE, ALG:SCAN:RATIO, or ALG:STATE. This change or "update" is considered to be pending until an update command is received. Several updates can be sent to the Update Queue, waiting for an update command to cause them to take effect synchronously. The update commands are ALG:UPDATE, and ALG:UPD:CHANNEL.
Update Queue	A list of scalar variable values, and/or buffer pointer values (for arrays, and swapping algorithms) that is built in response to updates (see Update). When an update command is sent, scalar values and pointer values are sent to their working locations.
User Function	A function callable from the Algorithm Language in the general form <*function_name*>**(** <*expression*> **)**. These user defined functions provide advanced mathematical capability to the Algorithm Language

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix D
Wiring and Noise Reduction Methods

Separating Digital and Analog SCP Signals

Signals with very fast rise time can cause interference with nearby signal paths. This is called cross-talk. Digital signals present this fast rise-time situation. Digital I/O signal lines that are very close to analog input signal lines can inject noise into them.

To minimize cross-talk you can maximize the distance between analog input and digital I/O signal lines. Figure D-1 shows that by installing analog input SCPs in positions 0 through 3, and analog output and digital I/O SCPs in positions 4 through 7, you can keep these types of signals separated by the width of the HP E1419 module. The signals are further isolated because they remain separated on the connector module as well. Note that in Figure D-1 , even though only 7 of the eight SCP positions are filled, the SCPs present are not installed contiguously, but are arranged to provide this digital/analog separation.

If you have to mix analog input and digital I/O SCPs on the same side, the following suggestions will help provide quieter analog measurements.

- Use analog input SCPs that provide filtering on the mixed side.
- Route only high level analog signals to the mixed side.

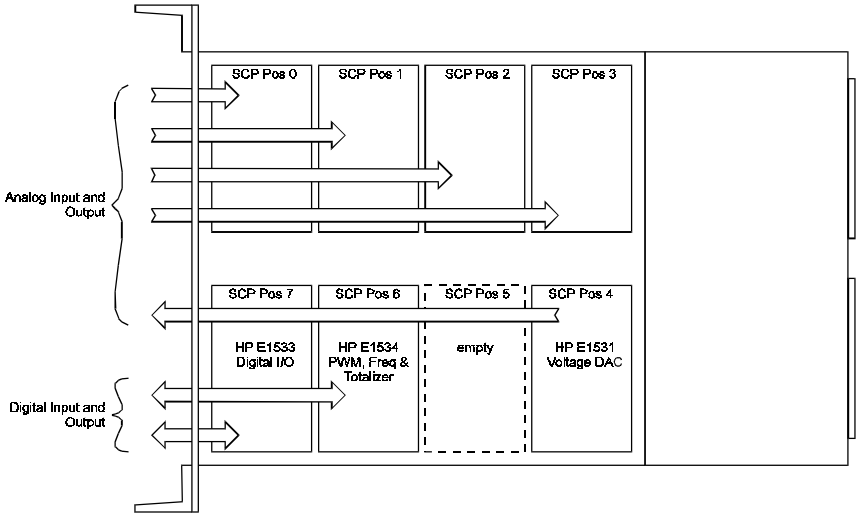


Figure D-1 Separating Analog and Digital Siganls

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Recommended Wiring and Noise Reduction Techniques

Unshielded signal wiring is very common in Data Acquisition applications. While this worked well for low speed integrating A/D measurements and/or for measuring high level signals, it does not work for high speed sampling A/Ds, particularly when measuring low level signals like thermocouples or strain gage bridge outputs. Unshielded wiring will pick up environmental noise, causing measurement errors. Shielded, twisted pair signal wiring, although it is expensive, is required for these measurements unless an even more expensive amplifier-at-the- signal-source or individual A/D at the source is used.

Generally, the shield should be connected to ground at the DUT and left open at the HP E1419. Floating DUTs or transducers are an exception. Connect the shield to HP E1419 GND or GRD terminals for this case, whichever gives the best performance. This will usually be the GND terminal. A single point shield to ground connection is required to prevent ground loops. This point should be as near to the noise source as possible and this is usually at the DUT.

Wiring Checklist

The following lists some recommended wiring techniques.

1. Use individually shielded, twisted-pair wiring for each channel.
2. Connect the shield of each wiring pair to the corresponding Guard (G) terminal on the Terminal Module .
3. The Terminal Module is shipped with the Ground-Guard (GND-GRD) shorting jumper installed for each channel. These may be left installed or removed, dependent on the following conditions:
   a. **Grounded Transducer with shield connected to ground at the transducer:** Low frequency ground loops (DC and/or 50/60Hz) can result if the shield is also grounded at the Terminal Module end. To prevent this, remove the GND-GRD jumper for that channel.
   b. **Floating Transducer with shield connected to the transducer at the source:** In this case, the best performance will most likely be achieved by leaving the GND-GRD jumper in place.
4. In general, the GND-GRD jumper can be left in place unless it is necessary to break low frequency (below 1 kHz) ground loops.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HP E1419 Guard Connections

The HP E1419 guard connection provides a 10 KΩ current limiting resistor between the guard terminals (G) and E1419 chassis ground for each 8 channel SCP bank. This is a safety device for the case where the Device Under Test (DUT) isn't actually floating, the shield is connected to the DUT and also connected to the HP E1419 guard terminal (G). The 10 KΩ resistor limits the ground loop current, which has been known to burn out shields. This also provides 20 KΩ isolation between shields between SCP banks which helps isolate the noise source.

Common Mode Voltage Limits

You must be very careful not to exceed the maximum common mode voltage referenced to the card chassis ground of ±16 volts (±60 volts with the HP E1513A Attenuator SCP). There is an exception to this when high frequency (1 kHz - 20 kHz) common mode noise is present (see "HP E1419 Noise Rejection" below). Also, if the DUT is not grounded, then the shield should be connected to the E1419 chassis ground.

When to Make Shield Connections

It is not always possible to state positively the best shield connection for all cases. Shield performance depends on the noise coupling mechanism which is very difficult to determine. The above recommendations are usually the best wiring method, but if feasible, experiment with shield connections to determine which provides the best performance for your installation and environment.

NOTE For a thorough, rigorous discussion of measurement noise, shielding, and filtering, see "Noise Reduction Techniques in Electronic Systems" by Henry W. Ott of Bell Laboratories, published by Wiley & Sons, ISBN 0-471-85068-3.

Noise Due to Inadequate Card Grounding

If either or both of the HP E1419 and HP E1482 (MXI Extender Modules) are not securely screwed into the VXIbus Mainframe, noise can be generated. Make sure that both screws (top and bottom) are screwed in tight. If not, it is possible that CVT data could be more noisy than FIFO data because the CVT is located in A24 space, the FIFO in A16 space; more lines moving could cause noisier readings.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HP E1419 Noise Rejection

See Figure D-2 for the following discussion.

Normal Mode Noise (Enm)

This noise is actually present at the signal source and is a differential noise (Hi to Lo). It is what is filtered out by the buffered filters on the HP E1502, E1503, E1508, and E1509 SCPs.

Common Mode Noise (Ecm)

This noise is common to both the Hi and Lo differential signal inputs. Low frequency Ecm is very effectively rejected by a good differential instrumentation amplifier, and it can be averaged out when measured through the Direct Input SCP (HP E1501). However, high frequency Ecm is rectified and generates an offset with the amplifier and filter SCPs (such as HP E1502, HP E1503, HP E1508, and HP E1509). This is since these SCPs have buffer-amplifiers on board and is a characteristic of amplifiers. The best way to deal with this is to prevent the noise from getting into the amplifier.

Keeping Common Mode Noise out of the Amplifier

Most common mode noise is about 60 Hz, so the differential amplifier rejection is very good. The amplifier Common Mode Noise characteristics are:

120 dB flat to 300 Hz, then 20 dB/octave rolloff

The HP E1419 amplifiers are selected for low gain error, offset, temperature drift, and low power. These characteristics are generally incompatible with good high frequency CMR performance. More expensive, high performance amplifiers can solve this problem, but since they aren't required for many systems, HP elected to handle this with the High Frequency Common Mode Filter option to the HP E1586A Remote Rack Panel (HP E1586 Option 001, RF Filter) discussed below.

Shielded, twisted pair lead wire generally does a good job of keeping high frequency common mode noise out of the amplifier, provided the shield is connected to the HP E1419 chassis ground through a very low impedance. (Not via the guard terminal - The HP E1419 guard terminal connection shown in the HP E1419 User's manual does not consider the high frequency Ecm problem, and is there to limit the shield current and to allow the DUT to float up to some DC common mode voltage subject to the maximum ±16 volt input specification limit.

This conflicts with the often recommended good practice of grounding the shield at the signal source and only at that point to eliminate line frequency ground loops, which can be high enough to burn up a shield. We

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

recommend that you follow this practice, and if you see high frequency common mode noise (or suspect it), tie the shield to the HP E1419 ground through a 0.1 µF capacitor. At high frequencies, this drives the shield voltage to 0 volts at the HP E1419 input. Due to inductive coupling to the signal leads, the Ecm voltage on the signal leads is also driven to zero.

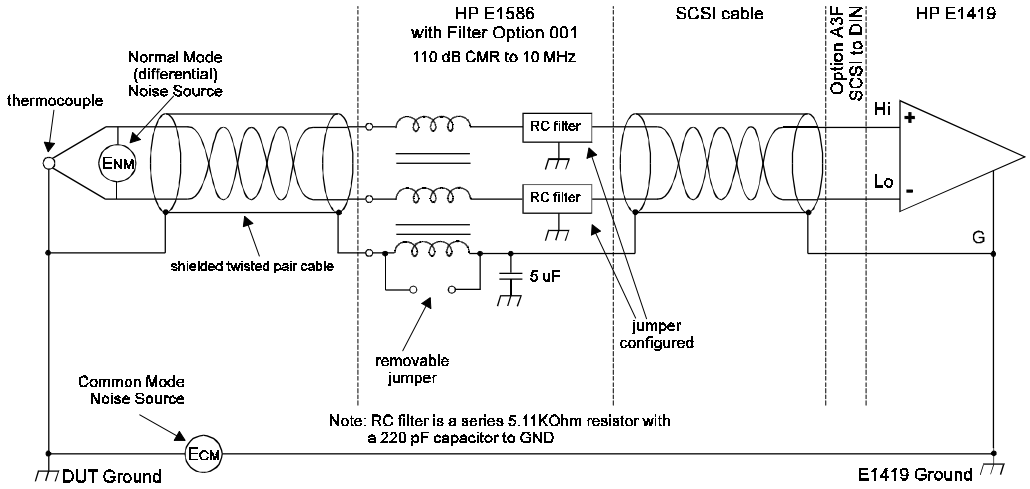


Figure D-2 HF Common Mode Filters

Reducing Common Mode Rejection Using Tri-Filar Transformers

One HP E1413 customer determined that greater than 100 dB CMR to 10 MHz was required to get good thermocouple (TC) measurements in his test environment. To accomplish this requires the use of tri-filar transformers which are an option to the HP E1586A Remote Rack Terminal Panel. (This also provides superior isothermal reference block performance for thermocouple measurements.) This works by virtue of the inductance in the shield connected winding presenting a significant impedance to high frequency common mode noise and forcing all the noise voltage to be dropped across the winding. The common mode noise at the input amplifier side of the winding is forced to 0 volts by virtue of the low impedance connection to the HP E1419 ground via the selectable short or parallel combination of 1 kΩ and 0.1 µF. The short can't be used in situations where there is a very high common mode voltage, (DC and/or AC) that could generate very large shield currents.

The tight coupling through the transformer windings into the signal Hi and Low leads, forces the common mode noise at the input amplifier side of those windings to 0 volts. This achieves the 110 dB to 10 MHz desired, keeping the high frequency common mode noise out of the amplifier, thus preventing the amplifier from rectifying this into an offset error.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This effectively does the same thing that shielded, twisted pair cable does, only better. It is especially effective if the shield connection to the HP E1419 ground can’t be a very low impedance due to large DC and/or low frequency common mode voltages.

The tri-filar transformers don’t limit the differential (normal mode) signal bandwidth. Thus, removing the requirement for "slowly varying signal voltages". The nature of the tri-filar transformer, or, more accurately, common-mode inductor, is that it provides a fairly high impedance to common mode signals, and a quite low impedance to differential mode signals. The ratio of common-mode impedance to differential-mode impedance for the transformer we use is ~ 3500:1. Thus, there is NO differential mode bandwidth penalty incurred by using the tri-filar transformers.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix E
Generating User Defined Functions

Introduction.

The HP E1419A MultifunctionPlus Measurement and Control Module has a limited set of mathematical operations such as add, subtract, multiply, and divide. Many control applications require functions such a square root for calculating flow rate or a trigonometric function to correctly transition motion of an actuator from a start to ending position. In order to represent a sine wave or other transcendental functions, one could use a power series expansion to approximate the function using a finite number of algebraic expressions. Since the above mentioned operations can take from 1.5usec to 4usec for each floating point calculation, a complex waveform such as sine(x) could take more than 100usec to get the desired result. A faster solution is desirable and available.

The HP E1419A provides a solution to approximating such complex waveforms by using a piece-wise linearization of virtually any complex waveform. The technique is simple. The CD ROM supplied with your HP E1419A contains the HP VEE program "fn_1419.vee" that builds user defined functions and loads them into the HP E1419A. The VEE module calls a 'C' function (source code supplied) that actually calculates 128 Mx+B segments over a specified range of values for the desired function. You supply the function; the program generates the segments in a table. The "fn_1419.vee" program can be merged into your HP VEE application program. Another HP VEE example program "eufn1419.vee" shows you how to apply "fn_1419.vee". Up to 32 functions can be created for use in algorithms. At runtime where the function is passed an 'x' value, the time to calculate the Mx+B function is approximately 18µsec.

The HP E1419A actually uses this technique to convert volts to temperature, strain, etc. The accuracy of the approximation is really based upon how well you select the range over which the table is built. For thermocouple temperature conversion, the HP E1419A fixes the range to the lowest A/D range(+/-64millivolts) so that small microvolt measurements yield the proper resolution of the actual temperature for a non-linear transducer. In addition, the HP E1419A permits you to create Custom Engineering Unit conversion for your transducer so that when the voltage measurement is actually made the EU conversion takes place(see SENS:FUNC:CUST). Algorithms deal with the resulting floating point numbers generated during the measurement phase and may require further complex mathematical operations to achieve the desired result.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

With some complex waveforms, you may actually want to break up the waveform into several functions in order to get the desired accuracy. For example, suppose you need to generate a square root function for both voltage and strain calculations. The voltages are only going to range from 0 to +/-16volts, worst case. The strain measurements return numbers in microstrain which range in the 1000's. Trying to represent the square root function over the entire range would severely impact the accuracy of the approximation. Remember, the entire range is broken up into only 128 segments of Mx+B operations. If you want accuracy, you MUST limit the range over which calculations are made. Many transcendental functions are simply used as a scaling multiplier. For example, a sine wave function is typically created over a range of 360 degrees or 2*PI radians. After which, the function repeats itself. It's a simple matter to make sure the 'x' term is scaled to this range before calculating the result. This concept should be used almost exclusively to obtain the best results.

Haversine Example.

The following is an example of creating a haversine function (a sine wave over the range of -PI/2 to PI/2). The resulting function represents a fairly accurate approximation of this non-linear waveform when you limit the range as indicated. Since the tables must be built upon binary boundaries(i.e. .125, .25, .5, 1, 2, 4, etc.) and since PI/2 is a number greater than 1 but less than 2, the next binary interval to include this range will be 2. Another requirement for building the table is that the waveform range MUST be centered around 0(i.e. symmetrical about the X-axis). If the desired function is not defined on one side or the other of the Y-axis, then the table is right or left shifted by the offset from X=0 and the table values are calculated correctly, but the table is built as though it were centered about the X-axis. For the most part, you can ignore these last couple of sentences if it does not make sense to you. The only reason its brought up here is that your accuracy may suffer the farther away from the X=0 point you get unless you understand what resolution is available and how much non-linearity is present in your waveform. We'll talk about that in the "Limitations" section, later.

Figure 1 shows the haversine function as stated above. This type of waveform is typical of the kind of acceleration and deceleration one wants when moving an object from one point to another. The desired beginning point would be the location at -PI/2 and the ending point would be at PI/2. With the desired range spread over +/- PI/2, the 128 segments are actually divided over the range of +/- 2. Therefore, the 128 Mx+B line segments are divided equally on both sides of X=0: 64 segments for 0..2 and 64 segments for -2..0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

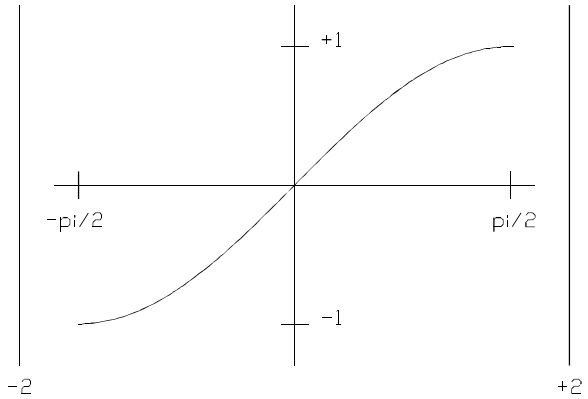


Figure 1 A Haversine Function

A typical use of this function would be to output an analog voltage or current at each Scan Trigger of the HP E1419A and over the range of the haversine. For example, suppose you wanted a new position of an analog output to move from 1ma to 3ma over a period of 100msec. If your TRIG:TIMER setting or your EXTernal trigger was set to 2msec, then you would want to force 50 intervals over the range of the haversine. This can be easily done by using a scalar variable to count the number of times the algorithm has executed and to scale the variable value to the -PI/2 to PI/2 range. 3ma is multiplied times the custom function result over each interval which will yield the shape of the haversine (.003*sin(x)+.001).

Try the "eufn1419.vee" example program and define some custom functions to illustrate this discussion.

Table 1 shows some examples of the accuracy of the custom function with various input values compared to an evaluation of the actual transcendental function found in 'C' or RMB. Please note that the Mx+B segments are located on boundaries specified by 2/64 on each side of X=0. This means that if you select the exact input value that was used for the beginning of each segment, you WILL get exactly the calculated value of that function at that point. Any point between segments will be an approximation dependent upon the linearity of that segment. Also note that values of X = 2 and X = -2 will result in Y=infinity.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

'C' sin(-1.570798)	-1.000000	'HP E1419A' sin(-1.570798)	-0.999905
'C' sin(-1.256639)	-0.951057	'HP E1419A' sin(-1.256639)	-0.950965
'C' sin(-0.942479)	-0.809018	'HP E1419A' sin(-0.942479)	-0.808944
'C' sin(-0.628319)	-0.587786	'HP E1419A' sin(-0.628319)	-0.587740
'C' sin(-0.314160)	-0.309017	'HP E1419A' sin(-0.314160)	-0.308998
'C' sin(0.000000)	0.000000	'HP E1419A' sin(0.000000)	0.000000
'C' sin(0.314160)	0.309017	'HP E1419A' sin(0.314160)	0.308998
'C' sin(0.628319)	0.587786	'HP E1419A' sin(0.628319)	0.587740
'C' sin(0.942479)	0.809018	'HP E1419A' sin(0.942479)	0.808944
'C' sin(1.256639)	0.951057	'HP E1419A' sin(1.256639)	0.950965
'C' sin(1.570798)	1.000000	'HP E1419A' sin(1.570798)	0.999905

Table 1. 'C' Sin(x) Vs. HP E1419A Haversine Function for Selected Points.

Limitations.

As stated earlier, there are limitations to using this custom function technique. These limitations are directly proportional to the non-linearity of the desired waveform. For example, suppose you wanted to represent the function X*X*X over a range of +/-1000. The resulting binary range would be +/-1024, and the segments would be partitioned at 1024/64 intervals. This means that every 16 units would yield an Mx+B calculation over that segment. As long as you input numbers VERY close to those cardinal points, you will get good results. Strictly speaking, you will get perfect results if you only calculate at the cardinal points, which may be reasonable for your application if you limit your input values to exactly those 128 points.

You may also shift the waveform anywhere along the X-axis, and Build_table() will provide the necessary offset calculations to generate the proper table. Be aware too that shifting the table out to greater magnitudes of X may also impact the precision of your results dependent upon the linearity of your waveform. Suffice it to say that you will get your best results and it will be easiest for you to grasp what your doing if you stay near the X=0 point since most of the results of your measurements will have 1e-6..16 values for volts.

One final note. You may see truncation errors in the fourth digit of your results. This is because only 15 bits of your input value is sent to the function. This occurs because the same technique used for Custom EU

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

conversion is used here, and the method assumes input values are from the 16 bit A/D (15 bits = sign bit). This is evident in Table 1 where the first and last entries return ±0.9999 rather than ±1. For most applications this accuracy should be more than adequate.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Index

!

*CAL?, how to use, 72
*RST
    and power-on defaults, 53
4-20 mA, adding sense circuits for, 41

A

A common error to avoid, 120
A complete thermocouple measurement command sequence, 65
A very simple first algorithm, 124
Abbreviated Commands, 182
ABORt subsystem, 189
abs(expression), 128
Access, bitfield, 131
Accessing I/O channels, 114
Accessing the E1419A's resources, 113
Accessories
    Rack Mount Terminal Panel, 44
Accuracy
    DC Volts, 334
    Sample timer, 333
    Temperature, 335
Adding settling delay for specific channels, 106
Adding terminal module components, 41
Additive-expression:, 134
Additive-operator:, 134
ADDRess
    MEM:VME:ADDR, 246
ADDRess?
    MEM:VME:ADDR?, 246
After INIT, 53
ALG:DEFINE in the programming sequence, 121
ALG:DEFINE's three data formats, 121
Algorithm definition, 75
Algorithm execution order, 120
Algorithm Language reference, 126
Algorithm language statement
    writecvt(), 116
    writefifo(), 117
Algorithm, A very simple first, 124
Algorithm, data acquisition, 125
Algorithm, exiting the, 128
Algorithm, process monitoring, 125
Algorithm, running the, 124
Algorithm, writing the, 124
Algorithm-definition:, 137
ALGorithm:FUNCtion:DEFine, 201
ALGorithm:OUTPut:DELay, 203
ALGorithm:OUTPut:DELay?, 204
ALGorithm:UPDate:CHANnel, 205
ALGorithm:UPDate:WINDow, 206
ALGorithm:UPDate:WINDow?, 207
ALGorithm:UPDate[:IMMediate], 204
ALGorithm[:EXPLicit]:ARRay, 191
ALGorithm[:EXPLicit]:ARRay?, 192
ALGorithm[:EXPLicit]:DEFine, 192
ALGorithm[:EXPLicit]:SCALar, 196
ALGorithm[:EXPLicit]:SCALar?, 197
ALGorithm[:EXPLicit]:SCAN:RATio, 197
ALGorithm[:EXPLicit]:SCAN:RATio?, 198
ALGorithm[:EXPLicit]:SIZe?, 199
ALGorithm[:EXPLicit]:TIMe?, 201
ALGorithm[:EXPLicit][:STATe], 199
ALGorithm[:EXPLicit][:STATe]?, 200
Algorithms
    disabling, 87
    enabling, 87
Algorithms, defining, 120
Algorithms, INITiating/Running, 81
Algorithms, non-control, 125
Algorithms, starting, 81
ALL?
    DATA:FIFO:ALL?, 266
AMPLitude
    OUTP:CURRent:AMPLitude, 249
    OUTPut:CURRent:AMPLitude?, 250
An example using the operation group, 93
APERture
    SENSe:FREQuency:APERture, 270
APERture?
    SENSe:FREQuency:APERture?, 271
Arithmetic operators, 127
Arm and trigger sources, 78
ARM subsystem, 208 - 210
ARM:SOURce, 209
ARM:SOURce?, 210
ARRay

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ALGorithm[:EXPLicit]:ARRay, 191
ARRay?
ALGorithm[:EXPLicit]:ARRay?, 192
Assigning values, 138
Assignment operator, 127
Attaching and removing the terminal module, 39
Attaching the HP E1415 terminal module, 39
Attaching the terminal module, 37
Autoranging, more on, 104
Available Power for SCPs, 333

B

Before INIT, 52
Bit-number:, 134
Bitfield access, 131
BLOCK), continuously reading the FIFO (FIFO mode, 84
Byte, enabling events to be reported in the status, 93
Byte, reading the status, 94

C

C language algorithms
defining, 74
CAL:CONF:RES, 212
CAL:CONF:VOLT, 213
CAL:SETup, 214
CAL:SETup?, 214
CAL:STORe, 215
CAL:TARE, 216
CAL:TARE and thermocouples, 100
CAL:TARE, resetting, 102
CAL:TARE:RESet, 218
CAL:TARE?, 218
CAL:VAL:RESistance, 219
CAL:VAL:VOLTage, 219
CAL:ZERO?, 220
CALibration subsystem, 211 - 221
Calibration, channel
*CAL?, 316
Calibration, control of, 19
Calling user defined functions, 118
Capability, maximum tare, 102
CAUTIONS
Loss of process control by algorithm, 189, 200, 310
Safe handling procedures, 15
Changing an algorithm while it’s running, 122
Changing gains, 102
Changing gains or filters, 102
Changing timer interval while scanning, 313
CHANnel
ALGorithm:UPDate:CHANnel, 205
Channel calibration
*CAL?, 316
Channels
defined input, 115
output, 57, 67, 115
SENSe:REFerence:CHANnels, 283
setting up analog input, 57
setting up digital input, 67
Channels, accessing I/O, 114
Channels, adding settling delay for specific, 106
Channels, input, 114
Channels, output, 114
Channels, special identifiers for, 127
Characteristics, settling, 104
Checking for problems, 105
CHECksum?
DIAG:CHECK?, 225
Clearing event registers, 96
Clearing the enable registers, 95
Coefficients, 86
Command
Abbreviated, 182
Implied, 183
Linking, 185
Separator, 182
Command Quick Reference, 326 - 332
Command Reference, Common
*CAL?, 316
*CLS, 317
*DMC, 317
*EMC, 317
*EMC?, 317
*ESE, 317
*ESE?, 318
*ESR?, 318
*GMC?, 318
*IDN?, 318
*LMC?, 319
*OPC, 319
*OPC?, 319
*PMC, 320
*RMC, 320
*RST, 320
*SRE, 321
*SRE?, 321
*STB?, 321
*TRG, 321
*TST?, 322
*WAI, 325
Command sequences, defined, 21
Comment lines, 141

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments:, 138
Common Command Format, 182
Common mode noise, 378
Common mode rejection, 334
Common mode voltage
    Maximum, 334
Common mode voltage limits, 377
Comparison operators, 127
Compensating for system offsets, 100
Compensation, thermocouple reference temperature, 64
Components, adding terminal module, 41
Compound-statement:, 136
CONDition
    SENSe:FUNC:CONDition, 271
    SOURce:FUNC[:SHAPe]:CONDition, 291
    STAT:OPER:CONDition?, 298
CONDition?
    STAT:QUES:CONDition?, 304
Conditional constructs, 128
Conditional execution, 139
Configuring programmable analog SCP parameters, 57
Configuring the enable registers, 93
Configuring the HP E1415, 13
Configuring the Reference Jumpers, 32
Configuring the transition filters, 93
Connection
    recommended, 34
    signals to channels, 34
Connections
    Guard, 377
Considerations, special, 102
Constant:, decimal, 133
Constant:, hexadecimal, 133
Constant:, octal, 133
Constructs, conditional, 128
Continuous Mode, 313
Continuously reading the FIFO (FIFO mode BLOCK), 84
Control, program flow, 128
Conversion, EU, 372
Conversion, linking channels to EU, 59
Conversions, custom EU, 67
Conversions, custom reference temperature EU, 99
Conversions, custom thermocouple EU, 99
COUNt?
    SENS:DATA:FIFO:COUNt?, 266
Counter, setting the trigger, 80
Creating and loading custom EU conversion tables, 98
Creating conversion tables, 100
CTYPe?
    SYST:CTYPe?, 308
Current Value Table
    SENSe:DATA:CVTable?, 264
CUSTom
    SENS:FUNC:CUSTom, 272
Custom EU conversion tables
    creating, 98
    loading, 98
Custom EU conversions, 67
Custom EU operation, 99
Custom EU tables, 99
Custom reference temperature EU conversions, 99
Custom thermocouple EU conversions, 99
CVT
    SENSe:DATA:CVTable?, 264
CVT elements, reading, 117
CVT elements, writing value to, 116
CVT, sending data to, 116

D

DATA
    FORMat:DATA, 233
    FORMat:DATA?, 234
Data acquisition algorithm, 125
Data structures, 130
Data types, 129
Data, retrieving algorithm, 83
DATA:FIFO:ALL?, 266
Decimal constant:, 133
Declaration initialization, 132
Declaration:, 136
Declarations:, 136
Declarator:, 135
Declaring variables, 138
Defaults
    power-on and *RST, 53
DEFine
    ALGorithm:FUNCtion:DEFine, 201
    ALGorithm[:EXPLicit]:DEFine, 192
    ROUT:SEQ:DEF?, 258
Defined input and output channels, 115
Defining algorithms, 120
Defining an algorithm for swapping, 122
Defining and accessing global variables, 115
Defining C language algorithms, 74
Defining data storage, 76
DELay
    ALGorithm:OUTPut:DELay, 203
DELay?
    ALGorithm:OUTPut:DELay?, 204
Detecting open transducers, 103
Determining an algorithm’s size, 123

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Determining first execution (First_loop), 115
Determining model
SCPI programming, 318
DIAG:CHECK?, 225
DIAG:CUST:REF:TEMP, 226
DIAG:INT:LINe, 228
DIAG:INT:LINe?, 228
DIAG:OTD[:STATe], 228
DIAG:OTD[:STATe]?, 229
DIAG:VERSion?, 230
DIAGnostic
DIAGnostic:CALibration:SETup[:MODE], 223
DIAGnostic:CALibration:SETup[:MODE]?, 223
DIAGnostic:CALibration:TARe:MODE, 224
DIAGnostic:CALibration:TARe:MODE?, 224
DIAGnostic:CUSTom:LINear, 225
DIAGnostic:CUSTom:PIECewise, 226
DIAGnostic:IEEE, 227
DIAGnostic:IEEE?, 227
DIAGnostic:QUERy:SCPREAD, 230
DIAGnostic:CALibration:SETup[:MODE], 223
DIAGnostic:CALibration:SETup[:MODE]?, 223
DIAGnostic:CALibration:TARe:MODE, 224
DIAGnostic:CALibration:TARe:MODE?, 224
DIAGnostic:CUSTom:LINear, 225
DIAGnostic:CUSTom:PIECewise, 226
DIAGnostic:IEEE, 227
DIAGnostic:IEEE?, 227
DIAGnostic:OTDetect, 103
DIAGnostic:QUERy:SCPREAD, 230
Digital evaluation of type float, 131
Directly, reading status groups, 95
Disabling flash memory access (optional), 19
Disabling the input protect feature (optional), 18
Does, what *CAL?, 72
Drivers; instrument, 21
DSP, 372

E

ENABle
STAT:OPER:ENABle, 299
STAT:QUES:ENABle, 304
ENABle?
STAT:OPER:ENABle?, 300
STAT:QUES:ENABle?, 305
Enabling and disabling algorithms, 87
Enabling events to be reported in the status byte, 93
Environment, the algorithm execution, 112
Equality-expression:, 135
Equality-operator:, 135
Error Messages, 363
Self Test, 366
ERRor?
SYST:ERRor?, 308
EU, 372
EU Conversion, 372
EVENt?
STAT:OPER:EVENt?, 300
STAT:QUES:EVENt?, 305
Example command sequence, 88
Example language usage, 111
Example programs, about, 21
Examples, operation status group, 94
Examples, questionable data group, 93
Examples, standard event group, 94
EXCitation
SENSe:STRain:EXCitation, 284
SENSe:STRain:EXCitation?, 285
Executing the programming model, 53
Execution, conditional, 139
Exiting the algorithm, 128
Expression-statement:, 136
Expression:, 135

F

Faceplate connector pin-signal lists, 27
FIFO, reading values from the, 117
FIFO, sending data to, 116
FIFO, time relationship of readings in, 117
FIFO, writing values to, 117
Filters, 102
Filters, adding circuits to terminal module, 41
Filters, configuring the transition, 93
Fixed width pulses at variable frequency (FM), 71
Fixing the problem, 105
Flash Memory, 372
Flash memory access, disabling, 19
Flash memory limited lifetime, 215
Floating point as integer, 131
FM:STATe
SOURce:FM:STATe, 290
FM:STATe?
SOURce:FM:STATe?, 291
Format
Common Command, 182
SCPI Command, 182
Format, specifying the data, 76
FORMat:DATA, 233
FORMat:DATA?, 234
Formats, ALG:DEFINE’s three data, 121
FREQuency
INPut:FILT:FREQ, 238

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SENSe:FUNCtion:FREQuency, 275
Frequency function, 68
Frequency, setting algorithm execution, 88
Frequency, setting filter cutoff, 58
FREQuency?
INP:FILT:FREQ?, 239
Function, frequency, 68
Function, setting input, 68
Function, static state (CONDition), 68, 70
Function, the main, 112
Function, totalizer, 68
Functions and statements, intrinsic
abs(expression), 128
interrupt(), 117, 128
max(expression1,expression2), 128
min(expression1,expression2), 128
writeboth(expression,cvt_element), 128
writecvt(expression,cvt_element), 116, 128
writefifo(expression), 117, 128
Functions, calling user defined, 118
Functions, linking output channels to, 67
Functions, setting output, 69
Functions:, 128

G

GAIN
channel, 316
INPut:GAIN, 241
GAIN?
INP:GAIN?, 242
Gains, setting SCP, 57
GFACtor
SENSe:STRain:GFACtor, 285
SENSe:STRain:GFACtor?, 286
Global variable definition, 74
Global variables, 132
accessing, 115
defining, 115
Glossary, 371
Grounding
Noise due to inadequate, 377
Group, an example using the operation, 93
Guard connections, 377

H

HALF?
SENS:DATA:FIFO:COUNt:HALF?, 267
SENS:DATA:FIFO:HALF?, 267
Hexadecimal constant:, 133
HINTS
for quiet measurements, 34
Read chapter 3 before chapter 4, 109
How to use *CAL?, 72
HP E1415, configuring the, 13
HP E1419A background operation, 96

I

Identifier:, 133
Identifiers, 126
IEEE +/- INF, 234
IMMediate
ALGorithm:UPDate[:IMMediate], 204
ARM:IMMediate, 209
INIT:IMM, 236
TRIG:IMMediate, 313
Impedance, input, 334
Implied Commands, 183
IMPORTANT!
Do use CAL:TARE for copper TC wiring, 100
Don’t use CAL:TARE for thermocouple wiring, 100
INF, IEEE, 234
INIT
after, 53
before, 52
Init-declarator-list:, 135
Init-declarator:, 135
INIT:IMM, 236
Initialization, declaration, 132
Initializing variables, 116
INITiate subsystem, 236
INITiating/Running algorithms, 81
INP:FILT:FREQ?, 239
INP:FILT:LPAS:STAT, 240
INP:FILT:LPAS:STAT?, 240
INP:GAIN?, 242
Input channels, 114
Input impedance, 334
Input protect feature, disabling, 18
INPut subsystem, 237 - 244
Input voltage, maximum, 334
INPut:DEB:TIME, 237
INPut:FILT:FREQ, 238
INPut:GAIN, 241
INPut:LOW, 242
INPut:LOW?, 243
INPut:POLarity, 243
INPut:POLarity?, 243
INPut:THReshold:LEVel?, 244
Inputs, setting up digital, 67
Installing signal conditioning plug-ons, 15
Instrument drivers, 21

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Integer evaluation of type float, 131
Integer values from type float, 131
Interrupt function, 117
Interrupt level, setting NOTE, 13
interrupt(), 117, 128
Interrupts
    updating the status system, 97
    VXI, 97
Intrinsic functions and statements
    abs(expression), 128
    interrupt(), 117, 128
    max(expression1,expression2), 128
    min(expression1,expression2), 128
    writeboth(expression,cvt_element), 128
    writecvt(expression,cvt_element), 116, 128
    writefifo(expression), 117, 128
Intrinsic-statement:, 136
Isothermal reference measurement, NOTE, 26

K

Keywords, special HP E1419A reserved, 126
Keywords, standard reserved, 126

L

Language syntax summary, 133
Language, overview of the algorithm, 110
Layout
    Terminal Module, 30
LEVel?
    INPut:THReshold, 244
Lifetime limitation, Flash memory, 215
Limits
    Common mode voltage, 377
LINe
    DIAG:INT:LINe, 228
LINe?
    DIAG:INT:LINe?, 228
Lines, comment, 141
Linking channels to EU conversion, 59
Linking Commands, 185
Linking output channels to functions, 67
Linking resistance measurements, 61
Linking strain measurements, 66
Linking temperature measurements, 62
Linking voltage measurements, 60
Lists
    Faceplate connector pin-signal , 27
Logical operators, 127
Logical-AND-expression:, 135
LOW
    INPut:LOW, 242
    INPut:LOW?, 243
Low-noise measurements, HINTS, 34

M

max(expression1,expression2), 128
Maximum
    Common mode voltage, 334
    Input voltage, 334
    Tare cal offset, 334
Maximum tare capability, 102
Measurement
    Ranges, 333
    Resolution, 333
Measurements
    terminal block considerations for TC, 33
Measurements, linking resistance, 61
Measurements, linking strain, 66
Measurements, linking temperature, 62
Measurements, linking voltage, 60
Measurements, reference measurement before thermocouple, 65
Measurements, thermocouple, 63
Measuring the reference temperature, 64
Meaurement
    Accuracy DC Volts, 334
MEM:VME:ADDR, 246
MEM:VME:ADDR?, 246
MEM:VME:SIZE, 246
MEM:VME:SIZE?, 247
MEM:VME:STATe, 247
MEM:VME:STATe?, 248
Messages, error, 363
min(expression1,expression2), 128
MODE
    SENS:DATA:FIFO:MODE, 268
    SENSe:TOTalize:RESet:MODE, 288
Mode, selecting the FIFO, 77
MODE?
    SENS:DATA:FIFO:MODE?, 269
    SENSe:TOTalize:RESet:MODE?, 289
Mode?, which FIFO, 84
Model, determining
    SCPI programming, 318
Modifier, the static, 129
Modifying running algorithm variables, 86
Modifying the terminal module circuit, 41
Module
    SCPs and Terminal, 28
Modules
    Terminal, 28

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

More on auto ranging, 104
Multiplicative-expression:, 134
Multiplicative-operator:, 134

N

NaN, 234
Noise
    Common mode, 378
    Normal mode, 378
Noise due to inadequate grounding, 377
Noise reduction with amplifier SCPs, NOTE, 106
Noise reduction, wiring techniques, 376
Noise Rejection, 378
Noisy measurements
    Quieting, 34
Non-Control algorithms, 125
Normal mode noise, 378
Not-a-Number, 234
NOTES
    *CAL? and CAL:TARE turns off then on OTD, 229
    *RST effect on custom EU tables, 99
    *TST? sets default ASC,7 data format, 234
    + & - overvoltage return format from FIFO, 266, 268, 270
    ALG:SCAN:RATIO vs. ALG:UPD, 198
    ALG:SIZE? return for undefined algorithm, 199
    ALG:STATE effective after ALG:UPDATE, 87
    ALG:STATE effective only after ALG:UPD, 199
    ALG:TIME? return for undefined algorithm, 201
    Algorithm Language case sensitivity, 127
    Algorithm Language reserved keywords, 126
    Algorithm source string terminated with null, 121
    Algorithm source string terminates with null, 194
    Algorithm swapping limitations, 196
    Algorithm Swapping restrictions, 123
    Algorithm variable declaration and assignment, 115
    Amplifier SCPs can reduce measurement noise, 106
    BASIC’s vs. ’C’s "is equal to" symbol, 139
    Bitfield access ’C’ vs. Algorithm Language, 131
    Cannot declare channel ID as variable, 127
    Combining SCPI commands, 186
    CVT contents after *RST, 265
    Decimal constants can be floating or integer, 133
    Default (*RST) Engineering Conversion, 60
    Define user function before algorithm calls , 118
    Do not CAL:TARE thermocouple wiring, 216
    Do use CAL:TARE for copper in TC wiring, 100
    Do use CAL:TARE for copper TC wiring, 216
    Don’t use CAL:TARE for thermocouple wiring, 100
    Flash memory limited lifetime, 102, 215
    Isothermal reference measurements, 26
    MEM subsystem vs. command module model, 245
    MEM subsystem vs. TRIG and INIT sequence, 245
    MEM system vs TRIG and INIT sequence, 232
    Memory required by an algorithm, 122
    Number of updates vs. ALG:UPD:WINDOW, 191, 196, 207
    Open transducer detect restrictions, 103
    OUTP:CURR:AMPL command, 59
    OUTP:CURR:AMPL for resistance measurements, 249
    OUTP:VOLT:AMPL command, 59
    Reference to noise reduction literature, 377
    Resistance temperature measurements, 62
    Saving time when doing channel calibration, 73
    Selecting manual range vs. SCP gains, 60
    Setting the interrupt level, 13
    Settings conflict, ARM:SOUR vs TRIG:SOUR, 208, 313
    Thermocouple reference temperature usage, 281, 284
    TRIGger:SOURce vs. ARM:SOURce, 79 - 80
    Warmup before executing *TST?, 367
    When algorithm variables are initialized, 132
NTRansition
    STAT:OPER:NTRansition, 300
    STAT:QUES:NTRansition, 305
NTRansition?
    STAT:OPER:NTRansition?, 301
    STAT:QUES:NTRansition?, 306

O

Octal constant:, 133
Offset
    A/D, 214, 316
    channel, 214, 316
Offsets, compensating for system, 100
Offsets, residual sensor, 101
Offsets, system wiring, 100
Operating model, 52
Operating sequence, 118
Operation, 72, 101
Operation and restrictions, 72
Operation status group examples, 94
Operation, custom EU, 99
Operation, HP E1419A background, 96
Operation, standard EU, 98
Operator, assignment, 127
Operator, unary arithmetic, 139
Operator, unary logical, 127
Operators, 127
Operators, arithmetic, 127
Operators, comparison, 127

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Operators, logical, 127
Operators, the arithmetic, 139
Operators, the comparison, 139
Operators, the logical, 139
Operators, unary, 127
Option A3F, 44
Order, algorithm execution, 120
OTD restrictions, NOTE, 103
OTDetect, DIAGnostic:OTDetect, 103
OUTP:CURRent:AMPLitude, 249
OUTP:CURRent:AMPLitude?, 250
OUTP:SHUNt, 253
OUTP:SHUNt?, 253
OUTP:TTLT :STATe, 255
OUTP:TTLT :STATe?, 255
Output channels, 114
OUTPut subsystem, 249 - 257
OUTPut:CURRent:STATe, 251
OUTPut:CURRent:STATe?, 251
OUTPut:POLarity, 252
OUTPut:POLarity?, 252
OUTPut:TTLTrg:SOURce, 254
OUTPut:TTLTrg:SOURce?, 254
OUTPut:TYPE, 255
OUTPut:TYPE?, 256
OUTPut:VOLTage:AMPLitude, 256
OUTPut:VOLTage:AMPLitude?, 257
Outputs, setting up digital, 68
Outputting trigger signals, 81
OVER), reading the latest FIFO values (FIFO mode, 86
Overall program structure, 142
Overall sequence, 118
Overloads, unexpected channel, 102
Overview of the algorithm language, 110

P

Parameter data and returned value types, 187
Parameters, configuring programmable analog SCP, 57
PART?
    SENS:DATA:FIFO:PART?, 269
Performing channel calibration (Important!), 72
PERiod
    SOURce:PULSe:PERiod, 293
PERiod?
    SOURce:PULSe:PERiod?, 294
Planning
    grouping channels to signal conditioning, 23
    planning wiring layout, 23
    sense vs. output SCPs, 25
    thermocouple wiring, 26
Plug-ons, installing signal conditioning, 15
Points
    ROUT:SEQ:POINts?, 259
POISson
    SENSe:STRain:POISson, 286
    SENSe:STRain:POISson?, 287
POLarity
    INPut:POLarity, 243
    OUTPut:POLarity, 252
Polarity, setting input, 67
Polarity, setting output, 69
POLarity?
    INPut:POLarity?, 243
    OUTPut:POLarity?, 252
Power Available for SCPs, 333
Power-on and *RST defaults, 53
Pre-setting algorithm variables, 75
PRESet
    STAT:PRESet, 303
Primary-expression:, 133
Problem, fixing the, 105
Problems, checking for, 105
Process monitoring algorithm, 125
Program flow control, 128
Program structure and syntax, 138
Programming model
    executing the, 53
Programming the trigger timer, 80
PTRansition
    STAT:OPER:PTRansition, 302
    STAT:QUES:PTRansition, 307
PTRansition?
    STAT:OPER:PTRansition?, 302
    STAT:QUES:PTRansition?, 307
PULSe
    SOURce:FUNC[:SHAPe]:PULSe, 292

Q

Questionable data group examples, 93
Quick Reference, Command, 326 - 332
Quiet measurements, HINTS, 34
Quieter readings with amplifier SCPs, NOTE, 106

R

Rack Mount Terminal Panel Accessories, 44
Ranges, measurement, 333
RATio
    ALGorithm[:EXPLicit]:SCAN:RATio, 197
RATio?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ALGorithm[:EXPLicit]:SCAN:RATio?, 198
Re-Execute *CAL? when:, 73
Reading algorithm variables, 83
Reading condition registers, 96
Reading CVT elements, 117
Reading event registers, 96
Reading status groups directly, 95
Reading the latest FIFO values (FIFO mode OVER), 86
Reading the status byte, 94
Reading values from the FIFO, 117
Recommended measurement connections, 34
REFerence
SENS:FUNC:CUST:REF, 273
SENS:REFerence, 281
Reference Jumpers
configuring the, 32
Reference junction, 32
Reference measurement before thermocouple measurements, 65
Reference temperature measurement, NOTE, 26
Reference temperature sensing, 31
Reference temperature sensing with the HP E1415, 31
Reference, Algorithm language, 126
Register, the status byte group's enable, 95
Registers, clearing event, 96
Registers, clearing the enable, 95
Registers, configuring the enable, 93
Registers, reading condition, 96
Registers, reading event, 96
Rejection
Noise, 378
Rejection, common mode, 334
Relational-expression:, 134
Relational-operator:, 135
Removing the HP E1415 terminal module, 39
RESet
*RST, 320
SENS:DATA:CVT:RESet, 265
SENS:DATA:FIFO:RESet, 270
Resetting CAL:TARE, 102
Residual sensor offsets, 101
RESistance
CAL:CONF:RES, 212
CAL:VAL:RESistance, 219
SENS:FUNC:RESistance, 275
Resolution, measurement, 333
Resources, accessing the E1419A's, 113
Restrictions, 72
Retrieving Algorithm Data, 83
ROUT:SEQ:DEF?, 258
ROUT:SEQ:POINts?, 259
ROUTe subsystem, 258 - 259
RTD and thermistor measurements, 62
Running the algorithm, 124
Running, changing an algorithm while it's, 122

S

Safe Handling, static discharge CAUTION, 15
SAMP:TIMer, 260
SAMP:TIMer?, 260
SAMPle subsystem, 260 - 261
sample timer, accuracy, 333
SCALar
ALGorithm[:EXPLicit]:SCALar, 196
SCALar?
ALGorithm[:EXPLicit]:SCALar?, 197
SCP, 372
grouping channels to signal conditioning, 23
sense vs. output SCPs, 25
SCP, Power Available, 333
SCP, setting the HP E1505 current source, 58
SCPI commands, 177
DIAGnostic:OTDetect, 103
Format, 182
SCPs and Terminal Module, 28
Selecting the FIFO mode, 77
Selecting the trigger source, 78
Selecting trigger timer arm source, 79
Selection-statement:, 136
Self test
and C-SCPI for MS-DOS (R), 322
how to read results, 322
Self Test, error messages, 366
Sending Data to the CVT and FIFO, 116
SENS:DATA:CVT:RESet, 265
SENS:DATA:FIFO:COUNt:HALF?, 267
SENS:DATA:FIFO:COUNt?, 266
SENS:DATA:FIFO:HALF?, 267
SENS:DATA:FIFO:MODE, 268
SENS:DATA:FIFO:MODE?, 269
SENS:DATA:FIFO:PART?, 269
SENS:DATA:FIFO:RESet, 270
SENS:FUNC:CUST:REF, 273
SENS:FUNC:CUST:TC, 274
SENS:FUNC:RESistance, 275
SENS:FUNC:STRain, 277
SENS:FUNC:TEMPerature, 278
SENS:FUNC:VOLTage, 280
SENS:REF:TEMPerature, 284
SENS:REFerence, 281
SENSe subsystem, 262 - 289
SENSe:CHANnel:SETTling, 263

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SENSe:CHANnel:SETTling?, 264
SENSe:DATA:CVTable?, 264
SENSe:FREQuency:APERture, 270
SENSe:FREQuency:APERture?, 271
SENSe:FUNC:CONDition, 271
SENSe:FUNC:CUSTom, 272
SENSe:FUNCtion:FREQuency, 275
SENSe:FUNCtion:TOTalize, 280
SENSe:REFerence:CHANnels, 283
SENSe:STRain:EXCitation, 284
SENSe:STRain:EXCitation?, 285
SENSe:STRain:GFACtor, 285
SENSe:STRain:GFACtor?, 286
SENSe:STRain:POISson, 286
SENSe:STRain:POISson?, 287
SENSe:STRain:UNSTrained, 287
SENSe:STRain:UNSTrained?, 288
SENSe:TOTalize:RESet:MODE, 288
SENSe:TOTalize:RESet:MODE?, 289
Sensing
    Reference temperature with the HP E1415, 31
Sensing 4-20 mA, 41
Separator, command, 182
Sequence, A complete thermocouple measurement command, 65
Sequence, ALG:DEFINE in the programming, 121
Sequence, example command, 88
Sequence, operating, 118
Sequence, overall, 118
Sequence, the operating, 82
Setting algorithm execution frequency, 88
Setting filter cutoff frequency, 58
Setting input function, 68
Setting input polarity, 67
Setting output drive type, 69
Setting output functions, 69
Setting output polarity, 69
Setting SCP gains, 57
Setting the HP E1505 current source SCP, 58
Setting the HP E1511 strain bridge SCP excitation voltage, 59
Setting the logical address switch, 14
Setting the trigger counter, 80
Setting up analog input and output channels, 57
Setting up digital input and output channels, 67
Setting up digital inputs, 67
Setting up digital outputs, 68
Setting up the trigger system, 78
Settings conflict
    ARM:SOUR vs TRIG:SOUR, 208, 313
SETTling
    SENSe:CHANnel:SETTling, 263
Settling characteristics, 104
SETTling?
    SENSe:CHANnel:SETTling?, 264
SETup
    CAL:SETup, 214
    CAL:SETup?, 214
Shield Connections
    When to make, 377
SHUNt
    OUTP:SHUNt, 253
    OUTPut:SHUNt?, 253
Signal, connection to channels, 34
Signals, outputting trigger, 81
SIZE
    MEM:VME:SIZE, 246
Size, determining an algorithm's, 123
SIZe?
    ALGorithm[:EXPLicit]:SIZe?, 199
    MEM:VME:SIZE?, 247
SOURce
    ARM:SOURce, 209
    ARM:SOURce?, 210
    OUTPut:TTLTrg:SOURce, 254
    TRIG:SOURce, 313
SOURce subsystem, 290 - 295
Source, selecting the trigger, 78
Source, selecting trigger timer arm, 79
SOURce:FM:STATe, 290
SOURce:FM:STATe?, 291
SOURce:FUNC[:SHAPe]:CONDition, 291
SOURce:FUNC[:SHAPe]:PULSe, 292
SOURce:FUNC[:SHAPe]:SQUare, 292
SOURce:PULM:STATe, 292
SOURce:PULM:STATe?, 293
SOURce:PULSe:PERiod, 293
SOURce:PULSe:PERiod?, 294
SOURce:PULSe:WIDTh, 294
SOURce:PULSe:WIDTh?, 294
SOURce?
    TRIG:SOURce?, 314
Sources
    arm, 78
    trigger, 78
Special considerations, 102
Special HP E1419A reserved keywords, 126
Special identifiers for channels, 127
Specifications, 333
Specifying the data format, 76
SQUare
    SOURce:FUNC[:SHAPe]:SQUare, 292
Standard Commands for Programmable Instruments, SCPI, 188

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Standard EU operation, 98
Standard event group examples, 94
Standard reserved keywords, 126
Starting algorithms, 81
STAT:OPER:CONDition?, 298
STAT:OPER:ENABle, 299
STAT:OPER:ENABle?, 300
STAT:OPER:EVENt?, 300
STAT:OPER:NTRansition, 300
STAT:OPER:NTRansition?, 301
STAT:OPER:PTRansition, 302
STAT:OPER:PTRansition?, 302
STAT:PRESet, 303
STAT:QUES:CONDition?, 304
STAT:QUES:ENABle, 304
STAT:QUES:ENABle?, 305
STAT:QUES:EVENt?, 305
STAT:QUES:NTRansition, 305
STAT:QUES:NTRansition?, 306
STAT:QUES:PTRansition, 307
STAT:QUES:PTRansition?, 307
STATe
    ALGorithm[:EXPLicit][:STATe], 199
    DIAG:OTD[:STATe], 228
    DIAG:OTD[:STATe]?, 229
    INP:FILT:LPAS:STATe, 240
    INP:FILT:LPAS:STATe?, 240
    MEM:VME:STATe, 247
    MEM:VME:STATe?, 248
    OUTPut:CURRent:STATe, 251
    OUTPut:CURRent:STATe?, 251
    SOURce:PULM:STATe, 292
STATe?
    ALGorithm[:EXPLicit][:STATe]?, 200
    SOURce:PULM:STATe?, 293
Statement, algorithm language
    writecvt(), 116
    writefifo(), 117
Statement-list:, 136
Statement:, 136
Statements and functions, intrinsic
    abs(expression), 128
    interrupt(), 117, 128
    max(expression1,expression2), 128
    min(expression1,expression2), 128
    writeboth(expression,cvt_element), 128
    writecvt(expression,cvt_element), 116, 128
    writefifo(expression), 117, 128
Statements:, 128
Static discharge safe handling, CAUTION, 15
Static state (CONDition) function, 68, 70
STATus subsystem, 296 - 307
Storage, defining data, 76
STORe
    CAL:STORe, 215
STRain
    SENS:FUNC:STRain, 277
Structure, overall program, 142
Structures, data, 130
Sub subsystem, 222 - 235, 245 - 248
Subsystem
    ABORT, 189
    ARM, 208 - 210
    CALibration, 211 - 221
    DIAGnostic, 222 - 230
    FETCh?, 231 - 232
    FORMat, 233 - 235
    INITiate, 236
    INPut, 237 - 244
    MEMory, 245 - 248
    OUTPut, 249 - 257
    ROUTe, 258 - 259
    SAMPle, 260 - 261
    SENSe, 262 - 289
    SOURce, 290 - 295
    STATus, 296 - 307
    SYSTem, 308 - 309
    TRIGger, 310 - 315
Summary, 100
Summary, language syntax, 133
Supplying the reference temperature, 65
Swapping, defining an algorithm for, 122
Switch, setting the logical address, 14
Symbols, the operations, 139
Syntax, Variable Command, 183
SYST:CTYPe?, 308
SYST:ERRor?, 308
SYST:VERSion?, 309
SYSTem subsystem, 308 - 309
System wiring offsets, 100
System, setting up the trigger, 78
System, using the status, 90

T

Tables, creating conversion, 100
Tables, custom EU, 99
TARE
    CAL:TARE:RESet, 218
    CAL:TARE?, 218
Tare cal offset, maximum, 334
TARE?
    CAL:TARE, 216

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TCouple
SENS:FUNC:CUST:TC, 274
Techniques
Wiring and noise reduction, 376
TEMPerature
DIAG:CUST:REF:TEMP, 226
SENS:FUNC:TEMPerature, 278
SENS:REF:TEMPerature, 284
Temperature accuracy, 335
Temperature, measuring the reference, 64
Temperature, supplying the reference, 65
Terminal block considerations for TC measurements, 33
Terminal Blocks, 373
Terminal Module, 373
Attaching and removing the HP E1415, 39
Attaching the HP E1415, 39
Removing the HP E1415, 39
Wiring and attaching the, 37
Terminal Module Layout, 30
Terminal module wiring maps, 42 - 43
Terminal modules, 28
The algorithm execution environment, 112
The arithmetic operators, 139
The comparison operators, 139
The logical operators, 139
The main function, 112
The operating sequence, 82
The operations symbols, 139
The static modifier, 129
The status byte group's enable register, 95
Thermistor and RTD measurements, 62
Thermocouple measurements, 63
Thermocouple reference temperature compensation, 64
Thermocouples and CAL:TARE, 100
TIME
INPut:DEB:TIME, 237
Time relationship of readings in FIFO, 117
TIMe?
ALGorithm[:EXPLicit]:TIMe?, 201
Timer
SAMP:TIMer, 260
SAMP:TIMer?, 260
TRIG:COUNt, 312
TRIG:TIMer, 314
Timer, programming the trigger, 80
TIMer?
TRIG:TIMer?, 315
TOTalize
SENSe:FUNCtion:TOTalize, 280
Totalizer function, 68
Transducers, detecting open, 103
TRIG:COUNt, 312
TRIG:COUNt?, 312
TRIG:IMMediate, 313
TRIG:SOURce, 313
TRIG:SOURce?, 314
TRIG:TIMer, 314
TRIG:TIMer?, 315
TRIGger subsystem, 310 - 315
trigger system
ABORt subsystem, 189
ARM subsystem, 208 - 210
INITiate subsystem, 236
TRIGger subsystem, 310 - 315
Trigger, variable width pulse per, 70
TTLTrg:SOURce
OUTPut:TTLTrg:SOURce?, 254
TTLTrg
OUTP:TTLT :STATe?, 255
OUTP:TTLTrg :STATe, 255
TYPe
OUTPut:TYPE, 255
Type, setting output drive, 69
TYPe?
OUTPut:TYPE?, 256
Types, data, 129

U

Unary arithmetic operator, 139
Unary logical operator, 127
Unary operators, 127
Unary-expression:, 134
Unary-operator:, 134
Unexpected channel offsets or overloads, 102
UNSTrained
SENSe:STRain:UNSTrained, 287
SENSe:STRain:UNSTrained?, 288
Updating the algorithm variables, 86
Updating the algorithm variables and coefficients, 86
Updating the status system and VXI interrupts, 97
Usage, example language, 111
Using the status system, 90

V

Value types
parameter data, 187
returned, 187
Values, assigning, 138
Variable Command Syntax, 183
Variable definition
global, 74

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Variable frequency square-wave output (FM), 71
Variable width pulse per trigger, 70
Variable width pulses at fixed frequency (PWM), 70
Variables
    reading directly, 84
Variables, declaring, 138
Variables, global, 132
Variables, initializing, 116
Variables, modifying running algorithm, 86
Variables, reading algorithm, 83
Verifying a successful configuration, 21
VERsion
    DIAG:VERSion?, 230
VERsion?
    SYST:VERSion?, 309
Voids Warranty
    Cutting Input Protect Jumper, 19
VOLTage
    CAL:CONF:VOLT, 213
    CAL:VALue:VOLTage, 219
    SENS:FUNC:VOLTage, 280
Voltage, setting the HP E1511 strain bridge SCP excitation, 59
VOLTage:AMPLitude
    OUTPut:VOLTage:AMPLitude, 256
    OUTPut:VOLTage:AMPLitude?, 257

W

Warranty
    Voided by cutting Input Protect Jumper, 19
What *CAL? does, 72
When to make shield connections, 377
When:, re-execute *CAL?, 73
Which FIFO mode?, 84
WIDTh
    SOURce:PULSe:WIDTh, 294
WIDTh?
    SOURce:PULSe:WIDTh?, 294
WINDow
    ALGorithm:UPDate:WINDow, 206
WINDow?
    ALGorithm:UPDate:WINDow?, 207
Wiring
    planning for thermocouple, 26
    planning layout, 23
    signal connection, 34
Wiring and attaching the terminal module, 37
Wiring maps
    Terminal Module, 42 - 43
Wiring techniques, for noise reduction, 376
Wiring the terminal module, 37
writeboth(expression,cvt_element), 128
writecvt(expression,cvt_element), 116, 128
writefifo(expression), 117, 128
Writing the algorithm, 124
Writing values to CVT elements, 116
Writing values to the FIFO, 117

Z

ZERO?
    CAL:ZERO?, 220

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP