HP 75000 SERIES C

64-Channel High Speed Scanning A/D HP E1413A and B

User's Manual

APPLICABILITY

This manual edition is intended for use with the following instrument drivers:

- Downloaded driver revision A.05.01 or later for HP Command Modules
- C-SCPI driver revision B.05.01 or later for HP 300 Series
- C-SCPI driver revision C.05.01 or later for MS-DOS®
- C-SCPI driver revision D.05.01 or later for HP 700 Series
- C-SCPI driver revision E.05.01 or later for LynxOS

HEWLETT PACKARD

Copyright © Hewlett-Packard Company, 1994

E1413-90003

Manual Part Number: E1413-90003
Microfiche Part Number: E1413-99003

Printed: MARCH 1994 Edition 1
Printed in U.S.A. E 0394

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 Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) other HP products.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

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (b)(3)(ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013. Hewlett-Packard Company; 3000 Hanover Street; Palo Alto, California 94304

Declaration of Conformity

According to ISO/IEC Guide 22 and EN 45014

The Hewlett-Packard Company declares that the HP E1413A conforms to the following Product Specifications.

Safety: IEC 1010-1 (1990)
CSA 234
UL 1244

EMC: CISPR 11:1990/EN 55011 (1991): Group1 Class A
IEC 801-2:1991/EN 50082-1 (1992): 4kVCD, 8kVAD
IEC 801-3:1984/EN 50082-1 (1992): 3 V/m
IEC 801-4:1988/EN 50082-1 (1992): 1kV

Hewlett-Packard Company
P.O. Box 301

Jean White

Q.A. Manager
May 1992

815 14th Street S.W.
Loveland, Colorado 80539 U.S.A

Printing History

The Printing History shown below lists all Editions and Updates of this manual and the printing date(s). The first printing 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 the current Edition of the manual. Updates are numbered sequentially starting with Update 1. When a new Edition is created, it contains all the Update information for the previous Edition. Each new Edition or Update also includes a revised copy of this printing history page. Many product updates or revisions do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

Edition 1 (Part Number E1413-90001) . . . . . MAY 1993
Edition 2 (Part Number E1413-90002) . . . . . SEPTEMBER 1993
Edition 3 (Part Number E1413-90003) . . . . . MARCH 1994

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.

Table of Contents

1. Getting Started

About this Chapter . . . 1-1
Configuring the HP E1413 . . . 1-1
Setting the Logical Address Switch . . . 1-2
Installing Signal Conditioning Plug-ons . . . 1-3
Disabling the Input Protect Feature (optional) . . . 1-7
Disabling Flash Memory Access (optional) . . . 1-7
Instrument Drivers . . . 1-9
About Example Programs . . . 1-9
Verifying a Successful Configuration . . . 1-10

2. Field Wiring

About This Chapter . . . 2-1
Planning Your Wiring Layout . . . 2-1
SCP Positions and Channel Numbers . . . 2-2
Sense SCPs and Output SCPs . . . 2-3
Planning for Thermocouple Measurements . . . 2-4
Reference Temperature Sensor Connections . . . 2-5
Recommended Measurement Connections . . . 2-6
The Terminal Module . . . 2-7
Opening and Wiring the Terminal Module . . . 2-8

3. Using the HP E1413

Module Description . . . 3-3
Programming Sequence . . . 3-4
Step 1 Setting up Signal Conditioning Plug-ons . . . 3-5
Setting SCP Gains . . . 3-5
Setting Filter Cutoff . . . 3-6
Setting Current Sources . . . 3-6
Step 2 Linking Channels to EU Conversion . . . 3-7
Linking Voltage Measurements . . . 3-8
Linking Resistance Measurements . . . 3-8
Linking Temperature Measurements . . . 3-10
Linking Strain Measurements . . . 3-13
Linking Custom EU Conversions . . . 3-14
Step 3 Performing Channel Calibration . . . 3-16
Step 4 Defining and Selecting the Scan Lists . . . 3-17
Defining the Scan Lists . . . 3-17
Selecting the Current Scan List . . . 3-17
Step 5 Setting the Sample Timer . . . 3-18
Step 6 Setting up the Trigger System . . . 3-19
The Trigger and Arm model . . . 3-19
Selecting the Trigger Source . . . 3-19
Selecting Timer Arm Source . . . 3-20
Setting the Trigger Counter . . . 3-21

Step 7 Specifying the Data Format . . . 3-21
Step 8 Selecting the FIFO Mode . . . 3-22
Step 9 Initiating the Trigger System . . . 3-22
Step 10 Retrieving Data . . . 3-23
Accessing the CVT . . . 3-23
Accessing the FIFO . . . 3-24
Additional Topics . . . 3-25
Example Program . . . 3-25
Example Command Sequence . . . 3-26

4. Understanding the HP E1413

Advanced FIFO Data Retrieval . . . 4-1
General Form of the Data Retrieval Section . . . 4-2
Choosing the Data Retrieval Method . . . 4-4
Controlling Data Conversion and Destination . . . 4-7
Understanding Scanning Modes . . . 4-8
Triggering and Scanning Modes . . . 4-11
Continuous (Free Run) Mode . . . 4-11
Timer Paced Scans . . . 4-12
Sequenced Scan Lists . . . 4-13
Externally Paced Scans . . . 4-14
Synchronizing Multiple Cards . . . 4-14
Continuous (Free-Run) Mode . . . 4-15
Internal Timer Based Scans . . . 4-16
Timer Based Scans at Different Rates . . . 4-17
Using the Status System . . . 4-21
Enabling Events to be Reported in the Status Byte . . . 4-22
Reading the Status Byte . . . 4-22
Clearing the Enable Registers . . . 4-23
The Status Byte Group's Enable Register . . . 4-23
Reading Status Groups Directly . . . 4-24
HP E1413 Background Operation . . . 4-25
Updating the Status System and VXI Interrupts . . . 4-25
Compensating for System Offsets . . . 4-27
Special Considerations . . . 4-28
Detecting Open Transducers . . . 4-29
HP E1413 Thermocouple Reference Compensation . . . 4-29
Reducing Settling Waits . . . 4-31

5. HP E1413 Command Reference

Using This Chapter . . . 5-1
Overall Command Index . . . 5-1
Command Fundamentals . . . 5-6
Common Command Format . . . 5-6
SCPI Command Format . . . 5-6
Linking Commands . . . 5-10
C-SCPI Data Types . . . 5-10
SCPI Command Reference . . . 5-12
ABORt . . . 5-13
ARM . . . 5-14
ARM[:IMMediate] . . . 5-15
ARM:SOURce . . . 5-15

ARM:SOURce? . . . 5-16
CALCulate . . . 5-17
CALCulate:AVERage[:STATe] . . . 5-18
CALCulate:AVERage[:STATe]? . . . 5-18
CALCulate:AVERage:COUNt . . . 5-18
CALCulate:AVERage:COUNt? . . . 5-19
CALCulate:CLIMits:FAIL[:CUMulative]? . . . 5-19
CALCulate:CLIMits:FAIL:CURRent? . . . 5-20
CALCulate:CLIMits:FLIMits[:CHANnels][:CUMulative]? . . . 5-20
CALCulate:CLIMits:FLIMits[:CHANnels]:CURRent? . . . 5-21
CALCulate:CLIMits:FLIMits:POINts[:CUMulative]? . . . 5-21
CALCulate:CLIMits:FLIMits:POINts:CURRent? . . . 5-21
CALCulate:LIMit[:STATe] . . . 5-22
CALCulate:LIMit[:STATe]? . . . 5-22
CALCulate:LIMit:FAIL[:CUMulative]? . . . 5-23
CALCulate:LIMit:FAIL:CURRent? . . . 5-23
CALCulate:LIMit:LOWer[:STATe] . . . 5-24
CALCulate:LIMit:LOWer[:STATe]? . . . 5-24
CALCulate:LIMit:LOWer:DATA . . . 5-24
CALCulate:LIMit:LOWer:DATA? . . . 5-25
CALCulate:LIMit:UPPer[:STATe] . . . 5-25
CALCulate:LIMit:UPPer[:STATe]? . . . 5-26
CALCulate:LIMit:UPPer:DATA . . . 5-26
CALCulate:LIMit:UPPer:DATA? . . . 5-27
CALibration . . . 5-28
CALibration:CONFigure:RESistance . . . 5-30
CALibration:CONFigure:VOLTage . . . 5-30
CALibration:SETup . . . 5-31
CALibration:SETup? . . . 5-31
CALibration:STORe . . . 5-32
CALibration:TARE . . . 5-33
CALibration:TARE:RESet . . . 5-35
CALibration:TARE? . . . 5-35
CALibration:VALue:RESistance . . . 5-35
CALibration:VALue:VOLTage . . . 5-36
CALibration:ZERO? . . . 5-37
DIAGnostic . . . 5-38
DIAGnostic:CHECksum? . . . 5-38
DIAGnostic:COMMand:SCPWRITE . . . 5-38
DIAGnostic:CUSTom:LINear . . . 5-39
DIAGnostic:CUSTom:PIECewise . . . 5-40
DIAGnostic:CUSTom:REFerence:TEMPerature . . . 5-41
DIAGnostic:INTerrupt[:LINe] . . . 5-41
DIAGnostic:INTerrupt[:LINe]? . . . 5-41
DIAGnostic:OTDetect[:STATe] . . . 5-42
DIAGnostic:OTDetect[:STATe]? . . . 5-43
DIAGnostic:QUERy:SCPREAD? . . . 5-43
DIAGnostic:VERSion? . . . 5-43
FETCh? . . . 5-45
FORMat . . . 5-47
FORMat[:DATA] . . . 5-47
FORMat[:DATA]? . . . 5-48
INITiate . . . 5-49

INITiate:CONTinuous 5-49
INITiate[:IMMediate] 5-50
INPut 5-51
INPut:FILTer[:LPASs]:FREQuency 5-51
INPut:FILTer[:LPASs]:FREQuency? 5-52
INPut:FILTer[:LPASs][:STATe] 5-52
INPut:FILTer[:LPASs][:STATe]? 5-53
INPut:GAIN 5-53
INPut:GAIN? 5-54
MEMory 5-55
MEMory:VME:ADDRess 5-56
MEMory:VME:ADDRess? 5-56
MEMory:VME:SIZE 5-56
MEMory:VME:SIZE? 5-57
MEMory:VME:STATe 5-57
MEMory:VME:STATe? 5-58
OUTPut 5-59
OUTPut:CURRent:AMPLitude 5-59
OUTPut:CURRent:AMPLitude? 5-60
OUTPut:CURRent[:STATe] 5-60
OUTPut:CURRent[:STATe]? 5-61
OUTPut:SHUNt 5-61
OUTPut:SHUNt? 5-62
OUTPut:TTLTrg:SOURce 5-62
OUTPut:TTLTrg:SOURce? 5-63
OUTPut:TTLTrg<n>[:STATe] 5-64
OUTPut:TTLTrg<n>[:STATe]? 5-64
ROUTe 5-65
ROUTe:SCAN 5-65
ROUTe:SEQuence:DEFine 5-66
ROUTe:SEQuence:DEFine? 5-68
ROUTe:SEQuence:POINts? 5-69
SAMPle 5-71
SAMPle:TIMer 5-71
SAMPle:TIMer? 5-72
[SENSe] 5-73
[SENSe:]DATA:CVTable? 5-74
[SENSe:]DATA:CVTable:RESet 5-75
[SENSe:]DATA:FIFO[:ALL]? 5-75
[SENSe:]DATA:FIFO:COUNt? 5-76
[SENSe:]DATA:FIFO:COUNt:HALF? 5-76
[SENSe:]DATA:FIFO:HALF? 5-77
[SENSe:]DATA:FIFO:MODE 5-78
[SENSe:]DATA:FIFO:MODE? 5-78
[SENSe:]DATA:FIFO:PART? 5-79
[SENSe:]DATA:FIFO:RESet 5-80
[SENSe:]FILTer[:LPASs][:STATe] 5-80
[SENSe:]FILTer[:LPASs][:STATe]? 5-80
[SENSe:]FUNCtion:CUSTom 5-81
[SENSe:]FUNCtion:CUSTom:REFerence 5-81
[SENSe:]FUNCtion:CUSTom:TCouple 5-83
[SENSe:]FUNCtion:RESistance 5-84
[SENSe:]FUNCtion:STRain:FBENding 5-85

[SENSe:]FUNCtion:STRain::FBPoisson . . . 5-85
[SENSe:]FUNCtion:STRain::FPOisson . . . 5-85
[SENSe:]FUNCtion:STRain::HBENding . . . 5-85
[SENSe:]FUNCtion:STRain::HPOisson . . . 5-85
[SENSe:]FUNCtion:STRain:[:QUARter] . . . 5-85
[SENSe:]FUNCtion:TEMPerature . . . 5-87
[SENSe:]FUNCtion:VOLTage . . . 5-88
[SENSe:]REFerence . . . 5-89
[SENSe:]REFerence:TEMPerature . . . 5-91
[SENSe:]STRain:EXCitation . . . 5-91
[SENSe:]STRain:EXCitation? . . . 5-92
[SENSe:]STRain:GFACtor . . . 5-93
[SENSe:]STRain:GFACtor? . . . 5-93
[SENSe:]STRain:POISson . . . 5-94
[SENSe:]STRain:POISson? . . . 5-94
[SENSe:]STRain:UNSTrained . . . 5-94
[SENSe:]STRain:UNSTrained? . . . 5-95
STATus . . . 5-96
STATus:OPERation:CONDition? . . . 5-98
STATus:OPERation:ENABle . . . 5-98
STATus:OPERation:ENABle? . . . 5-99
STATus:OPERation[:EVENt]? . . . 5-99
STATus:PRESet . . . 5-100
STATus:QUEStionable:CONDition? . . . 5-101
STATus:QUEStionable:ENABle . . . 5-101
STATus:QUEStionable:ENABle? . . . 5-102
STATus:QUEStionable[:EVENt]? . . . 5-102
SYSTem . . . 5-103
SYSTem:CTYPe? . . . 5-103
SYSTem:ERRor? . . . 5-103
SYSTem:VERSion? . . . 5-104
TRIGger . . . 5-105
TRIGger:COUNt . . . 5-106
TRIGger:COUNt? . . . 5-106
TRIGger[:IMMediate] . . . 5-107
TRIGger:SOURce . . . 5-107
TRIGger:SOURce? . . . 5-108
TRIGger:TIMer:MODE . . . 5-108
TRIGger:TIMer:MODE? . . . 5-109
TRIGger:TIMer[:PERiod] . . . 5-109
TRIGger:TIMer[:PERiod]? . . . 5-110
Common Command Reference . . . 5-111
*CAL? . . . 5-111
*CLS . . . 5-111
*ESE <*mask*> . . . 5-112
*ESE? . . . 5-112
*ESR? . . . 5-112
*IDN? . . . 5-112
*OPC . . . 5-113
*OPC? . . . 5-113
*RST . . . 5-113
*SRE <*mask*> . . . 5-114
*SRE? . . . 5-114

*STB? 5-114
*TRG 5-115
*TST? 5-115
*WAI 5-116
Command Quick Reference 5-117

6. Signal Conditioning Plug-on Manuals

A. Specifications

B. Error Messages

C. Glossary

D. Register-Based Programming

About this Appendix D-1
Table of Registers D-3
Register Addressing D-4
The Base Address D-4
Base Address with HP Command Modules D-5
Required VXI Registers D-6
The ID Register D-6
The Device Type Register D-6
The VXI Status Register D-7
The VXI Control Register D-8
The Offset Register D-8
The Query Response Register D-9
The Command and Parameter Registers D-10
Scan Status & Control Register D-10
Card Control Register D-13
Interrupt Configuration Register D-15
Interrupt Status Register D-15
Common Capabilities Register D-17
The Description Register D-18
The Subclass Register D-19
The FIFO MSW and LSW Registers D-20
The FIFO Status Register D-20
The FIFO Reading Count Register D-21
Trigger System Registers D-22
Software Trigger/ARM Register D-22
Trigger Timer Register D-22
The Trigger Mode Register D-22
Register Based Command Reference D-25
System Commands D-26
Calibration Commands D-31
Scan List Commands D-35
CVT Commands D-39
Trigger System Commands D-39
Debugging Commands D-40
Register Based Programming Fundamentals D-42

Resetting the HP E1413 . . . . . D-43
General Register Access D-44
Executing Register Based Commands D-44
Querying the Module D-46
Control Processor States D-48
Programming Sequence D-49
Reset Module to Default State D-49
Programming Module After Reset Sequence D-50

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 E1413 won't have to be removed from its mainframe later. Chapter contents include:

- Configuration the HP E1413 1-1
- About Example Programs 1-9
- Verifying a Successful Configuration 1-10

Configuring the HP E1413

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

- Setting the Logical Address Switch 1-2
- Installing Signal Conditioning Plug-ons 1-3
- Disabling the Input Protect Feature 1-7
- Disabling Flash Memory Access 1-7

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

Switch/Jumper	**Setting**
Logical Address Switch	24
Input Protect Jumper	Protected
Flash Memory Protect Jumper	PROG

NOTE Setting the VXIbus Interrupt Level: The HP E1413 uses a default VXIbus interrupt level of 1. The default setting is made 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.

Setting the Logical Address Switch

Follow the Figure 1-1 below and ignore any switch numbering printed on the Logical Address switch. When installing more than one HP E1413 in a single VXIbus Mainframe, set each instrument to a different Logical Address.

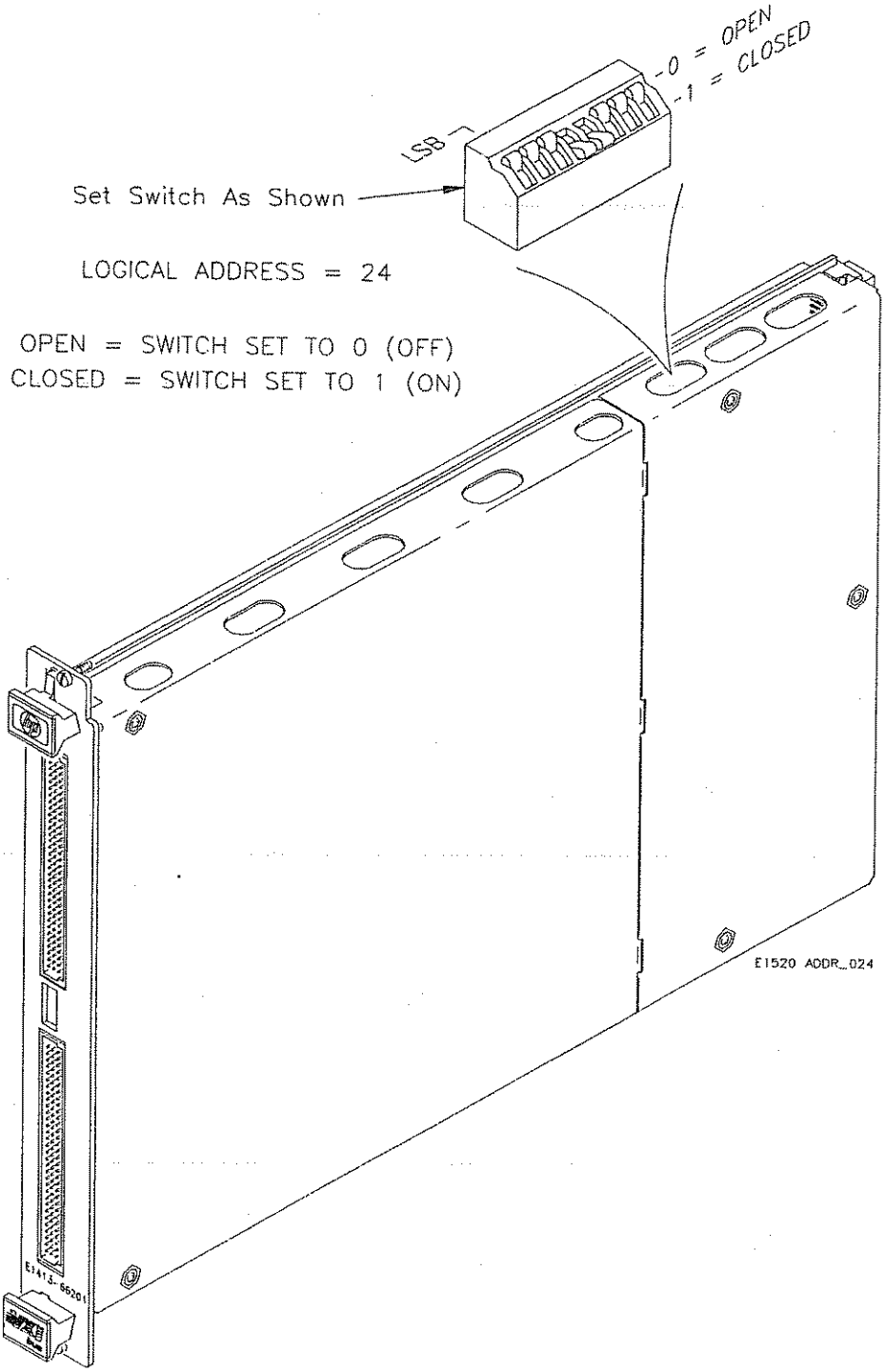


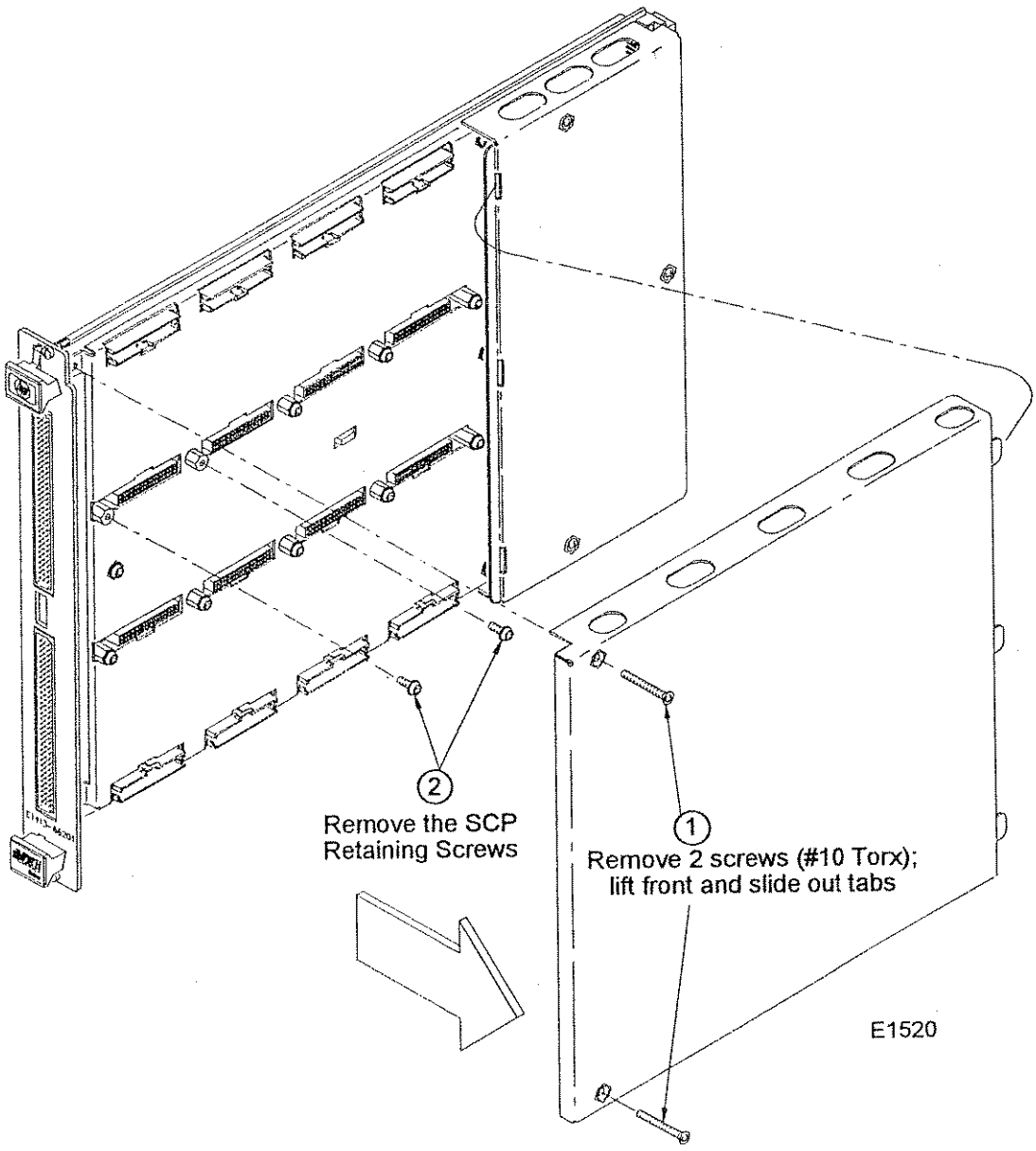
Figure 1-1 Setting Logical Address Switch

Installing Signal Conditioning Plug-ons

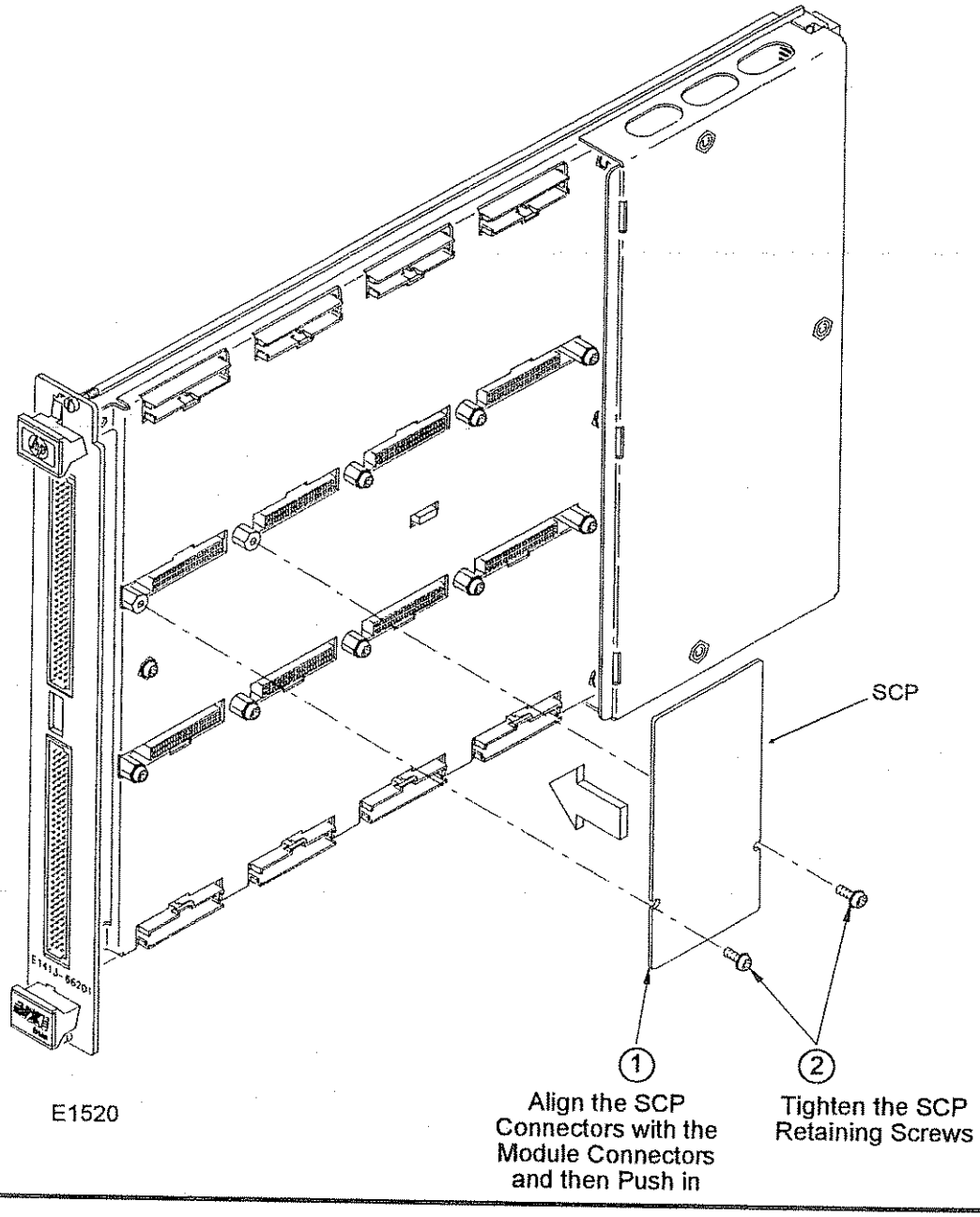
The following illustration shows the steps you'll use to install Signal Conditioning Modules.

CAUTION Use approved Static Discharge Safe handling procedures anytime you have the covers removed from the HP E1413 or are handling SCPs.

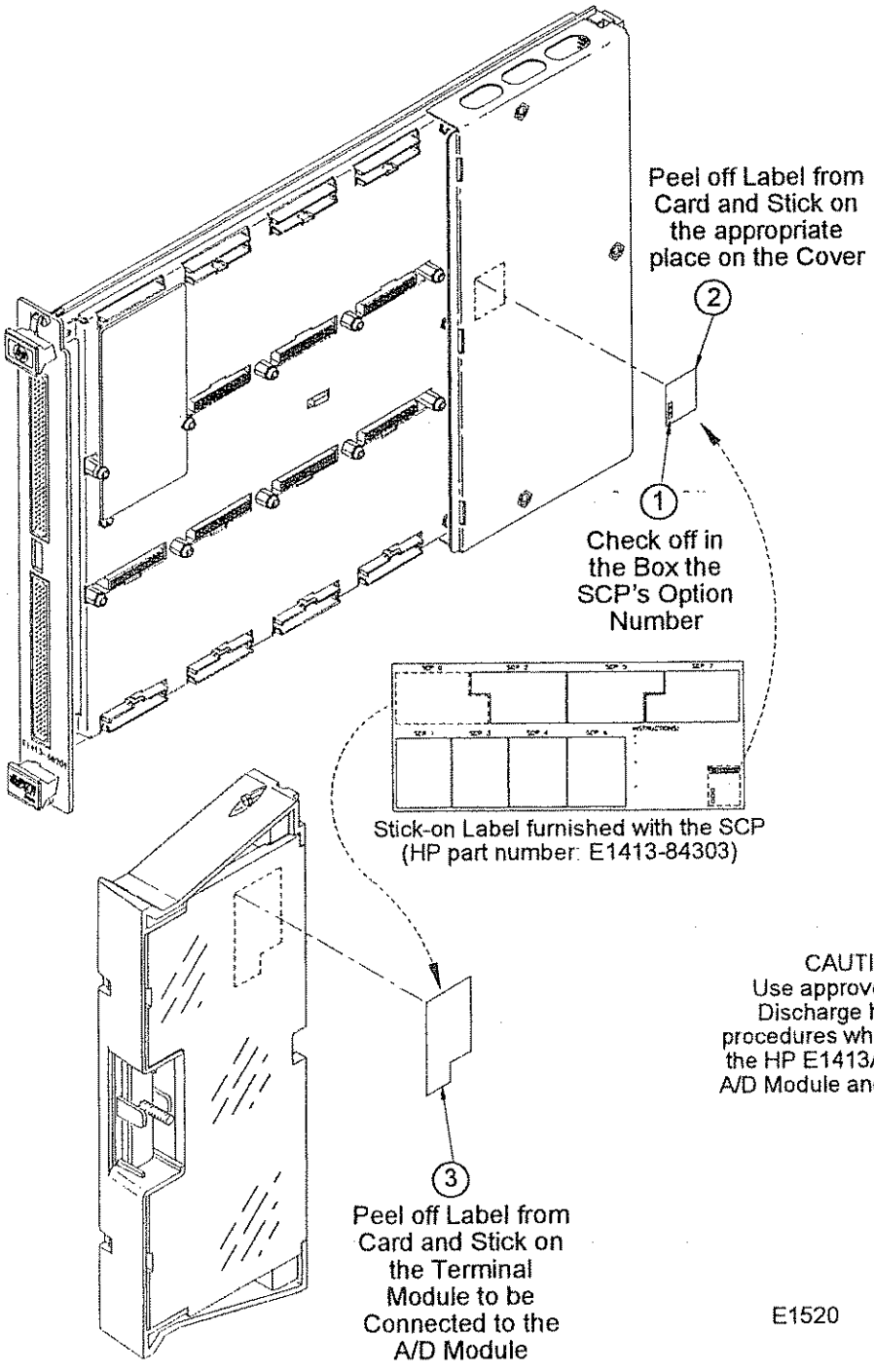
1 Installing SCPs: Removing the Cover



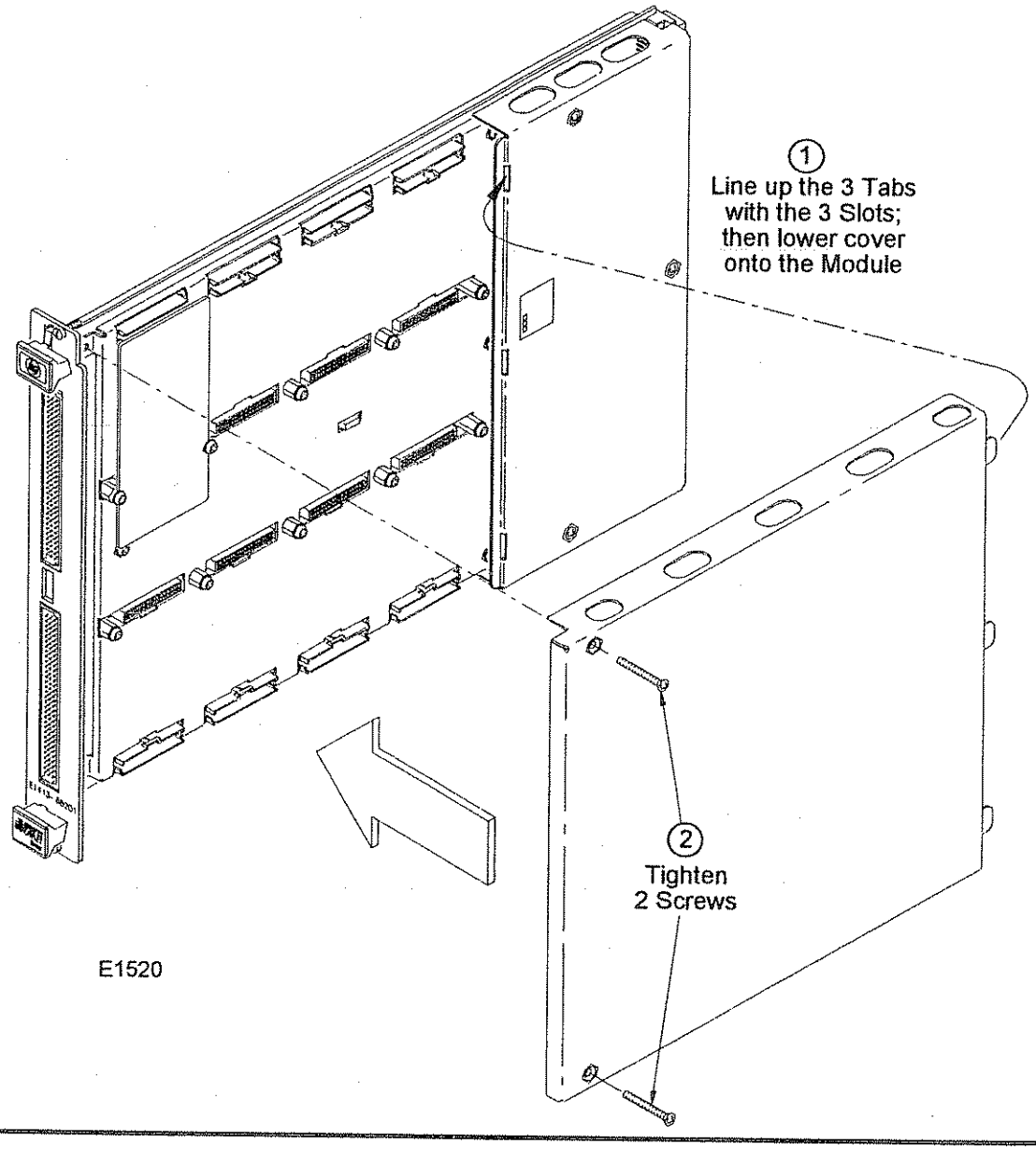
2 Installing SCPs



3 Installing SCPs: Labeling



4 Installing SCPs: Reinstalling the Cover



Disabling the Input Protect Feature (optional)

Disabling the Input Protect feature voids the HP E1413's warranty. The Input Protect feature allows the HP E1413 to open all channel input relays if any input's voltage exceeds ±19 volts. This feature will help to protect the card's Signal Conditioning Plug-ons, input multiplexer, 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 E1413, you may choose to disable the Input Protect feature.**

VOIDS WARRANTY

Disabling the Input Protection Feature voids the HP E1413'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:

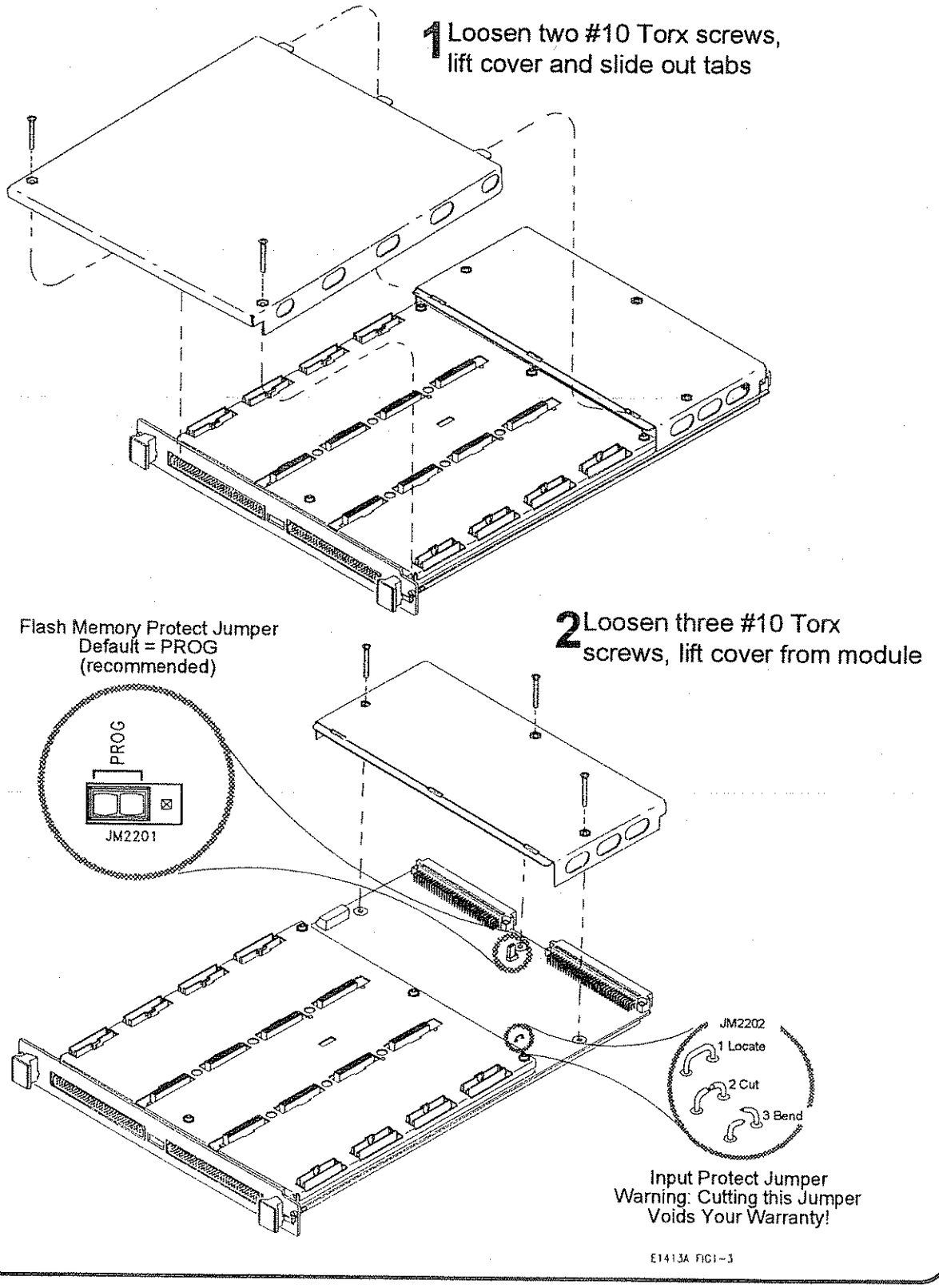
- The SCPI calibration command CAL:STORE ADC | TARE
- The register-based calibration commands STORECAL, and STORETAR
- Any application that installs firmware updates or makes any other modification to Flash Memory through the A24 window.

With the jumper in the "PROG" position, you can completely calibrate one or more HP E1413s without removing them from the application system. An HP E1413 calibrated in its working environment will in general be better calibrated than if it were calibrated separate from its application system.

Consider the multimeter used during the periodic calibration cycle as your transfer standard and have your Calibration Organization control unauthorized access to its calibration constants. See the"HP E1413 Service Manual" for complete information on HP E1413 periodic calibration.

If you want to limit access to the HP E1413'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.

Accessing and Locating JM2201 and JM2202



Instrument Drivers

If you will be using the HP E1413 with C-SCPI, the driver is supplied as an option to the C-SCPI product. Follow the C-SCPI documentation for use.

The HP E1405B/E1406 down-loadable driver is supplied with your HP E1413. See the manual for your HP Command module for down-loading procedures.

About Example Programs

Examples on Disc

All example programs mentioned by file name in this manual are available on the E1413 C-SCPI driver media (both DOS and HP-UX versions). Most of these example programs are also supplied in DOS C versions for users of HP Command Modules on the DOS formatted down-loadable driver disc for the command module (HP E1405B or HP E1406). An IBASIC version of the Verify program shown below is supplied on the LIF formatted down-loadable command module driver.

Examples on the HP Bulletin Board System

The C language examples supplied with the HP E1413 or optional C-SCPI drivers are also available by modem from Hewlett-Packard's Test & Measurement Help Line Bulletin Board (BBS) at 303-679-5978. Check the T&M BBS for the latest versions of these examples as well as additional helpful example programs. The T&M BBS supports 300, 1200, 2400, 9600, and 14,400 Baud modems. Set your communications parameters to 8-bits, no parity, and 1 stop bit. When you connect to the T&M BBS you will be guided to log-on and will receive introductory and tutorial messages. The file FILELIST.VXI contains the file name and one-line description of files available for the HP 75000 Series of VXI products. Download this file and use a word processor to search for "E1413".

Example Command Sequences

Where programming concepts are discussed in this manual, the commands to send to the HP E1413 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 E1413 SCPI commands and in which 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 4-5 for an example.

Typical C-SCPI Example program

The Verify program (file name verif.cs) is printed below to show a typical C-SCPI program for the HP E1413.

Verifying a Successful Configuration

An example C-SCPI (compiled-SCPI) program source is shown on the following pages. This program is included with your C-SCPI driver tape (file name *verif.cs*). The program uses the *IDN? query command to verify the HP E1413 is operational and responding to commands. The program also has an error checking function (*check_error()*). It is imortant to include an instrument error checking routine in your programs, particulary your first trial programs so you get instant feedback while you are learning about the HP E1413. After you run the C-SCPI preprocessor and then compile and load this program, type *verif* to run the example.

Typical C-SCPI program Example

```
/* verif.cs                                                              */
/*      1.) Prints the HP E1413A Module's identification, manufacturer, and  */
/*          revision number.                                              */
/*      2.) Prints the Signal Conditioning Plug-ons (SCPs) identification    */
/*          (if any) at each of the SCP positions.                        */
/*      3.) Takes voltage measurements on channels 100 to 163 and returns the */
/*          readings from the Current Value Table (CVT) and FIFO.         */

#include <stdio.h>
#include <cscpi.h>

      /* Defines module's logical address */
#define LADD "24"

      /* Declares module as a register device */
INST_DECL(e1413, "E1413A", REGISTER);

      /* Prototypes of functions declared later */

void  rst_clr( void );
void  id_scps( void );
void  start_ad( void );
void  get_readng( void );
void  prt_readng( float32 * );
int32 check_error( char * );

/*****************************************************************************/
main() /* Main function */
{
      char      read_id[80];

         /* Clear screen and announce program */
      printf("\033H\033J\n\n              Installation Verification Program\n\n");
      printf("\n\n                          Please Wait...");

         /* Start the register-based operating system for the module */
      INST_STARTUP();

         /* Enable communications to the module; check if successful */
      INST_OPEN(e1413, "vxi," LADD);
      if ( !e1413 )
      {
         printf("INST_OPEN failed (ladd = %s).Failure code is: %d\n",
               LADD,cscpi_open_error);
         exit(1);
      }

         /*  Read and print the module's identification */
      INST_QUERY(e1413, "*idn?", "", read_id);
      printf("\n\nInstrument ID: %s\n\n", read_id);

      rst_clr(); /* Function resets the module */
```

```
        id_scps();      /* Function checks for installed SCPs */

        start_ad();     /* Function sets up the module to make measurements */

        get_readng();   /* Function gets and prints readings */

        exit(0);
}
/*****************************************************************************/
void    rst_clr() /* Reset the A/D module to its power-on state */
{
        int16       opc_wait;

            /* Reset the module and wait until it resets */
        INST_QUERY(e1413, "*RST;*OPC?", "", &opc_wait);

            /* Check for module generated errors; exit if errors read */
        if (check_error("rst_clr"))
            exit(1);
}
/*****************************************************************************/
void    id_scps() /* Check ID of all installed SCPs */
{
        int16       scp_addr;
        char        scp_id[100];

            /* Get SCP identifications of all SCPs */
        printf("\nSCP Identifications:\n\n");
        for (scp_addr = 100; scp_addr <= 156; scp_addr += 8)
        {
            INST_QUERY(e1413, "SYST:CTYP? (@%d)", "%s", scp_addr, scp_id);
            printf("ID for SCP %d is %s\n", (scp_addr - 100) / 8, scp_id);
        }
}
/*****************************************************************************/
void    start_ad() /* Inititalize and trigger A/D then take readings. Default EU type is
        volts and scan list is LIST1 and is defined as all 64 channels */
{
        int16       opc_wait;

                /* Enable the Trigger System */
        INST_SEND(e1413, "INIT");

            /* Check for module generated errors; exit if errors read */
        if (check_error("start_ad (setup module)"))
            exit(1);

            /* Trigger the module to start the measurement process */
        INST_SEND(e1413, "TRIG");

            /* Check for module generated errors; exit if errors read */
        if (check_error("start_ad (trigger module)"))
            exit(1);
}
/*****************************************************************************/
void    get_readng() /* Get the module's readings */
{
        float32     read_data[64];
        char        wait_show[2];

            /* Wait to view previous screen */
        printf("\n\nPress 'Return' to continue");
        while(! gets(wait_show));

            /* Set format of returned data */
        INST_SEND(e1413, "FORMAT REAL,32");

            /* Get readings using FIFO */
        INST_QUERY(e1413, "DATA:FIFO:PART? 64", "", read_data);
```

```
        /* Print the readings */
    printf("\n\nFIFO data:\n\n");
    prt_readng(read_data);

        /* Wait here to view previous screen */
    printf("\n\nPress 'Return' to continue");
    while(! gets(wait_show));

        /* Get readings stored in the CVT */
    INST_QUERY(e1413, "DATA:CVT? (@100:163)", "", read_data);

        /* Print the readings */
    printf("\n\nCVT Data:\n\n");
    prt_readng(read_data);

    printf("\n\n");

        /* Check for module generated errors; exit if errors read */
    if (check_error("get_readng"))
        exit(1);
}
/*****************************************************************************/
void  prt_readng( float32 *read_data ) /* Display readings */
{
    int16    i;

    printf("ch  reading    ch  reading    ch  reading    ch  reading\n");
    printf("-- ---------  -- ---------  -- ---------  -- ---------\n");
    for(i = 0; i < 64; i += 4)
    {
        printf("%2d %13.6e  %2d %13.6e  %2d %13.6e  %2d %13.6e\n",
            i, read_data[i], i+1, read_data[i+1], i+2, read_data[i+2],
            i+3, read_data[i+3]);
    }
}
/*****************************************************************************/
int32  check_error( char *message ) /* Check for module generated errors */
{
    int16    error;
    char     err_out[256];

        /* Check for any errors */
    INST_QUERY(e1413, "SYST:ERR?", "", &error, err_out);

        /* If error is found, print out the error(s) */
    if (error)
    {
        while(error)
        {
            printf("Error %d,%s (in function %s)\n", error, err_out, message);
            INST_QUERY(e1413, "SYST:ERR?", "", &error, err_out);
        }
        return 1;
    }
    return 0;
}
```

Chapter 2
Field Wiring

About This Chapter

This chapter will help you understand how to plan your field wiring for the HP E1413 and how to physically connect your field wiring to the HP E1413's Terminal Module. The chapter will explain proper connection of analog signals to the E1413, both two-wire voltage type and four-wire resistance type measurements. Connections for other types of measurements such as strain, using the Bridge Completion SCP, can be found in the manual section for that specific SCP in the "SCP Manuals" section. Chapter contents include:

- Planning Wiring Layout for the HP E1413 2-1
- Recommended Measurement Connections 2-6
- Reference Temperature Sensor Connections 2-5
- The Terminal Module 2-7

Note An example C-SCPI program entitled **wiretest.cs** is included on the C-SCPI driver tape. When you have completed your field wiring, you can use this program to check for bad connections. The program performs "Open Transducer Detection" (see DIAG:OTD command in Chapter 5 for details) and continuously loops while performing measurements on all 64 channels.

Planning Your Wiring Layout

The first point to understand is that the HP E1413 makes no assumptions about the relationship between Signal Conditioning Plug-on (SCP) function and the position in the HP E1413 that it can occupy. You can put any type of SCP into any SCP position. There are however some factors you should consider when planning what mix of SCPs should be installed in each of your HP E1413s. The following discussions will help you understand these factors.

SCP Positions and Channel Numbers

The HP E1413 has a fixed relationship between Signal Conditioning Plug-on positions and the channels they connect to. Each of the eight SCP positions connect to eight channels. Figure 2-1 shows the channel number to SCP relationship.

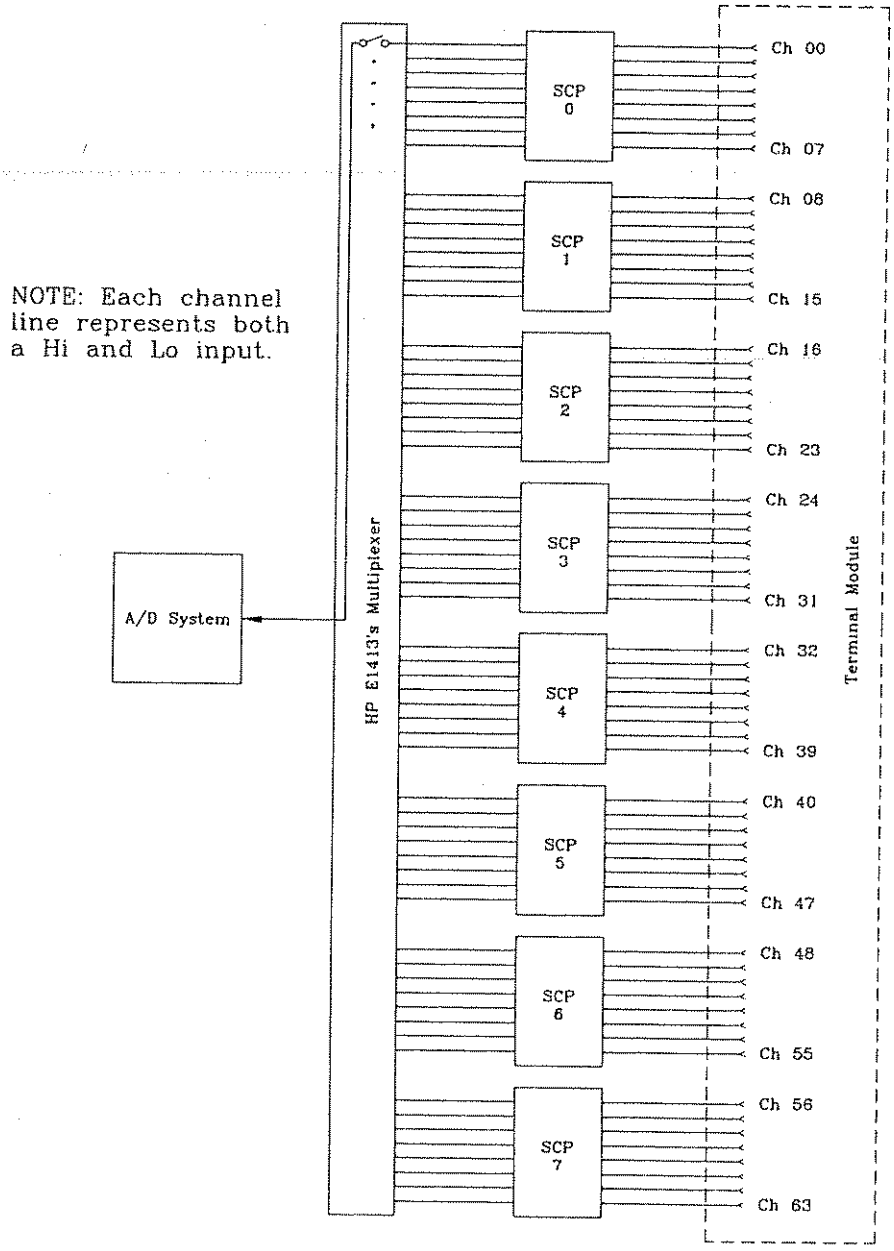


Figure 2-1 Channel Numbers at SCP Positions

Sense SCPs and Output SCPs

Some SCPs provide input signal conditioning (sense SCPs such as filters and amplifiers) while others provide stimulus to your measurement circuit (output SCPs such as current sources and strain bridge completion). In general, channels at output SCP positions are not used for external signal sensing but are paired with channels of a sense SCP. Two points to remember about mixing output and sense SCPs:

1. Paired SCPs (an output and a sense SCP) may reside in separate HP E1413s. SCP outputs are adjusted by *CAL? to be within a specific limit. The Engineering Unit (EU) conversion used for a sense channel will assume the calibrated value for the output channel.
2. Output SCPs while providing stimulus to your measurement circuit reduce the number of external sense channels available to your HP E1413.

Figure 2-2 illustrates an example of "pairing" output SCP channels with sense SCP channels (in this example, four-wire resistance measurements).

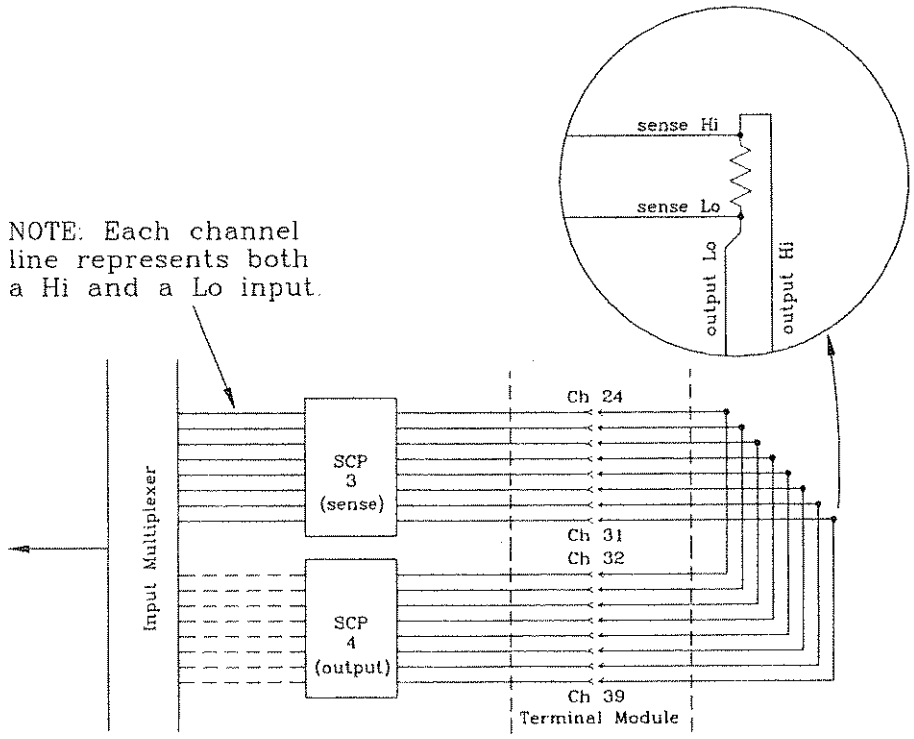


Figure 2-2 Pairing Output and Sense SCP Channels

Planning for Thermocouple Measurements

You can wire your thermocouples and your thermocouple reference temperature sensor to any of the HP E1413'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.

NOTE The isothermal reference temperature measurement made by an HP E1413 applies only to thermocouple measurements made by that instrument. In systems with multiple HP E1413s, each instrument must make its own reference measurements. The reference measurement made by one HP E1413 can not be used to compensate thermocouple measurements made by another HP E1413.

Reference Temperature Sensor Connections

The Terminal Module provides an on-board thermistor for sensing isothermal reference temperature of the terminal blocks. Also provided is a jumper set (JM1 in Figure 2-6) to route the HP E1413's on-board current source to a thermistor or RTD on a remote isothermal reference block. Figures 2-3 and 2-4 show connections for both local and remote sensing.

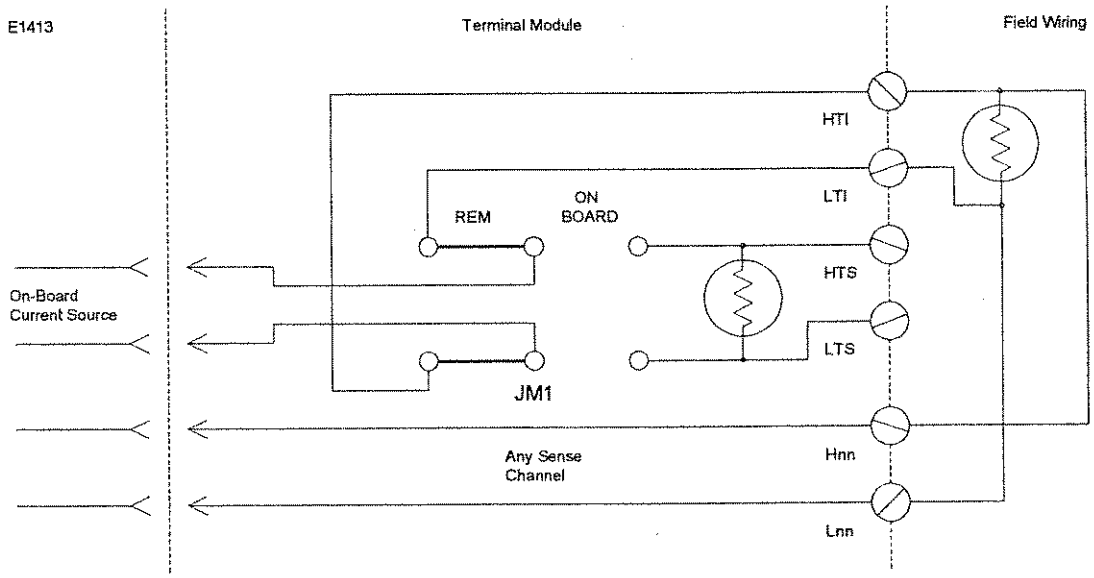


Figure 2-3 Remote Thermistor or RTD Connections

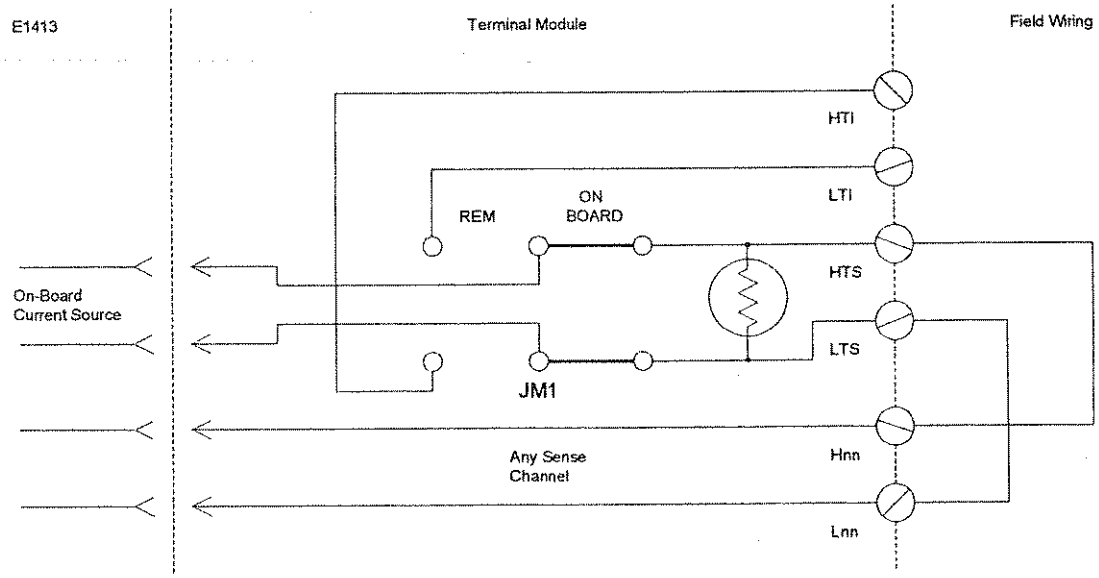


Figure 2-4 On-Board Thermistor Connections

Recommended Measurement Connections

The following illustration shows recommended connections for thermocouple and resistance (and resistance temperature) measurements.

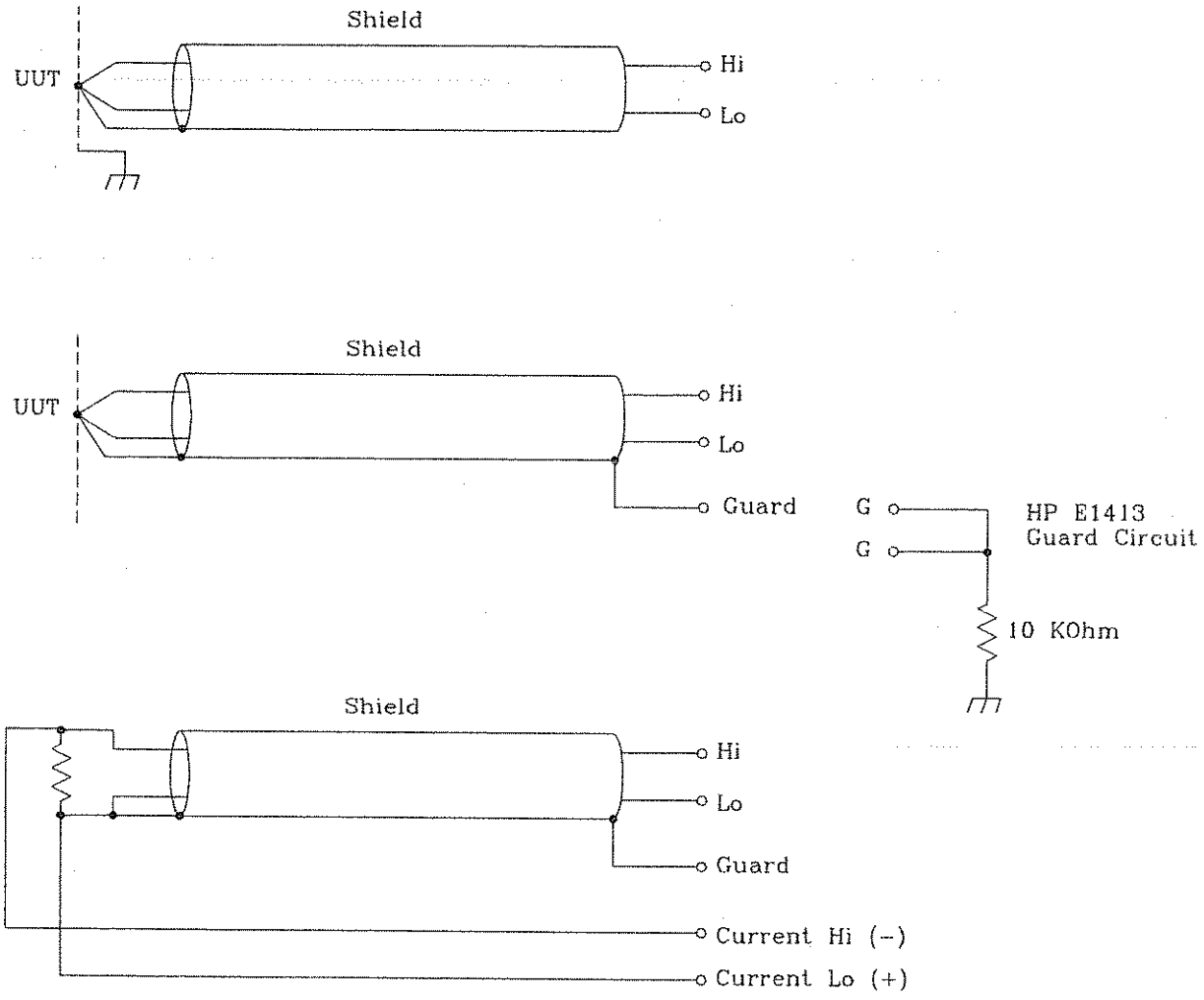


Figure 2-5 Recommended Signal Connections

The Terminal Module

The Terminal Module provides screw terminal connection to field wiring as well as a strain relief for the wiring bundle. The Terminal Module includes reference junction temperature sensing for thermocouple measurements. The same Terminal Module is used for all field wiring regardless of the mix of Signal Conditioning Plug-ons (SCPs) installed in the HP E1413. With each SCP a set of labels is supplied to map that SCP's channels to the Terminal Module's terminal blocks. See step three of "Installing SCPs" in Chapter 1.

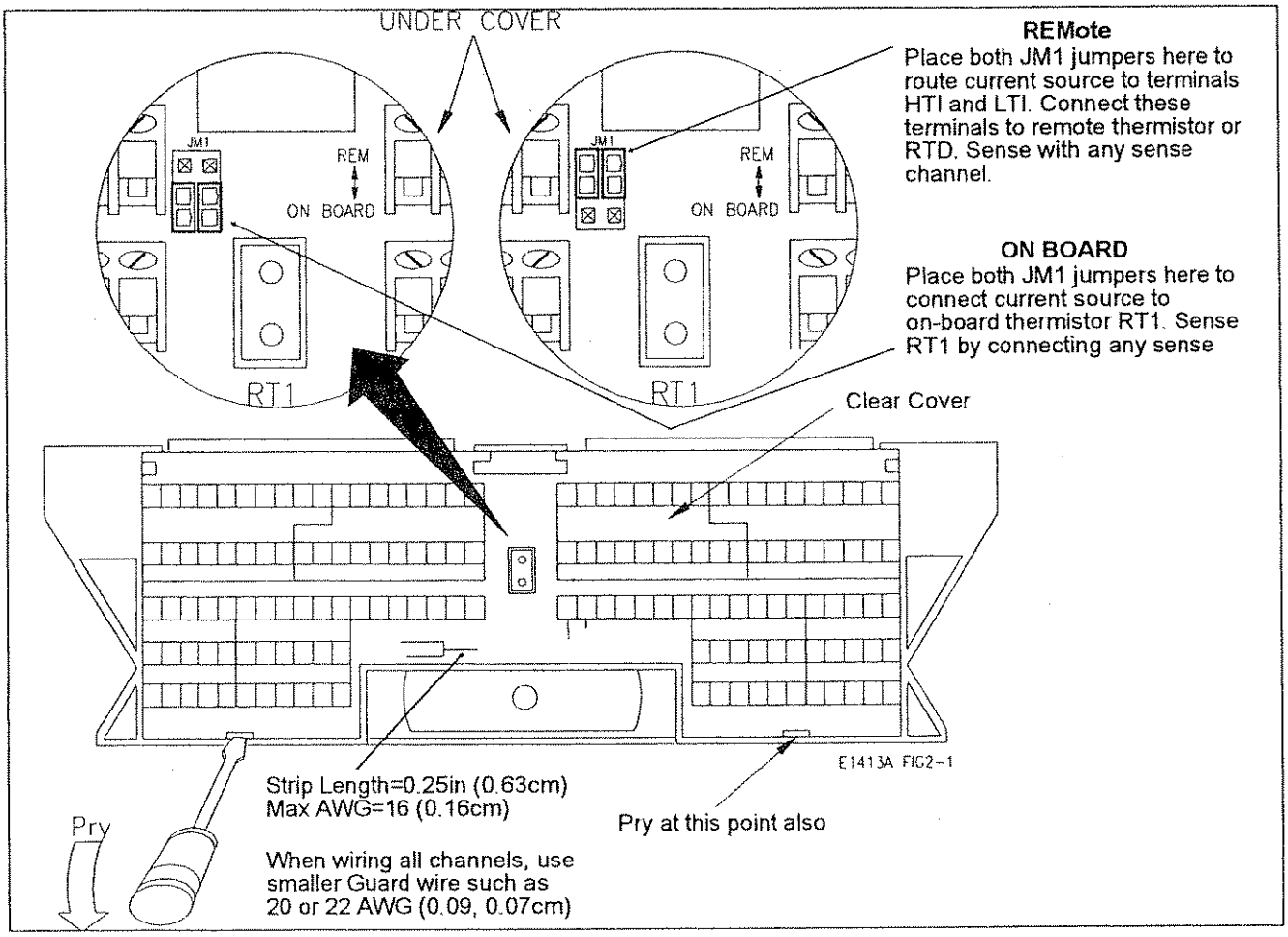


Figure 2-6 The Terminal Module

Opening and Wiring the Terminal Module

The following illustration shows how to open and wire the terminal module.

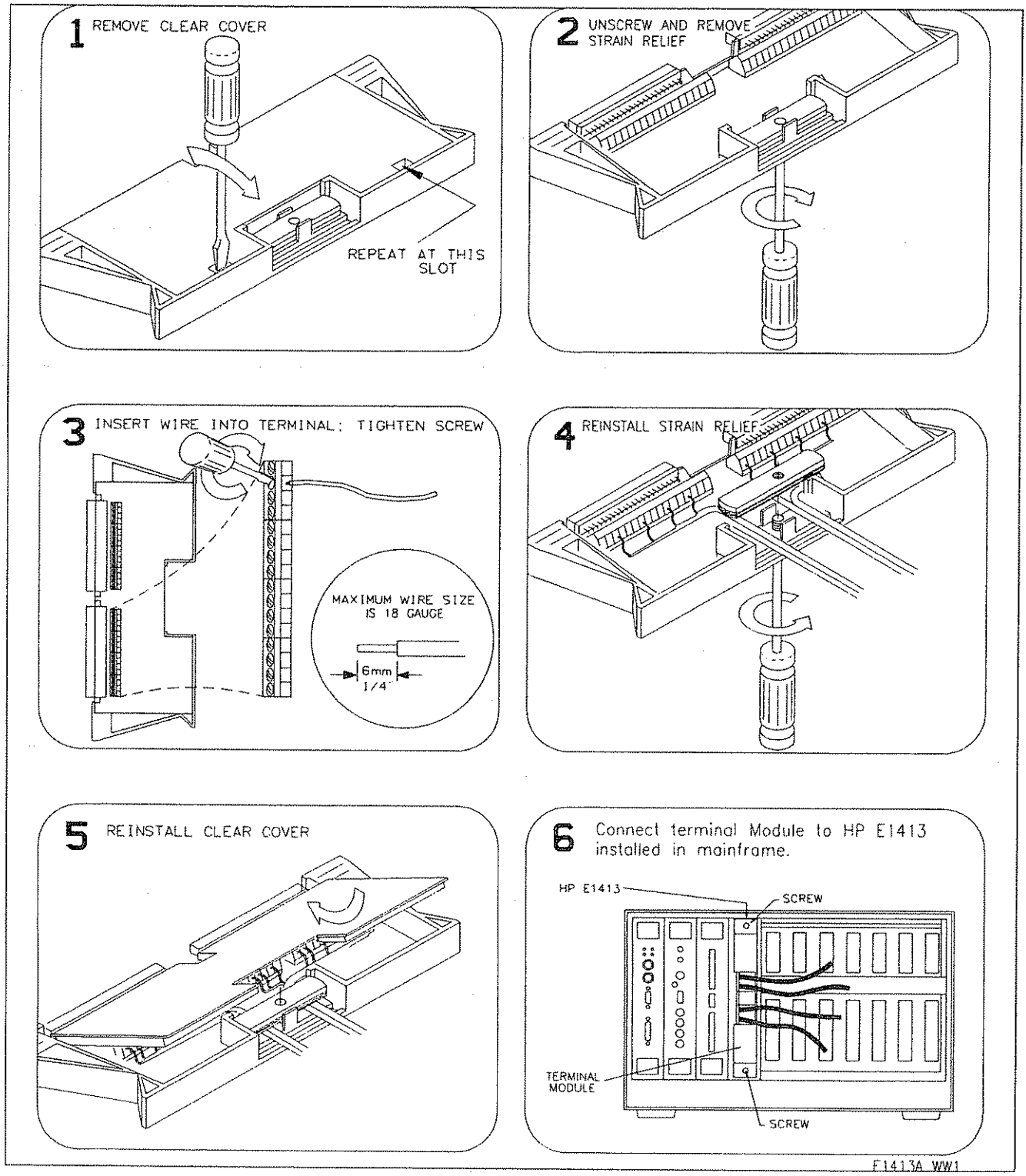


Figure 2-7 Opening And Wiring the Terminal Module

Chapter 3
Using the HP E1413

This chapter introduces programming the HP E1413 with the SCPI instrument language. Chapter contents include:

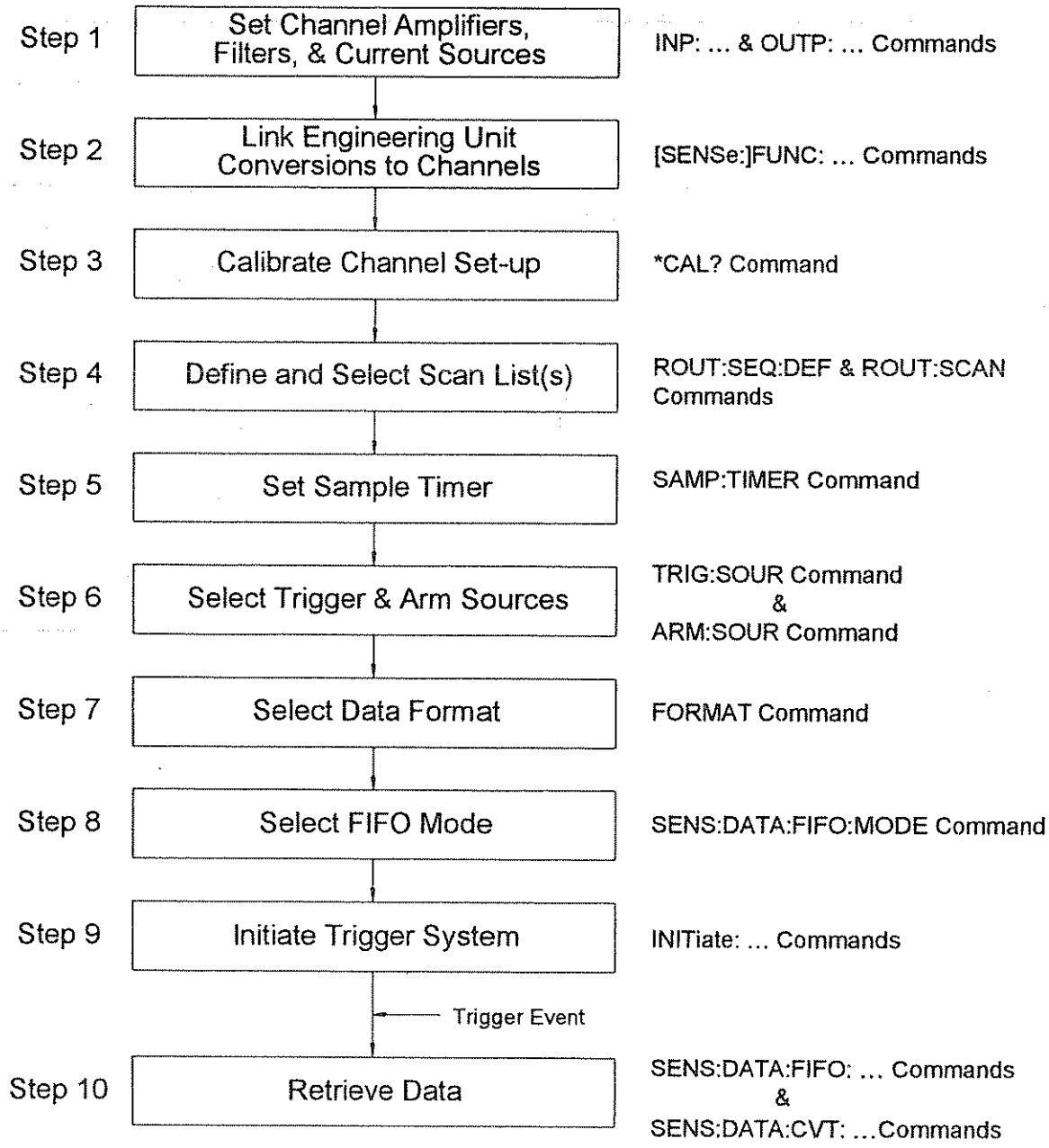
- Module description 3-3
- Fundamental Programming Sequence 3-4
  - Step 1 Setting up Signal Conditioning 3-5
  - Step 2 Linking Channels to EU Conversion 3-7
  - Step 3 Performing Channel Calibration 3-16
  - Step 4 Defining and select the Scan Lists 3-17
  - Step 5 Setting the Sample Timer 3-18
  - Step 6 Setting up the Trigger System 3-19
  - Step 7 Specifying the Data Format 3-21
  - Step 8 Selecting the FIFO Mode 3-22
  - Step 9 INITiating 3-22
  - Step 10 Retrieving Data 3-23
- Additional Topics 3-25
  - Example Program 3-25
  - Example Command Sequence 3-26

Module Description

The HP E1413 is a 64-channel high-speed scanning Analog-to-Digital Converter with optional per-channel signal conditioning in a single width VXIbus module. It scans up to 100,000 channels per second, while auto-ranging and converting the readings into Engineering Units (EU). The reading stream is routed to a 65,024 reading FIFO buffer, while the latest reading from each of the 64 channels can be quickly accessed from a Current Value Table (CVT). All readings are returned to the FIFO and CVT in IEEE 754, 32-Bit floating point format either with EU conversion or as input voltage. Channel selection is controlled by four Scan Lists which can contain up to 1,024 channel assignments each. Another list called the List of Lists can reference each of the Scan Lists (1 through 4) up to 1,024 times. The HP E1413 also provides on-board calibration sources to allow on-line, single command, all-channel calibration. The module can also compensate for system wiring offsets. The HP E1413 can accept any mix of up to eight Signal Conditioning Plug-ons (SCPs). All of these features can be accessed with the SCPI and Compiled SCPI instrument languages. The following sections describe step-by-step how to program the module.

Programming Sequence

The sequence of programming the HP E1413 is important and should be performed as the following flow chart indicates. Follow along with the SCPI Programming Overview Diagram fold-out on the first page of this chapter.



Step 1 Setting up Signal Conditioning Plug-ons

This section does not apply to non-programmable Signal Conditioning Plug-ons such as the Option 11, Straight-Through, or the Option 12, 7 Hz Passive Filter SCPs.

Setting SCP Gains

The gain command for 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 frequency selection. An example for the Option 13 programmable Amp&Filter SCP:

To set the SCP gain to 8 for channels 0, 4, 6, and 10 through 19 send:

```
INP:GAIN 8,(@100,104,106,110:119)
```

To set the SCP gain to 16 for channels 0 through 15, and to 64 for channels 16 through 23 send:

```
INP:GAIN 16,(@100:115)
INP:GAIN 64,(@116:123)
```

or to combine into a single command message:

```
INP:GAIN 16,(@100:115);GAIN 64,(@116:123)
```

Notes:

1. Because of the high bandwidth of the A/D Range Amplifier, the quietest low-level readings are attained by using the highest possible SCP channel gain with the lowest A/D gain (higher A/D *range*) setting appropriate to the measurement to be made. A/D range setting will be discussed in the next programming step.
2. If you are going to manually set the A/D range in the next programming Step (Linking Channels to EU Conversion), you must also take into account the SCP channel gains set in this programming step. In general most measurements can be made at full speed using auto-range. Auto-range will choose the optimum A/D range for the amplified signal level.

Setting Filter Cutoff

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 0, 4, 6, and 10 through 19 send:

```
INP:FILT:FREQ  10,(@100,104,106,110:119)
```

To set 10 Hz cutoff for channels 0 through 15, and 100 Hz cutoff for channels 16 through 23 send:

```
INP:FILT:FREQ  10,(@100:115)
INP:FILT:FREQ  100,(@116:123)
```

or to combine into a single command message

```
INP:FILT:FREQ  10,(@100:115);FREQ  100,(@116:123)
```

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

```
INP:FILT:STAT  ON,(@100:163)
```
*all channel's filters on (same as at *RST)*

```
INP:FILT:STAT  OFF,(@100, 123,146,163)
```
only channels 0, 23, 46, and 63 OFF

Setting Current Sources

Current Source SCPs supply excitation current for all resistance type measurements. These include resistance, and temperature measurements using resistance temperature sensors. The commands to control Current Source SCPs are **OUTPut:CURRent:AMPLitude** *<level>*,(@*<ch_list>*) and **OUTPut:CURRent[:STATe] <enable>**.

- The *level* parameter sets the current output level. It is specified in units of ADC and for the Option 15 SCP 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 0 through 9 to output 30 µA and channels 10 through 19 to output 488 µA:

```
OUTP:CURR  30e-6,(@100:109)
OUTP:CURR  488e-6,(@110:119)        separate command per output level
```

or to combine into a single command message:

```
OUTP:CURR  30e-6,(@100:109);CURR  488e-6,(@110:119)
```

Step 2 Linking Channels to EU Conversion

This step links each of the module's channels to a specific measurement type. 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:VOLTage *<range>*,(@*<ch_list>*) for voltage measurements

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

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

---

NOTE At Power-on and after *RST, the default EU Conversion is autorange voltage for all 64 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: 0, .0625, .25, 1, 4, 16, or AUTO. When not specified or set to zero, the module uses auto-range.

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

```
SENS:FUNC:VOLT 0,(@100:115)          0 for range means auto-range
```

To set channels 16 and 24 to the 16 volt range, and 32 through 47 to the .625 volt range:

```
SENS:FUNC:VOLT 16,(@116,124)
SENS:FUNC:VOLT .625,(@132:147)       must send a command per range
```

or to send both commands in a single command message:

```
SENS:FUNC:VOLT 16,(@116,124);VOLT .625,(@123:147)
```

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 selecting ranges that provide good resolution based on the input signal). In general, measurements can be made at full speed using auto-range. Auto-range will choose the optimum A/D range for the amplified signal level.

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 3-1

- 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 ADC and the choices for the Option 15 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 or set to zero, 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 3-1 for diagrams of these measurement connections.

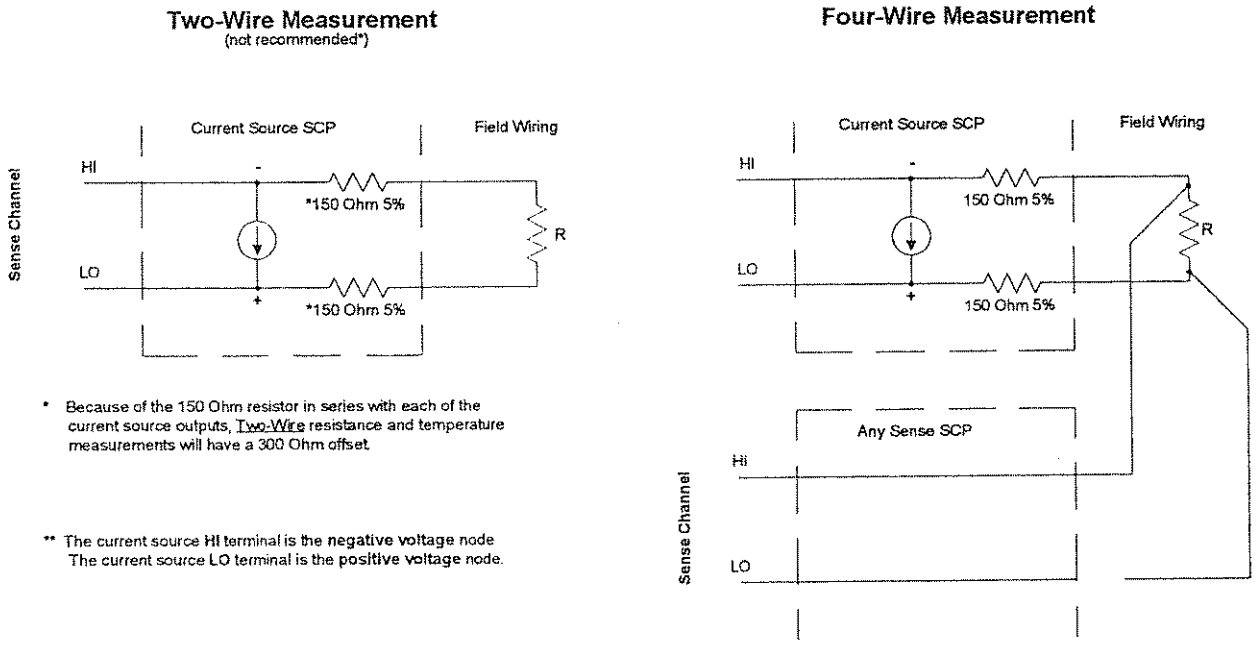


Figure 3-1 Resistance Measurement Sensing

To set channels 0 through 15 to measure resistances greater than 8,000 Ohms and set channels 16,20, and 24 through 31 to measure resistances less than 8K (in this case paired to current source SCP channels 32 through 57):

OUTP:CURR:AMPL 30e-6, (@132:147)

set 16 channels to output 30μA for 8KΩ or greater resistances

SENS:FUNC:RES 30e-6, (@100:115)

link channels 0 through 15 to resistance EU conversion (8KΩ or greater)

OUTP:CURR:AMPL 488e-6, (@148,149,150:157)

set 10 channels to output 488μA for less than 8KΩ resistances

SENS:FUNC:RES 488e-6, (@116,120,124:132)

link channels 16, 20 and 24 through 32 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 or set to zero, 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 3-1 in "Linking Resistance Measurements".

For resistance temperature measurements the *sub_type* parameter specifies:

- For RTDs; "85" or "92" (for 100 Ohm RTDs with 0.00385 or 0.00392 Ohms/Ohm/Degree C temperature coefficients respectively)
- For Thermistors; 2250, 5000, or 10000 (the nominal value of these devices at 25 degrees C)

NOTE 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:

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

NOTE 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 15 to measure temperature using 2,250 Ohm thermistors (in this case paired to current source SCP channels 16 through 31):

```
OUTP:CURR:AMPL 488e-6,(@116:131)
```

set excite current to 488μA on current SCP channels 16 through 31

```
SENS:FUNC:TEMP THER, 2250, (@100:115)
```

link channels 0 through 15 to temperature EU conversion for 2,250Ω thermistor

To set channels 32 through 47 to measure temperature using 10,000 Ohm thermistors (in this case paried to current source SCP channels 48 through 63):

```
OUTP:CURR:AMPL 30e-6,(@148:163)
```

set excite current to 30μA on current SCP channels 48 through 63

```
SENS:FUNC:TEMP THER, 10000, (@132:147)
```

link channels 32 through 47 to temperature EU conversion for 10,000Ω thermistor

To set channels 48 through 63 to measure temperature using 100 Ohm RTDs with a TC of .00385 Ohm/Ohm/°C (in this case paired to current source SCP channels 32 through 47):

```
OUTP:CURR:AMPL 488e-6,(@132:147)
```

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

```
SENS:FUNC:TEMP RTD, 85, (@148:163)
```

link channels 48 through 63 to temperature EU conversion for 100Ω RTDs with .00385 TC.

Thermocouple Measurements

Thermocouple measurements are voltage measurements that the EU conversion changes into temperature readings based on the *sub_type* parameter and latest reference temperature value. As mentioned in the "Linking Voltage Measurements" section, higher SCP channel gain provides quieter measurements. However, it is possible to specify a channel gain high enough to cause an A/D over-range reading for the highest temperature ranges for some thermocouples.

- 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 31 through 63 to measure temperature using type E thermocouples:

first measure or supply reference temperature for channels 31 through 63 (see below)

```
SENS:FUNC:TEMP TC, E, (@131:163)
```

NOTE A Scan List must include a reference temperature channel before related thermocouple channels, or the fixed reference temperature must have been supplied before starting to scan thermocouple channels (see following section)

Thermocouple Reference Temperature

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 a 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

The [SENSe:]REFerence <*type*>, <*sub_type*>,[<*range*>,](@<*ch_list*>) command links channels to the reference temperature EU conversion. When the channel is scanned, the reference temperature is measured and stored in the Reference Temperature Register, the FIFO, and the CVT. The reference value is applied to all subsequent thermocouple channel measurements until another reference temperature value is stored in the Reference Temperature Register.

- The *Ch_list* parameter specifies any sense channel that is connected to the reference temperature sensor.
- The *type* parameter can specify THERmistor, RTD, or CUSTom. This is a resistance temperature measurement and uses the on-board 122 µA current source for all *types*.
- The *sub_type* parameter must specify:

- For RTDs; "85" or "92" (for 100 Ohm RTDs with 0.00385 or 0.00392 Ohms/Ohm/Degree C temperature coefficients respectively)
- For Thermistors; only 5000 (See previous note on page 3-11)
- For CUSTom; only 1 (contact your HP Field Engineer for Custom EU algorithms)

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

To set channel 12 to measure the isothermal reference temperature on the HP E1413's Terminal Module (with built-in 5,000 Ohm thermistor) send

```
SENS:REF THER, 5000, (@112)          on-board thermistor connected to channel 12
```

The same command could set channel 12 to measure a 5K thermistor mounted on a remote reference block. See "Reference Temperature Thermistor Connections" on page 2-5 for connections.

Supplying a Fixed Reference Temperature

The [SENse:]REFerence:TEMPerature <*degrees_c*> command immediately stores the set temperature of a controlled temperature reference junction panel in the Reference Temperature Register. The value is applied to all subsequent thermocouple channel measurements until another reference temperature value is specified or measured.

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 E1413 Options 16 and 17). 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 Option 16 and 17 SCP 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

Command	Bridge Type
:FBENding	Full Bending 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 temperature EU conversion. *ch_list* does not specify channels on the Strain Bridge Completion SCP.

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

To link channels 23 through 30 to the quarter bridge strain EU conversion

SENS:FUNC:STR:QUAR (@123:130) *uses autorange*

Other commands used to set up strain measurements are:
[SENSe:]STRain:POISson
[SENSe:]STRain:EXCitation
[SENSe:]STRain:GFACtor
[SENSe:]STRain:UNSTrained

See the Command Reference Chapter 5 and the HP E1413 Options 16 and 17 manual for more information on strain measurements.

Linking Custom EU Conversions

There are two major steps to link channels to Custom EU Conversions:

1. Load the Custom EU Conversion table values into the E1413. These values will be in the form of a numeric array. (Contact your HP Field Engineer for assistance in generating the EU conversion table values.)

2. Link channels to the Custom EU Conversion table.

Loading Custom EU Tables

There is a specific location in the E1413's memory for each channel's EU Conversion table. When standard EU conversions are specified, the E1413 loads these locations with EU conversion tables copied from its non-volatile FLASH Memory. For Custom EU conversions you must load these table values using either of two SCPI commands.

Loading Tables for Linear Conversions

The **DIAGnostic:CUSTom:LINear** *<table_range>*,*<table_block>*, (@*<ch_list>*) command downloads a custom linear Engineering Unit Conversion table to the HP E1413 for each channel specified.

- *<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 input voltage that the table covers (from -*<table_range>* to +*<table_range>*). The value you specify must be within 5% of: .015625 | .03125 | .0625 | .125 | .25 | .5 | 1 | 2 | 4 | 8 | 16 | 32 | 64.

- *<ch_list>* specifies which channels will have this custom EU table loaded.

Usage Example

Your program puts table constants into array *table_block*

DIAG:CUST:PIEC *table_block*,1,(@132:163) *send table for chs 32-63 to HP E1413*

SENS:FUNC:CUST:PIEC 1,1,(@132:163) *link custom EU with chs 32-63 and set the 1V A/D range*

INITiate then TRIGger module

Loading Tables for Non Linear Conversions

The **DIAGnostic:CUSTom:PIECewise *<table_range>*,*<table_block>*,** (@*<ch_list>*) command downloads a custom piecewise Engineering Unit Conversion table to the HP E1413 for each channel specified.

- *<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 input voltage that the table covers (from -*<table_range>* to +*<table_range>*). The value you specify must be within 5% of: .015625 | .03125 | .0625 | .125 | .25 | .5 | 1 | 2 | 4 | 8 | 16 | 32 | 64.

- *<ch_list>* specifies which channels will have this custom EU table loaded.

Usage Example

Your program puts table constants into array *table_block*

DIAG:CUST:PIEC *table_block*,1,(@124:131) *send table for chs 24-31 to HP E1413*

SENS:FUNC:CUST:PIEC 1,1,(@124:131) *link custom EU with chs 24-31 and set the 1V A/D range*

INITiate then TRIGger module

Linking Custom EU Tables

The **[SENSe:]FUNCtion:CUSTom [*<range>*,](@*<ch_list>*)** command links chanels with the custom Engineering Unit Conversion table loaded with the DIAG:CUST:LINEAR or DIAG:CUST:PIECE commands.

- The <range> parameter selects one of the E1413's voltage ranges: .0625 | .25 | 1 | 4 | 16. To select a range, simply specify the range Specifying 0 selects the lowest range (.0625VDC). Specifying AUTO selects auto range. The default range (no range parameter specified) is auto range. 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.

Usage Example

```
program must put table constants into array table_block
DIAG:CUST:LIN 1,table_block,(@116:123)    send table to HP E1413 for chs 16-23
SENS:FUNC:CUST 1,(@124:131)               link custom EU with chs 24-31 and set the 1V A/D range.
INITiate then TRIGger module
```

Step 3 Performing Channel Calibration

At this point in the programming sequence, with the signal conditioning set-up and the EU conversions assigned to specific channels, a Channel Calibration should be performed.

To perform the Channel Calibration, send the command

```
*CAL?
now read the value returned (0 means successful; see *CAL? in Chapter 5 page 5-111 for other values)
```

The *CAL? command causes the module to calibrate A/D offset and gain, and all channel offsets. This takes several minutes to complete. The actual time it will take your HP E1413 to complete *CAL? depends on the mix of SCPs installed. Filtered channels (especially those with large voltages applied) take more time to settle than unfiltered channels.

To perform Channel Calibration on multiple HP E1413s, use the CAL:SETup command (see Chapter 5 for details).

NOTE Because Channel Calibration is dependent on the module set-up at the time of execution, changing the set-up will require another Channel Calibration.

Step 4 Defining and Selecting the Scan Lists

Defining the Scan Lists

In this programming step, you will create the actual list(s) of channels to be scanned. With the HP E1413 you can define up to 4 different Scan Lists each with its own channel-to-channel sample pacing (see Setting the Sample Timer). The Scan Lists specify the sequence that channels will be scanned. Each Scan List can contain up to 1,024 channel references, so channels can be referenced multiple times.

The command to define channel sequence for each Channel List is **ROUTe:SEQuence:DEFine <*scan_list*>, (@<*ch_list*>)**

- The *scan_list* parameter can be one of LIST1, LIST2, LIST3, LIST4 or ALL.
- The *ch_list* parameter must specify at least 2 channels. The channels can be specified in any order. The same channel can be specified more than once. A Scan List can contain as many as 1,024 channel specifiers.
- When ROUT:SEQ:DEF is executed the scan list specified is cleared and then defined according to <*ch_list*>. This means that the entire channel specification for a scan list must be sent in <*ch_list*> with a single ROUT:SEQ:DEF command.
- After a *RST command or at power-on Channel List 1 (LIST1) is predefined as ROUT:SEQ:DEF LIST1,(@100:163).

To define Scan List 1 as channels 0 through 31, 40, and 48 through 63

```
ROUT:SEQ:DEF LIST1, (@100:131,140,148:163)
```

Selecting the Current Scan List

When the module is triggered it will execute the Current Scan List. After the *RST command or at power-up, the Current Scan List is List 1. To select another scan list, execute the ROUTe:SCAN <*scan_list*> command.

- The *scan_list* parameter can specify "LIST1", "LIST2", "LIST3", or "LIST4"
- ROUT:SCAN is not sequence sensitive. It can be executed at any time, even while the module is scanning. Normally the specified scan list number becomes effective when the Trigger System moves from the Initiated State to the Waiting for Trigger State. If ROUT:SCAN is executed after this point, it will become effective for the next scan. If INIT:CONT is ON and TRIG:SOUR is IMM, ROUT:SCAN will generate an error. See Figure 3-2.

To select Scan List 2 as the current Scan List:

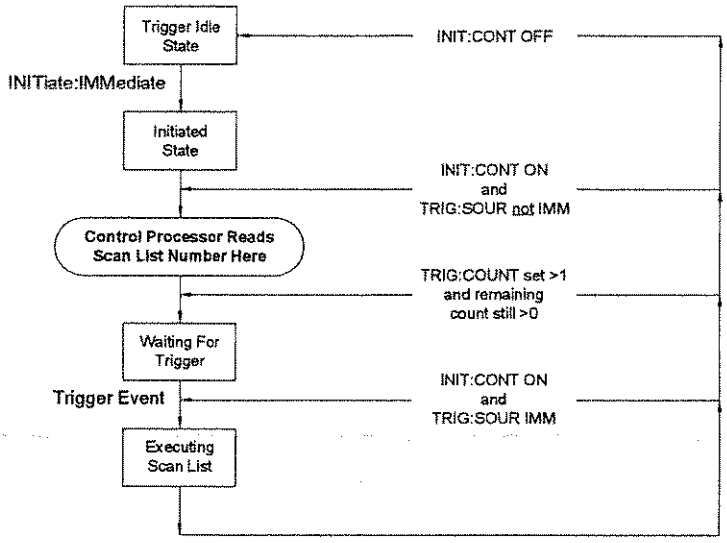


Figure 3-2 Event Sequence for ROUTe:SCAN

```
ROUT:SCAN LIST2          Current Scan List becomes 2 when module Initated to Waiting-for-Trigger
```

Step 5 Setting the Sample Timer

The SAMPle:TIMer *<scan_list>,<interval>* command individually programs each of the four Scan Lists for channel-to-channel sample pacing. The power-on default allows maximum sample rate.

- The *<scan_list>* parameter can specify LIST1, LIST2, LIST3, or LIST4.
- The *<interval>* parameter can specify 10.0e-6 (MIN) to 32.7680e-3 (MAX) seconds with a resolution of 0.5e-6 seconds.

To set the sample timer for Scan List 1 to .005 seconds

```
SAMP:TIM  LIST1,5ms      5 ms for each channel in scan list 1
```

Step 6 Setting up the Trigger System

The Trigger and Arm model

Figure 3-3 shows the trigger and arm model for the HP E1413. Note that when the Trigger Source selected is TIMer, the remaining sources become Arm Sources. An Arm Source allows you to specify an event that must occur in order to start the Trigger Timer.

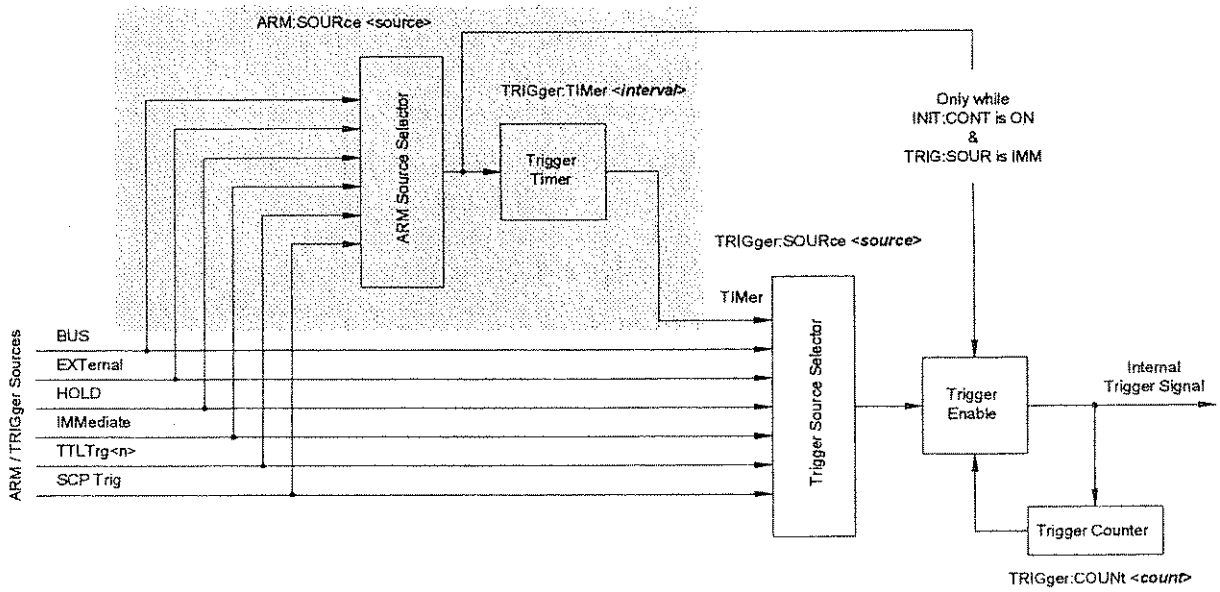


Figure 3-3 Logical Arm and Trigger Model

Selecting the Trigger Source

In order to start a measurement scan 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* parameter.

Parameter Value	Source of Trigger (after INITiate:... command)
BUS	TRIGger[:IMMediate], *TRG, GET (for HP-IB)
EXTernal	"Trig" 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 (ROUT:SEQ:DEF) in the active scan list (ROUT:SCAN) or a +3012, "Trigger Too Fast" error will be generated during the measurement scan. See the Command Reference for TRIG:TIM or SAMP:TIM for timing details.

To set the Trigger Source to a software trigger (TRIG:IMM, or *TRG)

```
TRIG:SOUR HOLD                    trigger on TRIG or *TRG
```

To set the trigger source to the external trigger input connection

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

To set the trigger source to the internal trigger interval timer

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

Selecting Timer Arm Source

Figure 3-3 shows that when the trigger source is the Trigger Timer (TRIG:SOUR TIM), the other trigger sources are used to arm the Trigger Timer. The command to select the Trigger Timer arm source is **ARM:SOURce** *<source>*.

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

Parameter Value	Source of Arm (after INITiate:... command)
BUS	ARM[:IMMediate]
EXTernal	"Trig" 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)

NOTES

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
```

Setting the Trigger Counter

The Trigger Counter controls how many trigger events will be allowed to start a measurement scan. 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 1 (returns to the Trigger Idle State after each scan). if TRIG:COUNT is set to INF, the trigger counter is disabled and the module will accept an unlimited number of trigger events.

To set the trigger counter to 100:

```
TRIG:COUNT 100          can trigger 100 times after a single INIT command.
```

Step 7 Specifying the Data Format

The format of the readings 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 readings will be converted as they are transferred from the CVT and FIFO.

- 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 (readings converted)
ASCii,7	means 7-bit ASCII (readings converted)
ASCii	same as above (the *RST condition)

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

```
FORMAT REAL,32
```

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

```
FORMAT  ASC,7
or
FORM ASC
```

Step 8 Selecting the FIFO Mode

The HP E1413's FIFO can operate in two modes:

- Blocking; The FIFO stops accepting readings when it becomes full (65,024 readings). Readings made after FIFO is full are discarded. The first reading to exceed 65,024 sets the STAT:QUES:COND? bit 10 (FIFO Overflowed), and an error message is put in Error Queue (read with SYS:ERR? command). No readings stored in FIFO are lost.

- Overwrite; When FIFO fills, the oldest readings in the FIFO are overwritten by the newest readings. Only the latest 65,024 readings are available.

The differences in these two modes is only significant if the rate of retrieving data from the FIFO is slower than the reading rate of the A/D.

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

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

Step 9 Initiating the Trigger System

The commands to initiate the HP E1413 are:

INITiate[:IMMediate]

and

INITiate:CONTinuous ON

The INITiate commands move the HP E1413 from the Trigger Idle State to the Waiting For Trigger State. When initiated, the instrument is ready to receive one (:IMMediate) or more (:CONTinuous ON) trigger events.

To initiate the instrument for the number of triggers specified with the TRIG:COUNT command:

INIT or INIT:IMM

To have the instrument continuously in the Waiting For Trigger State:

INIT:CONT ON

If INIT:CONT ON is and TRIG:SOUR is IMM, the module scans continuously until you execute **INIT:CONT OFF**.

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
- Have programmable SCP gain or Current Source SCP settings changed since the *CAL? command was executed?

Step 10 Retrieving Data

Each reading made during a scan is (by default) stored in two places:

- The 64 channel Current Value Table (CVT). As the name implies, the CVT contains the most current value read for each channel.
- The 65,024 reading FIFO Buffer. The FIFO allows the A/D to maintain its high reading rate while your program transfers readings from this buffer.

Readings can be continuously retrieved while the instrument is scanning.

Accessing the CVT

A single command provides access to the channel readings in the CVT:

[SENse:]DATA:CVTable? (@<ch_list>)

- The *<ch_list>* parameter specifies which channel value(s) to retrieve from the CVT.

NOTE 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 negative overvoltage return IEEE -INF. Refer to the FORMat command in Chapter 5 for the NaN, +INF, and -INF values for each data format.

To access the latest reading from each of the instrument's 64 channels

```
DATA:CVT? (@100:163)          completes when 64 readings have been accepted by an input statement
execute program input statement here    must input 64 readings
```

To reset the CVT (and set all values to NaN), send the command **[SENSe:]DATA:CVTable:RESet**.

Accessing the FIFO

The command we'll use here to access the contents of the FIFO buffer is **[SENSe:]DATA:FIFO:PART? <*n_readings*>**. This command returns the number of readings specified by <*n_readings*> (2,147,483,647 maximum). The command completes only after *n_readings* have been transferred. This in not the only FIFO data retrieval command. There are four other commands and they are covered in "Advanced FIFO Data Retrieval" in Chapter 4.

NOTE Channel readings which are a positive overvoltage return IEEE +INF and negative overvoltage return IEEE -INF.

To transfer readings from the instrument, simply set <*n_readings*> to the exact number of readings to be made during all scans. The following is a guide:

$$n\_readings = (number\ of\ channels\ in\ Scan\ List) \times (number\ of\ scans)$$

The *number of channels* is determined by ROUTE:SEQ:DEFINE command. The instrument executes one scan per trigger. The TRIGger:SOURce, TRIGger:COUNt, and the INITiate:CONTinuous mode determine how many times the list of channels will be scanned. When INIT:CONT is ON, the number of triggers generated by TRIG:SOURce and the specified TRIG:COUNT control the *number of scans*.

To transfer 1,024 readings from the instrument

```
DATA:FIFO:PART? 1024          completes after 1,024 readings
                              are accepted by an input
                              statement
execute program input statement here   must take all readings specified
                                       above.
```

Additional Topics

See Chapter 4 "Understanding the HP E1413" for:

- Advanced FIFO Data Retrieval
- Controlling Data Conversion and Destination
- Understanding Scanning Modes
- Triggering and Scanning Modes
- Using the Status System
- Error Trapping
- VXI Interrupts
- Compensating for System Offsets
- Detecting Open Transducers
- Thermocouple Reference Compensation
- Reducing Settling Waits

Example Program

An example C-SCPI program (filename **using.cs**) is located on the C-SCPI driver tape. This program puts together all of the steps discussed so far in this chapter.

Example Command Sequence

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

Reset the module

```
*RST
```

Setting up Signal Conditioning (step 1 only for programmable SCPs)

```
INPUT:FILTER:FREQUENCY 2,(@116:119)
INPUT:GAIN 64,(@116:119)
INPUT:GAIN 8,(@120:123)
```

link channels to EU conversions (step 2)

```
SENSE:FUNCTION:VOLTAGE AUTO,(@100:107)
SENSE:REFERENCE THER,5000,AUTO,(@108)
SENSE:FUNCTION:TEMPERATURE TC,T,AUTO,(@109:123)
```

execute channel calibration (step 3)

```
*CAL?
```

define and select scan list (step 4)

```
ROUTE:SEQUENCE:DEFINE LIST1,(@108:123,100:107)
ROUTE:SCAN LIST1                          this is a *RST default
```

set sample timer (step 5)

```
SAMPLE:TIMER LIST1,.00001
```

Configure the Trigger System (step 6)

```
ARM:SOURCE IMMEDIATE                      this is a *RST default
TRIGGER:COUNT 10
TRIGGER:TIMER .0007
TRIGGER:SOURCE TIMER
```

specify data format (step 7)

```
FORMAT REAL,32
```

select FIFO mode (step 8)

```
SENSE:DATA:FIFO:MODE BLOCK
```

initiate trigger system (step 9)

```
INITIATE:CONTINUOUS OFF                   this is a *RST default
INITIATE
```

retrieve data (step 10)

```
SENSE:DATA:FIFO:PART? <read_data>
```

Chapter 4
Understanding the HP E1413

This chapter introduces more advanced SCPI programming procedures for the HP E1413 and includes programming examples. Chapter contents include:

- Advanced FIFO Data Retrieval 4-1
- Controlling Data Conversion and Destination 4-7
- Understanding Scanning Modes 4-8
- Triggering and Scanning Modes 4-11
- Using the Status System 4-21
- HP E1413 Background Operation 4-25
- Updating the Status System and VXI Interrupts 4-25
- Compensating for System Offsets 4-27
- Detecting Open Transducers 4-29
- Thermocouple Reference Compensation 4-29
- Reducing Settling Waits 4-31

Advanced FIFO Data Retrieval

The discussion of FIFO data access in Chapter 3 ("Retrieving Data") assumed that the number of readings to be transferred from the HP E1413 is a known quantity. The DATA:FIFO:PART? command works well in that situation. This section is applicable when:

- You don't want the data retrieval section of your program to "need to know" the total number of readings to be transferred during scanning. Making the data Retrieval section independent of the configuration section reduces the complexity of your program.
- You can't define the total number of readings that will be transferred. This is very likely when the number of scans to be made is open-ended (INIT:CONT is ON or TRIG:COUNT is set to INF).
- Your system contains several HP E1413s and your program must execute as fast as possible to maintain a reading transfer rate greater than the total reading acquisition rate.

The commands used to access the contents of the FIFO buffer are:

FIFO Reading Transfer Commands

[SENSe:]DATA:FIFO[:ALL]? returns all readings in the FIFO. This command completes only after measuring stops or 65,024 readings have been transferred.

[SENSe:]DATA:FIFO:HALF? returns 32,768 readings ("half" of the FIFO capacity) when they become available. This command completes only after the 32,768 readings are transferred.

[SENSe:]DATA:FIFO:PART? *<n_readings>* returns the number of readings specified by *<n_readings>* (2,147,483,647 maximum). This command completes only after *n_readings* have been transferred.

FIFO Status Commands

[SENSe:]DATA:FIFO:COUNt? returns a count of the readings in the FIFO buffer. Use with the DATA:FIFO:PART? or DATA:FIFO:ALL? commands

[SENSe:]DATA:FIFO:COUNt:HALF? returns a 1 if the FIFO is at least half full (32,768 readings) or a 0 if not. Use with the DATA:FIFO:HALF? command.

All of these FIFO commands can execute while the instrument continues to take readings. Once a FIFO Reading Transfer command is executed, the instrument can not accept other commands until the transfer is complete as specified for each command above. The FIFO Status commands allow you to poll the instrument for availability of readings before executing a transfer command.

General Form of the Data Retrieval Section

Figure 4-1 shows program flow in the data retrieval section for a single HP E1413. The conditions before entering the data retrieval section are:

- Signal conditioning is set up
- Channel EU functions are set
- Scan List is defined
- Instrument initiated (INIT sets the Measuring bit checked in the first decision block in Figure 4-1)

It is not necessary that the instrument be triggered at this time since the program will loop waiting for readings to appear in the FIFO buffer.

While the instrument is still making measurements (Measuring Bit is 1), the program will stay on the left side of the flow chart.

By executing a FIFO status command, the program will determine when there are enough readings to transfer.

When there are enough (count depends in which status FIFO command used), flow will fall through to execute the FIFO data transfer command.

After transferring readings, program flow returns to the decision block which determines if the instrument has completed all measurements or is still measuring.

If measurements are complete, (Measuring Bit is 0), program flow goes to the right side of the diagram.

Here a FIFO status command is executed to determine if there are any readings remaining in the FIFO.

If readings are present, the program executes a FIFO data retrieval command to transfer the remaining readings.

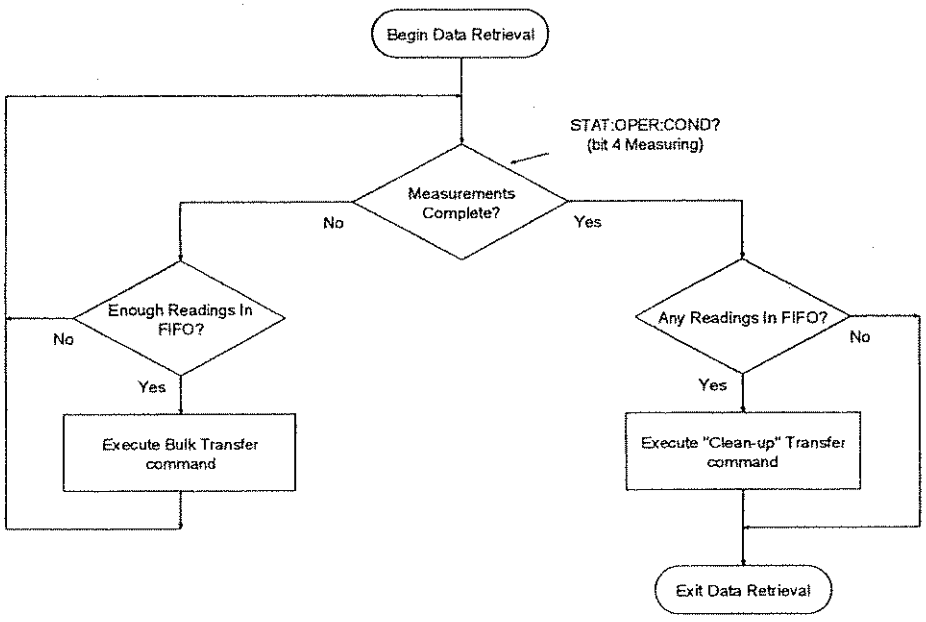


Figure 4-1 General Form of FIFO Data Retrieval

Choosing the Data Retrieval Method

There are two command sets available to retrieve large blocks of data as diagrammed in Figure 4-1.

Controlled Reading Count

The first allows you complete control of the number of readings returned: The FIFO commands used are:

DATA:FIFO:COUNT?	*to determine the number of readings in the FIFO*
DATA:FIFO:PART? <n_readings>	*to retrieve n_readings from the FIFO*

With these commands you can determine exactly how many readings are in the FIFO and transfer just that many. The program flow is just as described in the General Form section earlier. Figure 4-2 shows program flow with this command set filled in.

Note The example program *counted.cs* on your C-SCPI driver tape shows how to retrieve data using controlled reading count.

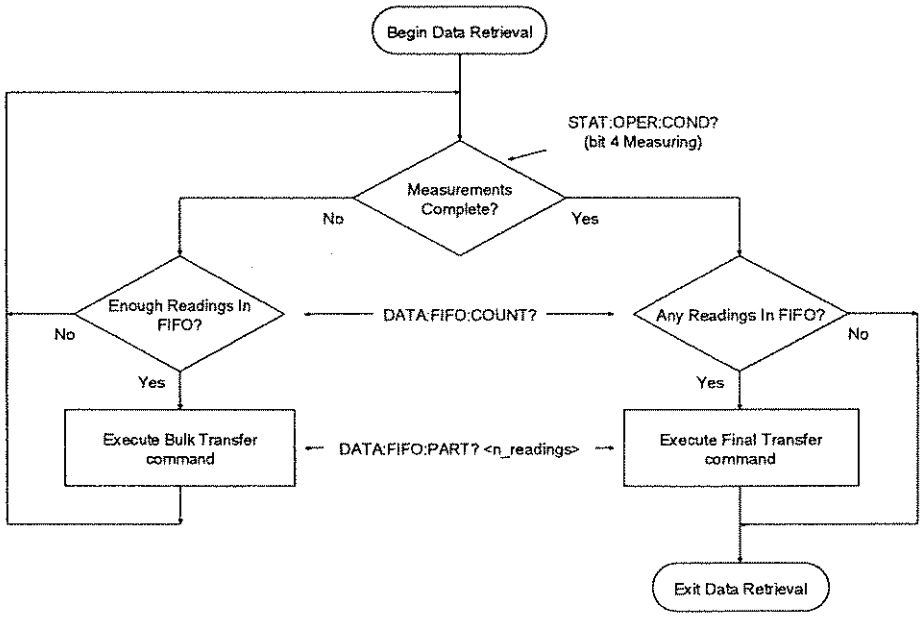


Figure 4-2 Controlling Reading Count

Example Command Sequence

```
    counted

    read_data is a 32-bit floating point array

    n_readings is an unsigned 16-bit integer
*RST
SENSE:FUNCTION:VOLTAGE AUTO,(@100:163)      this is a *RST default
ROUTE:SEQUENCE:DEFINE LIST1,(@100:163)      this is a *RST default
ROUTE:SCAN LIST1                            this is a *RST default
SAMPLE:TIMER LIST1,.00001
ARM:SOURCE IMM
TRIGGER:COUNT 16384
TRIGGER:TIMER .0007
TRIGGER:SOURCE TIMER
FORMAT REAL,32
INITIATE
    following loop reads number of readings in FIFO while module is scanning
loop while "measuring" bit is true          see STAT:OPER:COND bit 4
    DATA:FIFO:COUNT? <n_readings>
    if n_readings >= 16384
        DATA:FIFO:PART? <read_data>
    end if
end loop
    following checks for readings remaining in FIFO after "measuring" false
DATA:FIFO:COUNT? <n_readings>
if n_readings                               if any readings...
    DATA:FIFO:PART? <read_data>             get readings from FIFO
end if
```

Fastest Reading Transfer

This command set is made up of the following commands:

DATA:FIFO:COUNT:HALF?	*to determine if there are at least 32,768 readings in the FIFO*
DATA:FIFO:HALF?	*to transfer one half of the FIFO (32,768 readings)*
DATA:FIFO:ALL?	*to return the remaining readings from the FIFO* **after** *scanning is complete*

What makes this method fastest is that DATA:FIFO:HALF? and DATA:FIFO:ALL? have no parameters for the driver and instrument to process.

The program flow is almost the same as described in the General Form section earlier. The difference is in the right side of Figure 4-3 where there is no decision block to determine the number of remaining readings. Since DATA:FIFO:ALL? will complete even if there are no readings in the FIFO, you don't have to check for them.

Note The example program *fast.cs* on your C-SCPI driver tape shows how to retrieve data using the above commands.

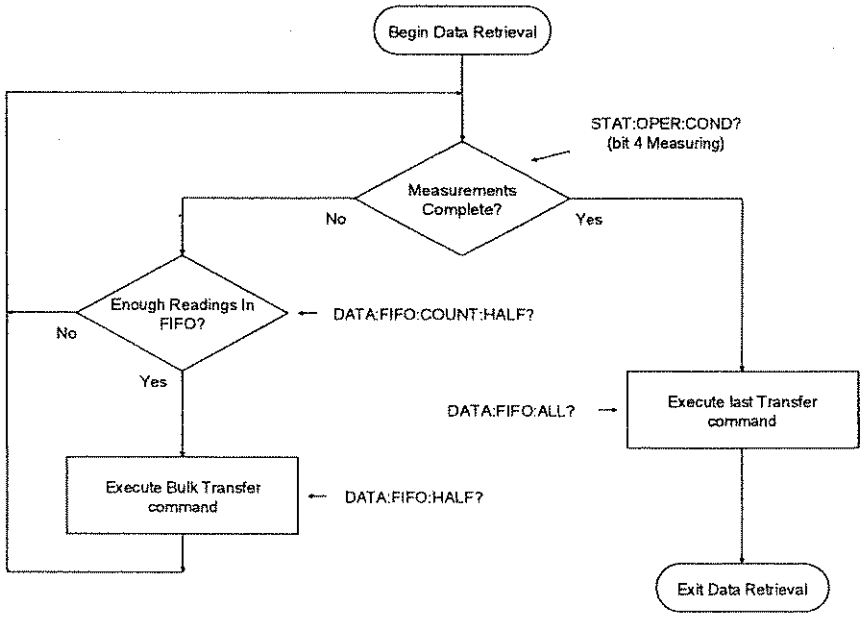


Figure 4-3 Fastest Reading Transfer

Example Command Sequence

```
    fast
    read_data is a 32-bit floating point array
    half_flag  is a 16-bit integer
*RST
SENSE:FUNCTION:VOLTAGE AUTO,(@100:163)        this is a *RST
                                              default
ROUTE:SEQUENCE:DEFINE LIST1,(@100:163)        this is a *RST default
ROUTE:SCAN LIST1                              this is a *RST default
SAMPLE:TIMER LIST1,.00001
ARM:SOURCE IMM
TRIGGER:COUNT 16384
TRIGGER:TIMER .001
TRIGGER:SOURCE TIMER
FORMAT REAL,32
INITIATE
    following loop reads half of the FIFO as long as the module is scanning
loop while "measuring" bit is true            see STAT:OPER:COND bit 4
    DATA:FIFO:COUNT:HALF? <half_flag>
    if half_flag
        DATA:FIFO:HALF? <read_data>
    end if
end loop
    following reads remaining data, if any, after scanning stops
DATA:FIFO:ALL? <read_data>
```

Controlling Data Conversion and Destination

For ROUT:SEQ:DEF, the SCPI Relative Channel specification form can be used to control the conversion and storage destination of data measured during a scan. The SCPI Relative Channel specification syntax is:

(@cc(nn, nn, nn:nn))

where cc = card number, and nn = channel number.

For the HP E1413, card number becomes the Channel Data Modifier. The value can range from 1 through 7. The value controls whether Engineering Unit conversion is performed and the internal destination of the resulting value. The following table explains the effect of the Channel Data Modifier:

Table 4-1 Channel Data Modifiers

Channel Data Modifier	Description
1	Perform EU conversion and store result in both FIFO buffer and Current Value Table
2*	Leave measurement as voltage and store result in both FIFO and CVT
3	Perform EU conversion and store result in CVT only
4*	Leave measurement as voltage and store in CVT only
5	Perform EU conversion and store result in FIFO only
6*	Leave measurement as voltage and store in FIFO only
7*	Leave measurement as voltage and don't store result in either FIFO or CVT. Use as dummy channel set

* Limit Checking is not performed for channels that are not converted to engineering units.

Both the standard and relative channel specification modes can be mixed within a Scan List definition. For example:

```
ROUTE:SEQ:DEF LIST1, (@100:115, 6(00:15))
```

This command specifies that the readings taken on channels 0 through 15 are to be converted into engineering units and stored in both the FIFO data buffer and the Current Value Table (CVT). In addition, channels 0 through 15 are to be read and the raw voltage values are to be added to the FIFO buffer. The FIFO will contain 16 converted readings and 16 voltage readings. The CVT will contain a converted reading for channels 0 through 15.

To scan channels 0 through 63, send EU converted readings to the FIFO, and send voltage readings to the CVT

```
ROUT:SEQ:DEF  LIST1,(@5(00:63),4(00:63))
```

Understanding Scanning Modes

The HP E1413 is a 64-channel Scanning Analog to Digital converter. The sequence of channels to scan are specified in a Scan List. Scanning is the only way the module makes measurements. A Scan list can contain as few as two channels and as many as 1,024 channels.

Once the module has been initiated and triggered it "executes" the Current Scan List and stores the resultant readings in the 64 channel Current Value

Table (CVT) and in the 65,023 reading FIFO Buffer (FIFO). Different sequences of channels can be specified in each of four Scan Lists. Any of the four scan lists that have been defined (channels specified) can be selected as the Current Scan List.

After being Initiated, the module executes the Current Scan List once for each trigger event. There are several choices for the trigger source:

- Several software trigger methods
- The VXIbus TTLTRG lines
- The external Trigger input
- An on-board Programmable Trigger Timer
- Continuously triggered (TRIG:SOUR IMMediate)

The module provides three main modes of scanning:

- The "Default Mode" is set when INITiate:CONTinuous is set to OFF, and the TRIGGER:COUNT is set to 1. Power-on and the *RST command also set this mode. The sequence of events for this mode is shown in Figure 4-4 A. Note that starting at the Trigger Idle State, you must execute an INITiate command to begin each scan. The sequence also shows that the Current Scan List (as set by ROUT:SCAN <*scan_list*>) becomes effective after the INIT command.

- The "Counted Mode" is set when INITiate:CONTinuous is set to OFF, and the TRIGger:COUNt is set greater than 1. In this mode a single INITiate command will allow as many scans as TRIG:COUNT specifies. Note that TRIG:COUNT can be from 2 to 32768 or INFinity. The sequence of events for this mode is shown in Figure 4-4 B. The sequence shows that the Current Scan List is set each time the module completes a scan. This allows the Current Scan List to be changed "on-the-fly" by executing the ROUTe:SCAN <*scan_list*> command while the module is scanning.

- The "Continuous Mode" is set when TRIGger:SOURce is set to IMMediate, and INITiate:CONTinuous is set to ON. Note that executing INIT:CONT ON not only sets the continuous mode but also initiates the module. To stop at the end of the current scan, you execute the INIT:CONT OFF command. The sequence of events for this mode is shown in Figure 4-4 C. The sequence shows that the Current Scan List is not read while the module is scanning. This means that the Current Scan List can not be changed "on-the-fly". However, by specifying LISTL as the Current Scan List before INIT:CONT ON, the sequence of scan lists specified in LISTL will be executed. This mode is the fastest since the module doesn't have to read the current scan list or check for the occurrence of a trigger event.

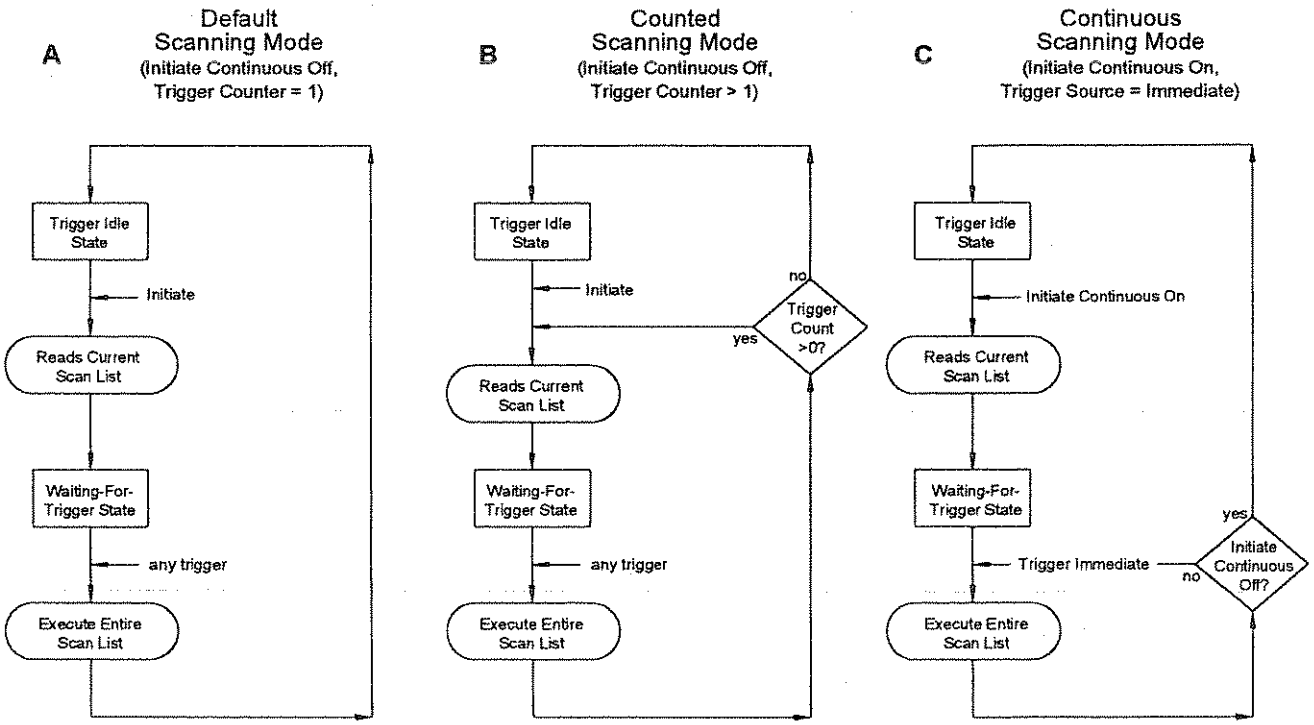


Figure 4-4 Scanning Modes

Note Example programs on your HP Command Module downloadable driver and C-SCPI driver media illustrate the three scanning modes.

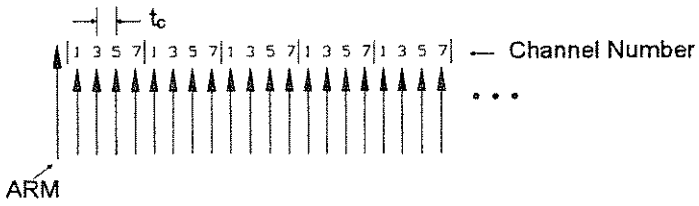
- Use program verif.cs for the Default Mode.
- Use program counted.cs for the Counted Mode.
- Use program through.cs for the Continuous Mode

Triggering and Scanning Modes

The following section shows the various ways the HP E1413 can control channel scanning, and how to select the trigger mode to accomplish it.

Continuous (Free Run) Mode

The Continuous Mode provides the fastest scanning. The time between channel measurements is controlled by the Trigger Timer interval and is consistent from channel-to-channel as well as from the last channel in the scan list back to the first channel in the list when the scan list is re-executed.



t_c is the sample time between channels.

The large arrow is the arming event or signal. The smaller arrows are sample triggers generated by the E1413 when stepping through the channel list.

Example Command Sequence

```
ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)        n can be 1 - 4
ROUT:SCAN LIST<n>
SAMP:TIM LIST<n>,<tc>
TRIG:SOUR IMM
ARM:SOUR < BUS | EXT | HOLD | IMM | TTLTRG<i>   i can be 0 - 7
INIT:CONT ON
```

Starting

ARM event, either hardware or software
Acquisition starts

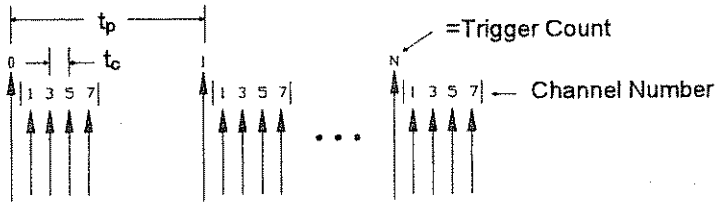
Stopping

```
INIT:CONT OFF                    Stops at end of Scan List
```

Timer Paced Scans

In this mode the Trigger Timer triggers each execution of a scan list. The trigger timer (t_p) can be set from .1ms up to 6.5 seconds. Of course t_p must allow time for all channels to be scanned before the next Timer trigger; $t_p > (channel\ count + 3) * t_c + 30\mu s$. The Sample timer controls t_c in the range from 10µs to 32ms.

The TRIGger:COUNt command controls the number of times any trigger will be accepted, from 1 to 32767 or INFinite. When the Trigger Count is exhausted, the module stops scanning until another INITiate - ARM - TRIG sequence is executed. Trig count defaults to 1.



t_c is the sample time between channels.

t_p is the pacing time between channel lists.

The large arrow is the pacing Trigger Timer. The smaller arrows are sample triggers generated by the E1413 when stepping through the channel list. The first large arrow is the arming event.

Example Command Sequence

ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)	*n can be 1 - 4*
ROUT:SCAN LIST<n>	
SAMP:TIM LIST<n>,<t_c>	*t_c can be 10µs to 32.768ms*
TRIG:SOUR TIMER	
TRIG:TIMER:PERIOD t_p	*t_p can be .1ms - 6.5536 s*
TRIG:COUNT<N>	*N can be 1 - 32768 or INF*
ARM:SOUR < BUS | EXT | HOLD | IMM | TTLTRG<i>	*i can be 0 - 7*
INIT	

Starting

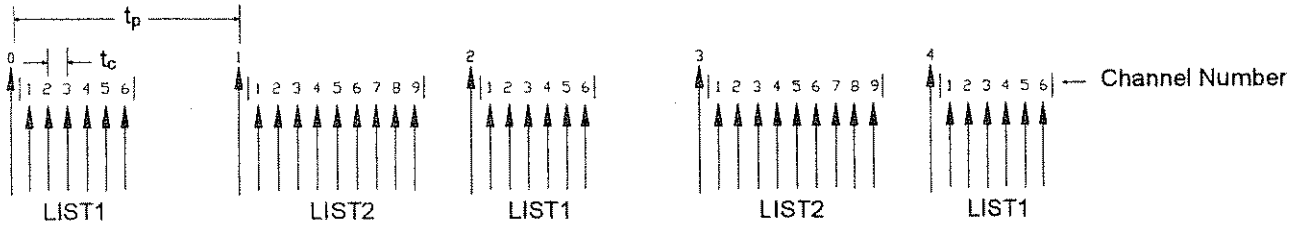
ARM event, either hardware or software
Acquisition starts

Stopping Before Trigger Count is Reached

TRIG:SOUR HOLD	*Stops at end of Scan List, still initiated*
TRIG:IMM	*One more scan, now idle*
or	
ABORT	*Stops immediately*

Sequenced Scan Lists

This mode uses the Automatic Scan List Sequencing feature (also known as List-of-Lists). Here, 2 to 4 Scan lists are defined with channels (up to 1024 channel numbers each), then these Scan Lists are executed in the sequence specified in the List-of-Lists (LISTL). When LISTL is controlling the sequencing of the Scan Lists (1 through 4), each Scan List specified in LISTL must contain at least 6 channels. In the example we scan channels 1 through 6 on every scan but only scan channels 7 through 9 every other scan.



t_c is the sample time between channels.

t_p is the pacing time between channel lists.

The large arrow is the pacing Trigger Timer. The smaller arrows are sample triggers generated by the E1413 when stepping through the channel list. The first large arrow is the arming event.

Example Command Sequence

ROUT:SEQ:DEF LIST1,(@101:106)	*first 6 channels*
ROUT:SEQ:DEF LIST2,(@101:109)	*add 7, 8 and 9*
ROUT:SEQ:DEF LISTL,(@1,2)	*List-of-Lists is List 1 and List 2*
ROUT:SCAN LISTL	*Use LISTL*
SAMP:TIM LISTL,<t_c>	*Sets samp time for lists in LISTL*
TRIG:SOUR TIMER	
TRIG:TIMER:PERIOD t_p	*t_p can be .1ms - 6.5536 s*
TRIG:COUNT<N>	*N can be 1 - 32768 or INF*
ARM:SOUR < BUS | EXT | HOLD | IMM | TTLTRG<i>	*i can be 0 - 7*
INIT	

Starting

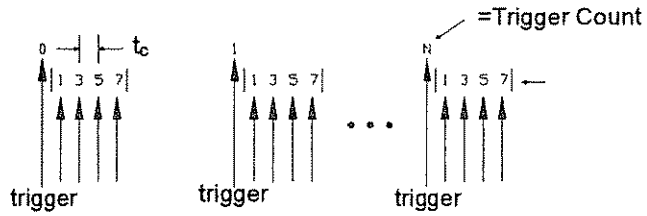
ARM event, either hardware or software
Acquisition starts

Stopping Before Trigger Count is Reached

TRIG:SOUR HOLD	*Stops at end of Scan, still initiated*
TRIG:IMM	*One more scan, now idle*
or	
ABORT	*Stops immediately, now idle*

Externally Paced Scans

This mode is much like the Timer Paced Scans except the trigger source is external to the HP E1413. The trigger could come from a software command such as TRIG, or a hardware trigger from either the External Trigger input or one of the VXIbus TTLTRG lines.



t_c is the sample time between channels.

The large arrows are the Trigger events. The smaller arrows are sample triggers generated by the E1413 when stepping through the channel list.

Example Command Sequence

ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)	*n can be 1 - 4*
ROUT:SCAN LIST<n>	
SAMP:TIM LIST<n>,<t_c>	*t_c can be 10µs to 32.768ms*
TRIG:SOUR < BUS | EXT | HOLD | TTLTRG<i>	*i can be 0 - 7*
TRIG:COUNT<N>	*N can be 1 - 32768 or INF*
INIT	

Starting

Trigger event, either hardware or software
Acquisition starts

Stopping Before Trigger Count is Reached

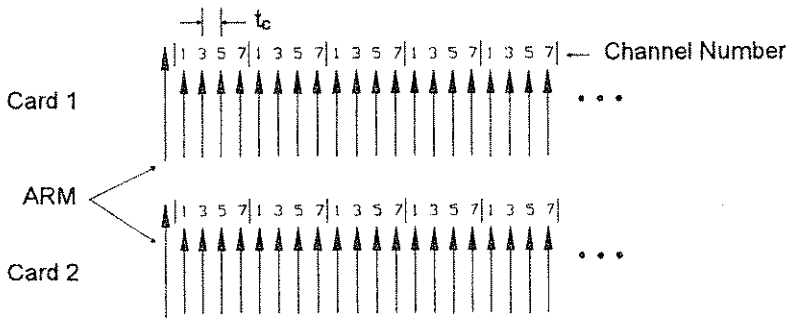
Remove Trigger Source	*Stops at end of Scan List, still initiated*
or	
TRIG:SOUR HOLD	*Stops at end of Scan List, still initiated*
TRIG:IMM	*One more scan, now idle*
or	
ABORT	*Stops immediately, now idle*

Synchronizing Multiple Cards

In this section we show you how to synchronize two or more HP E1413s. The examples show two cards but the principles may be extended to several.

Continuous (Free-Run) Mode

This example shows you how to start two or more cards scanning at the same time. It shows the cards running in the Continuous mode (TRIG:SOUR IMM and INIT:CONT ON).



t_c is the sample time between channels.

The large arrow is the arming event or signal. The smaller arrows are sample triggers generated by the E1413 when stepping through the channel list. Sample clocks on each card are not synchronized to each other

Example Command Sequence

Card 1

```
ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)          n can be 1 - 4
ROUT:SCAN LIST<n>
SAMP:TIM LIST<n>,<tc>
TRIG:SOUR IMM
ARM:SOUR < BUS | EXT | HOLD | IMM | TTLTRG<i>   i can be 0 - 7
INIT:CONT ON
```

Card 2

```
ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)          n can be 1 - 4
ROUT:SCAN LIST<n>
SAMP:TIM LIST<n>,<tc>
TRIG:SOUR IMM
ARM:SOUR < BUS | EXT | HOLD | IMM | TTLTRG<i>   i can be 0 - 7
INIT:CONT ON
```

Starting

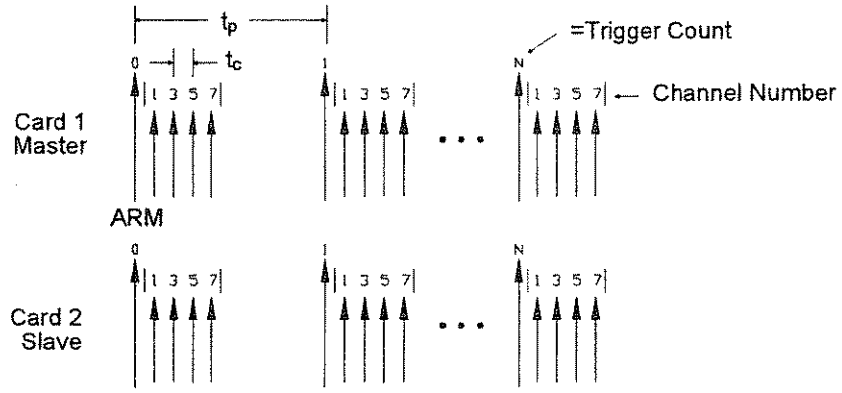
ARM event, either hardware or software
Acquisition starts

Stopping

```
INIT:CONT OFF        To both cards, stops at end of Scan List
```

Internal Timer Based Scans

In this case both cards are set up to be paced by their trigger timers. Card 1 is also set up to source a trigger on one of the VXIbus TTLTRG lines and Card 2 is set up to be triggered by that line. This means that triggering card 1 will start both cards.



t_c is the sample time between channels.

t_p is the pacing time between channel lists.

The large arrow is the pacing Trigger Timer. The smaller arrows are sample triggers generated by the E1413 when stepping through the channel list. The first large arrow is the arming event.

Example Command Sequence

Card 1

```
ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)          n can be 1 - 4
ROUT:SCAN LIST<n>
SAMP:TIM LIST<n>,<tc>                    tc can be 10µs to 32.768ms
TRIG:SOUR TIMER
TRIG:TIMER:PERIOD tp                          tp cn be .1ms - 6.5536 s
TRIG:COUNT<N>                                 N can be 1 - 32768 or INF
ARM:SOUR < BUS | EXT | HOLD | IMM | TTLTRG<i>      i can be 0 - 7
OUTP:TTLT:SOUR TRIG                  Will drive TTLT line if triggered
OUTP:TTLT<q>  ON                     Enable to drive selected ttltrg line
INIT
```

Card 2

```
ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)          n can be 1 - 4
ROUT:SCAN LIST<n>
SAMP:TIM LIST<n>,<tc>                    tc can be 10µs to 32.768ms
TRIG:SOUR TTLTRG<q>                  Selects trig sourced from card 1
TRIG:COUNT<N>                                 N can be 1 - 32768 or INF
INIT
```

Starting

ARM event, either hardware or software
Acquisition starts

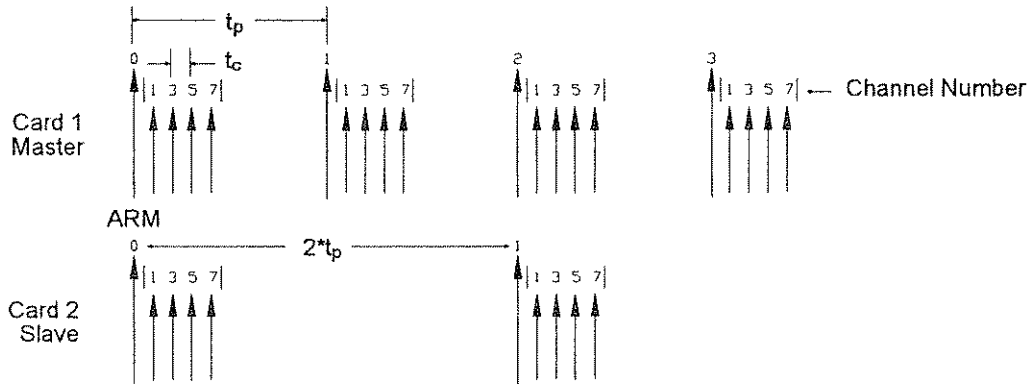
Stopping Before Trigger Count is Reached

TRIG:SOUR HOLD	*Stops at end of Scan List, still initiated*
TRIG:IMM	*One more scan, now idle*
or	
ABORT	*Stops immediately*

Timer Based Scans at Different Rates

This example shows how to use the E1413's internal FTRigger (First TRigger) event to source triggers onto a VXIbus TTLTRG line. FTRigger sets the module to source a trigger when the module receives its first trigger of N where N is set by TRIGGER:COUNT. The ratio of trigger outputs to trigger inputs is $\frac{1}{TRIG:COUNT}$.

Also shown is the TRIGger:TIMer:MODE command. When TRIG:TIM:MODE is set to SYNC, the Trigger Timer keeps running even after the trigger count is reached. When TRIG:TIM:MODE is set to ASYN (the default) the Trigger Timer stops and is reset each time the trigger count is reached. By using SYNC you can insure that trigger pacing stays consistent when the Trigger Count is reset and scanning restarts.



t_c is the sample time between channels.

t_p is the pacing time between channel lists.

The large arrow is the pacing Trigger Timer. The smaller arrows are sample triggers generated by the E1413 when stepping through the channel list. The first large arrow is the arming event.

Example Command Sequence

Card 1

ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)	*n can be 1 - 4*
ROUT:SCAN LIST<n>	
SAMP:TIM LIST<n>,<t_c>	*t_c can be 10μs to 32.768ms*
TRIG:SOUR TIMER	
TRIG:TIMER:PERIOD t_p	*t_p cn be .1ms - 6.5536 s*
TRIG:TIMER:MODE SYNC	*Keep Timer running during INIT:CONT ON*
TRIG:COUNT 2	
ARM:SOUR < BUS | EXT | HOLD | IMM | TTLTRG<i>	*i can be 0 - 7*
OUTP:TTLT:SOUR FTR	*1st trig will source TTLTRG*
OUTP:TTLT<q> ON	*Enable to drive selected ttltrg line*
INIT:CONT ON	

Card 2

ROUT:SEQ:DEF LIST<n>,(@101,103,105,107)	*n can be 1 - 4*
ROUT:SCAN LIST<n>	
SAMP:TIM LIST<n>,<t_c>	*t_c can be 10μs to 32.768ms*
TRIG:SOUR TTLTRG<q>	*Selects trig sourced from card 1*
TRIG:COUNT<N>	*N can be 1 - 32768 or INF*
INIT	

Starting

ARM event, either hardware or software
Acquisition starts

Stopping Before Trigger Count is Reached

INIT:CONT OFF	*To Master Card*
TRIG:SOUR HOLD	*Stops at end of Scan List, still initiated*
TRIG:IMM	*One more scan, now idle*
or	
ABORT	*Stops immediately*

Using the Status System

The HP E1413's Status System allows you to quickly poll a single register (the Status Byte) to see if any internal condition needs attention. Figure 4-5 shows the Status System in detail. Figure 4-5 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.

Note The C-SCPI program *status.scpi* on your C-SCPI driver tape, shows how to use the status system.

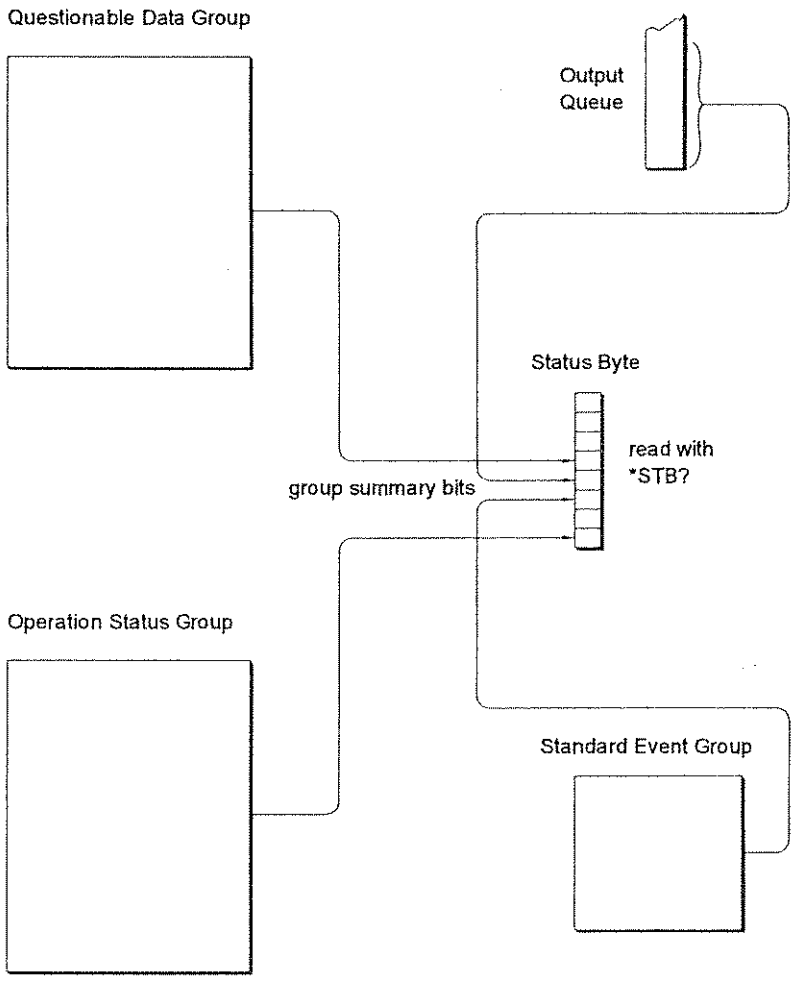


Figure 4-5 Simplified Status System Diagram

Enabling Events to be Reported in the Status Byte

In Figure 4-5 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;

```
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).

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 4-5 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.

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 E1413 Background Operation

The HP E1413 inherently runs its measurements and calibration 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.

The driver is required to setup the type of measurement to be run, and to unload data from the card after it has been acquired. Once the INIT[:IMM] or INIT:CONT ON commands are given, the HP E1413 is initiated and all functions of the trigger system and acquisition are controlled by its on-board control processor. The driver returns to waiting for user commands. No interrupts are required for the HP E1413 to complete its measurement.

While the module is measuring, the driver can be queried for its status, and data can be read from the FIFO and CVT. One command can be given which changes the measurement setup, i.e., the scan list can be changed using ROUT:SEQ:DEF. The INIT:CONT OFF command may also be given to force a continuous measurement to complete. Any other changes to the measurement setup will not be allowed until the measurement completes, or an ABORT command is given. Of course any commands or queries can be given to other instruments while the HP E1413 is actively measuring.

Updating the Status System and VXI Interrupts

The driver needs to update the status systems information whenever the status of the HP E1413 changes. This update is always done when the status system is accessed, or when CALibrate, INIT or ABORt commands are executed. Most of the bits in the OPER and QUES registers represent conditions which can change while the HP E1413 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 E1413, the HP E1413 interrupts can be used.

The HP E1413 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 E1413.
- Scan complete. The HP E1413 has finished a scan list.

- SCP trigger. A trigger was received from an SCP.
- FIFO half full. The FIFO contains at least 32768 readings.
- Limit Test was exceeded.
- Measurement complete. The trigger system exited the "Wait-For-Arm" state and is checking for INIT CONTinuous. This clears the Measuring bit in the OPER register.

These HP E1413 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. In the C-SCPI environment, there are times when interrupts from the HP E1413 could cause problems with system function calls, as described in the C-SCPI User's Guide. 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 STATus:OPER/QUES:ENABle command to be a part of the group summary bit, it will also enable the HP E1413 interrupt for that condition. If that bit is not enabled, the corresponding interrupt will be disabled.

Sending the STAT:PRESET will disable all the interrupts from the HP E1413.

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.

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().

See the Chapter 5 of the C-SCPI users guide for more details on the overlapped mode and using interrupts in the C-SCPI environment.

Compensating for System Offsets

System Wiring Offsets

The HP E1413 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.

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 like 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 a selecting a manual range below the range floor will return an Overload value (see table on page 5-48).

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.

NOTE The HP E1413'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 the CAL:TARE:RESet command. 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 5-48 for more on values returned for Overload readings.

Unexpected Channel Offsets or Overloads

This can occur when your E1413's Flash Memory contains CAL:TARE offset constants that are no longer appropriate for its current application. Execute CAL:TARE:RESET 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.

Detecting Open Transducers

Open transducer detection is implemented in Signal Conditioning Plug-ons. See "Detecting Open Transducers" in Section 6 "Signal Conditioning Plug-on Manuals".

HP E1413 Thermocouple Reference Compensation

The HP E1413 performs reference junction compensation automatically on all channels defined as thermocouple measurements by the FUNC:TEMP TC command. There are two ways to do reference junction compensation in the HP E1413:

1. You can define the Reference Junction temperature to be a fixed value using the SENSE:REF:TEMP command. Use this method if the copper-to-thermocouple wire junction is at a connection panel with active temperature control. These panels are sometimes called "Uniform Temperature Reference" (UTR) panels or sometimes "Isothermal Reference Panels".

   For example, if the UTR is specified to operate at 85 Degrees centigrade, the panel will contain heating elements which keep the panel at exactly 85 degrees. If you use an 85 degree panel, use the command "SENSE:REF:TEMP 85.0". Then all future thermocouple measurements will be corrected for this value.

2. You can measure the Reference Junction temperature during a scan, using 5K ohm Thermistors, RTDs (resistance thermal devices) or with another "Custom" absolute temperature measuring device. (The "Custom" device is pre-defined in the HP E1413 as a Type K thermocouple which has a thermally controlled ice point reference junction.) Reference temperature is measured during a scan by defining a channel to be a temperature reference channel using the SENSE:REF command, then including that channel in the scan list sequence before the thermocouple channels.

   For example, to use the 5K thermistor built into the HP E1413-66510 terminal block, do the following:

   a. Connect the built-in thermistor to a channel's input terminals on the terminal block. In this example, we will use channel 0. Connect wires from the terminal labeled "HTS" to the terminal labeled "H00", and connect a wire from the terminal labeled "LTS" to the terminal labeled "L00".

b. Be sure the HP E1413's current source is connected to excite the on-board 5K thermistor. This requires the two jumpers at JM1 to be in the "on-board" position (not the REMote position), see Figure 2-6.

c. Define channel 0 to be a 5K thermistor reference temperature channel. This requires the command SENSe:REF THER,5000,(@100).

d. Put Channel 0 into the scan list before any thermocouple channels. Do this using the ROUT:SEQ:DEF command. Each time channel 0 is measured, a new value for the reference temperature will be stored in the HP E1413, and used for the thermocouple channels that follow in the scan list. The measured reference temperature by default is sent to the CVT and FIFO. If you do not want the reference temperature reported, you can set the Channel Data Modifier for that channel to 7 (see Controlling Data Conversion and Destination on page 4-7).

The SCPI Command sequence should be:

```
FUNC:TEMP TC,E,(@108:115)
FUNC:TEMP TC,J,(@116:123)
SENS:REF THER,5000,(@100)
ROUT:SEQ:DEF LIST1,(@100,108:115,116:123)
INIT
TRIG
DATA:FIFO?
```

Reducing Settling Waits

While the HP E1413 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 (another 10 or 20 µs). The best solution is to insert settling time only for channels where it is required, rather than to increase sample time for all channels because of one or two problem channels. With the HP E1413 channels will be scanned in any order you specify including multiple times, and you can specify which readings to return and which to discard (ROUT:SEQ:DEF command). Using this capability you can set your scan sequence to measure input channels from lowest level to highest level. This will minimize channel settling problems. If you plan to continuously execute a scan list, wrapping around from the last channel to the first channel can present this high level signal to low level signal settling problem. To allow for additional channel settling time here, you can measure the low level channel two or more times but only return data from the final measurement of that channel. Figure 4-6 will illustrate this idea.

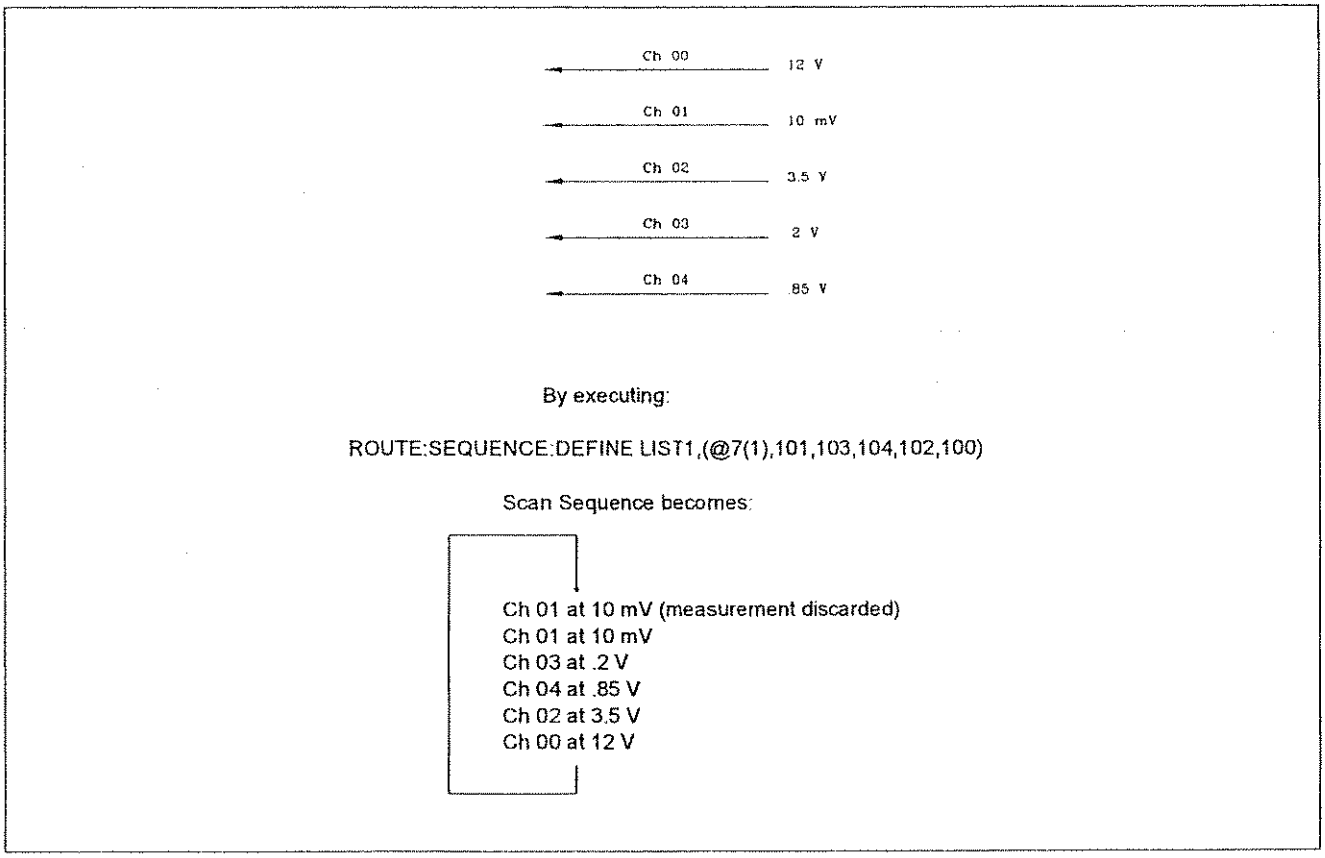


Figure 4-6 Recommended Scanning Sequence

Chapter 5
HP E1413 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 E1413.

- Overall Command Index 5-1
- Command Fundamentals 5-6
- SCPI Command Reference 5-12
- Common Command Reference 5-111
- Command Quick Reference 5-117

Overall Command Index

SCPI Commands

ABORt 5-13

ARM:IMMediate 5-15
ARM:SOURce BUS | EXT | HOLD | IMM | TTLTrg<n> 5-15
ARM:SOURce? 5-16

CALCulate:AVERage[:STATe] ON | OFF 5-18
CALCulate:AVERage[:STATe]? 5-18
CALCulate:AVERage:COUNt *<n>* 5-18
CALCulate:AVERage:COUNt? 5-19
CALCulate:CLIMits:FAIL[:CUMulative]? 5-19
CALCulate:CLIMits:FAIL:CURRent? 5-20
CALCulate:CLIMits:FLIMits[:CHANnels][:CUMulative]? 5-20
CALCulate:CLIMits:FLIMits[:CHANnels]:CURRent? 5-21
CALCulate:CLIMits:FLIMits:POINts[:CUMulative]? 5-21
CALCulate:CLIMits:FLIMits:POINts:CURRent? 5-21
CALCulate:LIMit[:STATe] ON | OFF,(@*<ch_list>*) 5-22
CALCulate:LIMit[:STATe]? (@*<channel>*) 5-22
CALCulate:LIMit:FAIL[:CUMulative]? (@*<channel>*) 5-23
CALCulate:LIMit:FAIL:CURRent? (@*<channel>*) 5-23
CALCulate:LIMit:LOWer[:STATe] ON | OFF,(@*<ch_list>*) 5-24
CALCulate:LIMit:LOWer[:STATe]? (@*<channel>*) 5-24
CALCulate:LIMit:LOWer:DATA *<lower_lim>*,(@*<ch_list>*) 5-24
CALCulate:LIMit:LOWer:DATA? (@*<channel>*) 5-25
CALCulate:LIMit:UPPer[:STATe] ON | OFF,(@*<ch_list>*) 5-25
CALCulate:LIMit:UPPer[:STATe]? (@*<channel>*) 5-26

CALCulate:LIMit:UPPer:DATA *<upper_lim>*,(@*<ch_list>*) . . . 5-26
CALCulate:LIMit:UPPer:DATA? (@*<channel>*) . . . 5-27

CALibration:CONFigure:RESistance . . . 5-30
CALibration:CONFigure:VOLTage *<range>* . . . 5-30
CALibration:SETup . . . 5-31
CALibration:SETup? . . . 5-31
CALibration:STORe ADC | TARE . . . 5-32
CALibration:TARE (@*<ch_list>*) . . . 5-33
CALibration:TARE:RESet . . . 5-35
CALibration:TARE? . . . 5-35
CALibration:VALue:RESistance *<ref_ohms>* . . . 5-35
CALibration:VALue:VOLTage *<ref_volts>* . . . 5-36
CALibration:ZERO? . . . 5-37

DIAGnostic:CHECksum? . . . 5-38
DIAGnostic:COMMand:SCPWRITE *<reg_addr>*,*<reg_data>* . . . 5-38
DIAGnostic:CUStom:LINear *<table_ad_range>*,*<table_block>*,(@*<ch_list>*) . . . 5-39
DIAGnostic:CUStom:PIECewise *<table_ad_range>*,*<table_block>*,(@*<ch_list>*) . . . 5-40
DIAGnostic:CUSTom:REFerence:TEMPerature . . . 5-41
DIAGnostic:INTerrupt[:LINe] *<int_line>* . . . 5-41
DIAGnostic:INTerrupt[:LINe]? . . . 5-41
DIAGnostic:OTDetect[:STATe] ON | OFF,(@*<ch_list>*) . . . 5-42
DIAGnostic:OTDetect[:STATe]? (@*<channel>*) . . . 5-43
DIAGnostic:QUERy:SCPREAD? *<reg_addr>* . . . 5-43
DIAGnostic:VERSion? . . . 5-43

FETCh? . . . 5-45

FORMat[:DATA] *<format>*[,*<size>*] . . . 5-47
FORMat[:DATA]? . . . 5-48

INITiate:CONTinuous ON | OFF . . . 5-49
INITiate[:IMMediate] . . . 5-50

INPut:FILTer[:LPASs]:FREQuency *<cutoff_freq>*,(@*<ch_list>*) . . . 5-51
INPut:FILTer[:LPASs]:FREQuency? (@*<channel>*) . . . 5-52
INPut:FILTer[:LPASs][:STATe] ON | OFF,(@*<ch_list>*) . . . 5-52
INPut:FILTer[:LPASs][:STATe]? (@*<channel>*) . . . 5-53
INPut:GAIN *<chan_gain>*,(@*<ch_list>*) . . . 5-53
INPut:GAIN? (@*<channel>*) . . . 5-54

MEMory:VME:ADDRess *<A24_address>* . . . 5-56
MEMory:VME:ADDRess? . . . 5-56
MEMory:VME:SIZE *<mem_size>* . . . 5-56
MEMory:VME:SIZE? . . . 5-57
MEMory:VME:STATe ON | OFF . . . 5-57
MEMory:VME:STATe? . . . 5-58

OUTPut:CURRent:AMPLitude <*amplitude*>,(@<*ch_list*>) 5-59
OUTPut:CURRent:AMPLitude? (@<*channel*>) 5-60
OUTPut:CURRent[:STATe] 1 | 0 | ON | OFF,(@<*ch_list*>) 5-60
OUTPut:CURRent[:STATe]? (@<*channel*>) 5-61
OUTPut:SHUNt 1 | 0,(@<*ch_list*>) 5-61
OUTPut:SHUNt? (@<*channel*>) 5-62
OUTPut:TTLTrg:SOURce TRIGger | FTRigger | SCPlugon | LIMit 5-62
OUTPut:TTLTrg:SOURce? 5-63
OUTPut:TTLTrg<n>[:STATe] ON | OFF 5-64
OUTPut:TTLTrg<n>[:STATe]? 5-64

ROUTe:SCAN <*scan_list*> 5-65
ROUTe:SEQuence:DEFine <*scan_list*>,(@<*ch_list*>) 5-66
ROUTe:SEQuence:DEFine? <*scan_list*>,CHANnel | MODifier 5-68
ROUTe:SEQuence:POINts? <*scan_list*> 5-69

SAMPle:TIMer <*scan_list*>,<*interval*> 5-71
SAMPle:TIMer? <*scan_list*> 5-72

[SENSe:]DATA:CVTable? (@<*ch_list*>) 5-74
[SENSe:]DATA:CVTable:RESet 5-75
[SENSe:]DATA:FIFO[:ALL]? 5-75
[SENSe:]DATA:FIFO:COUNT? 5-76
[SENSe:]DATA:FIFO:COUNT:HALF? 5-76
[SENSe:]DATA:FIFO:HALF? 5-77
[SENSe:]DATA:FIFO:MODE BLOCk | OVERwrite 5-78
[SENSe:]DATA:FIFO:MODE? 5-78
[SENSe:]DATA:FIFO:PART? <*n_readings*> 5-79
[SENSe:]DATA:FIFO:RESet 5-80
[SENSe:]FILTer[:LPASs][:STATe] ON | OFF 5-80
[SENSe:]FILTer[:LPASs][:STATe]? 5-80
[SENSe:]FUNCtion:CUSTom [<*range*>,](@<*ch_list*>) 5-81
[SENSe:]FUNCtion:CUSTom:REFerence [<*range*>,](@<*ch_list*>) 5-81
[SENSe:]FUNCtion:CUSTom:TCouple <*type*>,[<*range*>,](@<*ch_list*>) 5-83
[SENSe:]FUNCtion:RESistance <*excite_current*>,[<*range*>,] @<*ch_list*>) 5-84
[SENSe:]FUNCtion:STRain:FBENding [<*range*>,](@<*ch_list*>) 5-85
[SENSe:]FUNCtion:STRain:FBPoisson [<*range*>,](@<*ch_list*>) 5-85
[SENSe:]FUNCtion:STRain:FPOisson [<*range*>,](@<*ch_list*>) 5-85
[SENSe:]FUNCtion:STRain:HBENding [<*range*>,](@<*ch_list*>) 5-85
[SENSe:]FUNCtion:STRain:HPOisson [<*range*>,](@<*ch_list*>) 5-85
[SENSe:]FUNCtion:STRain[:QUARter] [<*range*>,](@<*ch_list*>) 5-85
[SENSe:]FUNCtion:TEMPerature <*sensor_type*>,<*sub_type*>,[<*range*>,] (@<*ch_list*>) 5-87
[SENSe:]FUNCtion:VOLTage[:DC] [<*range*>,](@<*ch_list*>) 5-88
[SENSe:]REFerence <*sensor_type*>,[<*sub_type*>,][<range>,] (@<*ch_list*>) 5-89
[SENSe:]REFerence:TEMPerature <*degrees_c*> 5-91
[SENSe:]STRain:EXCitation <*excite_v*>,(@<*ch_list*>) 5-91

[SENSe:]STRain:EXCitation? (@<*channel*>) . . . 5-92
[SENSe:]STRain:GFACtor <*gage_factor*>,(@<*ch_list*>) . . . 5-93
[SENSe:]STRain:GFACtor? (@<*channel*>) . . . 5-93
[SENSe:]STRain:POISson <*poisson_ratio*>,(@<*ch_list*>) . . . 5-94
[SENSe:]STRain:POISson? (@<*channel*>) . . . 5-94
[SENSe:]STRain:UNSTrained <*unstrained_v*>,(@<*ch_list*>) . . . 5-94
[SENSe:]STRain:UNSTrained? (@<*channel*>) . . . 5-95

STATus:OPERation:CONDition? . . . 5-98
STATus:OPERation:ENABle <*enable_mask*> . . . 5-98
STATus:OPERation:ENABle? . . . 5-99
STATus:OPERation[:EVENt]? . . . 5-99
STATus:PRESet . . . 5-100
STATus:QUEStionable:CONDition? . . . 5-101
STATus:QUEStionable:ENABle <*enable_mask*> . . . 5-101
STATus:QUEStionable:ENABle? . . . 5-102
STATus:QUEStionable[:EVENt]? . . . 5-102

SYSTem:CTYPe? (@<channel>) . . . 5-103
SYSTem:ERRor? . . . 5-103
SYSTem:VERSion? . . . 5-104

TRIGger:COUNt <*trig_count*> . . . 5-106
TRIGger:COUNt? . . . 5-106
TRIGger[:IMMediate] . . . 5-107
TRIGger:SOURce BUS | EXT | HOLD | IMM | TIMer | TTLTrg<n> . . . 5-107
TRIGger:SOURce? . . . 5-108
TRIGger:TIMer:MODE SYNChronous | ASYNchronous . . . 5-108
TRIGger:TIMer:MODE? . . . 5-109
TRIGger:TIMer[:PERiod] <*trig_interval*> . . . 5-109
TRIGger:TIMer[:PERiod]? . . . 5-110

Common Commands

*CAL? . . . 5-111
*CLS . . . 5-111
*ESE . . . 5-112
*ESE? . . . 5-112
*ESR? . . . 5-112
*IDN? . . . 5-112
*OPC . . . 5-113
*OPC? . . . 5-113
*RST . . . 5-113
*SRE . . . 5-114
*SRE? . . . 5-114
*STB? . . . 5-114
*TST? . . . 5-115

*WAI 5-116

Command Fundamentals

Commands are separated into two types: IEEE-488.2 Common Commands and SCPI Commands. The SCPI command set for the HP E1413 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 closing switches, making measurements, 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:

```
ROUTe
    :SCAN <scan_list>
    :SEQuence
        :DEFine <scan_list>,(@<ch_list>)
        :POINts? <scan_list>
```

ROUTe is the root command, :SCAN (with a parameter) and :SEQuence are second level commands, and :DEFine, and POINts? are third level commands.

Command Separator

A colon (:) always separates one command from the next lower level command as shown below:

```
ROUTE:SEQUENCE:POINTS? LIST1
```

Colons separate the root command from the second level command (ROUTE:SEQUENCE) and the second level from the third level (SEQUENCE:POINTS?). The parameter LIST1 is separated from the command by a space.

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.

For example, if the command syntax shows SEQuence, then SEQ and SEQUENCE are both acceptable forms. Other forms of SEQuence, such as SEQU or SEQUEN 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
    :CONTinuous  ON|OFF
    [: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 table 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}
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 ROUTe:SCAN *<scan_list>* command where *<scan_list>* can be LIST1, LIST2, LIST3, or LIST4.

Channel List (Standard Form)

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 E1413 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)

Channel List (Relative Form)

The SCPI Relative Channel specification syntax is:

(@cc(nn, nn, nn:nn))

where cc = card number, and nn = channel number.

Example:

0, 5, 6 through 32, and 45=(@1(0,5,6:32,45))

The Relative form has special meaning when used in the ROUTe:SEQuence:DEFine<scan_list>,(@<ch_list>) command. See page 5-67 for specific information.

Note that for both forms a channel list is always contained within (@ and). The Command Reference always shows the (@ and) punctuation:

(@*<ch_list>*)

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 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:

```
*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.

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)

SCPI Command Reference

The following section describes the SCPI commands for the HP E1413. 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.

The ABORt subsystem is a part of the HP E1413's trigger system. ABORt resets the trigger system from its Wait For Trigger state to its Trigger Idle state.

Subsystem Syntax

ABORt

Comments

- The instrument stops scanning immediately (scan list not completed).
- 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.
- If INITiate:CONTinuous is ON, then after ABORt sets the instrument to the Trigger Idle State, it immediately returns to the Waiting For Trigger State. If TRIG:SOUR is IMM then the module resumes scanning. INIT:CONT must be OFF to keep the instrument in the Trigger Idle State after an ABORT.
- The recommended method of ending the Continuous Scanning Mode is to execute INIT:CONT OFF and check the "Measuring" bit (bit 4) with STAT:OPER:COND?
- **Related Commands:** INITiate..., TRIGger...
- ***RST Condition:** TRIG:SOUR HOLD, INIT:CONT OFF

Usage

ABORT

If INITed, goes to Trigger Idle state. If scanning, stops and goes to Trigger Idle State (will return to Waiting for Trigger state if INIT:CONT is ON

With the HP E1413, when the TRIG:SOURCE is set to TIMer, an ARM event must occur to start the timer. When INIT:CONTINUOUS is set to ON and the TRIG:SOURCE is set to IMM (called the "Continuous Mode"), an ARM event must occur to start scanning. This can be something as simple as executing the ARM[:IMMediate] command, or it could be another event selected by ARM:SOURCE.

NOTE If TRIG:SOUR TIM is not set, and the Continous Mode (TRIG:SOUR IMM and INIT:CONT ON), is not set, the ARM:SOURce must be set to IMM or 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 or the Continuous Mode is set (INIT:CONT ON and TRIG:SOUR IMM).

Figure 5-3 shows the overall logical model of the Trigger System. The shaded area points out the ARM subsystem and Timer portion.

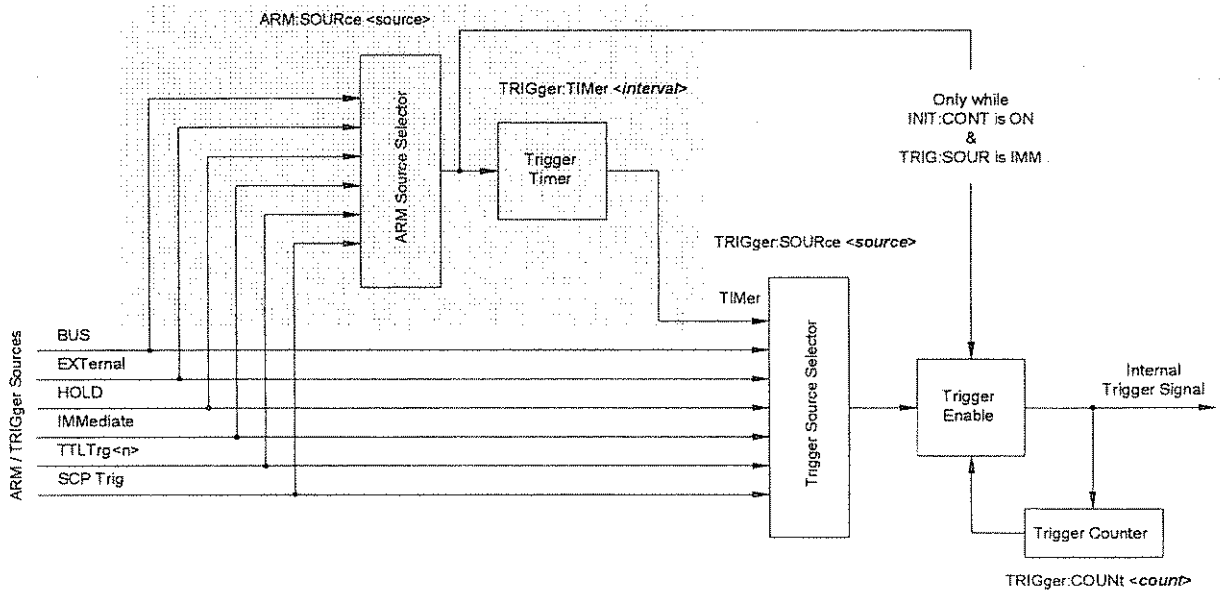


Figure 5-1 Logical Trigger Model

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
- ***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	"Trig" 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 on page 5-14.
- The Arm Source can be changed while the module is scanning (but not in the Continuous Mode). This allows changing the TRIG:TIM interval "on-the-fly". Change ARM:SOUR to BUS or HOLD, change TRIG:TIM, then return to original ARM:SOUR.

- While ARM:SOUR is IMM, you need only INITiate the trigger system to start a measurement scan.
- **Related Commands:** ARM:IMM, ARM:SOURCE?, INIT[:IMM], INIT:CONT ON, 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
```

The CALCulate subsystem allows testing channel measurements against limits, as well as performing data reduction and/or noise reduction by averaging measurements.

Subsystem Syntax

```
CALCulate
    :AVERage
        [:STATe] ON | OFF
        [:STATe]?
        :COUNt <n>
        :COUNt?
    :CLIMits
        :FAIL
            [:CUMulative]?
            :CURRent?
        FLIMits
            [:CHANnels]
                [:CUMulative]?
                :CURRent?
            :POINts
                [CUMulative]?
                :CURRent?
    :LIMit
        [:STATe] ON | OFF,(@<ch_list>)
        [:STATe]? (@<channel>)
        :FAIL
            [:CUMulative]? (@<channel>)
            :CURRent? (@<channel>)
        :LOWer
            [:STATe] ON | OFF,(@<ch_list>)
            [:STATe]? (@<channel>)
            :DATA <lower_lim>,(@<ch_list>)
            :DATA? (@<channel>)
        :UPPer
            [:STATe] ON | OFF,(@<ch_list>)
            [:STATe]? (@<channel>)
            :DATA <upper_lim>,(@<ch_list>)
            :DATA? (@<channel>)
```

CALCulate:AVERage[:STATe]

CALCulate:AVERage[:STATe] *<enable>* controls whether measurement averaging is enabled (ON) or disabled (OFF).

NOTES

1. When CALC:AVER is ON, an individual channel number must appear in a Scan List only once, and use of LISTL in not allowed.
2. When CALC:AVER is ON, channels must use manual ranging.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	discrete (string)	ON | OFF	none

Comments

- When averaging is on, each channel in the scan list(s) is measured a number of times (as set by CALC:AVER:COUNT). The average of those measurements becomes the reading stored for that channel.
- Related Commands: CALC:AVER:COUNT, CALC:AVER?
- ***RST Condition:** CALC:AVER OFF

Usage

```
CALC:AVER ON
```

Averaging for all scanned channels is ON

CALCulate:AVERage[:STATe]?

CALCulate:AVERage[:STATe]? returns a 1 if averaging is on, or a 0 if averaging is off.

Comments

- **Returned Value:** Numeric value either 1 or 0. The C-SCPI type is **int16**.
- **Related Commands:** CALC:AVER, CALC:AVER:COUNT

Usage

```
CALC:AVER?
```

A subsequent enter statement will return 0 or 1.

CALCulate:AVERage:COUNt

CALCulate:AVERage:COUNt *<n_readings>* sets the number of A/D measurements that will be averaged to produce a stored reading. The same count applies to all measured channels in all scan lists when CALC:AVER:STATE is set to ON.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
n_readings	numeric (int16)	2 | 4 | 8 | 16 | 32 | 64 | 128 | 256	none

Comments

- If INIT:CONT is OFF, *n_readings* must be less than or equal to TRIGger:COUNt.
- **Related Commands:** CALC:AVER:COUNT?, CALC:AVER ON | OFF
- ***RST Condition:** CALC:AVER:COUNT 2, CALC:AVER:STATE OFF

Usage

```
CALC:AVER:COUNT 8          8 measurements averaged per reading stored
```

CALCulate:AVERage:COUNt?

CALCulate:AVERage:COUNt? returns the value which sets the number of measurements averaged per stored reading.

Comments

- **Returned Value:** Numeric value, either 2, 4, 8, 16, 32, 64, 128, or 256. The C-SCPI type is **int16**.
- **Related Commands:** CALC:AVER:COUNT, CALC:AVER:STATE

Usage

```
CALC:AVER:COUNT?           A subsequent enter statement will return the value currently set
```

CALCulate:CLIMits:FAIL[:CUMulative]?

CALCulate:CLIMits:FAIL[:CUMulative]? returns the composite limit test status for all channels measured since the module was INITiated (CUMulative).

Comments

- If any channel has exceeded its limit test since the module was INITiated, the returned value will be 1. If no channel has exceeded its limit test, the returned value will be 0.
- This condition is also reported to bit 11 of the Operational Status Group and can generate a VXIbus interrupt.
- **Returned Value:** Numeric 0 or 1. The C-SCPI type is **int16**.
- **Related Commands:** CALC:CLIM:FLIM:CHAN?, CALC:CLIM:FLIM:POINTS?, CALC:LIMIT...

Usage CALC:CLIM:FAIL?

A subsequent enter statement will return 0 for no limit failures, or 1 for one or more limit failures.

CALCulate:CLIMits:FAIL:CURRent?

CALCulate:CLIMits:FAIL:CURRent? returns the composite limit test status for all channels measured in the last completed scan (CURRent).

Comments

- If any channel in the last completed scan has exceeded its limit test, the returned value will be 1. If no channel in the last completed scan has exceeded its limit test, the returned value will be 0.
- This condition is also reported to bit 11 of the Operational Status Group and can generate a VXIbus interrupt.
- **Returned Value:** Numeric 0 or 1. The C-SCPI type is **int16**.
- **Related Commands:** CALC:CLIM:FLIM:CHAN:...?, CALC:CLIM:FLIM:POINTS:...?, CALC:LIMIT:...

Usage CALC:CLIM:FAIL:CURR?

A subsequent enter statement will return 0 for no limit failures, or 1 for one or more limit failures.

CALCulate:CLIMits:FLIMits[:CHANnels][:CUMulative]?

CALCulate:CLIMits:FLIMits[:CHANnels][:CUMulative]? returns the individual channel limit test status for all channels measured since the module was INITiated (CUMulative).

Comments

- CALC:CLIM:FLIM:CHAN? returns 64 bits which report the status of each individual module channel. A binary one in a bit position of this "64-bit" value indicates that the channel associated with that bit position has exceeded its limit test.
- **Returned Value:** Returns 4 comma separated numeric values each representing 16 bits (a total of 64 bits - one for each channel). This "64-bit" value is returned with the least significant channel bits first, the most significant channel bits last (the C-SCPI data type is an **int16 array**).
- **Related Commands:** CALC:CLIM:FAIL?, CALC:CLIM:FAIL:CURR?, CALC:LIMIT...

Usage CALC:CLIM:FLIM:CHAN?

A subesequent enter statement for a 4 element array will return channel limit test status

CALCulate:CLIMits:FLIMits[:CHANnels]:CURRent?

CALCulate:CLIMits:FLIMits[:CHANnels]:CURRent? returns the individual channel limit test status for all channels measured in the last completed scan (CURRent).

Comments

- CALC:CLIM:FLIM:CHAN:CURR? returns 64 bits which report the status of each individual module channel. A binary one in a bit position of this "64-bit" value indicates that the channel associated with that bit position has exceeded its limit test.
- **Returned Value: Returned Value:** Returns 4 comma separated numeric values each representing 16 bits (a total of 64 bits - one for each channel). This "64-bit" value is returned with the least significant channel bits first, the most significant channel bits last (the C-SCPI data type is an **int16 array**).
- **Related Commands:** CALC:CLIM:FAIL?, CALC:CLIM:FAIL:CURR?, CALC:LIMIT...

Usage

CALC:CLIM:FLIM:CHAN:CURR?

A subesequent enter statement for a 4 element array will return channel limit test status

CALCulate:CLIMits:FLIMits:POINts[:CUMulative]?

CALCulate:CLIMits:FLIMits:POINts[:CUMulative]? returns the count of channels that exceeded their limit test since the module was INITiated (CUMulative).

Comments

- **Returned Value:** Numeric value from 0 through 64. The C-SCPI type is **int16**.
- **Related Commands:**

Usage

CALC:CLIM:FLIM:POINTS?

A subsequent enter statement will return the number of limit tests exceeded

CALCulate:CLIMits:FLIMits:POINts:CURRent?

CALCulate:CLIMits:FLIMits:POINts:CURRent? returns the count of channels that exceeded their limit test during the last completed scan (CURRent).

Comments

- **Returned Value:** Numeric value from 0 through 64. The C-SCPI type is **int16**.
- **Related Commands:**

Usage

CALC:CLIM:FLIM:POINTS:CURR?

A subsequent enter statement will return the number of limit tests exceeded

CALCulate:LIMit[:STATe]

CALCulate:LIMit[:STATe] *<enable>*,(@*<ch_list>*) enables or disables limit testing for the channels specified in *<ch_list>*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	discrete (string)	ON | OFF	none
ch_list	channel list (string)	100 - 163	none

Comments

- **Related Commands:** CALC:LIM[:STATE]?
- ***RST Condition:** CALC:LIM:STATE OFF

Usage

```
CALC:LIM ON,(@100:107)        turn on limit testing for channels 0 through 7
```

CALCulate:LIMit[:STATe]?

CALCulate:LIMit[:STATe]? (@*<channel>*) returns the state of limit testing for the channel specified in *<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 numeric 0 or 1. the C-SCPI type is **int16**.
- **Related Commands:** CALC:LIM[:STATE]

Usage

```
CALC:LIM:STATE? (@107)        returns state of limit testing for channel 7
```

CALCulate:LIMit:FAIL[:CUMulative]?

CALCulate:LIMit:FAIL[:CUMulative]? (@*<channel>*) returns the cumulative limit status for the channel specified by *<channel>*. A one (1) indicates that *<channel>* has exceeded its limit test since the module was INITiated (CUMulative).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- *<channel>* must specify a single channel.
- **Returned Value:** Numeric 1 or 0. The C-SCPI type is **int16**.
- **Related Commands:** CALC:LIM[:STATE], CALC:LIM:LOWer..., CALC:LIM:UPPer...

Usage

```
CALC:LIM:FAIL? (@102)
```

return Cumulative limit status for channel 2

CALCulate:LIMit:FAIL:CURRent?

CALCulate:LIMit:FAIL? (@*<channel>*) returns the current limit status for the channel specified by *<channel>*. A one (1) indicates that *<channel>* has exceeded its limit test during the last completed scan (CURRent).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
channel	channel list (string)	100 - 163	none

Comments

- *<channel>* must specify a single channel.
- **Returned Value:** Numeric 1 or 0. The C-SCPI type is **int16**.
- **Related Commands:** CALC:LIM[:STATE], CALC:LIM:LOWer..., CALC:LIM:UPPer...

Usage

```
CALC:LIM:FAIL:CURR? (@104)
```

return Current limit status for channel 4

CALCulate:LIMit:LOWer[:STATe]

CALCulate:LIMit:LOWer[:STATe] *<enable>*,(@*<ch_list>*) enables or disables testing of lower limits for channels specified in *<ch_list>*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
MIenable	discrete (string)	ON | OFF	none
ch_list	channel list (string)	100 - 163	none

Comments

- If CALC:LIM:LOW is OFF, an overrange will not cause a limit exceeded status. If CALC:LIM:LOW is ON, an overrange will cause a limit exceeded status regardless of the setting of CALC:LIM:LOW:DATA.
- **Related Commands:** CALC:LIM[:STATE], CALC:LIM:UPP[:STATE]
- ***RST Condition:** CALC:LIM:LOW OFF

Usage

```
CALC:LIM:LOW ON,(@100:107)
```

enable lower limit testing for channels 0 through 7

CALCulate:LIMit:LOWer[:STATe]?

CALCulate:LIMit:LOWer[:STATe]? (@*<channel>*) returns the state of lower limit testing for the channel specified by *<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.
- **Returned Value:** Numeric 0 or 1. C-SCPI type is **int16**
- **Related Commands:** CALC:LIM:LOW[:STATE]

Usage

```
CALC:LIM:LOW?(@104)
```

returns state of lower limit testing for channel 4

CALCulate:LIMit:LOWer:DATA

CALCulate:LIMit:LOWer:DATA *<lower_lim>*,(@*<ch_list>*) sets the lower limit value for channels specified in *<ch_list>*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
lower_lim	numeric (float32)	MIN or any valid float32	none
ch_list	channel list (string)	100 - 163	none

Comments

- A channel's lower limit value must be numerically lower than its upper limit value or an error will be generated when the module is INITiated.
- The lower limit is exceeded when the returned value is less than the value specified by <*lower_lim*>.
- **Related Commands:** CALC:LIM[:STATE], CALC:LIM:LOW[:STATE], CALC:LIM:UPP...
- ***RST Condition:** Lower limit for all channels set to -INFinity

Usage

```
CALC:LIM:LOW:DATA 3.75,(@102,105)
```

sets the lower limit for channels 2 and 5 to 3.75 VDC.

CALCulate:LIMit:LOWer:DATA?

CALCulate:LIMit:LOWer:DATA? (@<*channel*>) returns the lower limit value currently set for the channel specified by <*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.
- **Returned Value:** Numeric. The C-SCPI Type is **float32**.
- **Related Commands:** CALC:LIM:LOW:DATA

Usage

```
CALC:LIM:LOW:DATA? (@106)
```

return the lower limit set for channel 6

CALCulate:LIMit:UPPer[:STATe]

CALCulate:LIMit:UPPer[:STATe] <***enable***>,(@<***ch_list***>) enables or disables testing of upper limits for channels specified in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
MIenable	discrete (string)	ON | OFF	none
ch_list	channel list (string)	100 - 163	none

Comments

- If CALC:LIM:UPP is OFF, an overrange will not cause a limit exceeded status. If CALC:LIM:UPP is ON, an overrange will cause a limit exceeded status regardless of the setting of CALC:LIM:UPP:DATA.
- **Related Commands:** CALC:LIM[:STATE], CALC:LIM:LOW[:STATE]
- ***RST Condition:** CALC:LIM:UPP OFF

Usage

CALCulate:LIMit:UPPer[:STATe]?

CALCulate:LIMit:UPPer[:STATe]? (@<*channel*>) returns the state of upper limit testing for the channel specified by <*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.
- **Returned Value:** Numeric 0 or 1. C-SCPI type is **int16**
- **Related Commands:** CALC:LIM:UPP[:STATE]

Usage

CALC:LIM:UPP?(@104) *returns state of upper limit testing for channel 4*

CALCulate:LIMit:UPPer:DATA

CALCulate:LIMit:UPPer:DATA *<upper_lim>*,(@*<ch_list>*) sets the upper limit value for channels specified in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
upper_lim	numeric (float32)	MAX of any valid float32	none
ch_list	channel list (string)	100 - 163	none

Comments

- A channel's upper limit value value must be numerically higher than its lower limit value or an error will be generated when the module is INITiated.
- The upper limit is exceeded when the returned value is greater than the value specified by <*upper_lim*>.
- **Related Commands:** CALC:LIM[:STATE], CALC:LIM:UPP[:STATE], CALC:LIM:LOW...
- ***RST Condition:** Upper limit for all channels set to +INFinity

Usage

```
CALC:LIM:UPP:DATA 11.6,(@102,105)
```

sets the upper limit for channels 2 and 5 to 11.6 VDC.

CALCulate:LIMit:UPPer:DATA?

CALCulate:LIMit:UPPer:DATA? (@*<channel>*) returns the upper limit value currently set for the channel specified by <*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.
- **Returned Value:** Numeric. The C-SCPI Type is **float32**.
- **Related Commands:** CALC:LIM:UPP:DATA

Usage

```
CALC:LIM:UPP:DATA? (@107)
```

return the upper limit set for channel 7

The Calibration subsystem provides for two major categories of calibration.

1. "A/D Calibration"; In these procedures, an external multimeter is used to determine the actual voltage or resistance values of the HP E1413's internal calibration sources. The known values are then sent to the HP E1413 where they are stored and used to perform internal A/D calibration. These procedures each require a sequence of several commands from the CALibration subsystem (**CAL:CONFIG...**, **CAL:VALUE...**, and **CAL:STORE ADC**). For an actual calibration example see the HP E1413 Service Manual. Always execute *CAL? or a CAL:TARE operation after A/D Calibration.

2. "Working calibration", of which there are three levels (see Figure 5-2):

   - "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 E1413 where channel scanning speed is of primary importance, this function is performed only when the **CAL:ZERO?** command is executed.

   - "Channel Calibration"; This function corrects for offset and gain errors for each module channel. The internal current sources are also measured. 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?.

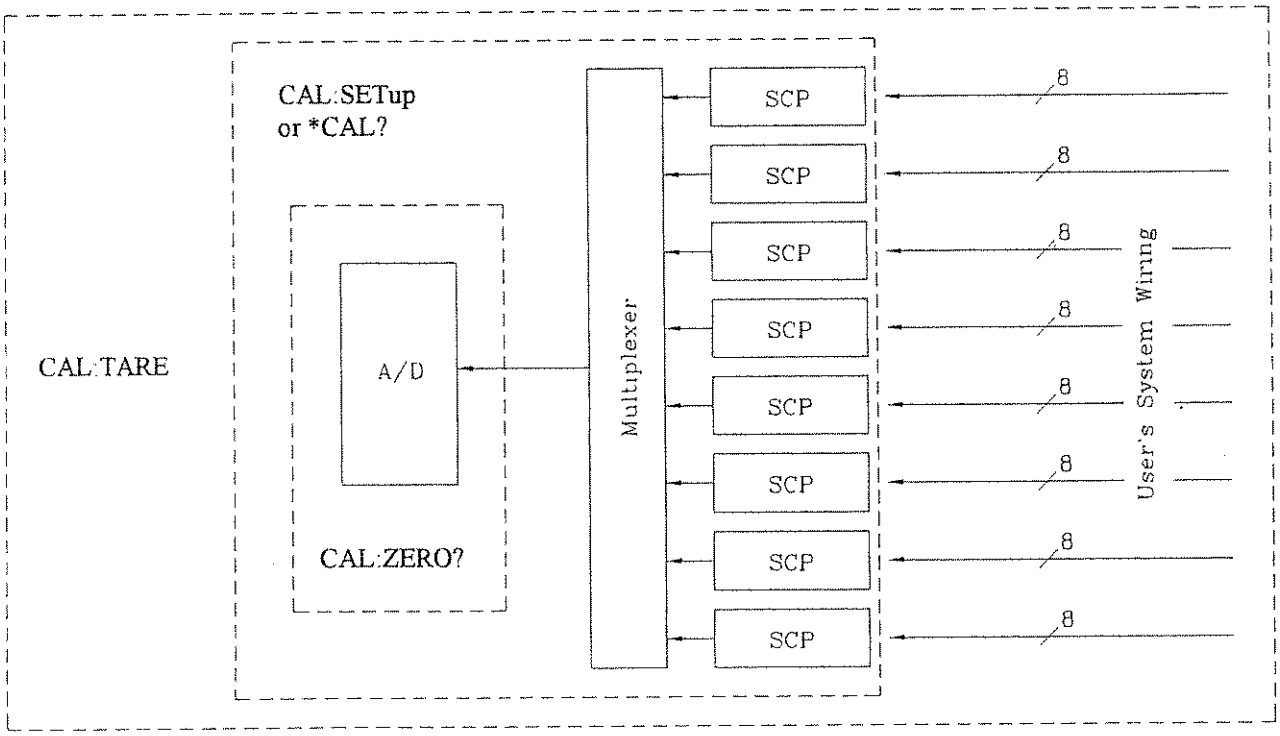


Figure 5-2 Levels of Working Calibration

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 reference resistor to the Calibration Bus. A four-wire measurement of the resistor can be 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.

Comments

- **Related Commands:** CAL:VAL:RES, CAL:STOR ADC

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 voltage reference to the Calibration Bus. A measurement of the source voltage can be 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 *range* parameter controls the 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 greater then 90% of the nominal value for each range, except the 16V range where the output is 10V.
- **Related Commands:** CAL:VAL:VOLT, STOR ADC

Usage

```
CAL:CONF:VOLT .25, ZERO         Voltage reference set to .25V range and zero output
CAL:CONF:VOLT .25, FSCALE       Set same range but full-scale output
```

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)
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 channel. The Channel Calibration function calibrates the A/D Offset, and the Gain/Offset for all 64 channels. This calibration is accomplished using internal calibration references.

- 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 E1413s 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?

Usage

```
CAL:SET                start SCP Calibration on 1st HP E1413
:                      start SCP Calibration on more HP E1413s
CAL:SET                start SCP Calibration on last HP E1413
CAL:SET?               query for results from 1st HP E1413
:                      query for results from more HP E1413s
CAL:SET?               query for results from last HP E1413
```

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. Also run *TST?
-2	No results available	No *CAL? or CAL:SETUP done

The C-SCPI type for this returned value is **int16**.

- **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 sets its Analog-to-Digital Converter calibration using the most recently measured CAL:VALues for voltage and resistance, and stores these to Flash Memory. When *type*=TARE, the module stores the most recently measured CAL:TARE channel offsets into Flash Memory.

NOTE The HP E1413'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 Enable 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.
- **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
```

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 4-27.

NOTES

1. The HP E1413'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.
2. 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 your program turns off OTD before executing CAL:TARE, it should also wait 1 minute for settling.

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 "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?.

- 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.

- Executing CAL:TARE sets the Calibrating bit (bit 0) in Operation Status Group. Executing CAL:TARE? resets the bit.

- **Related Commands:** CAL:TARE?, CAL:STOR TARE

- ***RST Condition:** ***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	No CAL:TARE done

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.
- **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 to the module for A/D calibration.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ref_ohms	numeric (float32)	± 5%	ohms

Comments

- ref_ohms must be within 5% of the nominal resistor value and may be specified in Kohm (kohm).
- A four-wire measurement of the resistor can be 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.
- Use the CAL:CONF:RES command to configure the reference resistor for measurement at the Calibration Bus connector.
- **Related Commands:** CAL:CONF:RES, CAL:STORE ADC

Usage

CAL:VALUE:RESISTANCE <measured value>

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 DC reference source to the module for A/D calibration.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ref_volts	numeric (float32)	must be within 10% of range nominal	volts

Comments

- ref_volts may be specified in millivolts (mv).
- A measurement of the source voltage can be 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.
- Use the CAL:CONF:VOLT command to configure the on-board voltage source for measurement at the Calibration 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.

- **Related Commands:** CAL:CONF:VOLT, CAL:STORE ADC

Usage

```
CAL:VALUE:VOLTAGE <measured value>
```

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.

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 out to the field wiring module connector.

- **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
```

The DIAGnostic subsystem allow 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
    :CHECksum?
    :COMMand
        :SCPWRITE <reg_addr>,<reg_data>
    :CUSTom
        :LINear <table_ad_range>,<table_block>,(@<ch_list>)
        :PIECewise <table_ad_range>,<table_block>,(@<ch_list>)
        :REFerence
            :TEMPerature
    :INTerrupt
        [:LINe] <intr_line>
        [:LINe]?
    :OTDetect
        [:STATe] ON | OFF,(@<ch_list>)
        [:STATe]? (@<channel>)
    :QUERy
        :SCPREAD? <reg_addr>
    :VERSion?
```

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:COMMand:SCPWRITE

DIAGnostic:COMMand:SCPWRITE *<reg_addr>,<reg_data>* writes data to custom Signal Conditioning Plug-on registers. Use to control custom SCPs created using the HP E1413 Option 14 Breadboard SCP.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
reg_addr	numeric (int32)	0-65,535	none
reg_data	numeric (int32)	0-65,535	none

Comments

- See Register Programming Section of your SCP Manual for parameter values
- **Related Commands:** DIAG:QUERy:SCPREAD

Usage

```
DIAG:COMM:SCPWRITE 696,3
```

For Option 15 Current Scource, set channel 7 on SCP 2 to 488mA with output enabled

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 E1413. 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 E1413 for chs 16-23
SENS:FUNC:CUST:LIN 1,1,(@116:123)       link custom EU with chs 16-23
```

INITiate then TRIGger module

DIAGnostic:CUSTom:PIECewise

DIAGnostic:CUSTom:PIECewise *<table_range>,<table_block>*, **(@***<ch_list>***)** downloads a custom piecewise Engineering Unit Conversion table (in *<table_block>*) to the HP E1413. 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 E1413*
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 <*tablerange*>,<*table_block*>,(@<*ch_list*>)

designate channel as reference

SENS:FUNC:CUST:REF <*range*>,(@<*ch_list*>)

set up scan list sequence (ch 0 in this case)

ROUT:SEQ:DEF: LIST1 (@100,100)

initiate, trigger, and retrieve data from FIFO

INIT;TRIG;SENS:DATA:FIFO?

dump reference temp register to FIFO

DIAG:CUST:REF:TEMP

read the diagnostic reference temperature value

SENS:DATA:FIFO?

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 E1413'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	discrete (string)	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).
- **Related Commands:** DIAG:OTDETECT:STATE?
- ***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
```

DIAGnostic:QUERy:SCPREAD?

DIAGnostic:QUERy:SCPREAD? *<reg_addr>* returns data word from a custom Signal Conditioning Plug-on register. Use to control custom SCPs created using the HP E1413 Option 14 Breadboard SCP.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
reg_addr	numeric (int32)	0-65,535	none

Comments

- See Register Programming Section of your SCP Manual for parameter values.
- **Returned Value:** returns numeric register value. C-SCPI type is **int32**.
- **Related Commands:** DIAG:COMM:SCPWRITE

Usage

```
DIAG:QUERY:SCPREAD? 192          read SCP's ID 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:** An example of the response string format is:
  HEWLETT-PACKARD,E1413,1413A00101,A.02.05,Thu Aug 5 9:38:07 MDT 1993
  The C-SCPI type is **string**.

- **Related Commands:** *IDN?

Usage

DIAG:VERS? *Returns version string as shown above*

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 5-9 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

Use Sequence

```
MEM:VME:ADDR #H300000
MEM:VME:SIZE #H100000          1M byte or 262144 readings
MEM:VME:STAT ON
      *
      *        (set up E1413 for scanning)
      *
TRIG:SOUR IMM                  let unit trigger on INIT
INIT                           program execution remains here until VME
                               memory is full or the HP E1413 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.

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.
- 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 5-9 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**).

- **Related Commands:** [SENSe:]DATA:FIFO:...?, [SENSe:]DATA:CVTable?, MEMory subsystem, and FETCh?
- ***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**.
- **Related Commands:** FORMAT
- ***RST Condition:** ASCII, 7

Usage

```
FORMAT?              Returns REAL, +32 | REAL, +64 | PACK, +64 | ASC, +7
```

The INITiate command subsystem moves the HP E1413 from the trigger idle state to the Initiated state. When initiated, the instrument is ready to receive one (:IMMediate) or more (:CONTinuous) trigger events. When triggered, one of four scan lists (specified by ROUTe:SCAN LIST1 through LIST4) will control the instrument. See the SENSe subsystem to specify scan list contents. See the TRIGger subsystem to specify the trigger source.

Subsystem Syntax

```
INITiate
    :CONTinuous ON | OFF
    [:IMMediate]
```

INITiate:CONTinuous

INITiate:CONTinuous <enable> . ON changes the trigger system from the Idle state to a continuous Wait For Trigger state. A trigger cycle occurs for each received trigger event. OFF cancels the CONTinuous mode and returns the instrument to the Trigger Idle state after the currently triggered scan is complete.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	discrete (string)	ON | OFF	none

Comments

- Executing INIT:CONT:ON 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 the ABORt command will reset the trigger system back to its Trigger Idle state and terminate any scan in progress. If INIT:CONT is ON, however, it will immediately return to Wait For Trigger state. If TRIG:SOUR is IMM also, then the module will resume scanning. Use INIT:CONT OFF to end a scan, then check the "Measuring" bit (bit 4) with STAT:OPER:COND?
- Sending INIT while the system is still in the Wait for Trigger state (already INITiated) will cause an error -213,"Init ignored".
- **Related Commands:** ABORt, CONFigure, TRIGger

- ***RST Condition:** Trigger system is in the Idle state.

Usage

```
INIT:CONTINUOUS ON          Set for continuous scanning through channel list on single trigger event
INIT:CONT OFF               Finish scan (if triggered) and return to Trigger Idle state AFTER the next trigger event if already armed
```

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. ROUTe:SCAN LIST1 through LIST4 specifies which of four scan lists to execute.

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 will reset the trigger system back to its Idle state and terminate any scan in progress.
- **Related Commands:** ABORt, CONFigure, TRIGger
- ***RST Condition:** Trigger system is in the Idle state.

Usage

```
INIT                        Both versions same function
INITIATE:IMMEDIATE
```

The INPut subsystem controls configuration of programmable *input* Signal Conditioning Plug-Ons (SCPs).

Subsystem Syntax

```
INPut
    :FILTer
        [:LPASs]
            :FREQuency <cutoff_freq>,(@<ch_list>)
            :FREQuency? (@<channel>)
            [:STATe] ON | OFF,(@<channel>)
            [:STATe]?
    :GAIN <chan_gain>,(@<ch_list>)
    :GAIN? (@<channel>)
```

INPut:FILTer[:LPASs]:FREQuency

INPut:FILTer[:LPASs]:FREQuency *<cutoff_freq>*,(@*<ch_list>*) sets the cutoff frequency of the filter on the specified channels.

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)	100 - 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.
- **Related Commands:** INP:FILT:FREQ?, INP:FILT:STAT ON | OFF
- ***RST Condition:** set to MIN

Usage

```
INP:FILT:FREQ 100,(@100:119)          Set cutoff frequency of 100 Hz for first 20 channels
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*.

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.
- **Returned Value:** Numeric value of Hz as set by the INP:FILT:FREQ command. The C-SCPI type is **float32**
- **Related Commands:** INP:FILT:LPAS, 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	discrete (string)	ON | OFF	none
ch_list	channel list (string)	100 - 163	none

Comments

- If the SCP has not yet been programmed, ON enables the SCP's default cutoff frequency.
- ***RST Condition:** ON

Usage

```
INPUT:FILTER:STATE ON,(@115,117)    Channels 115 and 117 return to previously
                                    set (or default) cutoff frequency
INP:FILT OFF,(@100:115)             Set channels 0-15 to "pass-through" state
```

INPut:FILTer[:LPASs][:STATe]?

INPut:FILTer[LPASs][:STATe]? (@<*channel*>) returns the currently set state of filtering for the specified channel.

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)	100 - 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.
- **Related Commands:** INP:GAIN?
- ***RST Condition:** gain set to 1

Usage

```
INP:GAIN 8,(@100:119)        Set gain of 8 for first 20 channels
INPUT:GAIN 64,(@155)         Set gain of 64 for channel 55
```

INPut:GAIN?

INPut:GAIN? (@<*channel*>) returns the gain currently set for *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.
- 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.**
- **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
```

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 E1413 for scanning)
        *
TRIG:SOUR IMM                  let unit trigger on INIT
INIT
*OPC?                          program execution remains here until VME
                               memory is full or the HP E1413 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.

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 E1413 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 *<size>* Specifies the number of bytes of VME memory to allocate for additional reading storage.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
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.
- size may be specified in decimal, hex (#H), octal (#Q), or binary(#B).
- bytes 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	1 | 0 | ON | OFF	none

Comments

- This command is only available in systems using an HP E1405B or HP E1406A command module.

- 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.
- **Related Commands:** MEMory subsystem, and FETCH?

Usage

```
MEM:VME:STAT?              Returns 1 for enabled, 0 for disabled
```

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>)
    :SHUNt 1 | 0 | ON | OFF,(@<ch_list>)
    :SHUNt? (@<channel>)
    :TTLTrg
        :SOURce TRIGger | FTRigger | SCPlugon | LIMit
        :SOURce?
    :TTLTrg<n>
        [:STATe] ON | OFF
        [:STATe]?
```

OUTPut:CURRent:AMPLitude

OUTPut:CURRent:AMPLitude *<amplitude>,(@<ch_list>)* sets the 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.

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)	100 - 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.
- **Related Commands:** *CAL, OUTP:CURR:AMPL?
- ***RST Condition:** MIN

Usage

```
OUTP:CURR:AMPL 488ua,(@116:123)
```
Set Current Source SCP at channels 16 through 23 to 488 μA

```
OUTP:CURR:AMPL 30E-6,(@105)
```
Set Current Source SCP at channel 5 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)	100 - 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.
- **Returned Value:** Numeric value of amplitude set. The C-SCPI type is **float32**.
- **Related Commands:** OUTP:CURR:AMPL

Usage

```
OUTP:CURR:AMPLITUDE? (@163)
```
Check SCP current set for channel 63 (returns +3.0E-5 or +4.88E-4)

OUTPut:CURRent[:STATe]

OUTPut:CURRent[:STATe] <*enable*>,(@<*ch_list*>) enables or disables current output on channels specified in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	numeric (uint16)	1 | 0 | ON | OFF	none
ch_list	channel list (string)	100 - 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.
- **Related Commands:** OUTP:CURR:AMPL, CAL:TARE
- ***RST Condition:** OUTP:CURR OFF (all channels)

Usage

```
OUTP:CURR OFF,(@100,108)          turn off current source channels 0 and 8
```

OUTPut:CURRent[:STATe]?

OUTPut:CURRent[:STATe]? (@*<channel>*) returns the state of the Current Source SCP channel specified by *<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 for enabled, 0 for disabled. C-SCPI type is **uint16**.
- **Related Commands:** OUTP:CURR:STATE, OUTP:CURR:AMPL

Usage

```
OUTP:CURR? (@108)                 query for state of Current SCP channel 8
execute enter statement here      enter query value, either 1 or 0
```

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)	100 - 163	none

Comments

- If *channel* specifies a non strain SCP, a 3007 "Invalid signal conditioning plug-on" error is generated.
- **Related Commands:** [SENSe:]FUNCtion:STRain..., [SENSe:]STRain...
- ***RST Condition:** OUTP:SHUNT 0 on all Strain SCP channels

Usage

```
OUTP:SHUNT 1,(@116:123)
```

add shunt resistance at channels 16 through 23

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)	100 - 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 CSCPI type is **uint16**.
- **Related Commands:** OUTP:SHUNT

Usage

```
OUTPUT:SHUNt? (@100)
```

Check status of shunt resistance on channel 0

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)	TRIGger | FTRigger | SCPlugon | LIMit	none

Comments • The following table explains the possible choices.

Parameter Value	Source of Trigger
FTRigger	Generated on the First TRigger of a multiple "counted scan" (set by TRIG:COUNT <*trig_count*>)
LIMit	Generated when a channel's limit test is exceeded
SCPlugon	Generated by a Signal Conditioning Plug-on (SCP)
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 passes through a scan list. 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 have one HP E1413 trigger (set to OUTP:TTLT:SOUR FTR and OUTP:TTLT<n> ON) a second HP E1413 (set to TRIG:SOUR TTLTRG<n>). The second module will be triggered in the ratio

$$\frac{1}{TRIG:COUNT_{module\#1}}.$$

  For an example of this arrangement, see Timer Based Scans at Different Rates on page 4-17.

- **Related Commands:** OUTP:TTLT<n>[:STATE], OUTP:TTLT:SOUR?, TRIG:SOUR, TRIG:COUNT

- ***RST Condition:** OUTP:TTLT:SOUR TRIG

Usage OUTP:TTLT:SOUR LIM — *toggle TTLTRGn line when limit test exceeded*

OUTPut:TTLTrg:SOURce?

OUTPut:TTLTrg:SOURce? returns the current ssetting for the TTLTRG line source.

Comments • **Returned Value:** Discrete, one of; TRIG, FTR, SCP, or LIM. C-SCPI type is **string**.

- **Related Commands:** OUTP:TTLT:SOUR

Usage OUTP:TTLT:SOUR? *enter statement will return on of FTR, LIM, 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	discrete (string)	ON | OFF	none

Comments

- Only one VXIbus TTLTRG line can be enabled simultaneously.
- **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>.

- **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
```

The ROUTe subsystem controls which of four Scan Lists will be executed at the next trigger event, and defines the channel content of each Scan List.

Subsystem Syntax

```
ROUTe
    :SCAN LIST1 | LIST2 | LIST3 | LIST4 | LISTL
    :SEQuence:
        :DEFine LIST1 | LIST2 | LIST3 | LIST4 | LISTL | ALL,(@<ch_list>)
        :DEFine?
        :POINts? LIST1 | LIST2 | LIST3 | LIST4 | LISTL
```

ROUTe:SCAN

ROUTe:SCAN *<scan_list>* establishes which of the four Scan Lists will be used for measurement scans.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
scan_list	discrete (string)	LIST1 - LIST4 | LISTL	none

Comments

- When ROUT:SCAN is LISTL, the module will automatically sequence through the scan lists specified by ROUT:SEQ:DEF LISTL,(@<lists>).

NOTES

1. When ROUT:SCAN selects LISTL (List-of-Lists), each scan list it specifies must contain at least 6 channels.
2. When CALC:AVER is ON, LISTL is not allowed.

- Normally the specified scan list number becomes effective when the Trigger System moves from the Initiated State to the Waiting for Trigger State. If ROUT:SCAN is executed after this point, it will become effective for the next scan. If INIT:CONT is ON and TRIG:SOUR is IMM, ROUT:SCAN will generate an error. Refer to Figure 5-2 for Trigger System event sequence.
- **Related Commands:** ABORt, INIT..., ROUT:SEQ:DEF
- ***RST Condition:** ROUT:SCAN LIST1.

Usage

```
ROUTE:SCAN LIST3
```

The next scan list executed will be LIST3

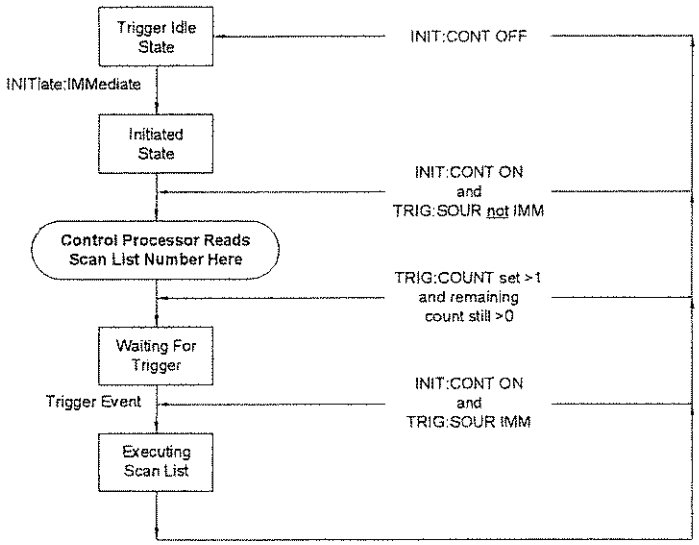


Figure 5-3 Event Sequence for ROUTe:SCAN

ROUTe:SEQuence:DEFine

ROUTe:SEQuence:DEFine *<scan_list>*,(@*<ch_list>*) define channel content and sequence for Scan List LIST<n>, or define the Scan List sequence for the List-of-Lists (LISTL). LISTL allows Automatic Scan List Sequencing.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
scan_list	discrete (string)	LIST1 - LIST4 | LISTL |ALL	none
ch_list	channel list (string)	for LIST1-4 or ALL, use 100 - 163 | <cc>(00 - 63) for LISTL, use 1-4	none

Comments

- Channels can be specified in any order.
- When CALC:AVER:STATE is OFF, channels can be specified multiple times in an individual Scan List. When CALC:AVER:STATE is ON, each channel must be appear only once in an individual Scan List.
- A channel list may contain as many as 1,024 entries.
- When ROUT:SCAN selects LIST1 through LIST4, the specified scan list can contain as few as two (2) channels (may be the same channel). When ROUT:SCAN selects LISTL, each Scan List specified in the List-of-Lists must contain at least 6 entries.

- When *scan_list is* ALL, the channel specification is copied to all four Scan Lists.

- For LISTL, the <*ch_list*> parameter can specify up to 1,024 Scan List numbers (1 through 4) in any order and any list number can be specified multiple times.

- For LIST1 - LIST4 or ALL, the SCPI Relative Channel specification form can be used to control the conversion, and storage destination of data measured during a scan. The SCPI Relative Channel specification syntax is:

  (@cc(nn, nn, nn:nn))
  where cc = card number, and nn = channel number.

  For the ROUT:SEQ:DEF command the card number becomes the Channel Data Modifier. The value can range from 1 through 7. The value controls whether Engineering Unit conversion is performed and the internal destination of the resulting value. The following table explains the effect of the Channel Data Modifier:

Channel Data Modifier	Description
1	Perform EU conversion and store result in both FIFO buffer and Current Value Table
2*	Leave measurement as voltage and store result in both FIFO and CVT
3	Perform EU conversion and store result in CVT only
4*	Leave measurement as voltage and store in CVT only
5	Perform EU conversion and store result in FIFO only
6*	Leave measurement as voltage and store in FIFO only
7*	Leave measurement as voltage and don't store result in either FIFO or CVT. Use as dummy channel set

* Limit Checking (CALC:LIM:...) can not be performed on channels that are not converted to Engineering Units.

- Both the standard and relative channel specification modes can be mixed within a Scan List definition. For example:

  ROUTE:SEQ:DEF LIST1,(@100:115, 6(00:15))

  specifies that the readings taken on channels 0 through 15 are to be converted into engineering units and stored in both the FIFO data buffer and the Current Value Table (CVT). In addition, channels 0 through 15 are to be read and the raw voltage values are to be added to the FIFO buffer. The FIFO will contain 16 converted readings and 16 voltage readings. The CVT will contain a converted reading for channels 0 through 15.

- After using ROUT:SEQ:DEF, use [SENSe:]FUNCtion:... to set each channel to a specific function and range.
- **Related Commands:** ROUT:SCAN
- ***RST Condition:** Scan List 1 = (@100:163), Scan Lists 2 through 4 have no channels assigned to them.

Usage

```
ROUT:SEQ:DEF  LIST2,(@100:131)
```
EU conversion to both FIFO and CVT (standard)

```
ROUT:SEQ:DEF  ALL,(@100:131)
```
Same channels and conversion as above but to all four Scan Lists

```
ROUT:SEQ:DEF  LIST1,(@2(00:31)
```
Voltage readings to both FIFO and CVT (relative channel specification using Channel Modifier

ROUTe:SEQuence:DEFine?

ROUTe:SEQuence:DEFine? *<scan_list>*[,*<mode>*] When *<scan_list>* is LIST1 - LIST4, returns either the sequence of channels or the sequence of Channel Data Modifiers for Scan List LIST<n>. When *<scan_list>* is LISTL, returns the sequence of scan list numbers.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
scan_list	discrete (string)	LIST1 - LIST4 | LISTL	none
mode	discrete (string)	CHANnel | MODifier	none

Comments

- The default for *mode* is CHAN.
- When *<scan_list>*is LIST1-4 and *mode* is CHAN, ROUT:SEQ:DEF? returns the sequence of channels assigned to *scan_list*.
- When *<scan_list>*is LIST1-4 and *mode* is MOD, ROUT:SEQ:DEF? returns the Channel Data Modifiers assigned to the current Scan List channels. The meaning of the Channel Data Modifier values are shown in the following table.

Channel Data Modifier	Description
1	Perform EU conversion and store result in both FIFO buffer and Current Value Table
2*	Leave measurement as voltage and store result in both FIFO and CVT
3	Perform EU conversion and store result in CVT only
4*	Leave measurement as voltage and store in CVT only
5	Perform EU conversion and store result in FIFO only
6*	Leave measurement as voltage and store in FIFO only
7*	Leave measurement as voltage and don't store result in either FIFO or CVT. Use as dummy channel set

* Limit Checking (CALC:LIM:...) can not be performed on channels that are not converted to Engineering Units.

- **Returned Value:** Returned values are 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 5-9 of this chapter. Each value is 2 bytes in length (the C-SCPI data type is an **int16 array**).

- **Related Commands:** ROUT:SEQ:DEF, ROUT:SCAN

- ***RST Condition:** Scan List 1 = (@100:163) (Channel Data Modifiers are all 1), Scan Lists 2 through 4 have no channels assigned to them, LISTL as no Scan Lists assigned to it.

Usage

```
ROUT:SEQ:DEF? LIST2, CHAN        query for channel sequence in LIST2
enter comma separated list of channels

ROUT:SEQ:DEF? LISTL              query for scan list sequence in LISTL
enter comma separated list of Scan Lists
```

ROUTe:SEQuence:POINts?

ROUTe:SEQuence:POINts? *<scan_list>* When *<scan_list>* is LIST1 - LIST4, returns the number of channels defined in the specified Scan List. When *<scan_list>* is LISTL, returns the number of Scan Lists defined.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
scan_list	discrete (string)	LIST1 - LIST4 | LISTL	none

Comments

- The number of channels returned with this command may not reflect the number of readings that will be sent to FIFO memory. This is because the Channel Data Modifier (see ROUT:SEQ:DEF command) of 3, 4 or 7 can be used which does not place the readings in FIFO.
- **Returned Value:** Numeric. The C-SCPI type is **int16**.
- **Related Commands:** ROUT:SEQ:DEF
- ***RST Condition:** Scan Lists contain zero channels.

Usage

```
ROUT:SEQ:DEF? LIST3
```

check number of channels defined for scan list 3 (returns a number between 0 and 1024)

The SAMPle subsystem provides commands to set and query the interval between channel measurements (pacing) for each of the four Scan Lists.

Subsystem Syntax

```
SAMPle
    :TIMer  LIST1 | LIST2 | LIST3 | LIST4 |LISTL | ALL,<interval>
    :TIMer?  LIST1 | LIST2 | LIST3 | LIST4 | LISTL
```

SAMPle:TIMer

SAMPle:TIMer *<scan_list>,<interval>* sets the time interval between channel measurements for the specified Scan List, or ALL four Scan Lists.When SAMP:TIM LISTL is specified, *<interval>* sets the time between channels for all scan lists defined in LISTL, overriding any timer settings for individual Scan Lists..

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
scan_list	discrete (string)	LIST1 - LIST4 | LISTL | ALL	none
interval	numeric (float32) (string)	1.0E-5 to 32.768E-3 | MIN | MAX	seconds

Comments

- The minimum *interval* is 10 µ seconds. The resolution for *interval* is .5µ second. ***interval*** may be specified in milliseconds (ms), or microseconds (us).
- 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.
- **Related Commands:** SAMP:TIMER?
- ***RST Condition:** Sample Timer for all Channel Lists set to 1.0E-5 seconds.

Usage

```
SAMPLE:TIMER  LIST3,1E-3
```

Pace measurements at 1 millisecond intervals for channels in Scan List 3

SAMPle:TIMer?

SAMPle:TIMer? *<scan_list>* returns the sample timer interval for the specified Channel List or List-of-Lists (LISTL).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
scan_list	discrete (string)	LIST1 - LIST4 | LISTL	none

Comments

- **Returned Value:** Numeric. The C-SCPI type is **float32**.
- **Related Commands:** SAMP:TIMER
- ***RST Condition:** Sample Timer for all Channel Lists set to 1.0E-5 seconds.

Usage

```
SAMPLE:TIMER? LIST4
```

Check the interval between channel measurements for scan list 4

Subsystem Syntax [SENSe:]

```
DATA
    :CVTable? (@<ch_list>)
        :RESet
    :FIFO
        [:ALL]?
        :COUNt?
            :HALF?
        :HALF?
        :MODE BLOCk | OVERwrite
        :MODE?
        :PART? <n_readings>
        :RESet
FILTer
    [:LPASs]
        [:STATe] ON | OFF
        [:STATe]?
FUNCtion
    :CUSTom [<range>,](@<ch_list>)
        :REFerence [<range>,](@<ch_list>)
        :TC <type>,[<range>,](@<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>)
    :VOLTage[:DC] [<range>,](@<ch_list>)
REFerence <sensor_type>, [<sub_type>,](@<ch_list>)
    :TEMPerature <degrees_celsius>
STRain
    :EXCitation <excite_v>,(@<ch_list>)
    :EXCitation?
    :GFACtor <gage_factor>,(@<ch_list>)
    :GFACtor? (@<channel>)
    :POISson <poisson_ratio>,(@<ch_list>)
    :POISson? (@<channel>)
    :UNSTrained <unstrained_v>,(@<ch_list>)
    :UNSTrained? (@<channel>)
```

[SENSe:]DATA:CVTable?

[SENSe:]DATA:CVTable? (@<*ch_list*>) returns from the Current Value Table the most recently measured values for the channels specified.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
ch_list	channel list	100 - 163	none

Comments

- [SENSe:]DATA:CVTable? (@<*ch_list*>) allows you to "view" the current channel values even while a long measurement scan is taking place.
- The Current Value Table is an area of A24 memory that contains a copy of the most recent measurement for each channel that has been measured since the last *RST or INIT command.
- The format of readings returned is set using the FORMat[:DATA] command
- **Returned Value:** 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. The C-SCPI data type is a **string array**.

  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 5-9 of this chapter. For REAL 32, each reading is 4 bytes in length (the C-SCPI data type is a **float32 array**). For REAL 64 and PACK 64, each reading is 8 bytes in length (the C-SCPI data type is a **float64 array**).

NOTE After *RST/Power-on, each channel location in the CVT contains the IEEE-754 value "Not-a-number" (NaN). Channels specified in the DATA:CVT? command that have not been measured during the scan will return the value 9.91E37.

Channel readings which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see table on page 5-48 for actual values for each data format).

- ***RST Condition:** CVT contains IEEE-754 "Not a Number".

Usage

```
SENS:DATA:CVT? (@1(00:63))      Return entire CVT (64 channels)
DATA:CVTABLE? (@108,110)        return latest values from channels 8 and 10
```

[SENSe:]DATA:CVTable:RESet

[SENSe:]DATA:CVTable:RESet sets all 64 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".
- **Related Commands:** SENSE:DATA:CVT?
- ***RST Condition:** SENSE:DATA:CVT:RESET

Usage SENSE:DATA:CVT:RESET *Clear the Current Value Table*

[SENSe:]DATA:FIFO[:ALL]?

[SENSe:]DATA:FIFO[:ALL]? returns all readings remaining in the FIFO buffer until all measurements are complete or until the number of readings returned exceeds FIFO buffer size (65,024).

Comments

- DATA:FIFO? may be used to acquire all readings (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 readings from the FIFO.
- The format of readings returned is set using the FORMat[:DATA] command.
- **Returned Value:** 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. The C-SCPI data type is a **string array**.

  REAL 32, REAL 64, and PACK 64, readings 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 5-9 of this chapter. For REAL 32, each reading is 4 bytes in length (the C-SCPI data type is a **float32 array**). For REAL 64 and PACK 64, each reading is 8 bytes in length (the C-SCPI data type is a **float64 array**).

NOTE Channel readings which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see table on page 5-48 for actual values for each data format).

- **Related Commands:** SENSE:DATA:FIFO:HALF?
- ***RST Condition:** FIFO is empty

Usage

```
DATA:FIFO?                          return all FIFO readings 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 readings currently in the FIFO buffer.

Comments

- DATA:FIFO:COUNT? is used to determine the number of readings 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 readings 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 readings), 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 readings*

[SENSe:]DATA:FIFO:HALF?

DATA:FIFO:HALF? returns 32,768 readings if the FIFO buffer is at least half-full. This command provides a fast means of acquiring blocks of readings from the buffer.

Comments

- For acquiring data from continuous scans, your application needs to execute a DATA:FIFO:HALF? command and a read statement often enough to keep up with the scan reading rate.
- Use the DATA:FIFO:ALL? command to acquire the readings remaining in the FIFO buffer after the last scan has completed.
- The format of readings returned is set using the FORMat[:DATA] command.
- **Returned Value:** 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. The C-SCPI data type is a **string array**.

  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 5-9 of this chapter. For REAL 32, each reading is 4 bytes in length (the C-SCPI data type is a **float32 array**). For REAL 64 and PACK 64, each reading is 8 bytes in length (the C-SCPI data type is a **float64 array**).

NOTE Channel readings which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see table on page 5-48 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 readings
```

[SENSe:]DATA:FIFO:MODE

[SENSe:]DATA:FIFO:MODE *<mode>* sets the mode of operation for the FIFO reading memory. The two valid modes are BLOCK and OVERWRITE. In Block Mode when the FIFO has been filled, new readings are discarded and will not be stored until the current FIFO data is read. In Overwrite Mode when the FIFO "fills up", new readings overwrite the oldest readings. In this situation, the FIFO contains only the latest 64K readings.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
mode	discrete (string)	BLOCk | OVERwrite	none

Comments

- In BLOCk mode, when FIFO becomes full and measurements are still being made, the new readings are discarded.
- **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_readings>* returns *n_readings* from the FIFO buffer.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
n_readings	numeric (int32)	1 - 2,147,483,647	none

Comments

- Use the DATA:FIFO:COUNT? command to determine the number of readings in the FIFO buffer.
- The format of readings returned is set using the FORMat[:DATA] command.
- **Returned Value:** 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. The C-SCPI data type is a **string array**.

  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 5-9 of this chapter. For REAL 32, each reading is 4 bytes in length (the C-SCPI data type is a **float32 array**). For REAL 64 and PACK 64, each reading is 8 bytes in length (the C-SCPI data type is a **float64 array**).

NOTE Channel readings which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see table on page 5-48 for actual values for each data format).

- **Related Commands:** DATA:FIFO:COUNT?
- ***RST Condition:** FIFO buffer empty

Usage

```
DATA:FIFO:PART? 256          return 256 readings from FIFO
```

[SENSe:]DATA:FIFO:RESet

[SENSe:]DATA:FIFO:RESet clears the FIFO of readings. The FIFO counter is reset to 0.

Comments

- **Related Commands:** SENSE:DATA:FIFO...
- ***RST Condition:** SENSE:DATA:FIFO:RESET

Usage

```
SENSE:DATA:FIFO:RESET          Clear the FIFO
```

[SENSe:]FILTer[:LPASs][:STATe]

[SENSe:]FILTer[:LPASs][:STATe] *<enable>* enables or disables the A/D's low-pass filter.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
enable	discrete (string)	ON | OFF	none

Comments

- The A/D low-pass filter when enabled affects all channels
- The filter's cut-off frequency is 12 KHz
- **Related Commands:** SENSE:FILTER:LPASS:STATE?
- ***RST Condition:** SENSE:FILTER:LPASS:STATE OFF

Usage

```
SENSE:FILTER ON                Enable A/D low-pass filter
```

[SENSe:]FILTer[:LPASs][:STATe]?

[SENSe:]FILTer[:LPASs][:STATe]? returns the current state of the A/D's low-pass filter.

Comments

- **Returned Value:** Returns numeric value 0 (off) or 1 (on). The C-SCPI type is **int16**.
- **Related Commands:** SENSE:FILTER:LPASS:STATE ON | OFF

Usage

```
SENSE:FILTER?                  enter statement returns either 1 or 0
```

[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 E1413 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 E1413 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 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.

- **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 E1413 for chs 16-23
SENS:FUNC:CUST 1,(@116:123)              link custom EU with chs 16-23
INITiate then TRIGger module
```

[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 FIFO and/or CVT, as well as the module's Reference Temperture 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 Channels with EU Conversions" in Chapter 3 for more information
- The **<*range*> parameter:** The HP E1413 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 E1413 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. ***range*** may be specified in millivollts (mv).
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. Since there is no performance penalty you can simply specify auto range.
- 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
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 Channels with EU Conversions" in Chapter 3 for more information
- The **<*range*> parameter:** The HP E1413 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 E1413 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. ***range*** may be specified in millivollts (mv).
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. Since there is no performance penalty you can simply specify auto range.
- 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

- ***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: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*.

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

- ***<range>* parameter**. The HP E1413 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 E1413 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. ***range*** may be specified in millivolts (mv).

- 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).

- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. Since there is no performance penalty, you can simply specify auto range.
- 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 Channels with EU Conversions" in Chapter 3 for more information
- **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)

[SENSe:]FUNCtion:STRain
:FBENding
:FBPoisson
:FPOisson
:HBENding
:HPOisson
[:QUARter]

Note on Syntax: Although the strain function is comprised of eight 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 Channels with EU Conversions" in Chapter 3 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)	100 - 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.
- The **<*range*> parameter:** The HP E1413 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 E1413 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. ***range*** may be specified in millivollts (mv).
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. Since there is no performance penalty you can simply specify auto range.
- The channel calibration command (*CAL?) calibrates the excitation voltage source on each Bridge Completion SCP.
- **Related Commands:** *CAL?, [SENSE:]STRAIN...
- ***RST Condition:** SENSE:FUNC:VOLT 0,(@100:163)

Usage

```
FUNC:STRAIN 1,(@100:,105,107)
```

[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 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

- See "Linking Channels with EU Conversions" in Chapter 3 for more information

- The ***<range>* parameter:** The HP E1413 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 E1413 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. ***range*** may be specified in millivollts (mv).

- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. Since there is no performance penalty you can simply specify auto range.

- 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.
- **Related Commands:** *CAL?, OUTP:CURR (for RTDs and Thermistors), SENS:REF, and SENS:REF:TEMP (for Thermocouples)
- ***RST Condition:** SENSE:FUNC:VOLT 0,(@100:163)

Usage

Link first 20 channels to the K type thermocouple temperature conversion

```
FUNC:TEMP  TCOUPLE,K,(@100:119)
```

Link channel 20 to measure reference temperature using 5K thermistor

```
FUNC:REF  THERM,5000,(@120)
```

Define Scan List 1 to measure reference channel first then TC channels

```
ROUTE:SEQ:DEFINE LIST1,(@120,100:119)
```

Link channels 21 - 39 to the 10K ohm thermistor temperature conversion, range is defaulted to auto-range

```
FUNC:TEMP  THER,10kohm,(@121:139)
```

[SENSe:]FUNCtion:VOLTage

[SENSe:]FUNCtion:VOLTage[:DC] [*<range>*,](@*<ch_list>*) links the specified channels to return DC voltage.

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

- *<range>* **parameter**. The HP E1413 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 E1413 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. ***range*** may be specified in millivolts (mv).

- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. Since there is no performance penalty you can simply specify auto range.
- 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 Channels with EU Conversions" in Chapter 3 for more information
- **Related Commands:** *CAL?, INPUT:GAIN...
- ***RST Condition:** SENSE:FUNC:VOLT 0,(@100:163)

Usage FUNC:VOLT (@140:163) *Channels 40 - 63 measure voltage in auto-range*

[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 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.

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 Channels with EU Conversions" in Chapter 3 for more information.
- *<range>* parameter. The HP E1413 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 E1413 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. ***range*** may be specified in millivolts (mv).
- 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 juntion temp and is not re-defineable.
- For <type> THERmistor, the ***<sub_type>*** parameter may be specified in ohms or kohm.
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. Since there is no performance penalty you can simply specify auto range.
- 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.
- **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: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 E1413 the temperature of a controlled temperature thermocouple reference junction.
- **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.
- **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 CSCPI type is **flt32**.
- *<channel>* must specify a single channel only.
- **Related Commands:** STRAIN:EXCitation

Usage

```
STRAIN:EXC? (@107)
enter statement here
```

query excitation voltage for channel 7

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

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 CSCPI 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
```

[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	type	100 - 163	none

Comments

- **Returned Value:** numeric value of the Poisson ratio. CSCPI 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
```

[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 CSCPI 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
```

The STATus subsystem communicates with the SCPI defined Operation and Questionable Data status register sets. Each is comprised of a Condition register, 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). Event registers contain latched representations of signal transition events from their Condition register. The HP E1413 has no programmable Transition Filter registers so any low-to-high transition in a Condition register will cause the corresponding bit in an Event register to be set to one. 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.

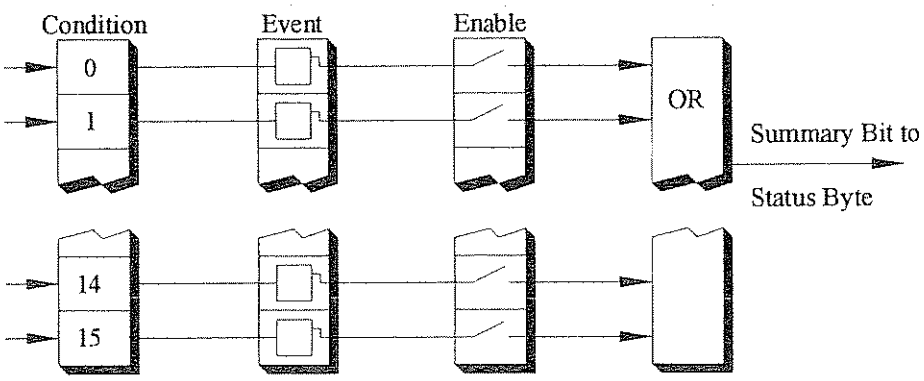


Figure 5-4 General Status Register Organization

Subsystem Syntax

```
STATus
    :OPERation
        :CONDition?
        :ENABle <enable_mask>
        :ENABle?
        [:EVENt]?
    :PRESet
    :QUEStionable
        :CONDition?
        :ENABle <enable_mask>
        :ENABle?
        [:EVENt]?
```

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.

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 E1413. The bit assignments are:

Bit # (dec value)	Bit Name	Description
0 (1)	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.
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 completed (may not indicate all measurements have been taken when INIT:CONT is ON or TRIG:COUNT >1).
9 (512)	SCP Trigger	An SCP has sourced a trigger event (future HP 1413 SCPs)
10 (1024)	FIFO Half Full	The FIFO contains at least 32,768 readings
11 (2048)	Limit Test Exceeded	One or more limit tests was exceeded
1-3, 5-7, 11-15		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.
- **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 0, 4, 8, 9, 10, or 11 are enabled, VXI card interrupts will occur as follows:

When the event corresponding to bit 0 or 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

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 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: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)		Bit Name	Description
0 - 7			Not used
8	(256)	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)	Trigger Too Fast	Scan not complete when another trigger event received.
10	(1024)	FIFO Overflowed	Attempt to store more than 65,024 readings in FIFO.
11	(2048)	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)	VME Memory Overflow	The number of readings 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).
14-15			Not used

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.
- **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

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 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

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,E1413 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.

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"*

The TRIGger command subsystem controls the behavior of the trigger system once it is initiated (see INITiate command subsystem).

Figure 5-5 shows the overall Trigger System model. The shaded area shows the ARM subsystem portion.

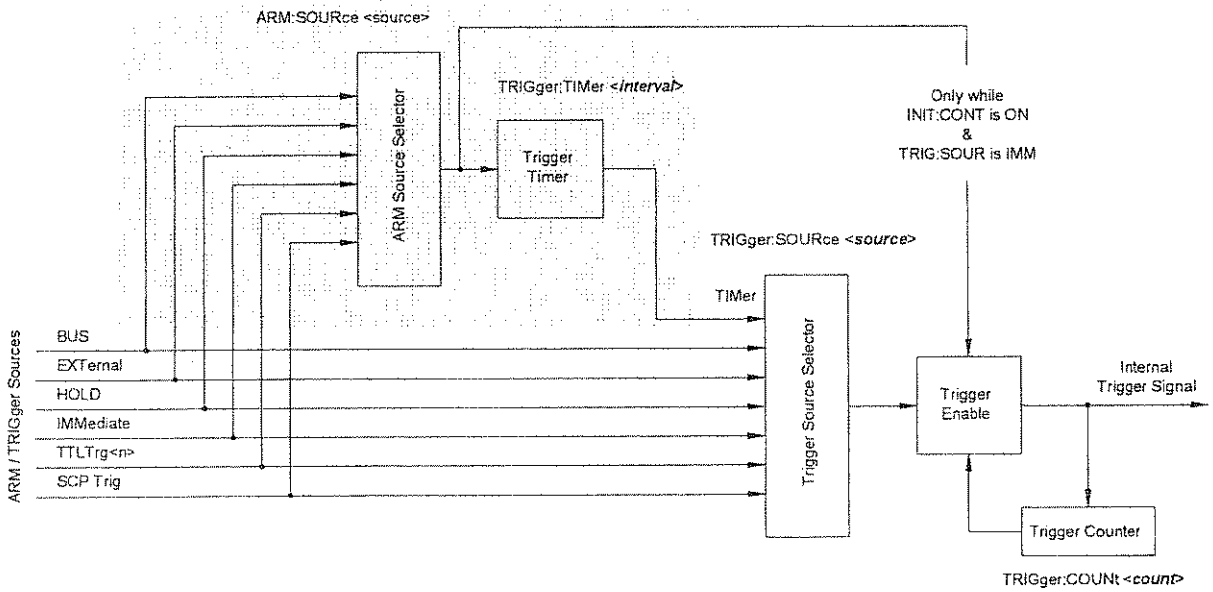


Figure 5-5 Logical Trigger Model

NOTE If TRIG:SOUR TIM is not set, and the Continous Mode (TRIG:SOUR IMM and INIT:CONT ON), is not set the ARM:SOURce must be set to IMM or an Error -221,"Settings conflict" will be generated.

Subsystem Syntax

```
TRIGger
    :COUNt <trig_count>
    :COUNt?
    [:IMMediate]
    :SOURce BUS | EXTernal | HOLD | SCP | IMMediate | TIMer | TTLTrg<n>
    :SOURce?
    :TIMer
        :MODE SYNChronous | ASYNchronous
        :MODE?
        [:PERiod] <trig_interval>
        [:PERiod]?
```

TRIGger:COUNt

TRIGger:COUNt ***<trig_count>*** sets the number of times a trigger event will be accepted after the module is INITiated. When *trig_count* is exhausted, the trigger system returns to the Trigger Idle State, waiting to be re-INITiated.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
Trig_count	numeric (uint16) (string)	0 to 65535 | MIN | MAX | INF	none

Comments

- When *trig_count* is set to 0 or INF, the trigger counter is disabled. Once the module is INITiated, it will stay in the Waiting for Trigger State, and accept an unlimited number of trigger events. ABORT will return the module to the Trigger Idle State.
- The default count is 1
- **Related Commands:** TRIG:COUNT?
- ***RST Condition:** TRIG:COUNT 1

Usage

```
TRIG:COUNT 10        Set the module to make 10 passes through a Scan List
TRIG:COUNT 0         Set the module to accept unlimited triggers
```

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 1

Usage

```
TRIG:COUNT?                    Query for trigger count setting
enter statement                Returns the TRIG:COUNT setting
```

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 specified source.

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	"Trig" signal on terminal module
HOLD	TRIGger[:IMMediate]
IMMediate	The trigger signal is always true (continuous triggering)
SCP	SCP Trigger Bus (future HP or SCP Breadboard)
TIMer	The internal trigger interval timer
TTLTrg<n>	The VXIbus TTLTRG lines (n=0 through 7)

- See NOTE on page 5-105
- The Trigger Source can be changed while the module is scanning (but not in the Continuous Mode). This allows changing the TRIG:TIM interval "on-the-fly". Change TRIG:SOUR to BUS or HOLD, change TRIG:TIM, then return to TRIG:SOUR TIM.
- While TRIG:SOUR is IMM, you need only INITiate the trigger system to start a measurement scan.
- **Related Commands:** ABORt, INITiate, *TRG
- ***RST Condition:** TRIG:SOUR HOLD

Usage TRIG:SOUR BUS *Trigger with TRIG command*

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 E1413 to return trigger source configuration*

TRIGger:TIMer:MODE

TRIGger:TIMer:MODE *<trimer_mode>* configures the trigger timer for either synchronous or asynchronous operation.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
trimer_mode	discrete (string)	SYNChronous | ASYNchronous	none

Comments

- The ASYNchronous mode causes the trigger timer to stop when the trigger count is exhausted, then restart when the module is re-INITiated. If INIT:CONT is ON, the asynchronous mode could cause jitter in the timing between the first measurement of one scan and the first measurement of the next scan. With the synchronous mode selected, the trigger timer continues to run while the module returns to the INITiated state. This insures that scans are evenly paced.
- **Related Commands:** TRIG:TIM:MODE?
- ***RST Condition:** TRIG:TIM:MODE ASYNCHRONOUS

Usage TRIG:TIM:MODE SYNC *Change trig timer mode to synchronous*

TRIGger:TIMer:MODE?

TRIGger:TIMer:MODE? returns the currently set mode for the trigger timer.

Comments

- **Returned Value:** Discrete, either SYNCH or ASYN. The C-SCPI type is **string**.
- **Related Commands:** TRIG:TIM:MODE
- ***RST Condition:** TRIG:TIM:MODE ASYNCHRONOUS

Usage TRIG:TIM:MODE? *Query returns either SYNC ar ASYN*

TRIGger:TIMer[:PERiod]

TRIGger:TIMer[:PERiod] *<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
interval	numeric (float32) (string)	1.0E-4 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 1.0E-3 seconds. ***interval*** may be specified in milliseconds (ms), or microseconds (us).

NOTES

1. TRIG:TIM can be executed while scanning is in progress. Change TRIG:SOUR to HOLD or BUS, execute TRIG:TIM *<interval>*, then return TRIG:SOUR to TIM.
2. To avoid a "Trigger Too Fast" error following INIT, the TRIG:TIM interval must be greater than: ((number of channels in scan list+3)*SAMP:TIM)+30µs.

- **Related Commands:** TRIG:SOUR TIMER, ARM:SOUR, ARM:IMM, INIT, TRIG:SOUR?
- ***RST Condition:** TRIG:TIM 1.0E-3

Usage

```
TRIG:TIMER 1.0E-2          Set the module to make a measurement scan every 10 mS
TRIG:TIMER 1               Set the module to make a measurement scan once 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
```

The following reference discusses 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.

- **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**.

- **Related Commands:** CALibration:SETup, CALibration:SETup?
- Executing this command **does not** alter the module's programmed state (function, range etc.).

NOTE If Open TransducerDetect (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.

*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).

*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**.

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**.

*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)

The *IDN? command returns the following command string for the HP E1413:
HEWLETT-PACKARD,E1413,*<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.

*OPC

Operation Complete. Causes an instrument to set bit 0 (Operation Complete Message) in the Standard Event Status Register when all pending operations 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 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?

*RST

Reset Command. Resets the HP E1413 as follows:

- Sets all four scan lists to their default states:
  - Scan List 1= ROUT:SEQ:DEF (@100:163),
    Scan List 2 through 4 are zero length *(undefined)*
  - SENSE:FUNC:VOLT AUTO,(@100:163) (all channels DCV, auto-range)
- Sets the trigger system as follows:
  - TRIGGER:SOURCE HOLD
  - TRIGGER:TIMER 1E-4
  - TRIGGER:COUNT 1
  - 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

*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" earlier in this chapter 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.

*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. The HP E1413 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 E1413 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?) See Error Messages in Appendix B

The C-SCPI type for this returned value is **int16**.

- Following *TST?, the module is placed in the *RST state.
- *TST? performs the following tests on the HP E1413 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
50-53:	Checks internal calibration source
54-55:	Checks gain calibration DAC
56-57:	Checks that autorange works
58-58:	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

*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.

The following tables summarize SCPI and IEEE-488.2 Common (*) commands for the HP E1413A 64-Channel High-Speed A/D.

SCPI Command Quick Reference	
Command	Description
ABORt	Stops scanning immediately and sets trigger system to idle state (scan lists are unaffected)
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
CALCulate	
:AVERage	
[:STATe] ON | OFF	Enables/disables channel measurement averaging
[:STATe]?	Returns the state of channel averaging
:COUNt *<n>*	Sets the number of measurements averaged to produce a stored reading
:COUNt?	Returns the current setting of measurements per reading
:CLIMits	
:FAIL	
[:CUMulative]?	Returns composite limit test status since module was INITed
:CURRent?	Returns composite limit test status for last completed scan list
:FLIMits	
[:CHANnels]	
[:CUMulative]?	Returns all channel's limit test status since module was INITed
:CURRent?	Returns all channel's limit test satus for last completed scan list
:POINts	
[:CUMulative]?	Returns count of channels exceeding limit tests since module INITed
:CURRent?	Returns count of channels exceeding limit tests for last completed scan
:LIMit	
[:STATe] ON | OFF,(@*<ch_list>*)	Enables/diasables all limit testing
[:STATe]? (@*<channel>*)	Returns state of limit testing
:FAIL	
[:CUMulative]? (@*<channel>*)	Returns limit test status for *channel* since module was INITed
:CURRent? (@*<channel>*)	Returns limit test status for *channel* for last completed scan list
:LOWer	
[:STATe] ON | OFF,(@*<ch_list>*)	Enables/diasables lower limit testing
[:STATe]? (@*<channel>*)	Returns state of lower limit testing
:DATA <lower_lim>,(@*<ch_list>*)	Sets lower limit for specified channels
:DATA? (@*<channel>*)	Returns lower limit setting for specified channel
:UPPer	
[:STATe] ON | OFF,(@*<ch_list>*)	Enables/diasables upper limit testing
[:STATe]? (@*<channel>*)	Returns state of upper limit testing
:DATA <lower_lim>,(@*<ch_list>*)	Sets upper limit for specified channels
:DATA? (@*<channel>*)	Returns upper limit setting for specified channel
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)

SCPI Command Quick Reference	
Command	Description
:STORe ADC | TARE	Store cal constants to Flash RAM for either A/D calibration or those generated by the CAL:TARE command
: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	
: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
:D32	
[:STATe] ON | OFF	Enables/disables D32 VXIbus data transfers (OFF=D16 transfers)
[:STATe]?	Returns state of D32 VXIbus transfers
: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#, revision #, and date e.g. HEWLETT-PACKARD,E1413A,1413A00001,A.01.00, Wed Jul 08 11:06:22 MDT 1992
FETCh?	Return readings stored in VME Memory (format set by FORM cmd)
FORMat	
[:DATA] <*format*>[, <*size*>] ASCii[, 7] PACKed[, 64] REAL[, 32] REAL, 64	Set format for response data from [SENSe:]DATA? Seven bit ASCII format (not as fast as 32-bit because of conversion) Same as REAL, 64 except NaN, +INF, and -INF format compatible with HP BASIC IEEE 32-bit floating point (requires no conversion so is fastest) 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	
:CONTinuous ON | OFF	When ON, module returns to Waiting for Trigger state after each scan When OFF, module goes to Trigger Idle state after each scan
[:IMMediate]	Put module in Waiting for Trigger state (ready to make one scan)
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

SCPI Command Quick Reference	
Command	**Description**
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 the Current Source SCP channel to 488μA (MAX) or 30μA (MIN)
: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
: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
:VOLTage	
:AMPLitude *<amplitude>*,(@*<ch_list>*)	Set the Strain Bridge excitation voltage (Option 21)
:AMPLitude? (@*<channel>*)	Returns the setting of the setting of excitation voltage on SCP
:STATe ON | OFF,(@*<ch_list>*)	Enable or disable the voltage source on SCP channels
:STATe? (@*<channel>*)	Returns the state of the voltage source SCP channel
ROUTe	
:SCAN LIST1 | LIST2 | LIST3 | LIST4 | LISTL	Selects the Scan List to be used in the next measurement set
:SEQuence	
:DEFine LIST1 | LIST2 | LIST3 | LIST4 | LISTL | ALL,(@*<ch_list>*)	Specify order of channel measurements for Scan LISTn or all Scan Lists
:POINts? LIST1 | LIST2 | LIST3 | LIST4 | LISTL	Returns the number of points (channels) in Scan LISTn
SAMPle	
:TIMer LIST1 | LIST2 | LIST3 | LIST4 | LISTL | ALL,*<interval>*	Sets the time interval in seconds between channel measurements for Scan LISTn, or all four Scan Lists
:TIMer? LIST1 | LIST2 | LIST3 | LIST4 | LISTL	Returns the timer interval in seconds for Scan LISTn
[SENSe:]	
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
FILTer	
[:LPASs]	
[:STATe] ON | OFF	Enables/disables A/D's 6 KHz low-pass filter
[:STATe]?	Returns the state of the A/D's low-pass filter
FUNCtion	Equate a function and range with groups of channels

SCPI Command Quick Reference	
Command	**Description**
: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>
:RESistance *<excite_current>*,[*<range>*,](@*<ch_list>*)	Configure channels for two-wire resistance measurement
: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 | EEX | J | K | N | S | T THERmistor, 2250 | 5000 | 10000	two-wire RTDs thermocouples two-wire thermistors
:TEMPerature *<sensor_type>*,*<sub_type>*, [*<range>*,](@*<ch_list>*)	Configure channels for temperature measurement types above: excitation current comes from Current Output SCP.
: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:
: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
:EXCitation? (@*<channel>*)	Returns the Excitation Voltage set for *<channel>*
: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
: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>*
STATus	
:OPERation	Operation Status Group: Bit assignments; 0=Calibrating, 4=Measuring, 8=Scan Complete, 10=FIFO Half Full
: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
: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

SCPI Command Quick Reference	
Command	Description
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
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
:MODE ASYNchronous | SYNChronous	ASYNC; Trigger Timer runs only while module is INITed, SYNC; Trigger Timer runs continouosly
:MODE?	Returns setting of Trigger Timer Mode
[:PERiod] <*trig_interval*>	Sets the interval between scan triggers when TRIG:SOUR is TIMer
[:PERiod]?	Returns setting of trigger timer

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,E1413A,1413A<*serial number*>,<*revision number*>
	*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.
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.
	*WAI	Wait to Complete	

Appendix A
Specifications

These specifications reflect the performance of the HP E1413 with the HP E1413 Option 11 Straight-Through Signal Conditioning Plug-on. The performance of the HP E1413 with other SCPs is found in the Specifications section of that SCP's manual.

Measurement ranges

DC Volts	(Opt 11 or 12) ±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	(Opt 15 with opt 11) 512 ohms to 131 Kohms FS
Strain	25,000 με or limit of linear range of strain gage

Measurement resolution

16 bits (including sign)

Maximum reading rate

100K rdgs/sec divided by the number of channels in the scan - for example,
100K rdgs/sec ÷ 64 channels = 1.56K rdgs/sec/ch
100K rdgs/sec ÷ 16 channels = 6.25K rdgs/sec/ch

Maximum input voltage
(Normal mode plus common mode)

Operating: < ±16 V peak Damage level: > ±42 V peak

Maximum common mode voltage

Operating: < ±16 V peak Damage level: > ±42 V peak

Common mode rejection

0 to 60Hz -105dB

Input impedance

greater than 100 Mohm differential

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

Measurement accuracy
DC Volts

(90 days) 23°C ±1°C (with *CAL? done after 1 hr warm up and CAL:ZERO? within 5 min.). If autoranging is ON, add ±.02% FS to accuracy 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

* [SENSe:]FILTer[:LPASs][:STATe] ON (max scan rate - 100 rdgs/sec/channel)

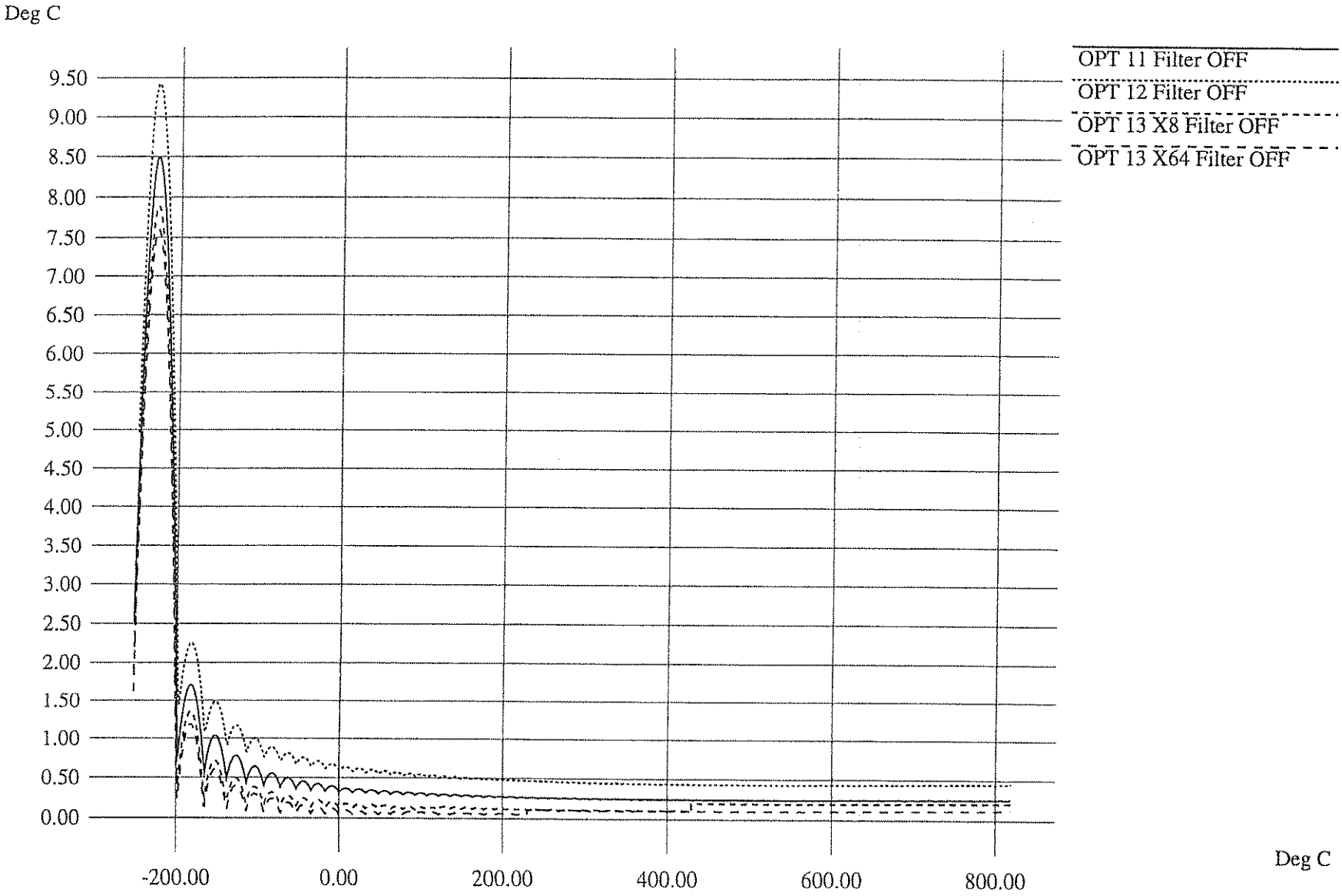
Temperature Coefficients: Gain - 10ppm/°C. Offset - (0 - 40°C) .14µV/°C, (40 - 55°C) .8µV+.38µV/°C

Temperature Accuracy

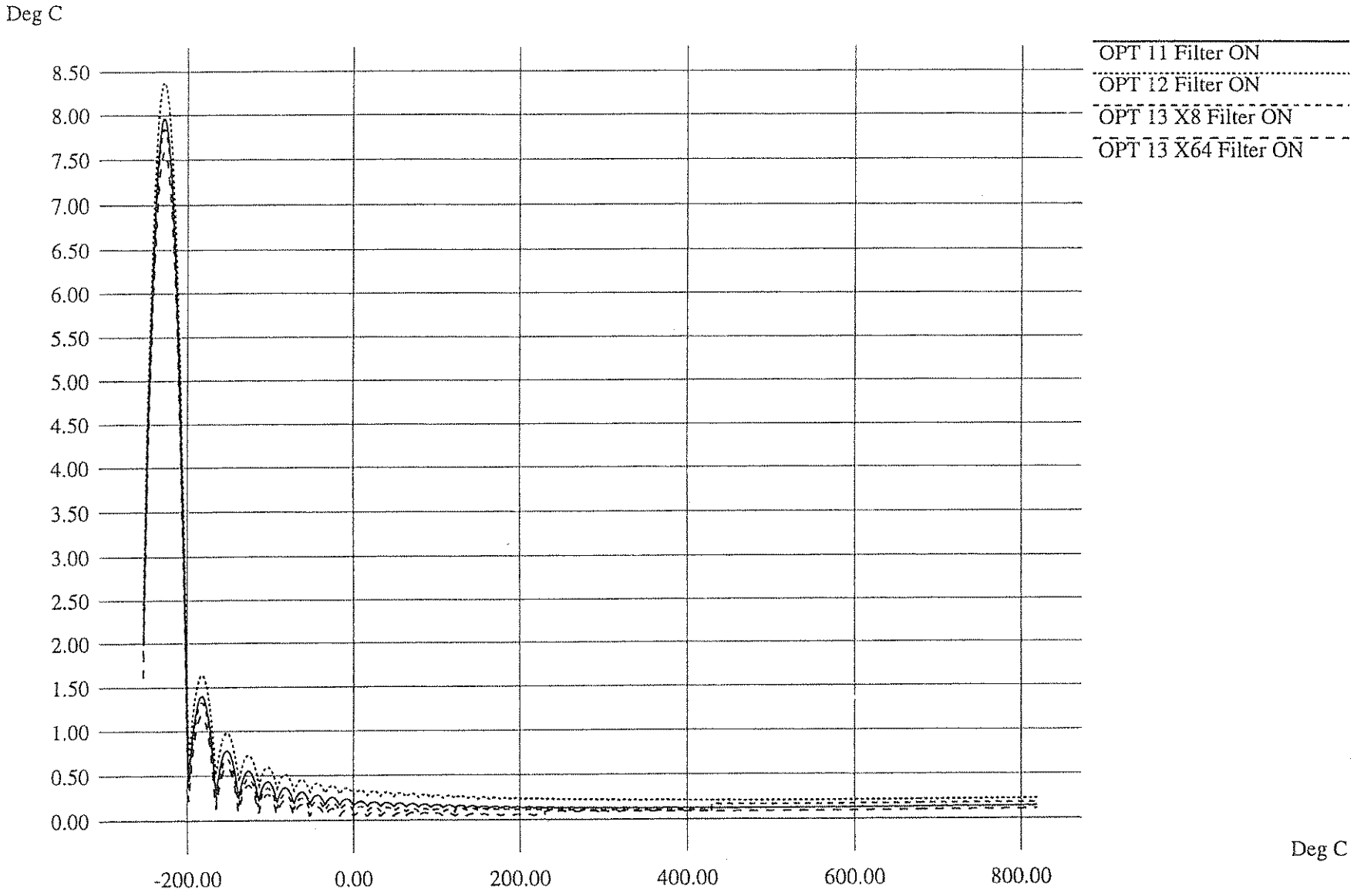
The following temperature accuracy curves include instrument and firmware linearization errors. The linearization algorithm used is based on the IPTS-68(78) standard transducer curves. Add your transducer accuracy to determine total measurement error.

- Thermocouple Type E (-200 to 800°C) A-4,5
- Thermocouple Type E (0 to 800°C) A-6,7
- Thermocouple Type EEXtended A-8,9
- Thermocouple Type J A-10,11
- Thermocouple Type K A-12,13
- Thermocouple Type R A-14,15
- Thermocouple Type S A-16,17
- Thermocouple Type T A-18,19
- Reference Thermistor 5K A-20,21
- Reference RTD 100Ω A-22,23
- RTD 100Ω A-24,25
- Thermistor 2250Ω A-26,27
- Thermistor 5KΩ A-28,29
- Thermistor 10KΩ A-30,31

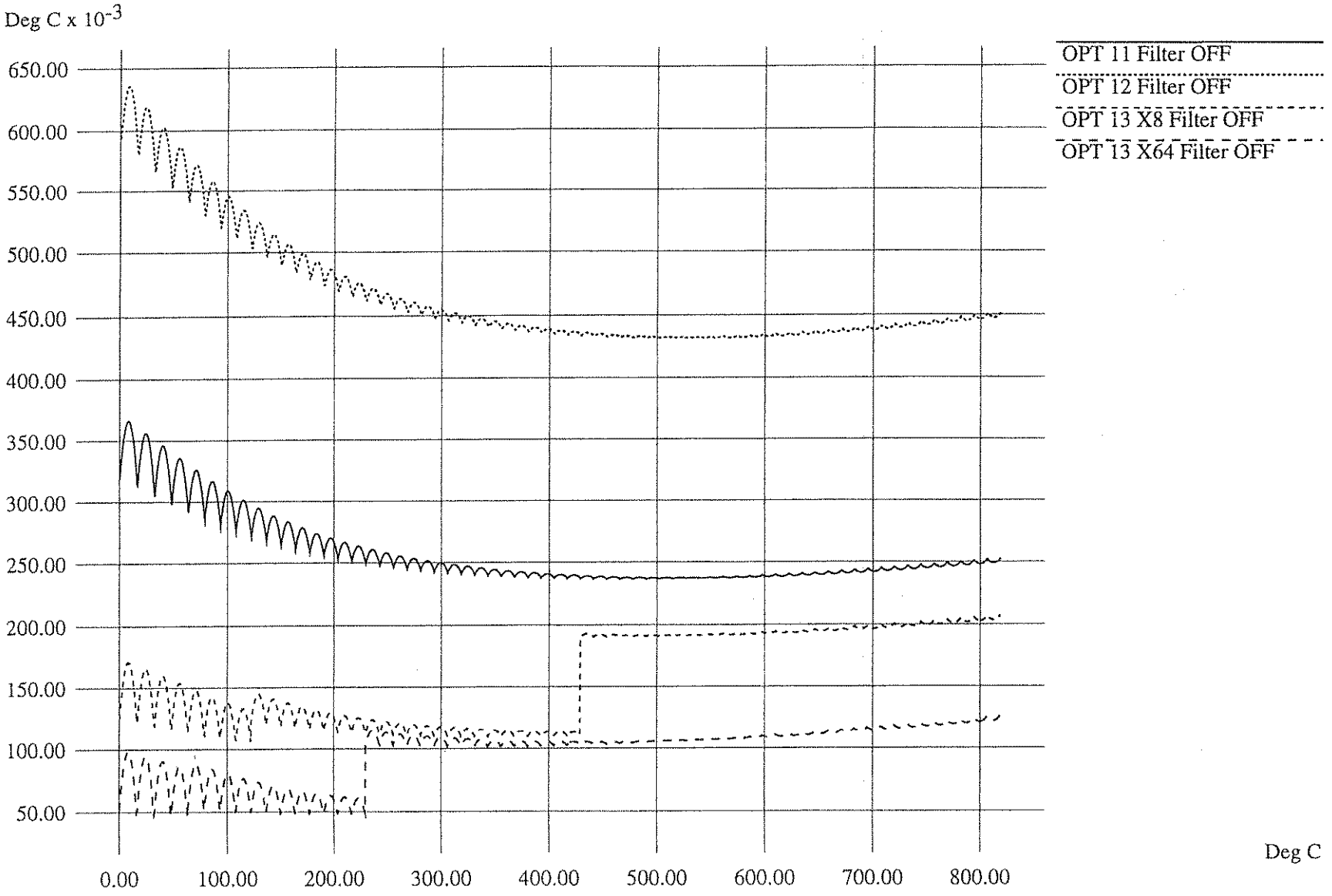
Type E


Deg C
9.50
9.00
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
-200.00
0.00
200.00
400.00
600.00
800.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF

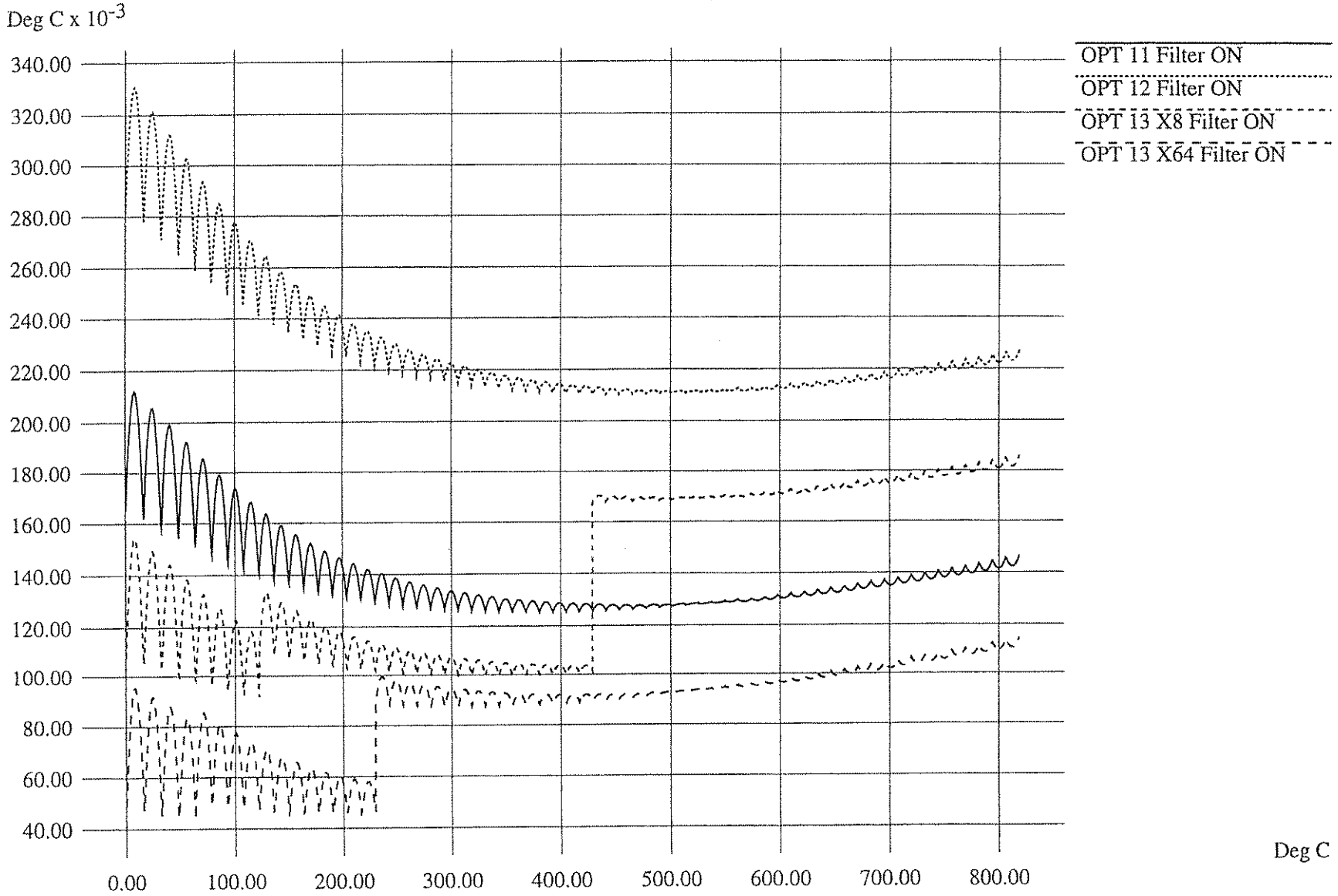
Type E


Deg C
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
-200.00
0.00
200.00
400.00
600.00
800.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

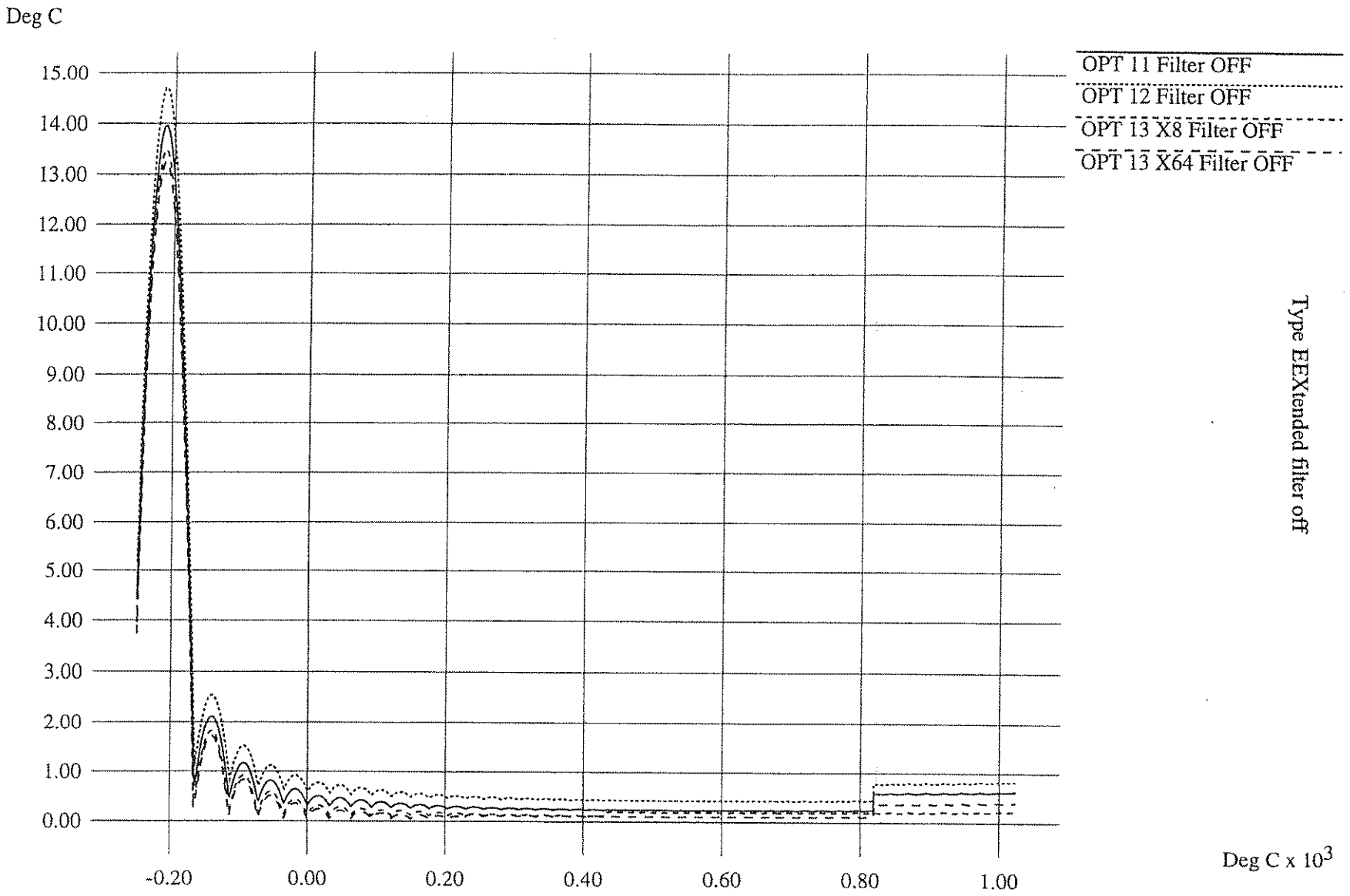
Type E


Deg C x 10^{-3}
650.00
600.00
550.00
500.00
450.00
400.00
350.00
300.00
250.00
200.00
150.00
100.00
50.00
0.00
100.00
200.00
300.00
400.00
500.00
600.00
700.00
800.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF

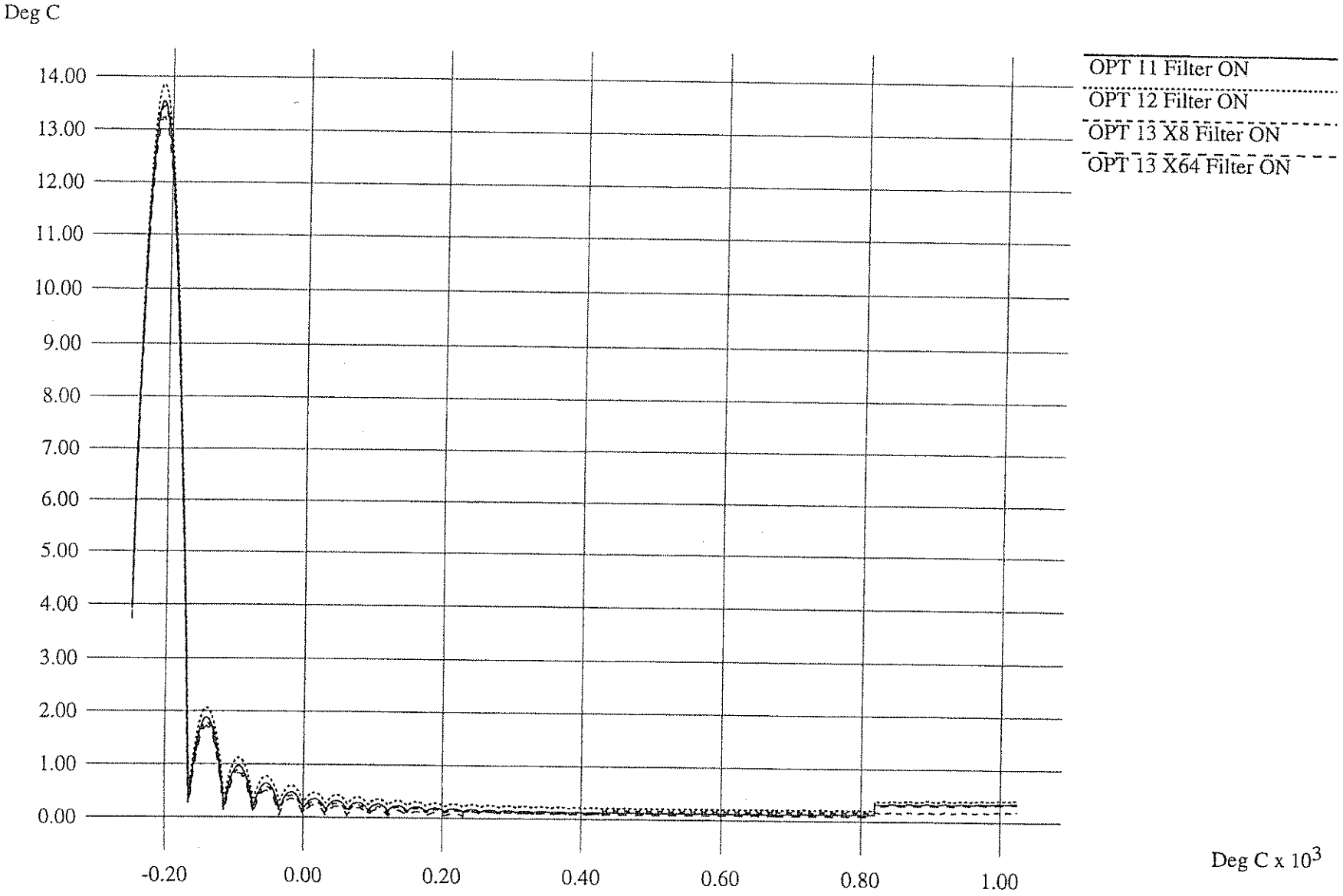
Type E


Deg C x 10^{-3}
340.00
320.00
300.00
280.00
260.00
240.00
220.00
200.00
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
0.00
100.00
200.00
300.00
400.00
500.00
600.00
700.00
800.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

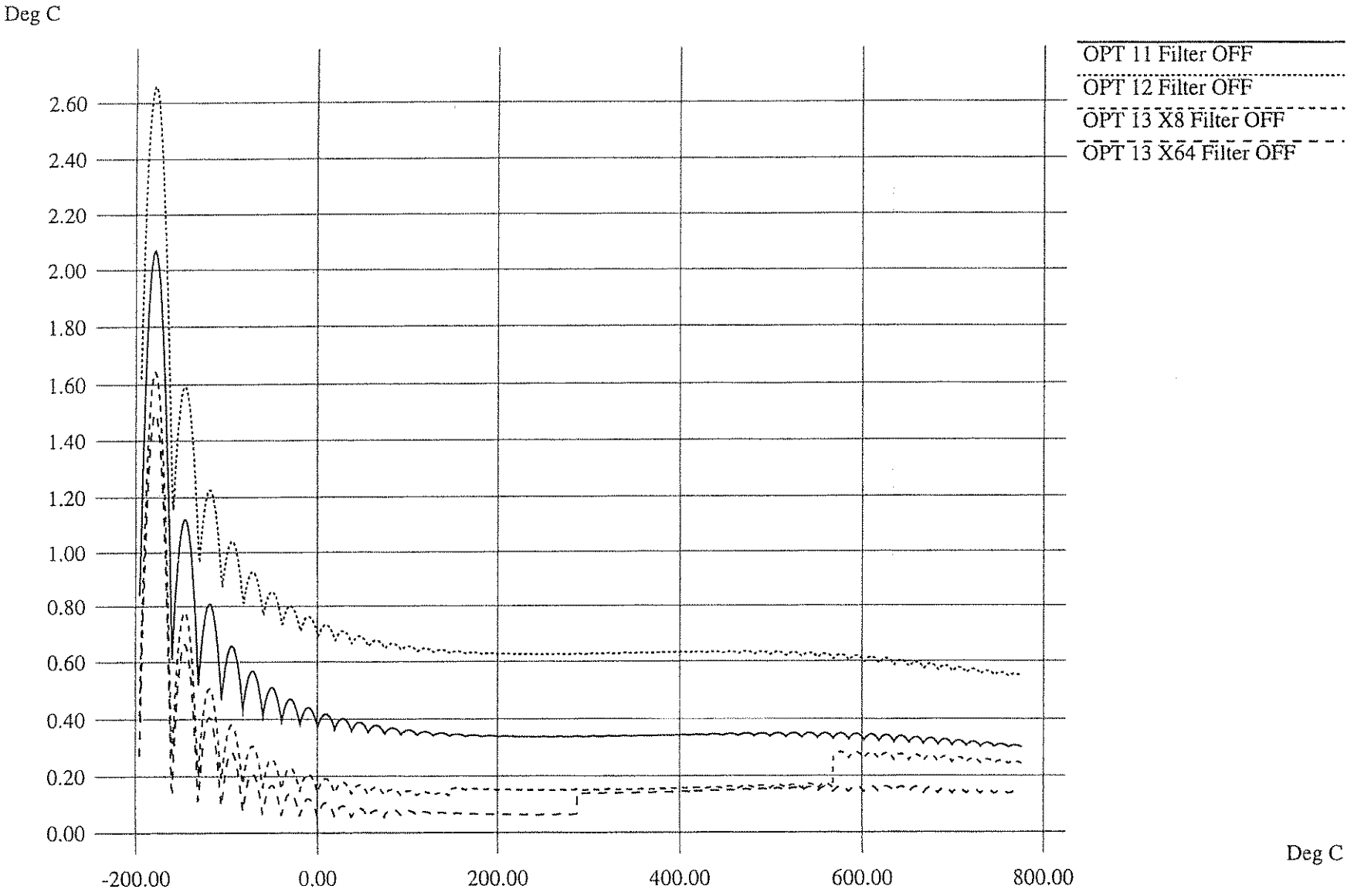
Type E Extended



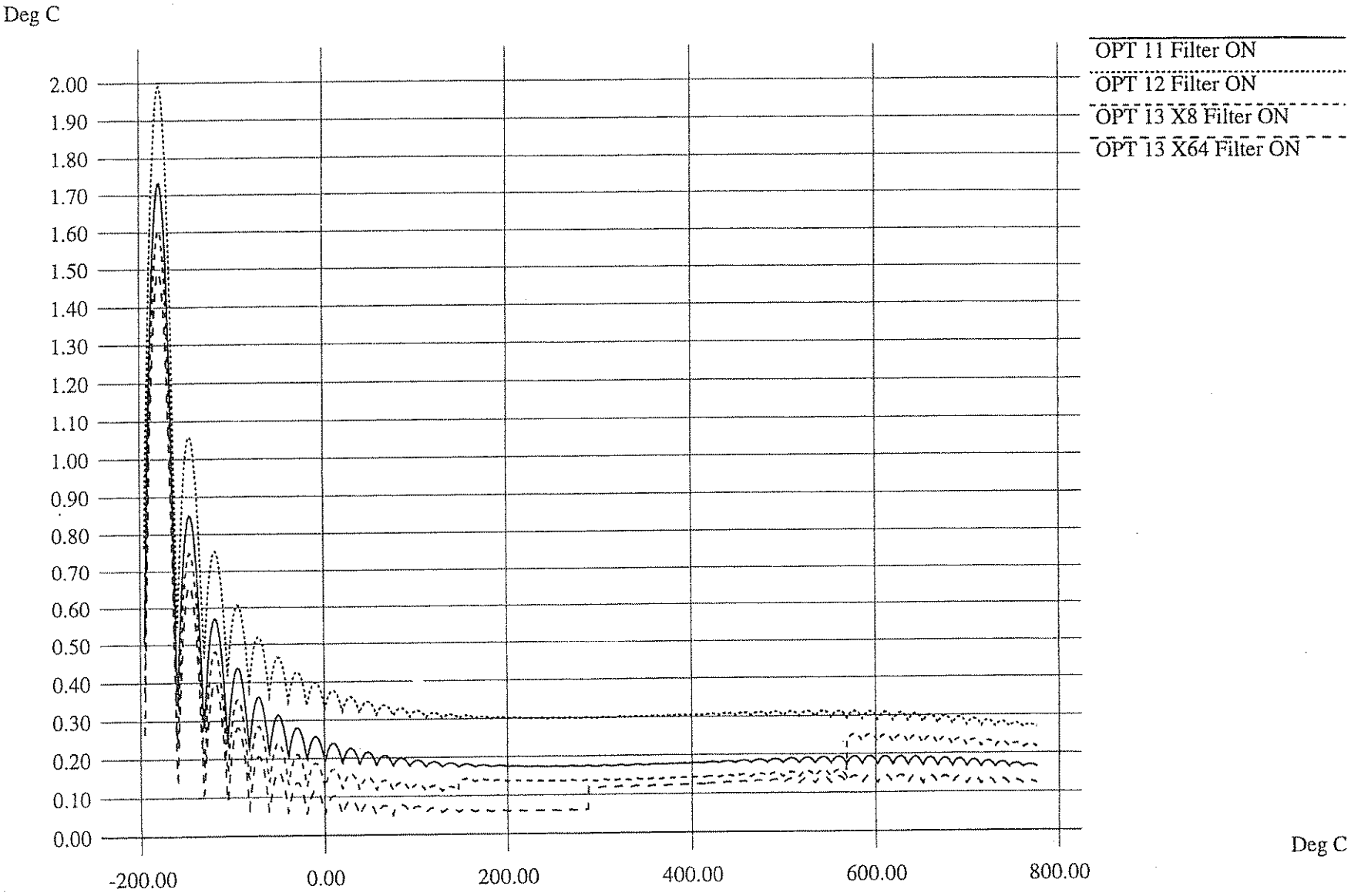
Type E Extended



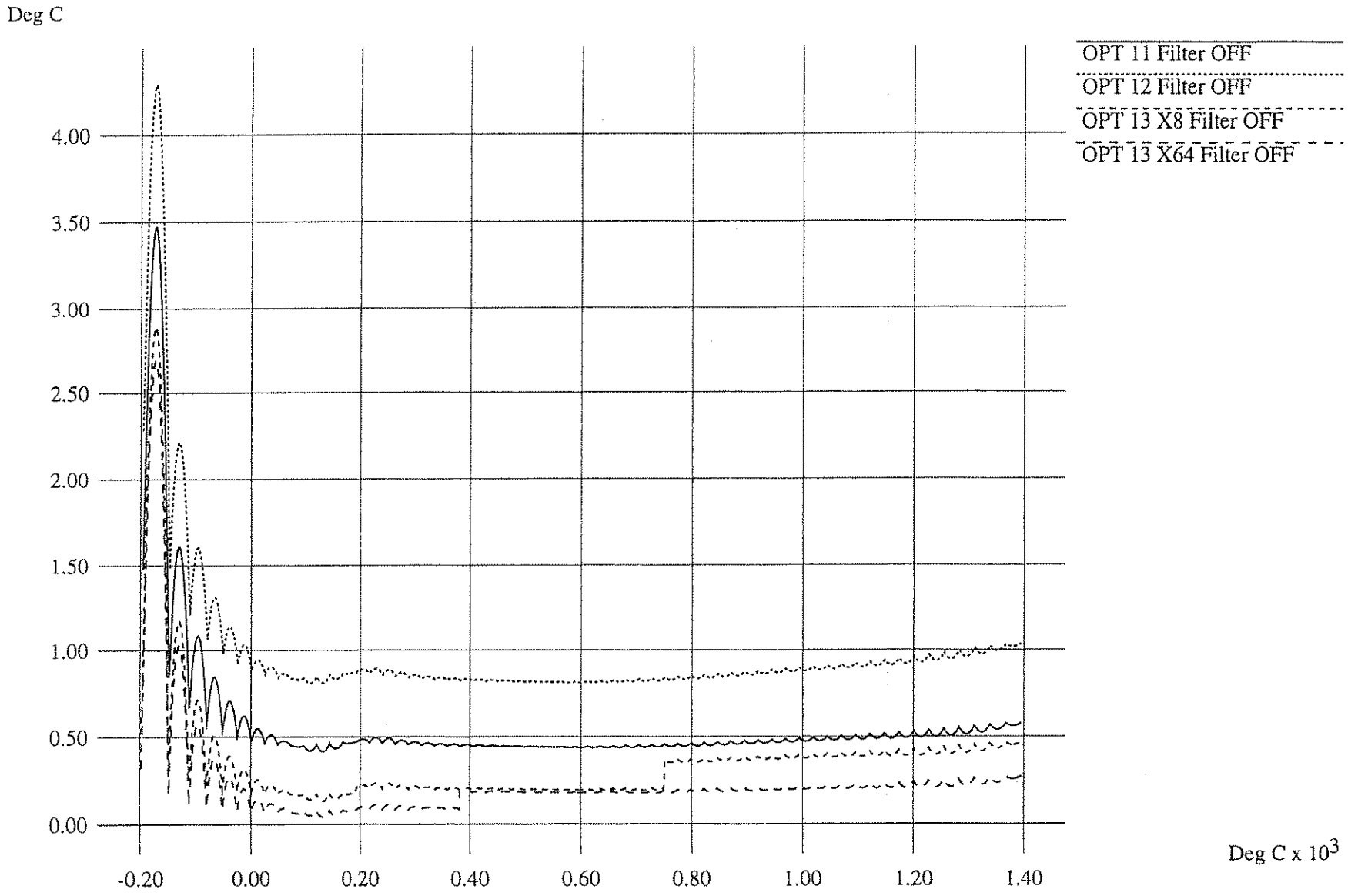
Type J


Deg C
2.60
2.40
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
-200.00
0.00
200.00
400.00
600.00
800.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF

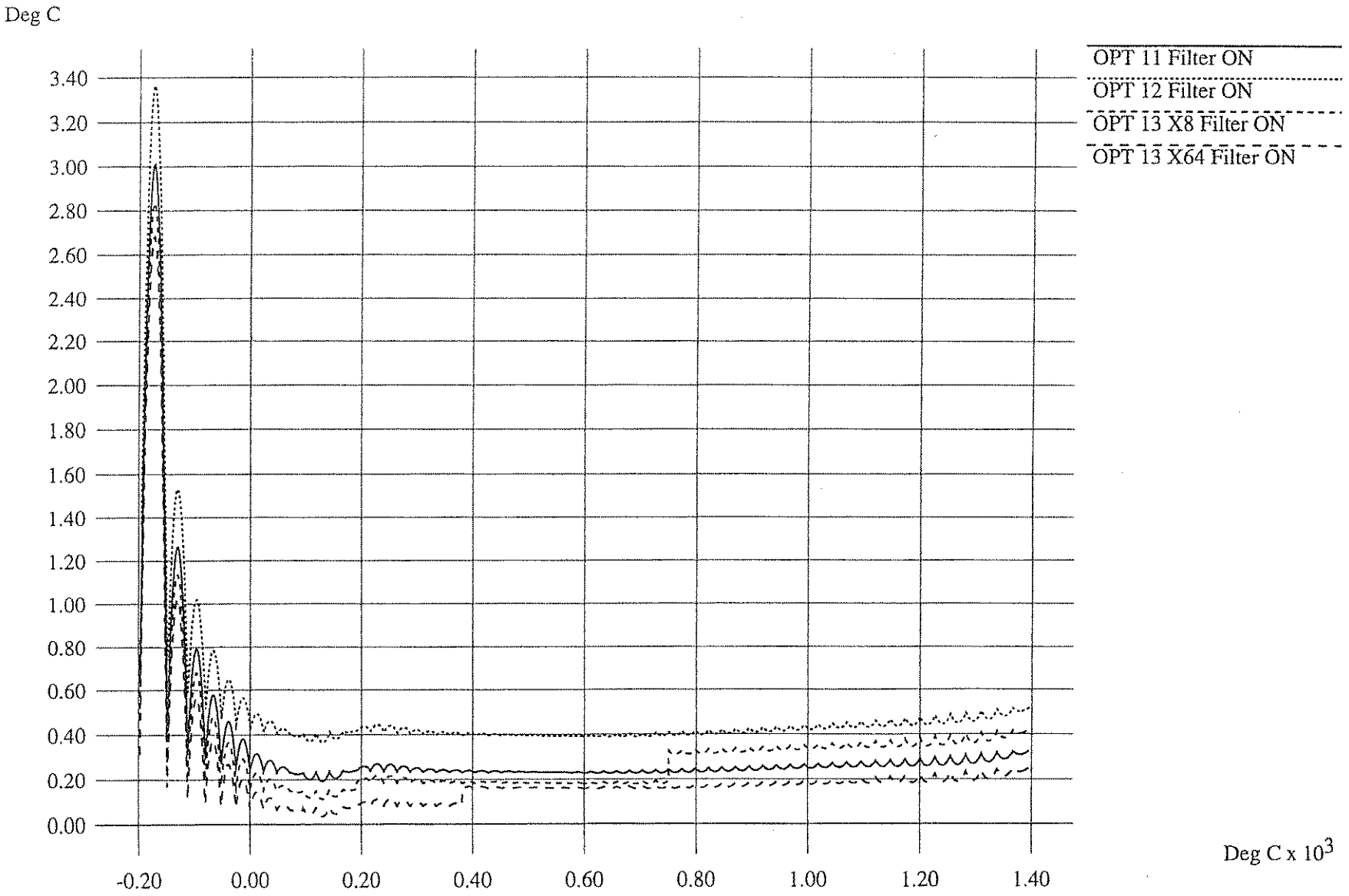
Type J


Deg C
2.00
1.90
1.80
1.70
1.60
1.50
1.40
1.30
1.20
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
-200.00
0.00
200.00
400.00
600.00
800.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

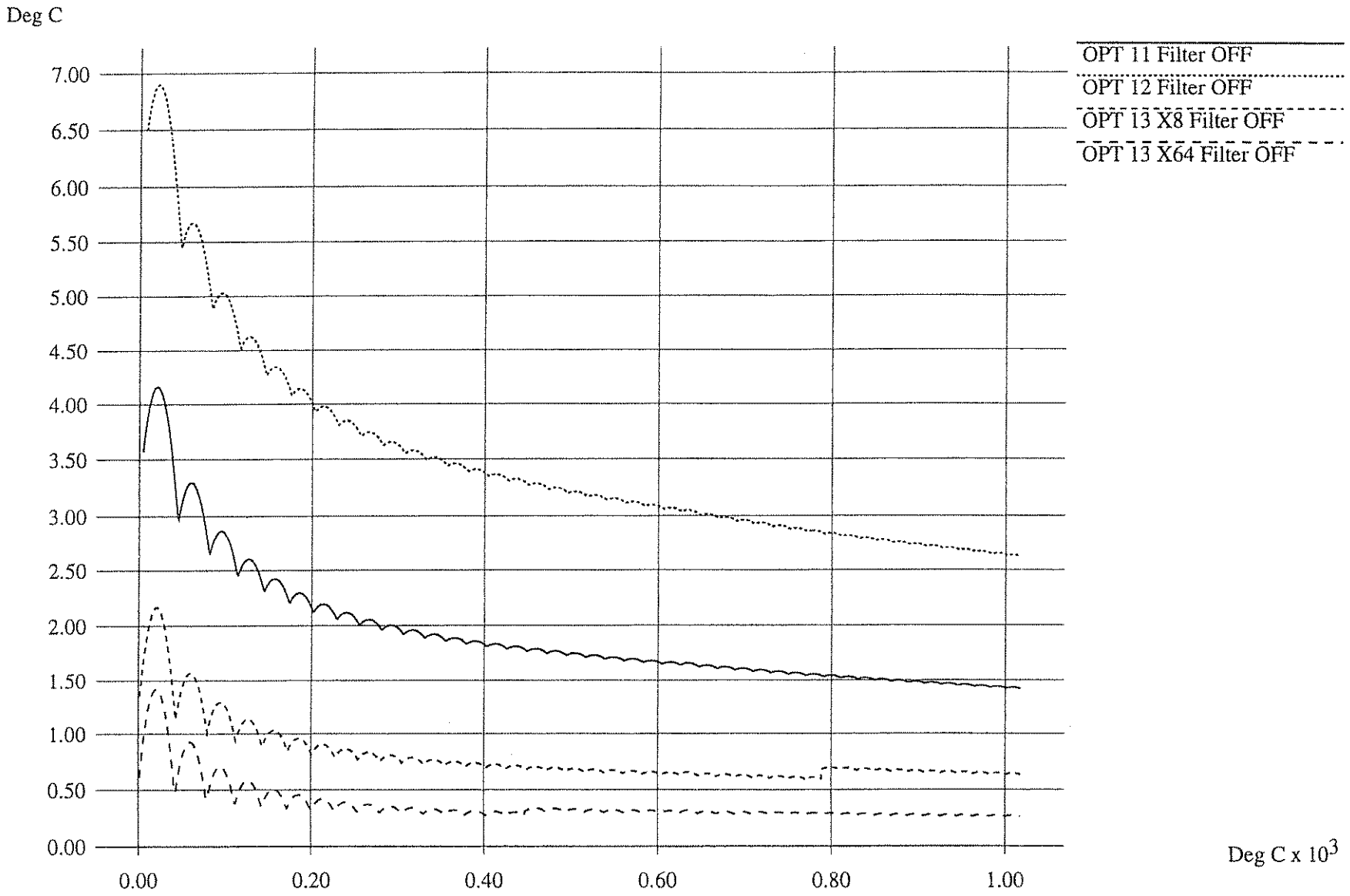
Type K


Deg C
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
-0.20
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF
Deg C x 10^3

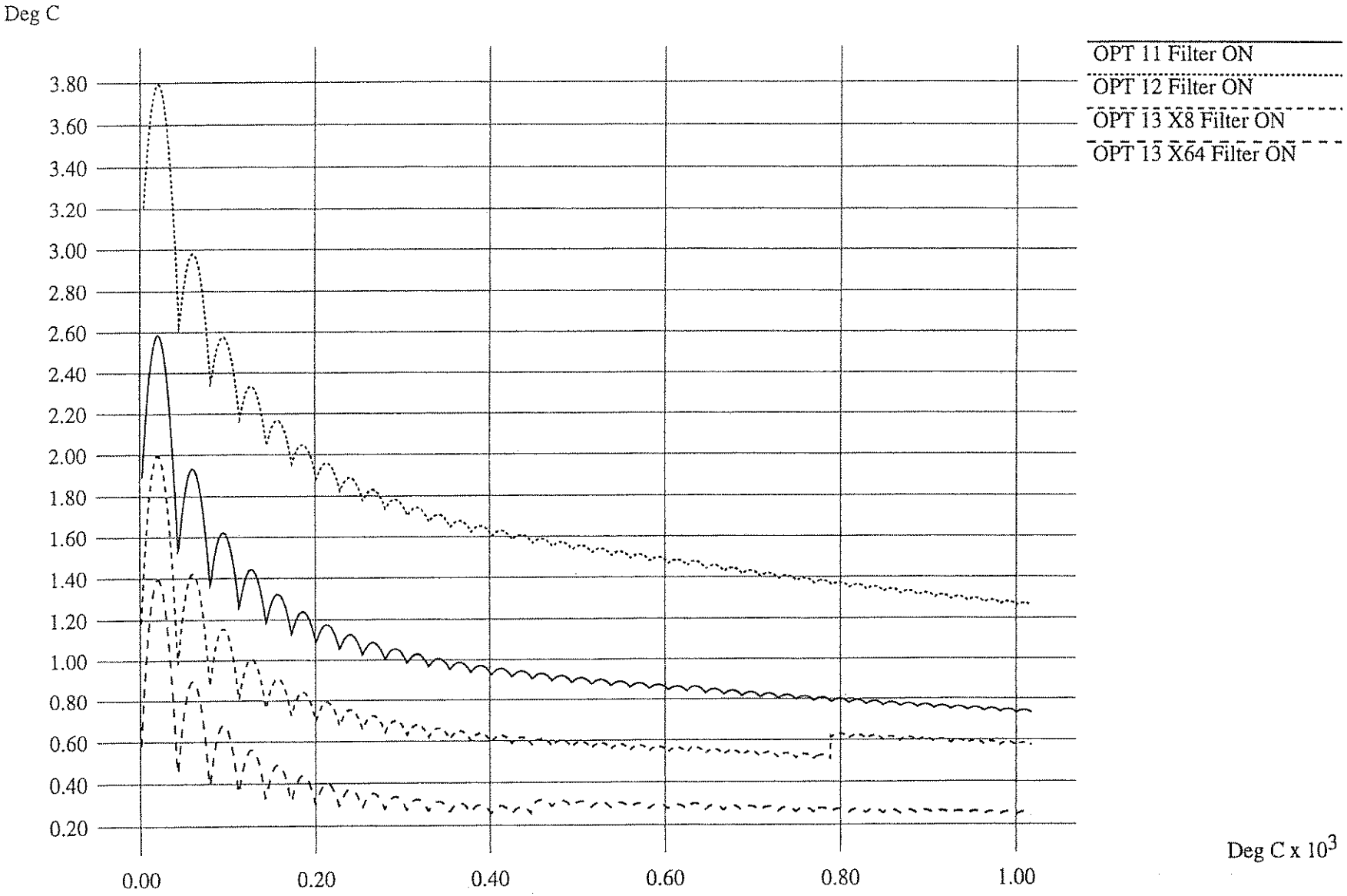
Type K


Deg C
3.40
3.20
3.00
2.80
2.60
2.40
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
-0.20
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON
Deg C x 10^3

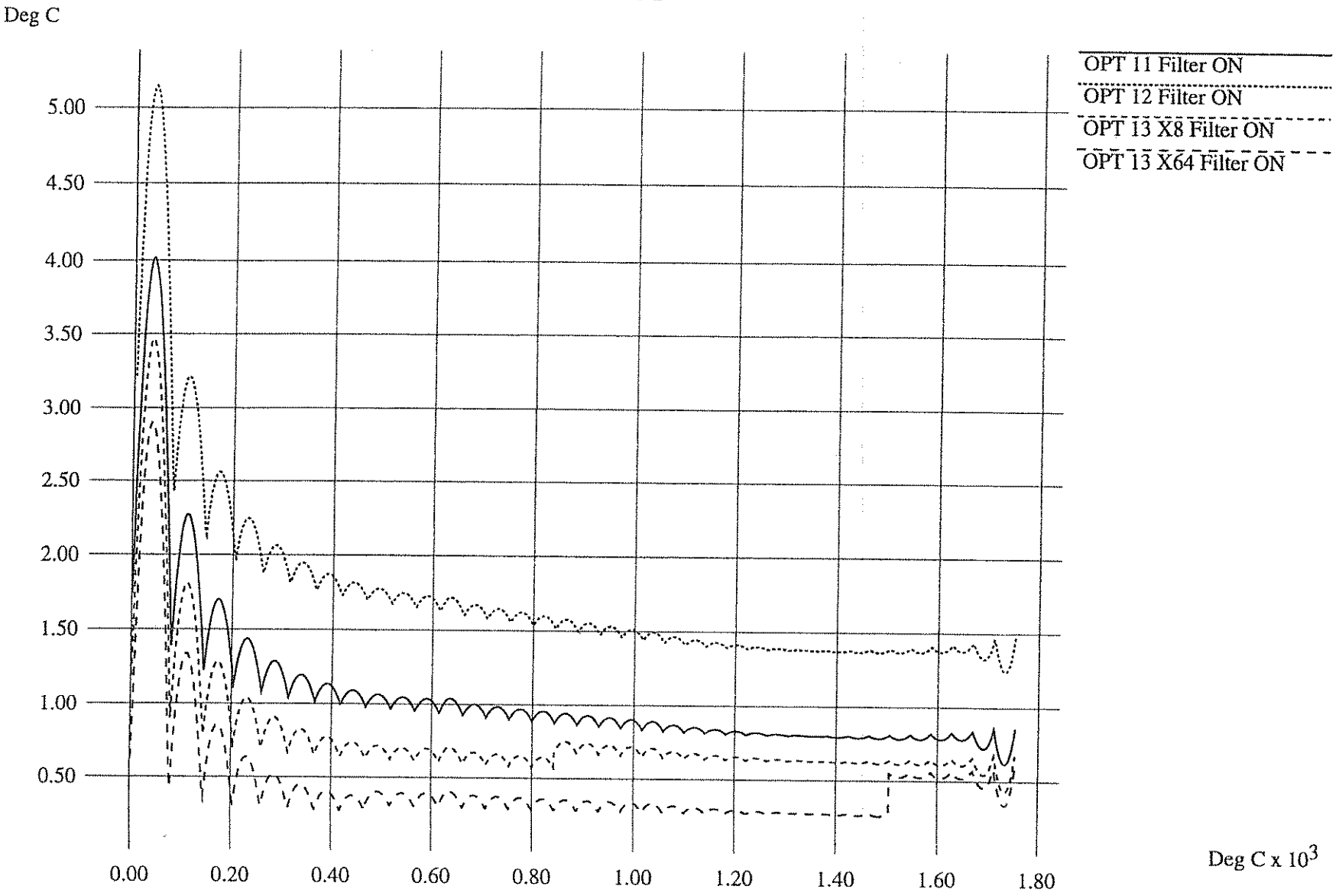
Type R


Deg C
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.00
0.20
0.40
0.60
0.80
1.00
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF
Deg C x 10^3

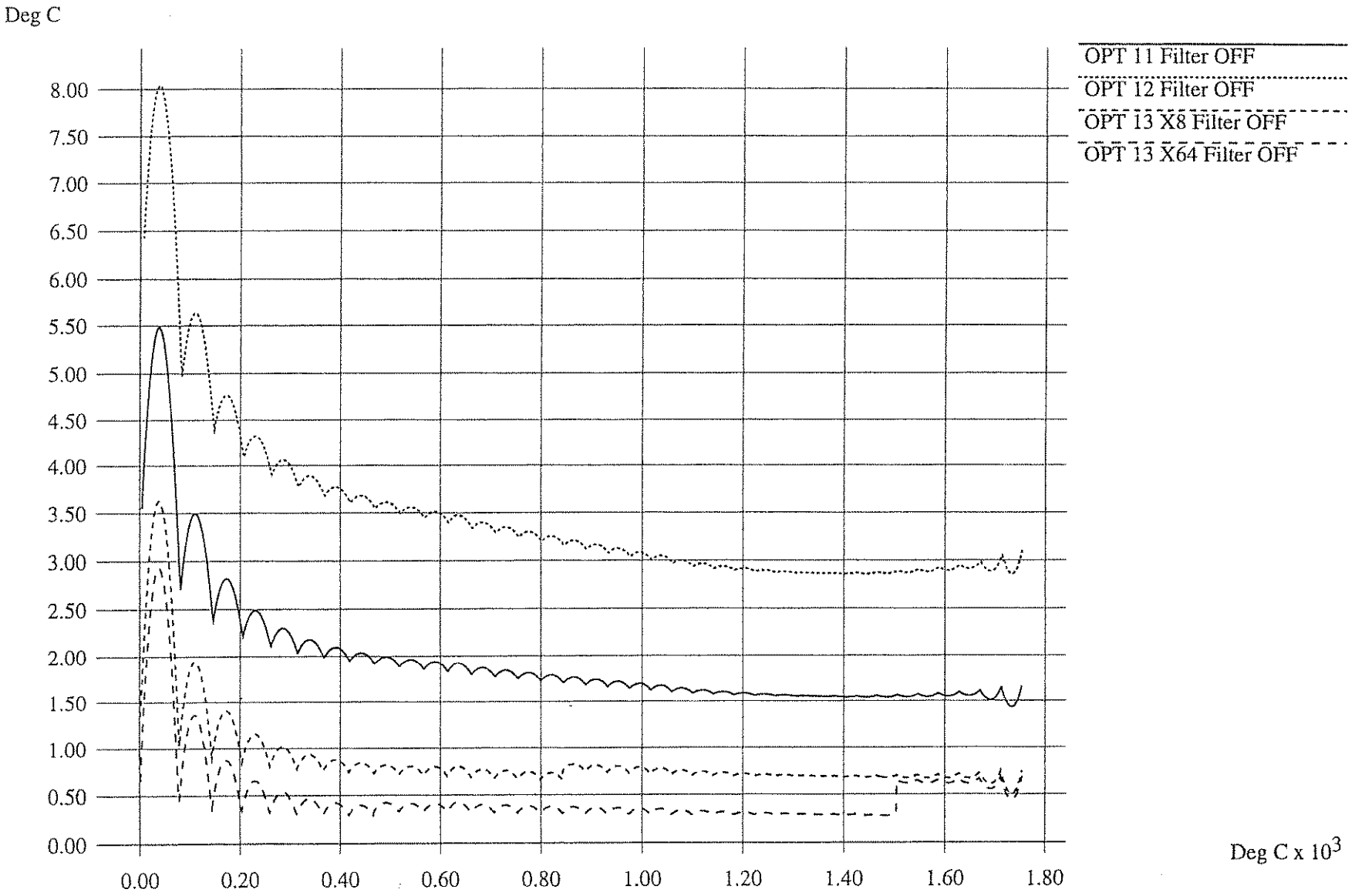
Type R


Deg C
3.80
3.60
3.40
3.20
3.00
2.80
2.60
2.40
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.20
0.40
0.60
0.80
1.00
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON
Deg C x 10^3

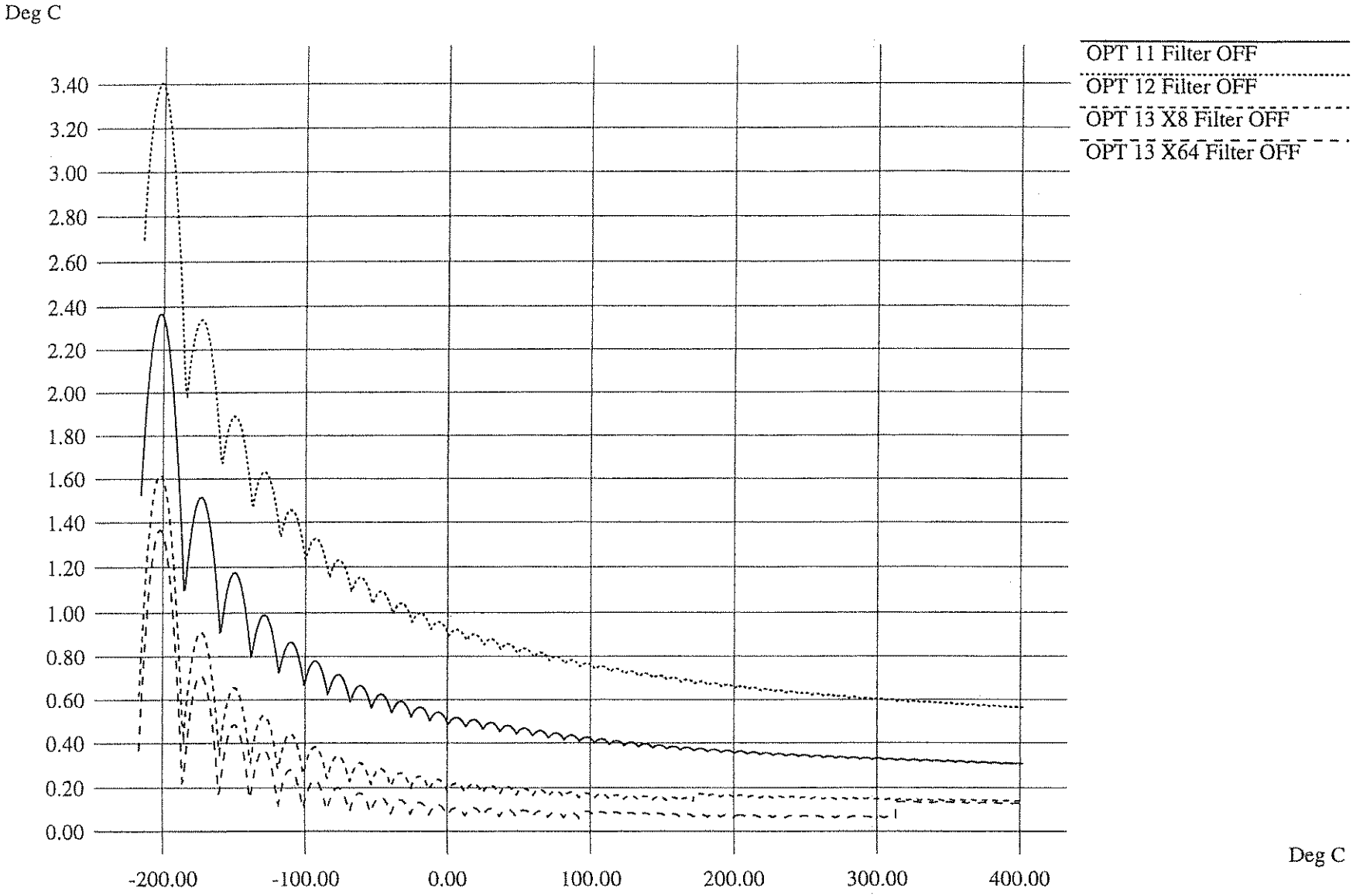
Type S

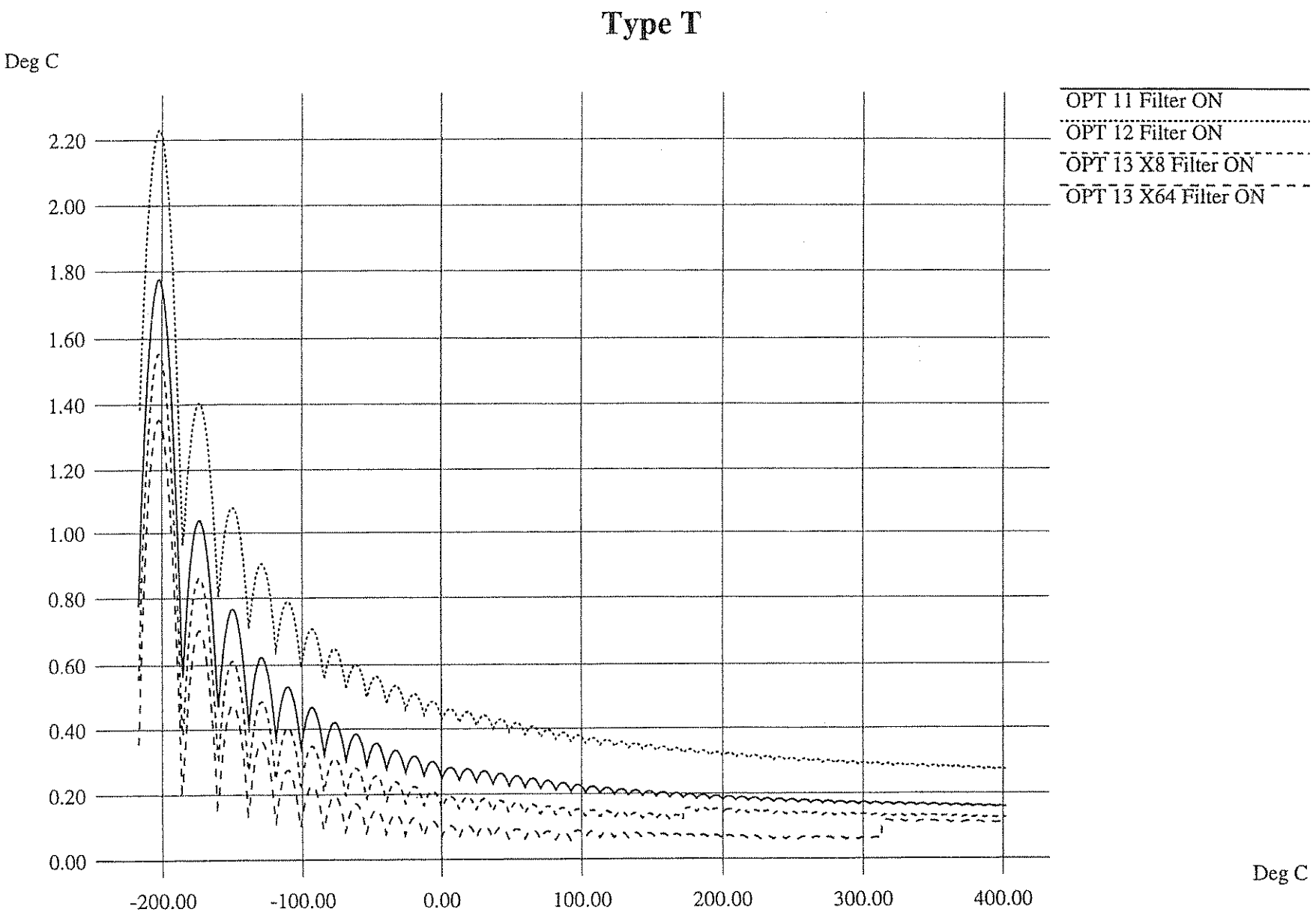

Deg C
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
1.60
1.80
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON
Deg C x 10^3

Type S

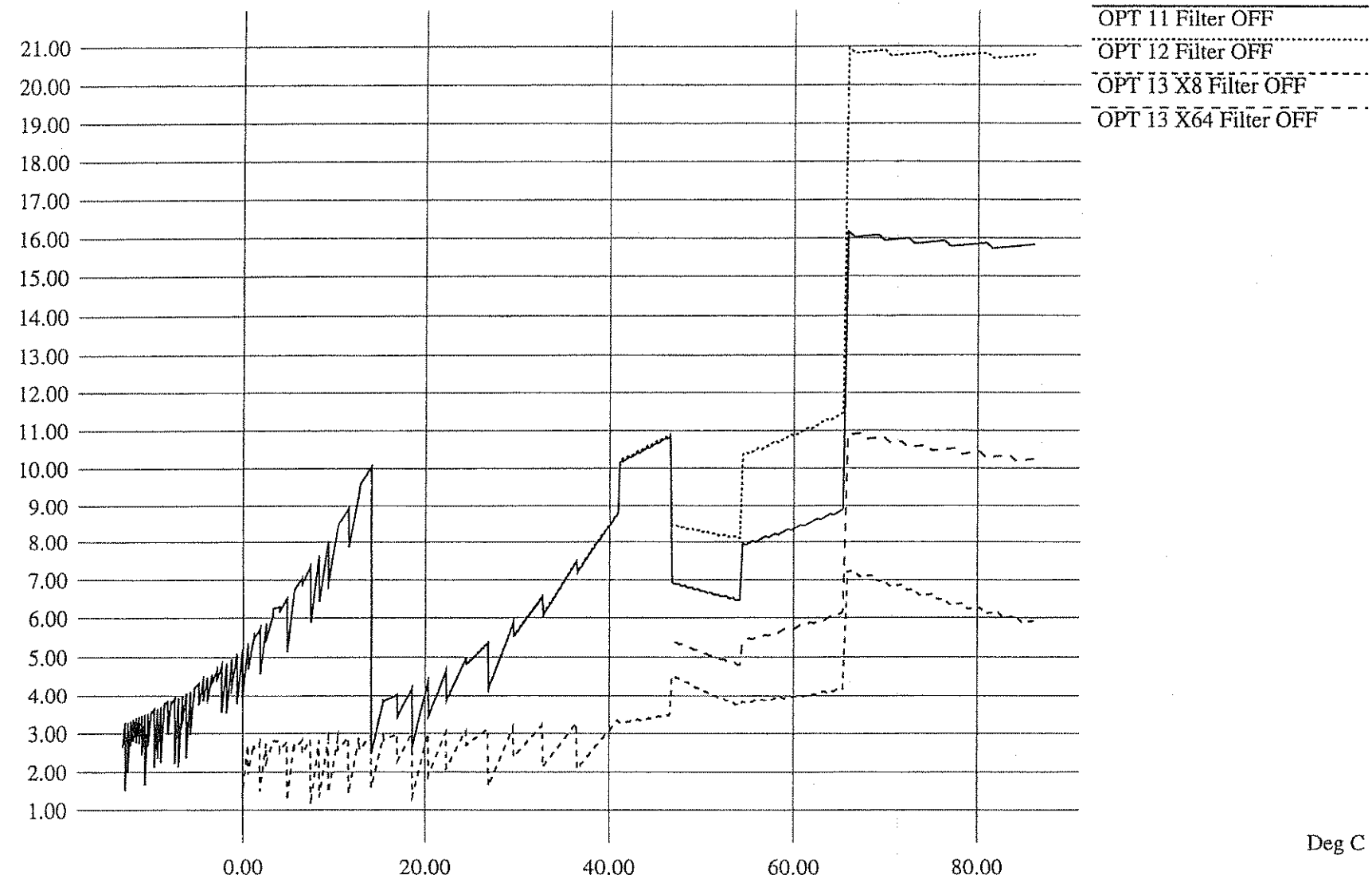

Deg C
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
1.60
1.80
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF
Deg C x 10^3

Type T

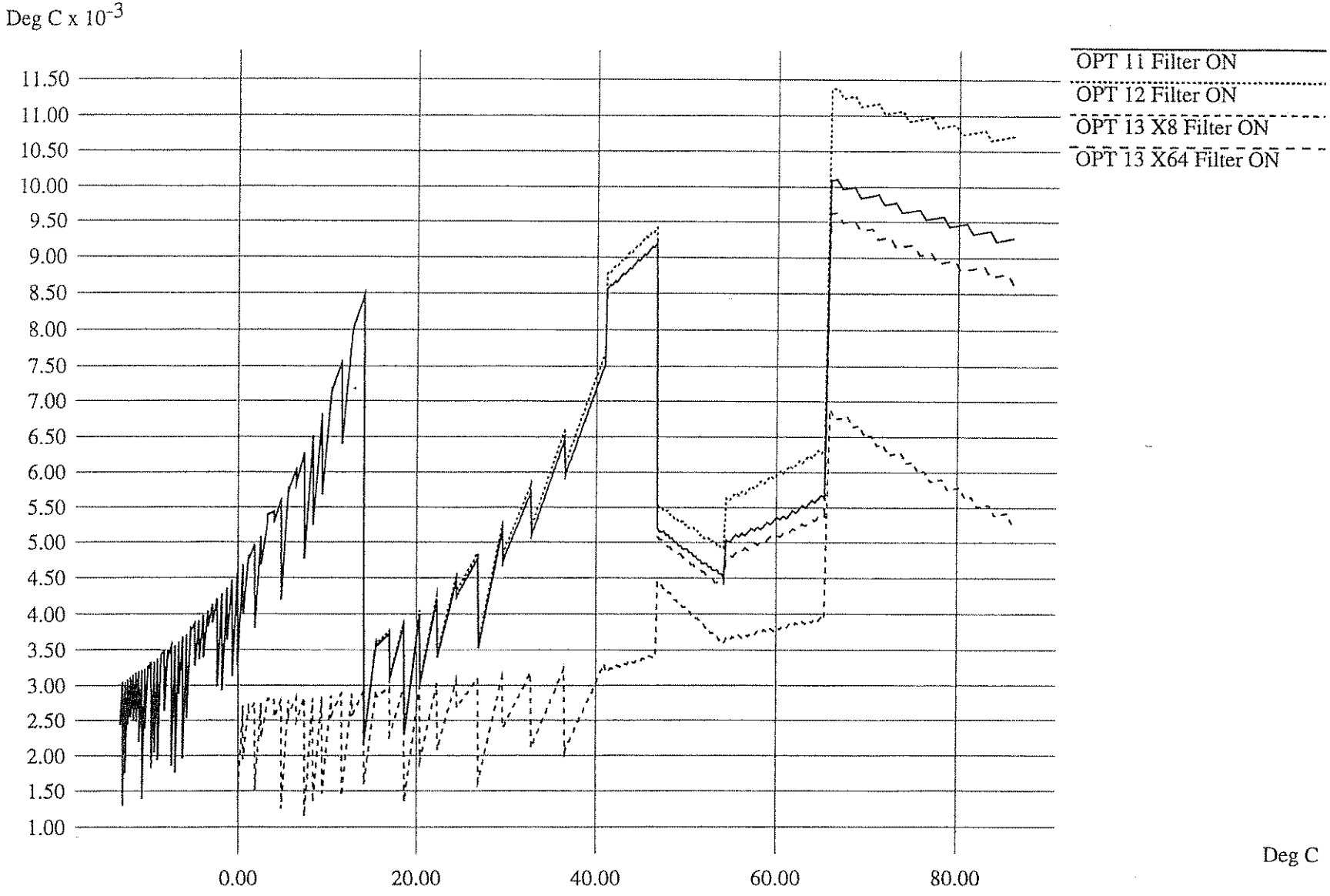

Deg C
3.40
3.20
3.00
2.80
2.60
2.40
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
-200.00
-100.00
0.00
100.00
200.00
300.00
400.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF


Type T
Deg C
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
-200.00
-100.00
0.00
100.00
200.00
300.00
400.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

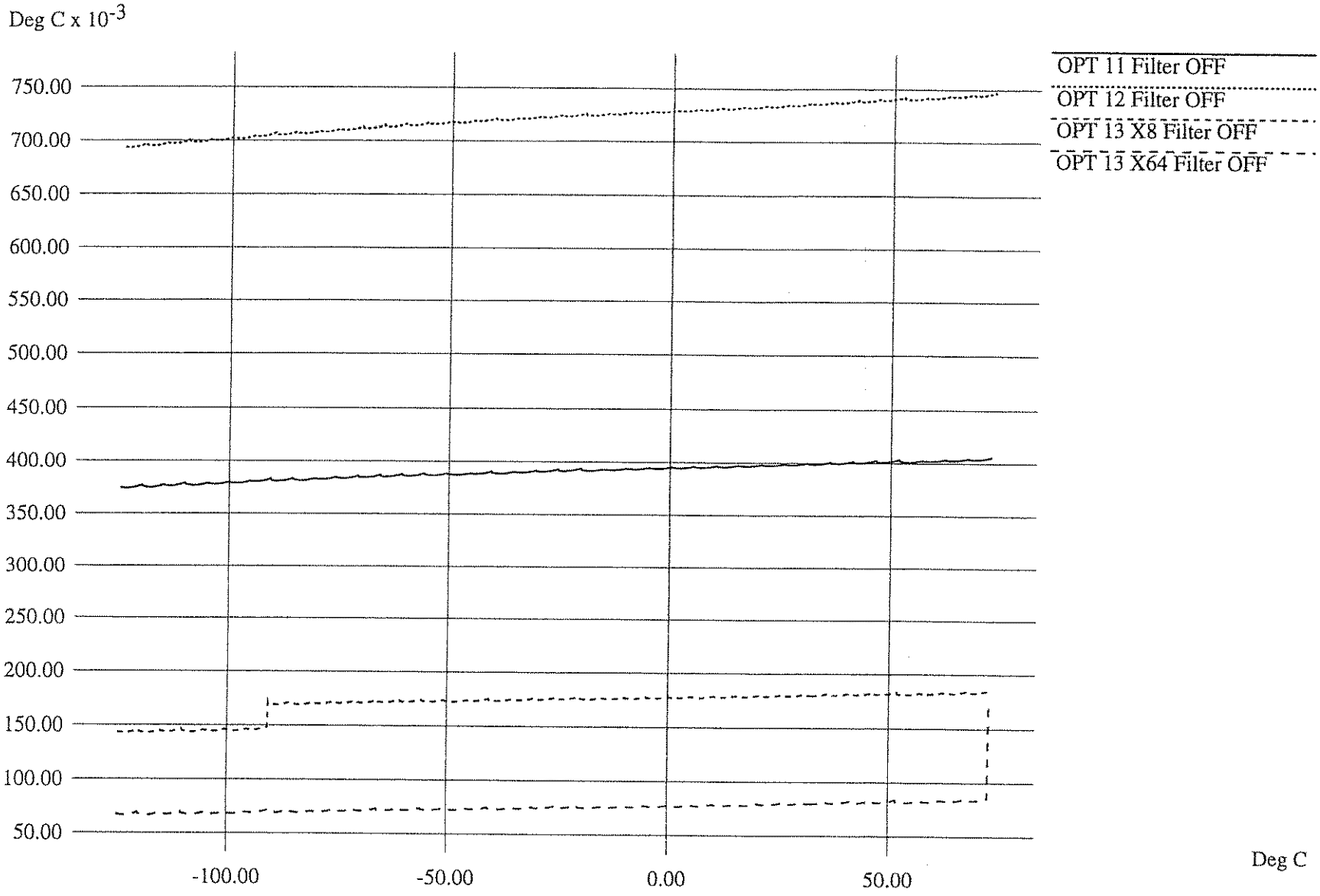
5K Therm REF


Deg C x 10^{-3}
21.00
20.00
19.00
18.00
17.00
16.00
15.00
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
20.00
40.00
60.00
80.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF

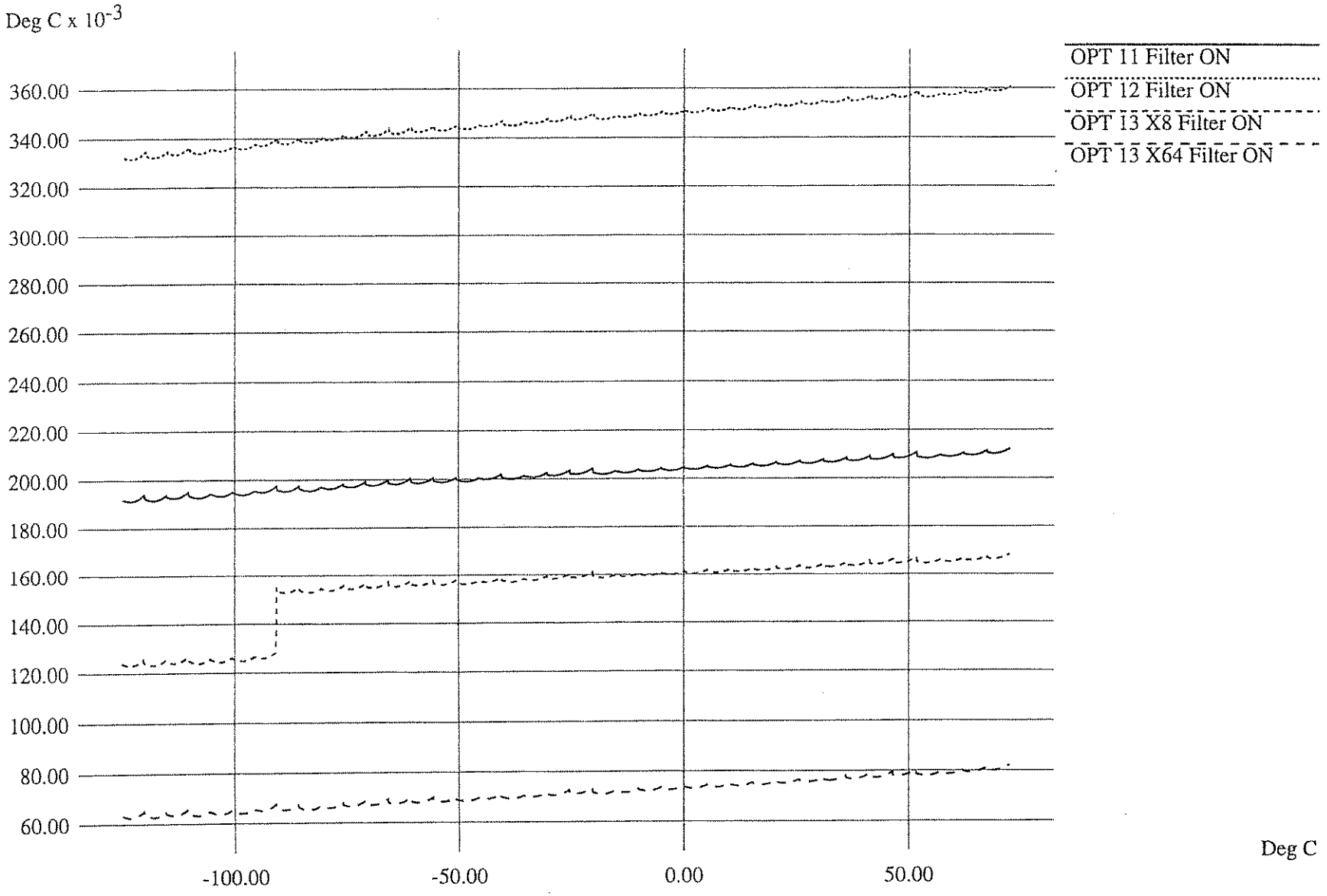
5K Therm REF


Deg C x 10^{-3}
11.50
11.00
10.50
10.00
9.50
9.00
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.00
20.00
40.00
60.00
80.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

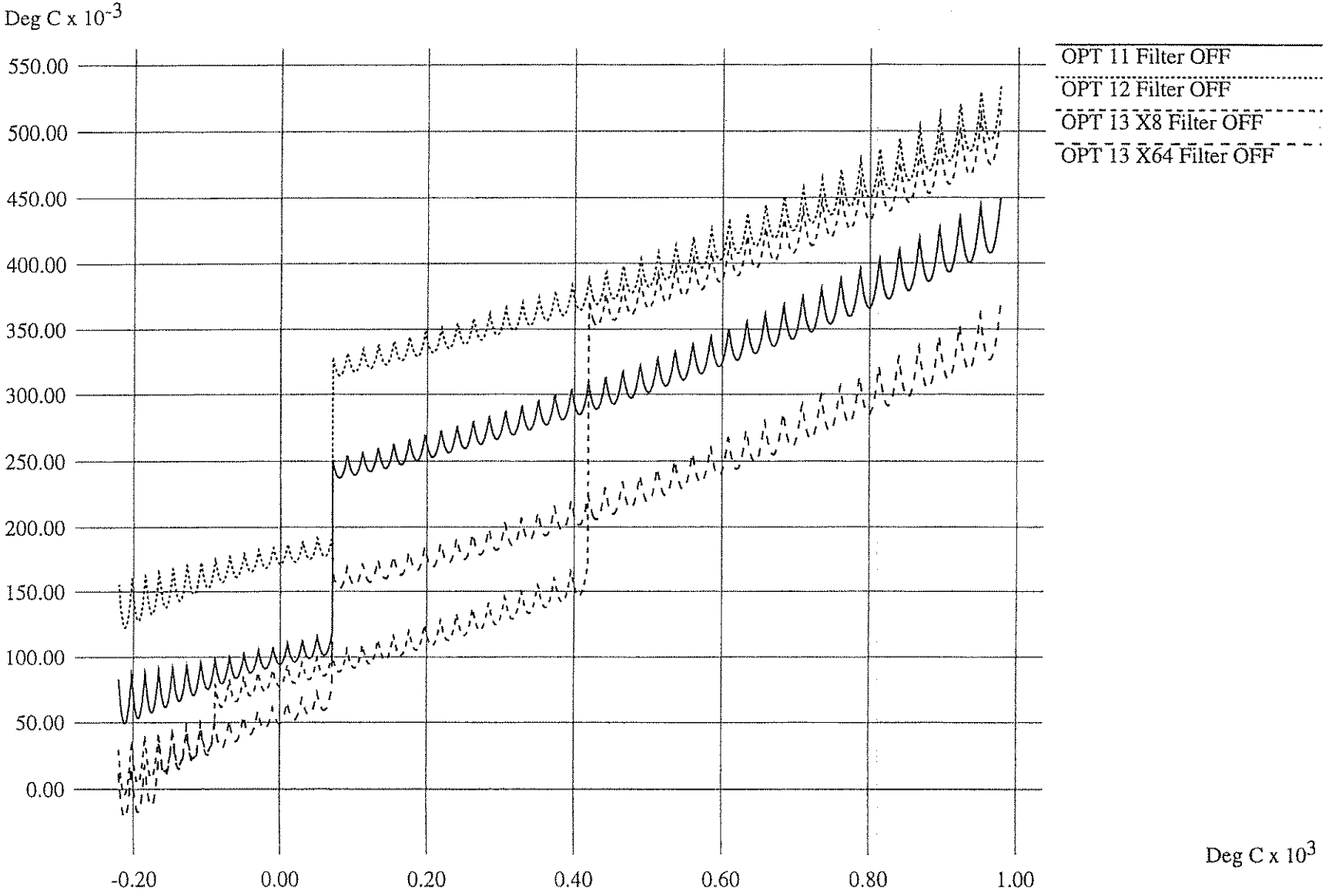
RTD REF


Deg C x 10^{-3}
750.00
700.00
650.00
600.00
550.00
500.00
450.00
400.00
350.00
300.00
250.00
200.00
150.00
100.00
50.00
-100.00
-50.00
0.00
50.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF

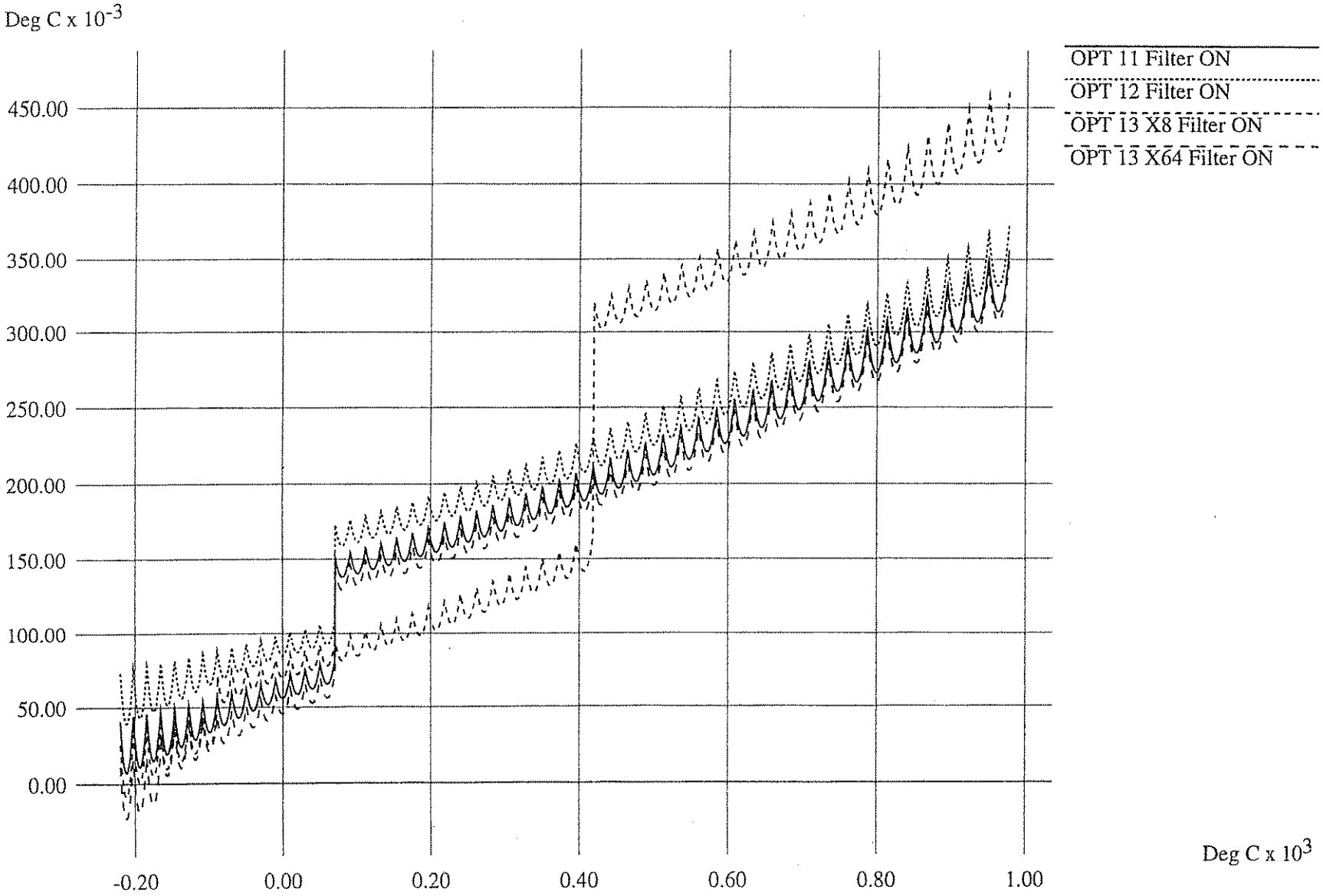
RTD REF


Deg C x 10^{-3}
360.00
340.00
320.00
300.00
280.00
260.00
240.00
220.00
200.00
180.00
160.00
140.00
120.00
100.00
80.00
60.00
-100.00
-50.00
0.00
50.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

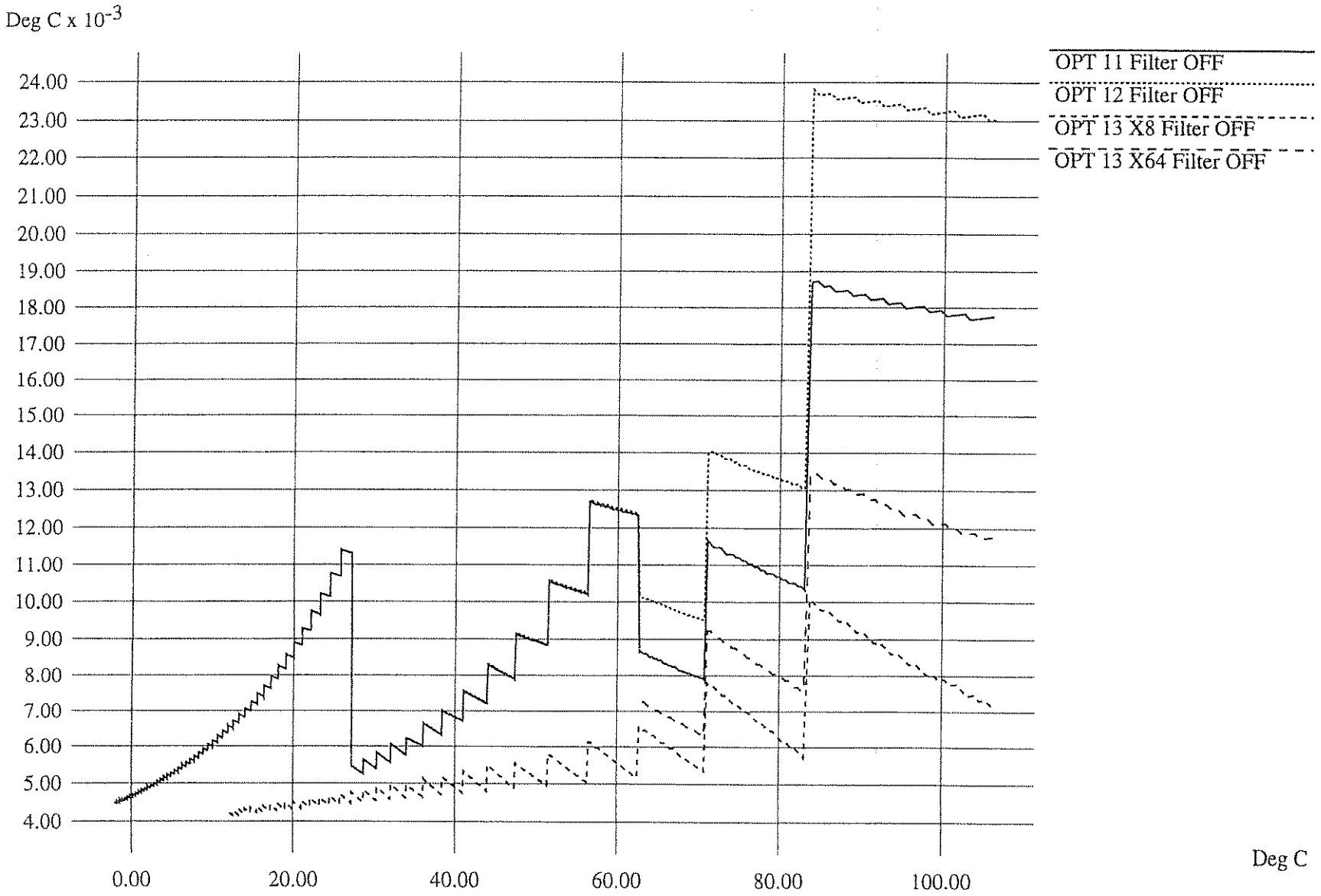
RTD

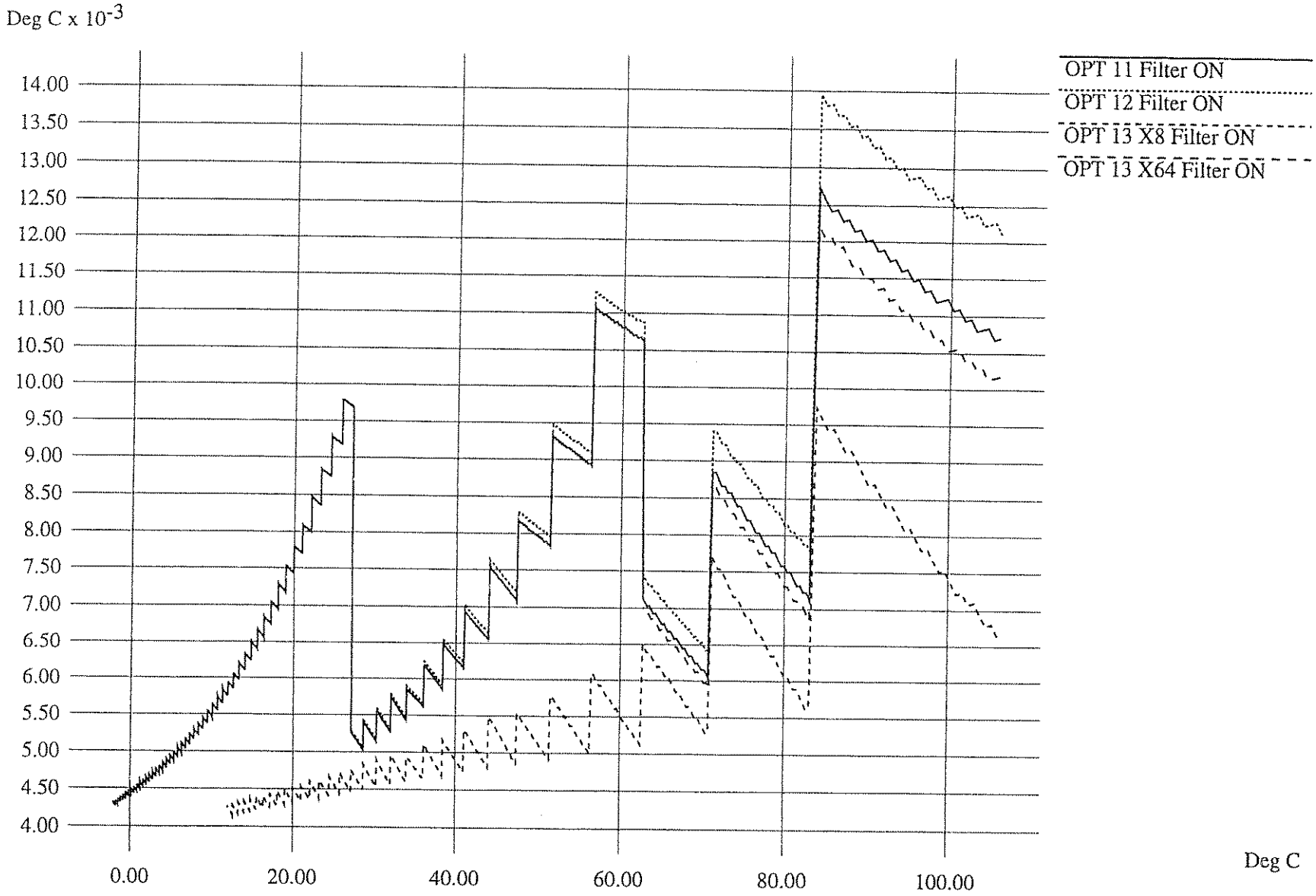

Deg C x 10^{-3}
550.00
500.00
450.00
400.00
350.00
300.00
250.00
200.00
150.00
100.00
50.00
0.00
-0.20
0.00
0.20
0.40
0.60
0.80
1.00
Deg C x 10^{3}
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF

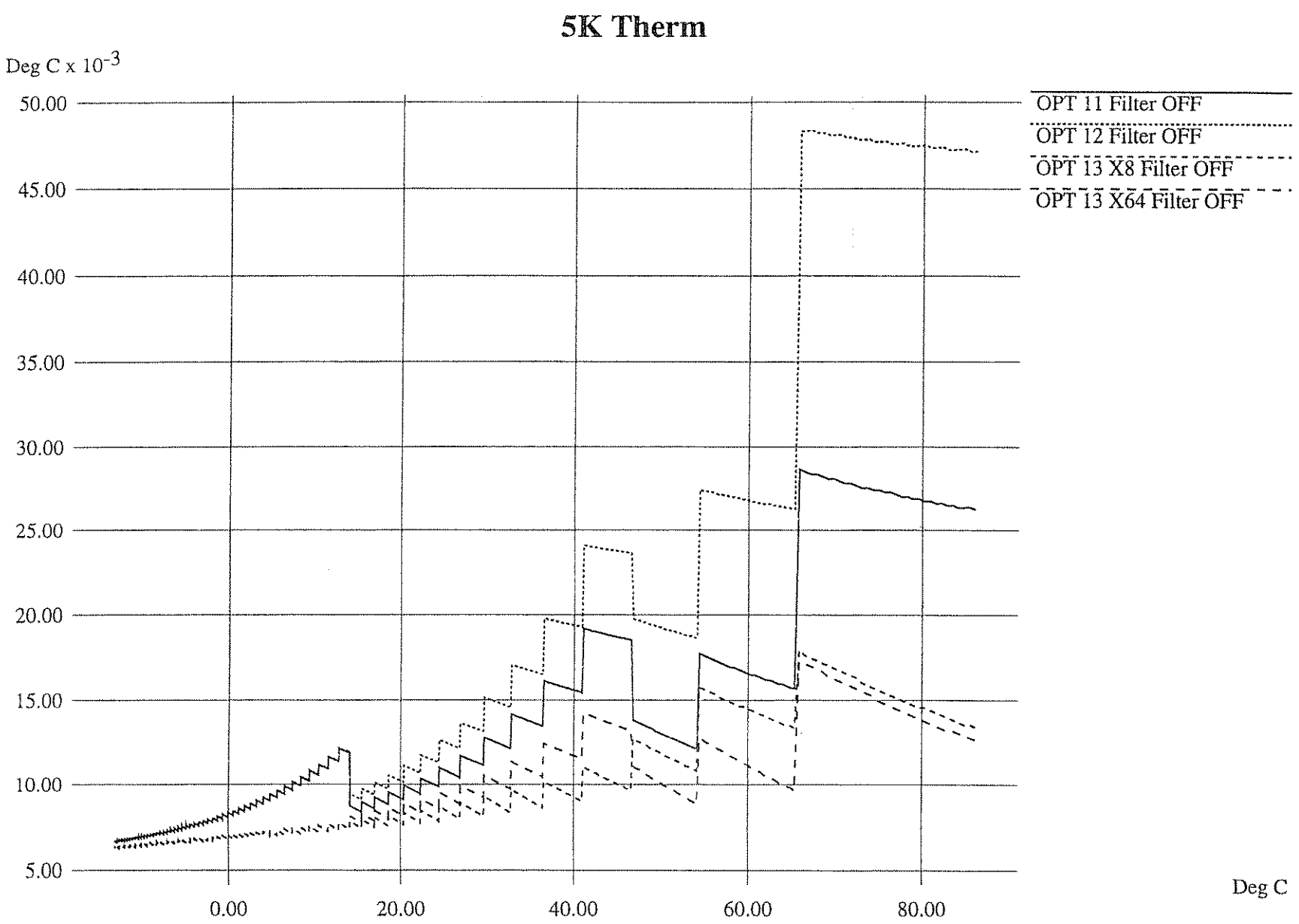
RTD

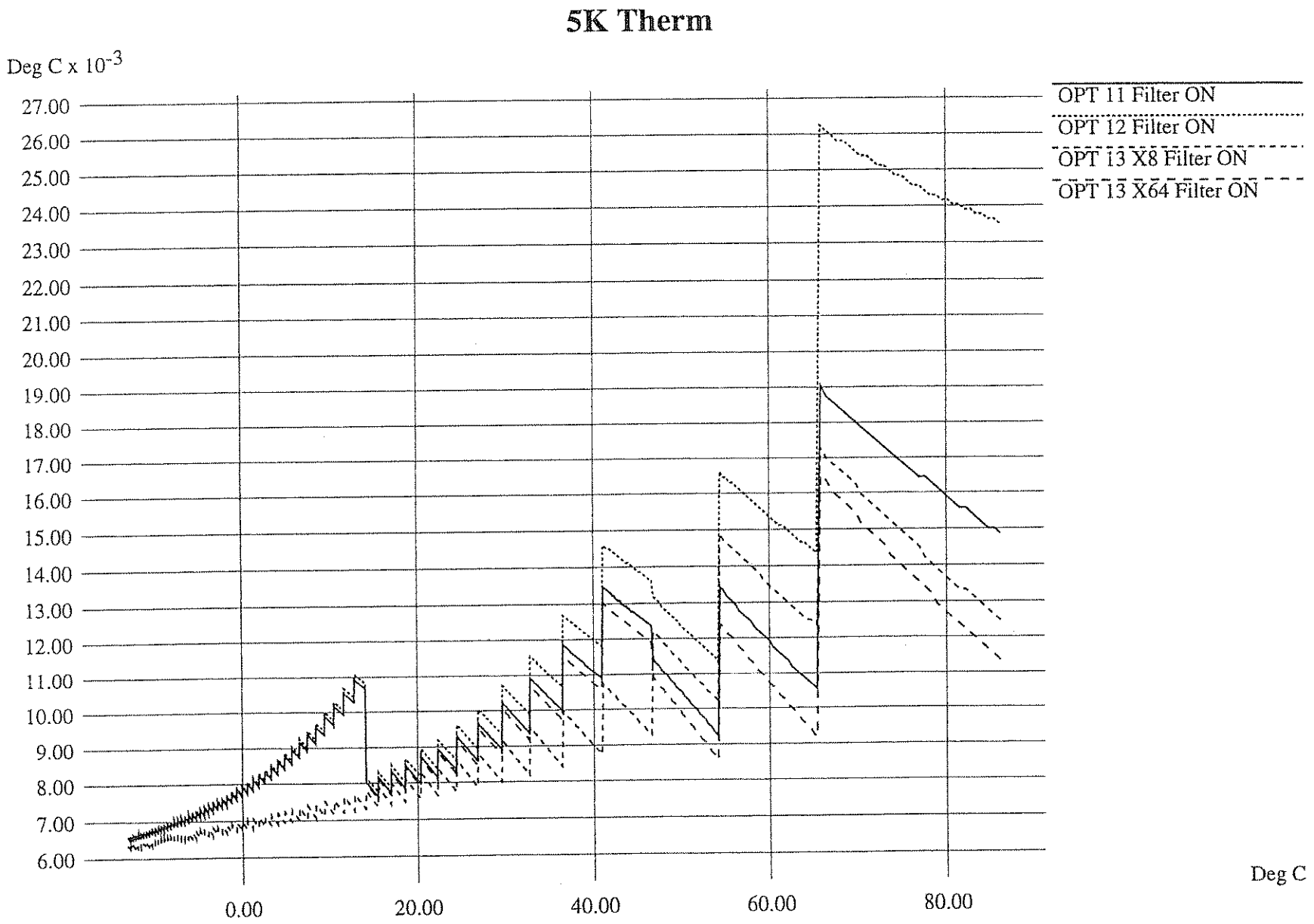

Deg C x 10^{-3}
450.00
400.00
350.00
300.00
250.00
200.00
150.00
100.00
50.00
0.00
-0.20
0.00
0.20
0.40
0.60
0.80
1.00
Deg C x 10^{3}
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

2252 Therm

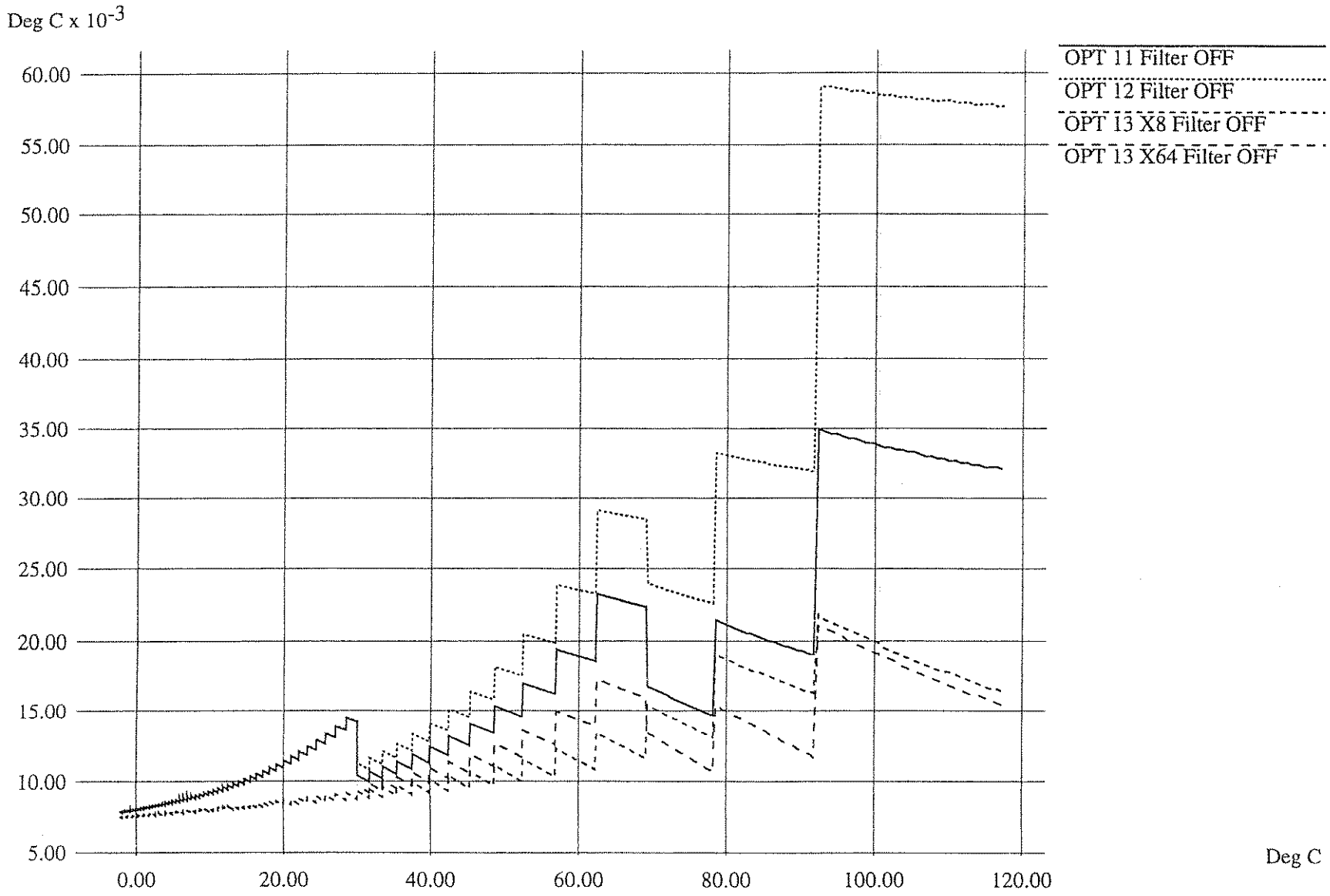

Deg C x 10^{-3}
24.00
23.00
22.00
21.00
20.00
19.00
18.00
17.00
16.00
15.00
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
5.00
4.00
0.00
20.00
40.00
60.00
80.00
100.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF


2252 Therm
Deg C x 10-3
14.00
13.50
13.00
12.50
12.00
11.50
11.00
10.50
10.00
9.50
9.00
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
0.00
20.00
40.00
60.00
80.00
100.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

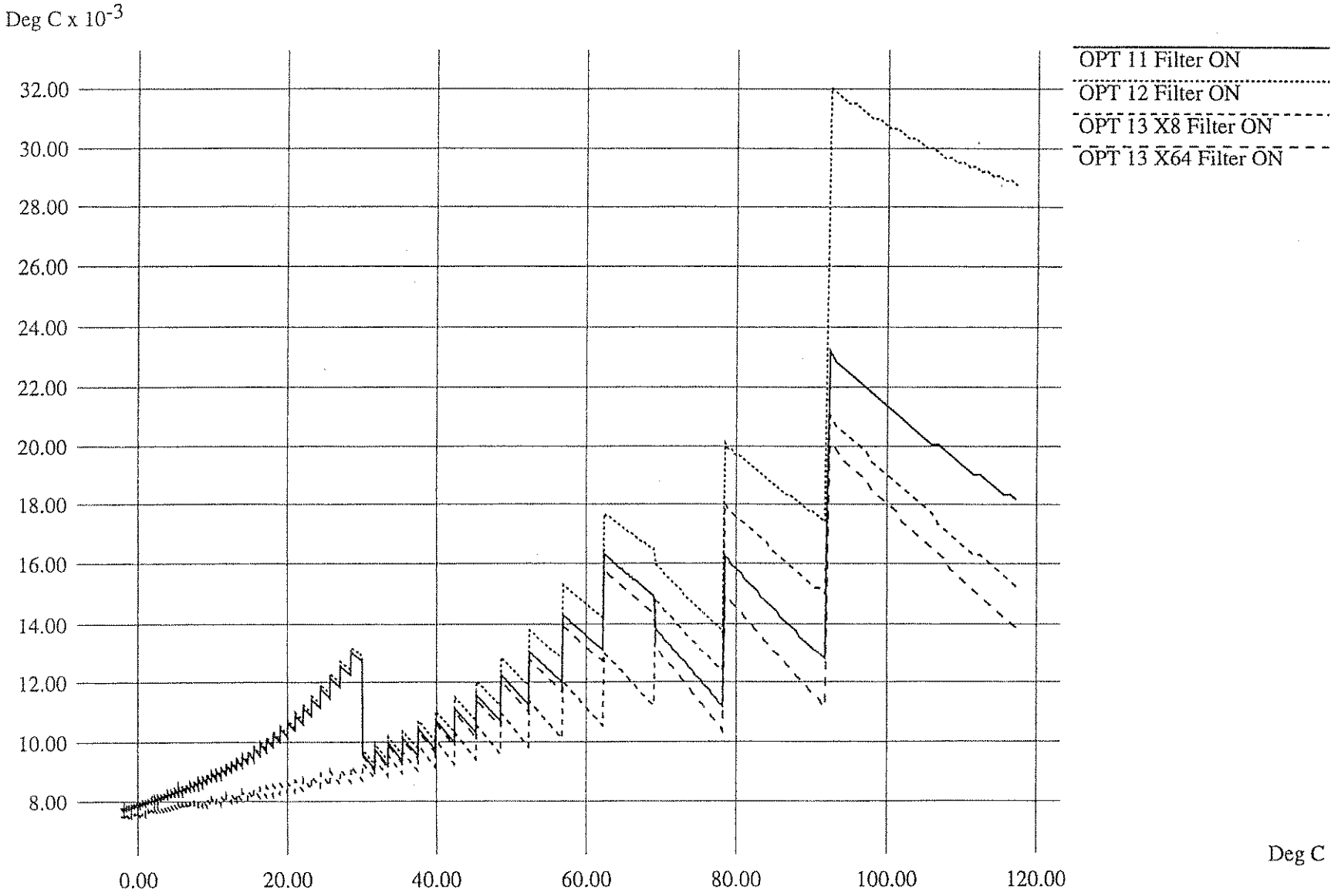

5K Therm
Deg C x 10^{-3}
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
20.00
40.00
60.00
80.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF


5K Therm
Deg C x 10^-3
27.00
26.00
25.00
24.00
23.00
22.00
21.00
20.00
19.00
18.00
17.00
16.00
15.00
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
0.00
20.00
40.00
60.00
80.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

10K Therm


Deg C x 10^{-3}
60.00
55.00
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
20.00
40.00
60.00
80.00
100.00
120.00
Deg C
OPT 11 Filter OFF
OPT 12 Filter OFF
OPT 13 X8 Filter OFF
OPT 13 X64 Filter OFF

10K Therm


Deg C x 10^{-3}
32.00
30.00
28.00
26.00
24.00
22.00
20.00
18.00
16.00
14.00
12.00
10.00
8.00
0.00
20.00
40.00
60.00
80.00
100.00
120.00
Deg C
OPT 11 Filter ON
OPT 12 Filter ON
OPT 13 X8 Filter ON
OPT 13 X64 Filter ON

Appendix B
Error Messages

Possible Error Messages:

-108	'Parameter not allowed'.
-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'.
2001	'Invalid channel number'.
2003	'Invalid word address'.
2007	'Bus error'.
2008	'Scan list not initialized'.
2009	'Too many channels in channel list'.
2016	'Byte count is not a multiple of two'.
3000	'Illegal while initiated'.

3001	'Illegal while continuous'.
3004	'Illegal command. CAL:CONF not sent'. Incorrect sequence of calibration commands. Send CAL:CONF command before CAL:VAL command.
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 have 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'.
3026	'Calibration failed'.
3027	'Unable to map A24 VXI memory'.
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'.
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 E1413 reading requires 4 bytes.

3052 'Self test failed. Test info in FIFO'. A value of 1 through 99 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 a 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 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 cousing the problem.

For tests 72, and 74-76, 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.

Refer to the Command Reference under *TST? for a list of module functions tested.

NOTES

1. HP E1413 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 E1413 C-SCPI driver for MS-DOS®.
2. 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.

3058	'Hardware does not have D32, S/H, or new trigger capabilities'. Module's serial number is earlier than 3313A00530.
3059	'DSP firmware does not have LISTL, LIMIT, or AVERAGE capability'. Firmware (stored in Flash Memory) must be updated to version A.03.00 or later.
3060	'Lower limit exceeds upper limit on one or more channels'. CALC:LIM:LOW:DATA set higer than channel's CAL:LIM:UPP:DATA
3061	'Can't change scan list while running with AVERAGE on'
3062	'LISTL not allowed with AVERAGE on'
3063	'Duplicate channels in list not allowed with AVERAGE on'
3064	'TRIG:COUNT must be = CALC:AVER:COUNT if INIT:CONT not on'
3065	'Scan list used in LISTL list must have at least 6 channels'
3066	'Autorange not allowed with AVERAGE on'
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'

Appendix C
Glossary

The following terms are either unique to Hewlett-Packard's VXIbus modules or are unique to the HP E1413.

Control Processor	The Digital Signal Processor chip that performs all of the HP E1413's internal hardware control functions as well as performing the EU Conversion process.
Continuous Mode	Fastest scanning mode. In effect when TRIGger:SOURce is set to IMM and INITitiate:CONTinuous is set to ON.
DSP	Same as Control Processor
EU	Engineering Units
EU Conversion	Engineering Unit Conversion: Converting binary A/D readings into readings of voltage, resistance, temperature, strain, or pressure.
Flash or Flash Memory	Non-volatile semiconductor memory used by the HP E1413 to store its control firmware and calibration constants
SCP	Signal Conditioning Plug-on: The HP E1413's internal plug-on signal conditioners.
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 E1413's front panel.

Appendix D
Register-Based Programming

About this Appendix

The HP E1413 64 Channel Scanning A/D module is a register-based module which does not support the VXIbus word serial protocol. When a SCPI command is sent to the module, the HP E1406A Command Module (Series C) parses the command and programs the module at the register level. The same is true for HP Compiled SCPI programming in the C language. In this case the SCPI commands are pre-processed by C-SCPI and replaced with function calls to driver libraries that perform register programming.

This appendix contains the information you need for register-based programming. The contents include:

- Table of Registers D-3
- Register Addressing D-4
- Register Based Command Reference D-25

  System Commands
  - AVGRDGS D-26
  - ERRFLAGS? D-27
  - FILTER D-27
  - FILTER? D-28
  - UP_LIMIT D-28
  - LOW_LIMIT D-28
  - NO_LIMIT D-28
  - NULL D-28
  - REVCODE? D-28
  - SCBREAD? D-28
  - SCBWRITE D-29

  Calibration Commands
  - ADGAIN D-31
  - ADZERO? D-32
  - CARDCAL D-32
  - READTEMP D-33
  - REFTEMP D-33
  - RESCAL D-33
  - RESIST D-33
  - SOURCE D-33
  - SPANHI D-33
  - SPANLO D-34
  - STORECAL D-34
  - STORETAR D-34
  - TAREAPPEND D-34
  - TARECAL? D-35

- TARENULL D-34
- UNHOOK D-35

Scan List Commands
- ADVRATEn D-35
- ADVRATEL D-35
- APPENDn D-35
- APPENDL D-36
- ASSIGN D-37
- NEWn D-38
- NEWL D-38
- SCPGAINS D-39
- SCPCHAR D-39

CVT Commands
- CVTINIT D-39

Trigger Commands
- ARM D-39
- SCPTRIGEN D-39
- TRIGCOUNT D-40

Debugging Commands
- AVERAGE D-40
- DSPEEK? D-40
- DSPOKE D-40
- PSPEEK? D-41

• Register Based Programming Fundamentals D-42
• Programming Sequence D-49

Table of Registers

Category	Address	Read Registers (return)	Write Registers	See Page...
Required VXI Registers	Base + 00_{16}	ID Register		D-6
	Base + 02_{16}	Device Type		D-6
	Base + 04_{16}	VXI Status Register	VXI Control Register	D-7, D-8
	Base + 06_{16}	Offset Register		D-8
Register-Based Command and Response Registers	Base + 08_{16}	Query Response Register	Command Register	D-9, D-10
	Base + $0A_{16}$		Parameter Register #1	D-10
	Base + $0C_{16}$		Parameter Register #2	D-10
	Base + $0E_{16}$		Parameter Register #3	D-10
Scan & Card Control Registers	Base + 10_{16}	Scan Status and Control Register		D-11, D-12
	Base + 12_{16}	Card Control Register		D-13
Interrupt Registers	Base + 14_{16}	Interrupt Configuration Register		D-15
	Base + 16_{16}	Interrupt Status Register		D-15
Virtual Instrument Registers	Base + $1A_{16}$	Common Capabilities Reg		D-17
	Base + $1C_{16}$	Description Register		D-18
	Base + $1E_{16}$	Subclass Register		D-11
FIFO Registers	Base + 20_{16}	FIFO MSW		D-20
	Base + 22_{16}	FIFO LSW		D-20
	Base + 24_{16}	FIFO Status Register		D-20
	Base + $2A_{16}$	FIFO Reading Count Reg		D-21
Trigger Registers	Base + 26_{16}		Software Trigger Register	D-22
	Base + $2C_{16}$	Trigger Timer Register		D-22
	Base + $2E_{16}$	Trigger Mode Register		D-22
IEEE-488.1 Interface Registers (HP E1414 only)	Base + 30_{16}	HP-IB Interrupt Status 0	HP-IB Interrupt Mask 0	
	Base + 32_{16}	HP-IB Interrupt Status 1	HP-IB Interrupt Mask 1	
	Base + 34_{16}	Address Status		
	Base + 36_{16}	Bus Status	Auxiliary CMD	
	Base + 38_{16}	Address for external switch	Address Register	
	Base + $3A_{16}$		Serial Poll	
	Base + $3C_{16}$	CMD Pass Through	Parallel Poll	
	Base + $3E_{16}$	Data In	Data Out	

Register Addressing

Register addresses for register-based devices are located in the upper 25% of the VXI A16 address space. Every VXI device (up to 256) is allocated a 64 (40_{16}) byte block of addresses. Figure D-1 shows the register address location within the A16 address space.

The Base Address

When you are reading or writing to a register, a hexadecimal or decimal address is specified. This address consists of a base address plus a register offset.

The base address is computed as:

49,152+LADDR*64

or

$C000_{16}$+$LADDR_{16}$*40_{16}

where 49,152 ($C000_{16}$) is the starting location of the register addresses, LADDR is the module's logical address, and 64 (40_{16}) is the number of address bytes per VXI device. For example, the HP E1413's factory set logical address is 24_{10} (18_{16}). If this address is not changed, the module will have a base address of:

49,152+24*64
49,152+1,536=50,688

or

$C000_{16}$+18_{16}*40_{16}
$C000_{16}$+600_{16}=$C600_{16}$

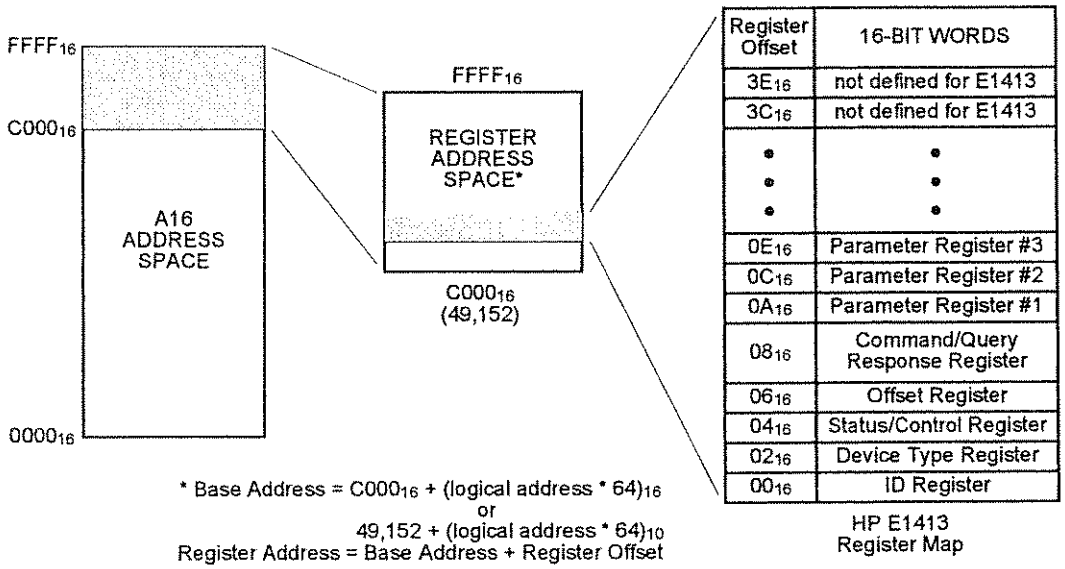


Figure D-1. E1413 Registers within A16 Address Space.

Base Address with HP Command Modules

To calculate the base address when using the HP E1413 with HP Command Modules (HP E1405B/E1406) add $1F0000_{16}$ or $2,031,616_{10}$ to the VXIbus Register Base Address.

Module Base Address =

$1FC000_{16}$ + logical address$_{16}$ * 40_{16}
or
$2,080,768_{10}$ + logical address$_{10}$ * 64_{10}

Register Address = Base Address + Register Offset

Required VXI Registers

The required VXI registers include ID, Device Type, VXI Status, and VXI Control registers.

The ID Register — Base + 00_{16}

Read only, returns $4FFF_{16}$: The module's ID register indicates the classification, addressing mode, and the manufacturer of the device.

Address	15	14	13	12	11-0
Base + 00_{16}	Device Class		Address Mode		Manufacture ID

Device Classification: Bits 15 and 14 classify a device as one of the following:

0 0 Memory device
0 1 Extended device
1 0 Message-based device
1 1 Register-based device

The HP E1413 and HP E1414 are extended devices.

Addressing mode: Bits 13 and 12 indicate the addressing mode used by the device:

0 0 A16/A24 address mode
0 1 A16/A32 address mode
1 0 RESERVED
1 1 A16 address mode

The HP E1413 and HP E1414 both use the A16/A24 address mode.

Manufacturer ID: Bits 11 through 0 identify the manufacturer of the device. Hewlett-Packard's ID number is 4095, which corresponds to bits 11-0 being set to "1".

The Device Type Register — Base + 02_{16}

Read only, returns $51C4_{16}$-E1413A $51C5_{16}$-E1414A: The Device Type register contains the required memory and model code for the HP E1413/14. The HP E1413 returns $51C4_{16}$, the HP E1414 returns $51C5_{16}$.

Address	15-12	11-0
Base + 02_{16}	Required Memory Code	Model Code

Required Memory: The HP E1413 and HP E1414 both support A24 address space. The size of the A24 space is specified as,

$$\text{A24 address space} = 2^{(23 - \text{Required Memory Code})}$$

Given that each module's A24 space is 256 Kbytes, the Required Memory field is 5_{16}.

Model Code: The model code for the HP E1413 is $1C4_{16}$. The model code for the HP E1414 is $1C5_{16}$.

The VXI Status Register — Base + 04_{16}

Read only: The Status register indicates when commands and output data can be sent, when query response and input data is available, and the error status of the module.

Address	15	14	13-8	7	6	5-4	3	2	1	0
Base + 04_{16}	A24 Active	MODID*	unused	DONE	NOERR	unused	Ready	Passed	Query/Resp Ready	Cmd/Parm Ready

A24 Active: A one (1) in this field indicates that the card's A24 address space can be accessed. This bit reflects the state of the Control Register's A24 Enable bit.

MODID*: A one (1) in this field indicates that the card is not selected via the P2 MODID line. A zero (0) indicates that the device is selected by a high state on the P2 MODID line.

DONE: A zero (0) in bit 7 indicates that the module is processing a command and its parameters. Bit 7 is set to a one (1) to indicate that the command is finished.

The validity of this bit is determined by bit 0, and in turn, bit 7 determines the validity of bit 1. See "Status Bit Precedence" for more information.

NOERR: A zero (0) in bit 6 indicates that an error has occurred.

Ready: A zero (0) in bit 3 with a one (1) in bit 2 indicates that the module has not completed its initialization process.

Passed: A zero (0) in bit 2 indicates that the module is executing a reset or has failed it self-test. A one (1) in bit 2 indicates that the reset has finished or the self-test passed.

Query/Resp Ready: A one (1) in bit 1 indicates that data returned by a query command is available in the Query Response Register. The bit is cleared (0) when the register is read.

Cmd/Parm Ready: A one (1) in bit 0 indicates that a command or parameter op code can be written to the Command or Parameter Register. The bit is cleared (0) when the command register is written to.

Status Bit Precedence

Certain status bits indicate the validity of other bits in the Status Register. This solves race condition between the selected bits.

When Cmd/Parm Ready is zero (0), DONE is invalid. This allows the module to clear DONE to indicate that a command is being processed.

When DONE is zero (0), NOERR and Query/Resp Ready are invalid. This allows the module to set those bits to the correct states based on the conditions they represent.

The VXI Control Register — Base + 04_{16}

Write only: The Control register is used to reset the module, and to disable the module from driving the SYSFAIL line.

Address	15	14-2	1	0
Base + 04_{16}	A24 Enable	unused	SYSINH	RESET

A24 Enable: Writing a one (1) to bit 15 enables accesses to the A24 address space. Writing a zero (0) disables accesses to the A24 address space.

SYSINH: Writing a one (1) to bit 1 prevents the module from asserting the SYSFAIL line. Writing a zero (0) allows the module to assert SYSFAIL.

RESET: Writing a one (1) to bit 0 resets the module. Writing a zero (0) turns the reset function off. While bit 0 is 1, the module is held in the reset state.

The Offset Register — Base + 06_{16}

Read/Write: The offset register defines the base address of the module's A24 address space. Because the HP E1413/14's A24 address space is 256 Kbytes, only the bits 15 through 10 are used.

Address	15-10	9-0
Base + 06_{16}	A24 Offset	unused

NOTE When reading the offset register, bits 9 through 0 always return zeros (0).

A24 Offset: These bits are the six most significant bits of the device's A24 base address. The 18 least significant bits are all 0.

Register-Based Command and Response Registers

The Query Response Register — Base + 08_{16}

Read only: When the module is sent a query command, the reply is sent to the Query Response Register. See Register-Based Command Reference starting on page D-25.

Address	15-0
Base + 08_{16}	Query Response

Some query commands may return multiple query responses.

The Command and Parameter Registers — Base + 08_{16}-$0E_{16}$

Write only: Commands and their parameters are written to the Command and Parameter registers. See Register-Based Command Reference starting on page D-25.

Command Register

Address	15-0
Base + 08_{16}	Command Op code

Parameter Register #1

Address	15-0
Base + $0A_{16}$	Parameter #1 Op code

Parameter Register #2

Address	15-0
Base + $0C_{16}$	Parameter #2 Op code

Parameter Register #3

Address	15-0
Base + $0E_{16}$	Parameter #3 Op code

NOTE The parameters associated with a command must always be written before the actual command op code. No command requires more than three parameter words. Upon a write to the Command Register, the Cmd/Parm Ready bit is deasserted (set LOW). It is subsequently asserted after the control processor has removed the command word and associated parameters from the parameter register buffers and is ready to receive another command.

Scan and Card Control Registers

Scan Status & Control Register — Base + 10_{16}

Read/Write: This register controls and provides status of the execution of scan lists. Note that bits 15 through 8 are read-only and are used to report scan status, and bits 7 through 0 are read/write and are used to control the scan.

The Scan Control Bits

Writing to these bits (7-0) controls the arming and execution of scan lists. When this register is read for scan status, bits 7-0 return their currently set values.

Address	15-8	7	6	5	4	3-2	1-0
Base + 10_{16}	*Scan Status*	Abort	List of Lists	Auto-arm	Free-run	Reserved	Next List

Scan Status: Writing to the Scan Status bits has no effect.

Abort: Toggling this bit high then back low halts acquisition without completing the current scan list. Data in the FIFO is undisturbed. The number of valid readings in the FIFO can be determined from the FIFO Reading Count register. To flush the FIFO, assert and deassert the FIFOCLR* bit in the Card Control register.

NOTE While the contents of registers are not disturbed, Abort resets all configuration accomplished by executing register based commands. Clearing both the Auto-arm and Free-run bits will end continuous scans when the current scan in complete without disturbing other configuration.

NOTE The control processor does not return to the command mode until the Abort bit is reset low again. Leaving the Abort bit high will cause the control processor to stay in the Abort state. See Control Processor States on page D-48

List of Lists: Writing a one (1) enables the "List of Lists". While enabled, the Next List field is ignored. Writing a zero (0 is the default) re-enables the Next List field.

Autoarm: When the module finishes executing a scan list, it will arm to receive another trigger if this bit is set. The Next List field will be used to determine the scan list to be executed when the next trigger is received. The Autoarm bit is ignored in Free-run Mode.

Free-run Mode: When this bit is set, the scan list is executed continuously after an initial trigger. Changes to the Next List bits after trigger won't be effective until scanning is stopped. If this bit is cleared when a scan is in progress, the acquisition will be halted at the end of the scan list. Acquisition may be halted immediately by asserting the abort bit, or by resetting the card.

Reserved: Always write zeros to these bits.

Next List: This field denotes which of four previously defined scan lists will be used for the next arm/trigger sequence. Note that in Free-run Mode, this field is read only once.

The Scan Status Bits

Bits 15-8 are read-only and show the present state of the scan process. Writing to this register has no effect on bits 15-8.

Address	15	14	13	12	11-10	9-8	7-0
Base + 10_{16}	Running*	Armed	Initiated	Free-run	Reserved	Current List	*Scan Control*

NOTE Check for Running* high and Initiated low (both deasserted) before writing to the Scan Control, Card Control, or Trigger Mode Registers.

Running*: This bit is low (0) when the module is executing a scan list or register based MEAS? command. The card cannot accept commands while scanning (but other A16 registers may be accessed as usual). Normally, one should avoid A24 card accesses except for reads of the Current Value Table when this bit is low. Accessing other portions of the module's A24 address space could interfere with the scan.

Armed: When asserted (1), this bit indicates that the module is initiated and is waiting for a scan trigger. Armed is deasserted after the trigger is recognized by the control processor. In Autoarm mode, Armed toggles.

Initiated: When asserted (1) this bit indicates that the module is initiated. Initiated is deasserted only when the scan or scans (in the case of Trigger Count >1) are complete.

Free-run Mode: This bit is asserted (1) when the module is armed or scanning in Free-run Mode. It may be different than the corresponding bit in the Scan Control Register, because it indicates processor state rather than register contents.

Reserved: These bits are reserved for future use. Their value is indeterminate, and should not be relied upon.

Current List: When Running* is asserted (0), this field shows the scan list presently being executed. When Running* is deasserted, but Armed is asserted (1), this field shows the scan list which will be executed upon receipt of a trigger. Under other circumstances, these bits are undefined.

Card Control Register — Base + 12_{16}

Read/Write: The card control register controls various functions on the card.

Address	15	14	13	12	11	10-8	7-0
Base + 12_{16}	Unused	FIFO Mode	unused	FIFO Clear	VPPEN	A24 Window	Open Transducer Detect

FIFO Mode: Writing this bit to a (0) sets the Block mode, writing a (1) sets the Overwrite mode. Block mode stops readings when the FIFO fills. Overwrite mode over writes oldest readings with new readings, leaving the latest 64K readings available.

FIFO Clear: Setting this bit to a (1) reset the FIFO. The FIFO will remain in reset until this bit is set to zero (0).

CAUTION

VPPEN: Setting this bit to a (1) turns on the programming voltage to the onboard flash memory. Setting this bit to zero (0) turns off this voltage. VPPEN must be set high to store calibration constants or download card control firmware to the module. In addition, the movable jumper JM2101 must be set to the Enable position.

A24 Window: These three bits allow A24 access to various 256 Kbyte pages of the control processor's memory. The following table and Figure D-2 show the relationship between bits 10-8 and the memory page mapped.

Control Processor Memory to A24 Mapping

Bit 10	Bit 9	Bit 8	A24 Window shows:
0	0	0	Current Value Table (default)
0	0	1	DSP Reserved, and RAM pages 1 to 3
0	1	0	RAM pages 4 to 7 (HP E1414 only)
0	1	1	RAM pages 8 to 11 (HP E1414 only)
1	1	0	Flash Memory (writes require jumper JM2201 to enabled position)
1	1	1	A/D RAM and DSP I/O Space

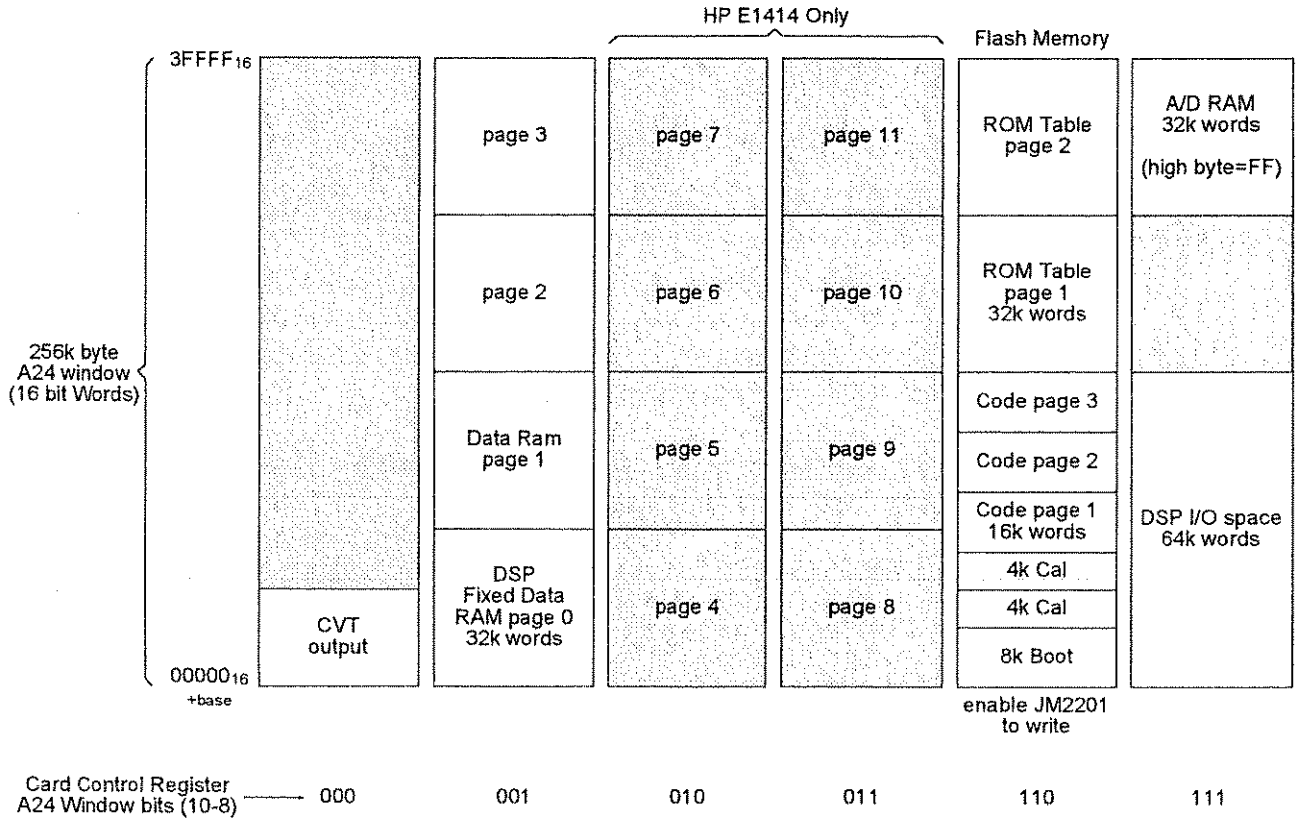


Figure D-2 A24 Memory Mapping

NOTE Normally bits 10-8 are zero (0), so that the Current Value Table can be accessed through A24 address space.

Open Transducer Detect: Writing a one (1) to a bit enables open transducer detect on the particular signal conditioning module. Writing a zero (0) to a bit disables open transducer detect.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCP 7	SCP 6	SCP 5	SCP 4	SCP 3	SCP 2	SCP 1	SCP 0

Interrupt System Registers

Interrupt Configuration Register — Base + 14_{16}

Read/Write: The Interrupt Configuration Register is used to select the VXI interrupt level to be used when the module interrupts, and controls which events are enabled to cause a VXI interrupt.

Address	15	14	13	12	11	10	9	8	7-3	2-0
Base + 14_{16}	Limit Test Exceeded	SCP Trig (E1413) HP-IB Interrupt (E1414)	Meas/Cal Complete	Trig Too Fast	Over V Detect	Scan Complete	FIFO over-flowed	FIFO Half full	unused	Interrupt Level

Interrupt Mask (bits 15-8): Writing a one (1) to a bit enables that bits interrupt. Writing a zero (0) to a bit disables that bits interrupt.

Interrupt Level: This field sets the interrupt level that the module will interrupt on. Setting the interrupt level to 0 will disable the card from interrupting.

Bit 2	Bit 1	Bit 0	Interrupt Level:
0	0	0	Disabled
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

Interrupt Status Register — Base + 16_{16}

Read/Write: The Interrupt Status Register returns the state of all sources of interrupt regardless of state of the Interrupt Mask Register. There are two types of interrupts, level and strobed. Level interrupts are interrupts that persist from an interrupt source and requires the host to service the interrupt through reads and writes to the interrupt source. Strobed interrupts are interrupts that do not persist and require the host to acknowledge the interrupt by writing a one (1) back to the appropriate bit in the Interrupt Status Register.

Level Interrupts:

- FIFO Half Full
- HP-IB Interrupt (HP E1414 only)

Strobed Interrupts:

- FIFO Overflowed
- Meas Complete
- Over voltage Detect
- Trigger Too Fast
- SCP Trigger
- Limit Test Exceeded

Address	15	14	13	12	11	10	9	8	7-0
Base + 16_{16}	Limit Test Exceeded	SCP Trig (1413) HP-IB Interrupt (1414 only)	Meas/Cal Complete	Trig Too Fast	Over V Detect	Scan Complete	FIFO Over-flowed	FIFO Half Full	unused

A one (1) in a bit indicates that the interrupt source is asserted. A zero (0) in a bit indicates that the interrupt source is deasserted.

Limit Test Exceeded: (HP E1413 only) This bit will be set to one (1) at the end of each Scan List in which a channel has exceeded its limit test.

SCP Trigger: (HP E1413 only) A one (1) in this bit indicates that a Signal Conditioning Plug-on (SCP) has triggered. A zero (0) indicates that an SCP has not triggered.

HP-IB Interrupt: (HP E1414 only) A one (1) in this bit indicates that the HP-IB chip is asserting its interrupt. A zero (0) indicates that the HP-IB chip is not asserting its interrupt.

Measurement/Calibration Complete: A one (1) indicates that the measurement/calibration has completed.

Trigger Too Fast: A one (1) in this bit indicates that a trigger too fast condition has been detected. A zero (0) indicates that no trigger too fast condition has occurred.

Over Voltage Detect: A one (1) indicates that an SCP over voltage condition has occurred. A zero (0) indicates that no over voltage condition has occurred.

Scan Complete: A one indicates that the module has completed a scan. If the module is in the Free-run mode, Scan Complete is asserted at the end of each scan.

FIFO Overflowed: A one (1) indicates that the FIFO has overflowed

FIFO Half Full: A one (1) indicates that the FIFO is half. To clear this condition, the FIFO is read until the FIFO is less than half full.

Virtual Instrument Registers

The Virtual Instrument registers are used to identify what virtual instrument functionality the HP E1413/14 modules have.

Common Capabilities Register — Base + $1A_{16}$

Read only, returns 50_{16}: This register indicates whether the HP E1413 and HP E1414 support certain optional registers.

Address	15-7	6	5	4	3	2	1	0
Base + $1A_{16}$	0	Send Count	Rec Count	IRQ 1	IRQ 2	Send Frame	Rec Frame	Slave Addr

Send Count: A one (1) indicates that the device has a Send Count register. A zero (0) indicates that the Send Count register locations have some device specific function(s).

Rec Count: A one (1) indicates that the device has a Receive Count register. A zero (0) indicates that the Send Count register locations have some device specific function(s).

IRQ 1: A one (1) indicates that the device has the IRQ Config 1 register. A zero (0) indicates that the IRQ Config 1 register location has some device specific function(s).

IRQ 2: A one (1) indicates that the device has the IRQ Config 2 register. A zero (0) indicates that the IRQ Config 0 register location has some device specific function(s).

Send Frame: A one (1) indicates that the device has the Send Frame register. A zero (0) indicates that the Send Frame register location has some device specific function(s).

Receive Frame: A one (1) indicates that the device has the Receive Frame register. A zero (0) indicates that the Receive Frame register location has some device specific function(s).

Slave Addr: A one (1) indicates that the device has the Slave Address register. A zero (0) indicates that the Slave Address register location has some device specific function(s).

The Description Register — Base + $1C_{16}$

Read only, returns $FA60_{16}$: The Description Register returns more information on identification and functionality of the HP E1413 and HP E1414. Note that the Send Data, Send Count, and Send Status registers correspond to the FIFO MSW, FIFO LSW, FIFO Reading Count, and FIFO Status registers.

Address	15-14	13-11	10-8	7	6	5	4	3	2	1	0
Base + $1C_{16}$	Rev	Sub Type	Mod Type	0	A16	VME Slave Send	VME Slave Rec	VME Mast Send	VME Mast Rec	LBUS Send	LBUS Rec

Rev: All ones indicates the current revision of the module.

Sub Type: all bits always set to ones (1).

Mod Type: The Mod Type indicates what type of module the HP E1413 and E1414 are.

Bit 10	Bit 9	Bit 8	Meaning
0	1	1	Signal Acquisition/Conditioning Front Ends
0	1	0	Analog/Interval to Digital Conversion
1	1	0	Data Storage
1	1	0	Data Processing
1	0	0	Digital to Analog Conversion
1	0	1	Signal Distribution/Enhancement Rear Ends
0	0	0	Reserved
0	0	1	Other

A16: A one (1) indicates that the Virtual Instrument Basic Registers, (Send Data, Send Count, Send Status, Receive Data, Receive Count, and Receive Status), are in the A16 address space. A zero (0) indicates that these registers are in A24 or A32 address space starting a address offset 8_{16}. The Send Data, Send Count, and Send Status registers are the FIFO MSW and LSW Registers, FIFO Reading Count, and FIFO Status Registers respectively.

VME Slave Send: A one (1) indicates that the device has the capability to send data via its VMEbus slave registers, and implements both the Send

Data and Send Status registers. A zero indicates that this capability is absent on this device.

VME Slave Rec: A one (1) indicates that the device has the capability to receive data via its VMEbus slave registers, and implements both the Receive Data and Receive Status registers. A zero indicates that this capability is absent on this device.

VME Mast Send: A one (1) indicates that the device is a VMEbus master having the capability to send data to another Virtual Instrument device's Receive Data register. A zero indicates that this capability is absent on this device.

VME Mast Receive: A one (1) indicates that the device is a VMEbus master having the capability to receive data from another Virtual Instrument device's Send Data register. A zero indicates that this capability is absent on this device.

LBUS Send: A one (1) indicates that this device is capable of sending Local Bus data. A zero (0) indicates that it is unable to send data over the Local Bus.

LBUS Rec: A one (1) indicates that this device is capable of receiving Local Bus data. A zero (0) indicates that it is unable to receive data over the Local Bus.

The Subclass Register — Base + $1E_{16}$

Read only, returns $7FFF_{16}$: The subclass register identifies the HP E1413 and HP E1414 as HP virtual instrument devices.

FIFO Registers

The FIFO registers consist of the FIFO MSW, FIFO LSW, the FIFO Status, and FIFO Reading Count Registers.

The FIFO MSW and LSW Registers — Base + 20_{16} and 22_{16}

Read only: IEEE 32-bit readings from the FIFO are read from these registers. The FIFO MSW register returns the most significant word of the reading, and the FIFO LSW register returns the least significant word.

FIFO MSW Register

Address	15-0
Base + 20_{16}	Most Significant Word

FIFO LSW Register

Address	15-0
Base + 22_{16}	Least Significant Word

NOTE Before reading these registers, the FIFO Data Ready bit in the FIFO Status Register must be asserted. If these registers are read while the FIFO Data Ready bit is deasserted, a bus error will occur.

The FIFO Status Register — Base + 24_{16}

Read only: The FIFO Status Register indicates when the FIFO is not empty, and when a block of 32,768 readings is in the FIFO.

Address	15-11	10-9	8-7	6-2	1	0
Base + 24_{16}	all 0s	Count Width (16 bits=102)	Data Width (32 bits=11_2)	Block Size (32,768=01111_2)	Block Ready	FIFO Ready

Count Width: These bits indicate the width of the FIFO Reading Count register. The count width is 16 bits.

Bit 10	Bit 9	Meaning
0	0	No Count Register
0	1	8 bits
1	0	16 bits
1	1	32 bits

Data Width: These bits indicate the data width of the FIFO registers. The data width is 32 bits.

Bit 8	Bit 7	Meaning
0	0	Reserved
0	1	8 bits
1	0	16 bits
1	1	32 bits

Block Size: These bits indicate the number of readings that may be read from the FIFO registers whenever the Block Ready bit is set. The number of readings is encoded in binary such that Number_of_readings = 2^n, where n is the value encoded in the Block Size field. For the HP E1413/14, n is equal to 15, which makes the block size 32,768 readings.

Block Ready: A one (1) indicates that there is a block of data available to be read from the FIFO registers. The number of readings that can be read is encoded in the Block Size field. A zero (0) indicates that less than a full block of readings is available.

FIFO Ready: A one (1) indicates that there is at least one reading in the FIFO. A zero (0) indicates that there is no valid data in the FIFO.

The FIFO Reading Count Register — Base + $2A_{16}$

Read only: The FIFO Reading Count Register contains the number of 32-bit readings in the FIFO.

Address	15-0
Base + $2A_{16}$	Number of readings in FIFO

Trigger System Registers

Software Trigger/ARM Register — Base + 26_{16}

Write only: A write to this register triggers a measurement scan if the trigger system is enabled and software trigger is selected as a trigger source. If the above is true and bit 9 of the Trigger Mode Register is set, a write to this register arms (starts) the trigger timer.

Address	15-0
Base + 26_{16}	write any data

Trigger Timer Register — Base + $2C_{16}$

Read/Write: A write to this register sets the time interval between timer triggers. The resolution is 1.0E-4 seconds. The minimum interval is 1.0E-4 seconds. The trigger interval in seconds = $(1.0E\text{-}4 * reg\_value) + 1.0E\text{-}4$

Address	15-0
Base + $2C_{16}$	trigger interval (0-65535)

The Trigger Mode Register — Base + $2E_{16}$

Read/Write: The Trigger Mode Register is used configure the trigger system.

Address	15	14-13	12-11	10	9	8-6	5-4	3-0
Base + $2E_{16}$	undefined (write 0)	SCP Trig Source	Output Trigger Source	Trig Timer Continuous	Select Trigger Timer	Trig Out	Trig Mode	Trig/Arm Source

SCP Trigger Source: Selects the source to drive the SCP trigger line.

Bit 14	Bit 13	SCP Trigger Source
0	0	None
0	1	Trigger In
1	0	Reserved
1	1	Reserved

Output Trigger Source: Selects the source driving the TTLTRGn line when the Trig Mode is "Output".

Bit 12	Bit 11	Output Trigger Source
0	0	Trigger In
0	1	First Trigger
1	0	Limit Test Exceeded
1	1	SCP Trigger

Trig Timer Continuous: A one (1) causes the trigger timer to run continuously after the first arm event (synchronous mode). When this bit is zero (0), the trigger timer will run only while the HP E1413 is initiated (asynchronous mode).

Select Trigger Timer: A one (1) selects the Trigger Timer as the trigger source. The Trig/Arm Source becomes the Trigger Timer Arm Source. A zero (0) selects the trigger source as set by the Trig/Arm source bits.

Trigger Out: The Trig Out field is used to select one of the TTLTRG* trigger for the trigger system to source.

Bit 8	Bit 7	Bit 6	Trigger Output:
0	0	0	TTLTRG0
0	0	1	TTLTRG1
0	1	0	TTLTRG2
0	1	1	TTLTRG3
1	0	0	TTLTRG4
1	0	1	TTLTRG5
1	1	0	TTLTRG6
1	1	1	TTLTRG7

Trig Mode: The Trig Mode field is used to select the trigger protocol.

Bit 5	Bit 4	Trigger Output:
0	0	Synchronous
0	1	Semi-synchronous
1	0	Asynchronous
1	1	Output

Trig/Arm Source: The Trig/Arm Source field is used to select the trigger or Arm source. When Bit 9 (Select Trigger Timer) is set to 0, the sources in the following table are the trigger sources. When Bit 9 is set to 1 (timer is trigger source), the following become ARM sources for the Trigger Timer.

Bit 3	Bit 2	Bit 1	Bit 0	Trigger/Arm Source:
0	0	0	0	TTLTRG0
0	0	0	1	TTLTRG1
0	0	1	0	TTLTRG2
0	0	1	1	TTLTRG3
0	1	0	0	TTLTRG4
0	1	0	1	TTLTRG5
0	1	1	0	TTLTRG6
0	1	1	1	TTLTRG7
1	0	0	0	Software
1	0	0	1	External
1	0	1	0	SCP
1	0	1	1	Trigger Immediate
1	1	X	X	none selected

Using the Trigger Mode Register

To support the various trigger modes the correct values must be set in each of the fields of the Trigger Mode Register. Below are examples settings for the trigger modes supported.

Sync: To support the VXI trigger mode sync, the Trigger Mode field is set to "Sync", and the Trigger Source field set to one of the TTLTRG* lines. It does not matter what the Trig Out field is set to. See the VXI document for more information on this trigger protocol.

Semi-Sync: To support the VXI trigger mode Semi-sync, the Trigger Mode field is set to "Semi-sync", the Trigger Source field set to one of the TTLTRG* lines, and the Trig Out field set to the same TTLTRG* line, which was set in the Trigger Source field. See the VXI document for more information on this trigger protocol.

Async: To support the VXI trigger mode Async, the Trigger Mode field is set to "Async", the Trigger Source is set to the lower line of a TTLTRG* line pair, and the Trig Out field is set to the higher line of the TTLTRG* line pair. See the VXI document for more information on this trigger protocol.

Output: This is not one of the defined VXI trigger protocols. This mode is used when it is necessary to trigger other modules in the system when another module has been triggered. To set up for this mode, the Trigger Mode field is set to "Output". The Trigger Out field is set to the TTLTRG trigger line that is to be triggered. And, the Trigger Source field is set to one of the trigger sources other than the one set in the Trigger out field.

Register Based Command Reference

This section describes the low level register commands of the HP E1413 module. Commands are sent to the module through the Command Register (08_{16}) and Parameter Registers ($0A_{16}$, $0C_{16}$, $0E_{16}$). Commands to the card are either a command word only **or** a command word and one to three parameter words. When a command includes parameters, the command word is sent only **after** the parameter word(s) have been sent. For query commands, data is returned through the Query Response Register (08_{16}). There is only one Query Response Register, there is no response buffering. Therefore any response left in the Query Response Register will be lost (over written) by the next query command. Always read query responses before executing the next query command. For example the value 9.9E+37 is $7E94F56A_{16}$, 7E9416 is sent to the lower numbered parameter register, while F56A16 goes to the higher numbered parameter register.

Note on Number Format

Where commands have the parameter names <highword> and <lowword>, the values taken together represent a 32-bit floating point value in the Motorola format. That is, the high 16-bit word goes to the lower numbered parameter register, the low 16-bit word goes to the higher numbered parameter register. For example the value $9.9E+37_{10}$ is $7E94F56A_{16}$, $7E94_{16}$ is sent to the lower numbered parameter register, while $F56A_{16}$ goes to the higher numbered parameter register.

Note that the ASCII command names shown here are a descriptive convenience only. The actual command is a sixteen-bit word, shown to the right of the command name. Where the command name ends with a lower-case "n," bits 1-0 of the command word control which of four possible scan lists are referenced by the command. The Register Based Commands are divided into the following categories:

- System Commands
  - AVGRDGS D-26
  - ERRFLAGS? D-27
  - FILTER D-27
  - FILTER? D-28
  - UP_LIMIT D-28
  - LOW_LIMIT D-28
  - NO_LIMIT D-28
  - NULL D-28
  - REVCODE? D-28
  - SCBREAD? D-28
  - SCBWRITE D-29
- Calibration Commands

- ADGAIN D-31
- ADZERO? D-32
- CARDCAL D-32
- CARDCAL? D-33
- READTEMP D-33
- REFTEMP D-33
- RESCAL D-33
- RESIST D-33
- SOURCE D-33
- SPANHI D-33
- SPANLO D-34
- STORECAL D-34
- STORETAR D-34
- TAREAPPEND D-34
- TARECAL D-34
- TARECAL? D-35
- TARENULL D-34
- UNHOOK D-35

• Scan List Commands
- ADVRATEn D-35
- ADVRATEL D-35
- APPENDn D-35
- APPENDL D-36
- ASSIGN D-37
- NEWn D-38
- NEWL D-38
- SCPGAINS D-39
- SCPCHAR D-39

• CVT Commands
- CVTINIT D-39

• Trigger Commands
- ARM D-39
- SCPTRIGEN D-39
- TRIGCOUNT D-40

• Debugging Commands
- AVERAGE? D-40
- DSPEEK? D-40
- DSPOKE D-40
- PSPEEK? D-41

System Commands

The System commands provide for checking errors, and communicating with the Signal Conditioning Plug-ons.

AVGRDGS <enable> <n_readings> 0060_{16}

This command sets the number of readings to be averaged for each reading stored in the FIFO or CVT. <enable> is 1 for enabled, 0 for disabled. It takes n scan triggers to obtain the n readings to average. No channel should

be repeated in the scan list. Valid values for <n_readings> are 2, 4, 8, 16, 32, 64, 128, and 256.

ERRFLAGS? 2020$_{16}$

This command returns the contents of the Error Flag Word in the VXI Query Response Register. The Error Flag Word provides additional information about the previously executed command.

NOTE Any other command will update/add the contents of the Error Flag Word. Error flag bits accumulate in the Error Flag Word until an ERRFLAGS? query is processed. ERRFLAGS? resets all flag bits to zero.

The following error flags are defined:

Bit	Description
0	DSP-Unrecognized command code
1	DSP-Parameter out of range
2	DSP-Flash rom erase failure
3	DSP-Programming voltage not present
4	DSP-Invalid SCP gain value
5	DSP-Invalid *CAL? constant or checksum. *CAL? required.
6	DSP-Couldn't cal some channels
7	DSP-Re-Zero of ADC failed
8	DSP-Invalid Tare CAL constant or checksum
9	DSP-Invalid Factory CAL constant or checksum
10	DSP-DAC adjustment went to limit
11	DSP Status--Do *CAL?
12	DSP-Overvoltage on input
13	DSP-reserved error condition
14	DSP-ADC hardware failure
15	DSP-reserved error condition

Bits in the Error Flag Word are asserted when high.

NOTE If the module does not pass its Self Test, the ERRFLAGS? command can not be executed. However, if the control processor is able, it will place the Error Flag Word into the Query Response Register.

FILTER <enable> 2300$_{16}$

Enables (1) or disables (0) the A/D filter.

FILTER? 2310_{16}

Returns the status of the A/D filter in the Query Response Register. 1 means enabled, 0 means disabled.

UP_LIMIT <channel> <highword> <lowword> 0030_{16}

Sets the upper limit for limit-checking readings from the channel specified by <channel>. <highword> and <lowword> combine to become a 32-bit Motorola format floating point value.

LOW_LIMIT <channel> <highword> <lowword> 0040_{16}

Sets the lower limit for limit-checking readings from the channel specified by <channel>. <highword> and <lowword> combine to become a 32-bit Motorola format floating point value.

NO_LIMIT <channel> 0050_{16}

Turns off limit checking for the channel specified by <channel>.

NULL 0000_{16}

This command takes no actions and causes no errors..

REVCODE? 0020_{16}

Returns the revision code for the module's firmware stored in its flash memory.

SCBREAD? <regaddr> 0800_{16}

Returns a sixteen bit value which is the contents of the requested register on the SCP Bus. <regaddr> is an 11-bit address, which has the two formats shown below.

Accessing SCP Channel Registers

10	9	8-6	5-3	2-0
CAL	CHN=1	Plug-on#	Chan_addr	Reg_addr

Accessing Whole SCP Registers

10	9	8-6	5-3	2-0
CAL	CHN=0	Plug-on#	Reg_addr	

To access registers which apply to the whole SCP, set the CHN bit to zero. To access registers for individual channels, set CHN bit to one. The CAL bit

is zero when accessing SCPs, and one when accessing registers which control the calibration relays.

SCBWRITE <regaddr> <word> 0810_{16}

Writes the sixteen bit value, <word>, to a location on the SCP Bus determined by <regaddr>. The format of <regaddr> is documented above.

Required Signal Conditioning Plug-on Registers

Signal Conditioning Plug-ons have several required registers. There are two major categories of SCP registers: Whole SCP registers (CHN bit=0) in which the register has effect on the entire SCP; and Channel registers (CHN bit=1) which effect only a single SCP channel.

Whole SCP Registers (CHN bit=0)

ID Register (Read, regaddr=00ppp000000$_2$)

15-8	7-0
ID	SUB ID

Sub ID is an eight-bit field which identifies the particular version of the SCP. Substantially similar SCPs may be available in several versions, distinguished by filter cutoff frequency or some other parameter which does not affect the software driver. The eight-bit ID field determines which software driver to use. Simple SCPs may return the same number in both fields.

Scale register (Read/Write, regaddr=00ppp000001$_2$)

15-4	3	2-0
don't care	Sign	Scale

The low four bits of this register are significant. They consist of a sign plus a three-bit integer. Thus -7 <= SCALE <= 7. This is the weight of the least significant bit in the Channel Gain Registers. See "Channel Gain," below, for details. SCALE may be fixed, or it may be programmable.

Channel Registers (CHN bit=1)

Gain Registers (Read/Write, regaddr=01pppccc001$_2$)

15-4	3-0
don't care	Gain

The low four bits of this register are significant. They consist of an unsigned integer, 0 <= CHGAIN <= 15. This number, in combination with SCALE, describes how voltage measurements on the particular channel should be translated into actual volts. This is described in the following paragraph. CHGAIN may be fixed, or it may be programmable.

Channel Gain:
Channel gain is computed as a LEFT SHIFT value obtained by multiplying channel gain bit value by the value in the Plug-on Scale Register. The resultant value is the channel's (base two) gain exponent. For each channel, compute:

SHIFT = SCALE * CHGAIN

This is the number of binary left shifts represented by the gain or attenuation through the SCP.

Example 1: An SCP module has SCALE=0010b, and CHGAIN programmable. Then if CHGAIN=0000, the channel has unity gain (SHIFT=0). If CHGAIN=0001b, the channel has a gain of x4 (SHIFT=2). If CHGAIN=0010b, the channel has a gain of x16, and so forth.

Example 2: An SCP module has SCALE=0000b (fixed). Since SHIFT is always zero, the SCP module has gain permanently fixed at unity for all channels.

Example 3: An SCP module has CHGAIN=0001b (fixed) for all channels. The bottom three bits of SCALE are programmable; the sign bit is fixed at 1. Then the SCP is a programmable attenuator, and attenuation is programmed (for the entire SCP) as follows:

Scale	Shift	function
1000_2	0	unity gain
1001_2	-1	divide by 2
1010_2	-2	divide by 4
1011_2	-3	divide by 8 etc.

Note that the E1413 does not support attenuation by greater than 2, due to fixed-point arithmetic limits. This does not prevent SCPs from implementing larger attenuation factors, but such factors won't work correctly on the HP E1413. Configuring an SCP for a larger attenuation will cause an error during execution of the CARDCAL command.

NOTE The Control Processor needs to "know" the SCP channel gain settings to properly perform an EU conversion for each channel. The SCPGAINS command reads all channel gains and must be executed once the gains are set and scan lists are defined (see ASSIGN and APPEND commands.

Calibration Commands

The Calibration commands provide for four levels of calibration:

- "A/D Calibration"; In these procedures, an external multimeter is used to determine the actual voltage or resistance values of the HP E1413's internal calibration sources. The known values are then sent to the HP E1413 where they are stored and used to perform internal A/D calibration. These procedures each require a sequence of several calibration commands.

- "Channel Calibration"; This function corrects for offset and gain errors for each module channel. The internal current sources are also measured. 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 the single command **CARDCAL**.

- "A/D converter 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 E1413 where channel scanning speed is of primary importance, this function is performed only when the **ADZERO**, and **CARDCAL** commands are executed. **ADZERO** is much faster than **CARDCAL**

- "Channel Tare"; This function (**TARECAL**) 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 (**STORETAR**) 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.

ADGAIN <range> 1020_{16}

Generates the Gain Correction Constant for the indicated A/D range. The "hi" and "lo" values of the on-board calibration source (for the given range) must previously have been measured with a transfer-quality external voltmeter, and the measured values must have been given to the card using

the SPANHI and SPANLO commands. The following is the recommended command sequence for each <range>:

1. SOURCE <range> <0>
2. [measure source voltage with external voltmeter on "autorange"]
3. SPANLO <range> <Volts (msw)> <Volts (lsw)>
4. [execute "range hold" function on external voltmeter]
5. SOURCE <range> <1>
6. [measure source voltage with external voltmeter]
7. SPANHI <range> <Volts (msw)> <Volts (lsw)>
8. ADGAIN <range>

The Gain Correction Constant is loaded into the A/D subsystem's register file (and is kept in CPU data RAM), but is not stored in Flash Memory until receipt of a STORECAL command.

ADZERO? 1010_{16}

Generates the A/D Coarse Offset, and the A/D Fine Offset correction constants for each A/D range, based on the card's internal short. ADZERO is intended to be executed more frequently than CARDCAL. It executes much more quickly, because it only updates constants associated with the A/D Converter offset voltage. This command corrects any short-term drift due to temperature changes in the A/D Converter. This command sets the calibration relays to their normal (measurement) position at its conclusion.

CARDCAL 1000_{16}

CARDCAL calibrates the HP E1413 for gain and offset voltage of the Signal Conditioning Plug-ons (SCPs). Every channel containing in input-type SCP is calibrated for every usable A/D measurement range. If programmable SCPs are present, the CARDCAL command calibrates for the present SCP gain and filter settings. CARDCAL calibrates the on-card current source, using the resistor value stored by the RESCAL command. CARDCAL then reloads the A/D subsystem's register file with the new calibration constants. The derived channel constants are valid only for the current SCP module settings. If these settings are changed, the CARDCAL command should be executed again before making further measurements. When execution is complete, CARDCAL sets the calibration relays to their normal (measurement) position.

NOTE Apply power to the HP E1413 and allow 1 hour for its temperature to stabilize before issuing the CARDCAL command.

CARDCAL? 1030_{16}

Returns the results of the latest CARDCAL command via the Query Response Register (base+08_{16}). The meaning of the bits in the response is the same as those in the Error flag Word (see table on page D-27)

READTEMP 2000_{16}

HP E1413 only: Reads the value of the reference junction temperature (either measured or set with the REFTEMP command) . The value is returned in the FIFO buffer as a 32 bit IEEE floating-point number (degrees C). Temperature range is ±128 °C.

REFTEMP <highword> <lowword> 2010_{16}

Sets the reference temperature value used for thermocouple compensation. The concatenation of <highword> and <lowword> is an IEEE floating point number in Motorola format interpreted as degrees Celsius. The principal use of this command is in situations where the reference junction is servo-controlled to a fixed value and is never measured.

RESCAL <highword> <lowword> 1040_{16}

Tells the CPU the value of the on-card resistance reference, as measured by an external DMM. The concatenation of <highword> and <lowword> is an IEEE floating point number in Motorola format representing resistance in ohms. Nominal value of on-card resistor is 7500.0 Ohms

RESIST 4020_{16}

The Calibration Bus Relay and the Ohm Relay switched to allow four-wire-ohms measurement of the on-card reference resistor by an external DMM using Calibration Bus connectors.

SOURCE <range> <lowhigh> 4010_{16}

Configures the Calibration Bus Relay and associated multiplexers to allow the on-board calibration source to be measured by an external voltmeter. The output voltage depends on the parameters <range> and <lowhigh>. If <lowhigh> = 0, the output voltage is approximately zero volts. If <lowhigh> =1, the output voltage is approximately 90% of A/D full scale for the indicated <range>.

SPANHI <range> <highword> <lowword> 4080_{16}

Tells the CPU the voltage of the on-card source, as measured by an external DMM after the source has been set up by a SOURCE <range> <1>

command. The concatenation of <highword> and <lowword> is an IEEE floating point number in Motorola format representing the source voltage.

SPANLO <range> <highword> <lowword> 4040_{16}

Tells the CPU the voltage of the on-card source, as measured by an external DMM after the source has been set up by a SOURCE <range> <0> command. The concatenation of <highword> and <lowword> is an IEEE floating point number in Motorola format representing the source voltage.

STORECAL 8000_{16}

If the Flash Memory Enable jumper (JM2201) is enabled, calibration constants which apply to the A/D subsystem, and channel gain and offset are copied into Flash Memory.

STORETAR 8010_{16}

HP E1413 only: If the Flash Memory Enable jumper (JM2201) is set to enable, the Channel Tare offset constants are copied into Flash Memory.

TARENULL 3100_{16}

HP E1413 only: Clears the Tare Channel List used by the TARECAL command. Add channels to the list with TAREAPPEND.

TAREAPPEND <channel> 3110_{16}

HP E1413 only: Adds *<channel>* to the Tare Channel List. The Tare Channel List is cleared with the TARENULL command. The Tare Channel List is used by the TARECAL command. Issue this command repeatedly, once for each channel requiring TARECAL.

TARECAL 1080_{16}

HP E1413 only: Derives the offset constants necessary to make the channels included in the Tare Channel List read zero. Use TAREAPPEND to define the Tare Channel List. These values are stored in RAM, and loaded into the A/D subsystem's register file. They are not copied into Flash Memory until receipt of the STORETAR command. A CARDCAL command should always be executed before TARECAL in order to decouple internal offsets from external (tare) offsets. TARECAL can also be used for limited-range nulling of strain gages. One would not normally store the tare constants in this case.

TARECAL? 1090_{16}

Returns results of the most recent TARECAL command in the Query Response Register. The meaning of bits in the response are the same as the Error Flags (see table on page D-27).

UNHOOK 4000_{16}

Disconnects the Calibration Bus from the connector panel and sets the cal relays to their measure positions. The bus is reconnected by a SOURCE, or RESIST command. Note that the ADZERO and CARDCAL commands also do this as a side effect.

Scan List Commands

These commands are used to assign a conversion algorithm to each module channel; establish which channels are to be assigned to each scan list; and establish the pace of measurements during the scan.

ADVRATEn <clocks> 0200_{16} - 0203_{16}

Sets the interval between successive samples for scan list n. The sample rate is 2 MHz / (<clocks> + 1). The minimum value for <clocks> is 19 (decimal), corresponding 10 µS advance interval. For values smaller than 19, 19 will be used.

ADVRATEL <clocks> 0204_{16}

Sets the interval between successive samples for all scan lists in the "List of Lists. The sample rate is 2 MHz / (<clocks> + 1). The minimum value for <clocks> is 19 (decimal), corresponding 10 µS advance interval. For values smaller than 19, 19 will be used.

APPENDn <channel> <range> <data_flags> 0110_{16} - 0113_{16}

Adds an item to scan list n where n is 0 - 3 and is determined by the last op-code digit. <channel> is the channel on which to make the measurement. <range> is an integer from -1 to 4, where 4 indicates the highest range. A value of 0FFFFh is also allowed, and indicates autorange. The <data_flags> parameter has the format ddc, where dd are destination bits and c is the conversion flag bit. The destination bits control the disposition of the measurement data, and the conversion flag determines whether the measurement is converted as specified in the ASSIGN statement for that channel, or reported in raw volts. This is summarized in the table which follows:

Data Flags (Bits 2-0)	Meaning
$00x_2$	No data destination (ref. temperature can be updated)
$01x_2$	Data destination is the Current Value Table (CVT)
$10x_2$	Data destination is First-In-First-Out Buffer (FIFO)
$11x_2$	Data destination is both the CVT and the FIFO
$xx1_2$	Measurement is reported in Engineering Units (EU)
$xx0_2$	Measurement is reported in voltage (with reference junction compensation where appropriate)

x=don't care

Example values of <data_flags>:
0007_{16} = Convert to Engineering Units and store in FIFO and CVT
0005_{16} = Convert and store in FIFO only
0002_{16} = No conversion (leave as voltage) and store in CVT only

APPENDL <list> $011A_{16}$

Adds a scan list to the "List of Scan Lists". <list> is the scan list to add and can take on the value 0 through 3.

ASSIGN <channel> <conversion> <compensation> 0010_{16}

Loads the specified channel's linearization table area with the correct values for the indicated conversion. If <conversion> is non zero, it indicates a standard linearization, such as a RTD or a type K thermocouple. Thermocouple conversions incorporate reference junction compensation. Some conversions are specifically for transducers measuring a reference junction. If <conversion> is zero, no conversion is performed and the channel returns voltage. The supported conversions are:

Code	EU Conversion	Use Range:
0_{16}	Volts	0-4 | $FFFF_{16}$
1_{16}	Reserved straight-line conversion	
2_{16}	Ohms, 31 µA	$FFFF_{16}$
3_{16}	Ohms, 488 µA	$FFFF_{16}$
4_{16}	Reference, Thermistor, 5K, 122 µA	3
5_{16}	Reference, RTD, 85, 121 µA	0
6_{16}	Reference, RTD, 92, 121 µA	0
7_{16}	Reference, Custom	0
8_{16}	Thermocouple, Type J	0
9_{16}	Thermocouple, Type K	0
A_{16}	Thermocouple, Type T	0
B_{16}	Thermocouple, Type E (high accuracy)	0
C_{16}	Thermocouple, Type E (extended range)	1
D_{16}	Thermocouple, Type R	0
E_{16}	Thermocouple, Type S	0
F_{16}	Thermocouple, Custom, no compensation	0
10_{16}	Thermistor, 2.25K, 488 µA	$FFFF_{16}$
11_{16}	Thermistor, 5K, 31 µA	$FFFF_{16}$
12_{16}	Thermistor, 10K, 31 µA	$FFFF_{16}$
13_{16}	RTD, 85, 488 µA	0
14_{16}	RTD, 92, 488 µA	0
15_{16}	Thermocouple, Type N	0
16_{16}-$1E_{16}$	Reserved (not used)	
$1F_{16}$	Custom downloaded on a per-channel basis	

The <compensation> parameter has the following values for thermocouple reference compensation:

Code	EU Conversion
0	No Compensation
1	J Type Compensation
2	K Type Compensation
3	T Type Compensation
4	E Type Compensation
5	N Type Compensation
6	R Type Compensation
7	S Type Compensation

NOTES

1. The Control Processor needs to "know" the SCP channel gain settings to properly perform an EU conversion for each channel. The SCPGAINS command reads all channel gains and must be executed once the gains are set and scan lists are defined.

2. Thermocouple measurements (except for conversion code F) are always compensated for the current reference temperature.

3. There are two types of conversions. Most conversions involve a piece-wise linear interpolation from tables stored in RAM. An alternative conversion method consists of a simple $(MX+B)2^E$ computation. This "straight-line" computation is valid for the entire measurement range of the A/D, and is used for reading voltage and resistance.

4. It is possible to download tables into RAM by direct memory access from A24. If a custom downloaded conversion will be used for channel <n>, the table for that channel should be loaded into the EU Conversion RAM at the address corresponding to that channel number.

5. When custom downloaded tables are present, that channel should be ASSIGNed a conversion type of $001F_{16}$. This conversion type code instructs the EU Converter to use the RAM table as-is, and not load a library table into that space.

6. All Piece wise EU Conversions have 16-bit dynamic range. Linear EU conversions have full dynamic range of the HP E1413.

7. Piece wise EU Conversions use y=Mx+B, where x is the low-order 9 bits, and M and B coefficients are fetched based on the high-order 7 bits of the A/D converter reading.

8. Linear EU Conversions use y=Mx+B, where x is entire 16-bit A/D reading, M and B are constants.

NEWn **0100_{16} - 0103_{16}**

Initializes scan list n to null. Any previous scan items assigned to this list are lost.

NEWL **0104_{16}**

Initializes "List of Lists" to null. Any previous Scan Lists assigned to this list are lost.

SCPCHAR <scp_number> <attributes> 0830_{16}

Sends an SCP attributes word to the Control Processor describing the characteristics of the SCP at installed at <*scp_number*>. The format of the attributes word is

15	14-6	5	4	3	2	1	0
SCP Present	Not Assigned	Gain is 1 ±2ppm	SCP can output signals	SCP can sense voltage	SCP has programmable current	SCP has programmable cutoff frequency	SCP has programmable gain

SCPGAINS 0820_{16}

Causes the Control Processor to access SCP registers to determine the channel gain settings for each ASSIGNed channel. The channel gain settings are used to relate A/D output to input voltage for EU conversions and voltage readings. Execute SCPGAINS command after establishing channel to EU conversion linkage with the ASSIGN command.

CVT Commands

CVTINIT 0420_{16}

This command causes the entire Current Value Table to be loaded with IEEE-754 Not-A-Number (7FFF FFFF hex).

Trigger System Commands

ARM <custom> 0400_{16}

This command causes the DSP to load its runtime code into fast RAM, and pre-settle the first channel of the scan list designated in the "Next List" field of the Scan Control Register. The processor then enters the "Armed" state, and indicates that it is ready to accept a scan trigger by asserting the "Armed" bit in the Scan Status Register.. The <custom> parameter should normally contain zero. If <custom> is set to a value of one, the Control Processor will load 2k of user code from location 0C00h in its Data Space RAM in place of the normal runtime routine.

SCPTRIGEN <enable> 0860_{16}

When <enable> is one (1) SCP triggers are enabled, when zero (0) SCP triggers are disabled.

TRIGCOUNT <count> 2330_{16}

Sets the number of trigger events that will be accepted. The value of <count> can be 0 to 65535. The default value is 1. Setting <count> to 0 allows unlimited trigger events to be accepted.

Debugging Commands

AVERAGE <channel> <range> <m> 0480_{16}

Takes 2^m readings (1<m<16) on the specified channel and range, and returns the mean value, and mean-squared value as IEEE floating point numbers in the FIFO buffer. The mean is returned first. No conversions are supported: the results are in raw volts, and volts^2 respectively. Reference junction compensation is not used, and the Current Value Table is unaffected. If <m> is greater than 7, the readings are taken over an even number of line cycles at 50 and 60 Hz, e.g. <m>=8 yields 256 samples spaced over .1 second. The mean-squared value is intended to be used to calculate the rms noise on a channel as follows:

$$rms\ noise=\sqrt{(mean\ squared\ value)-(mean\ value)^2}$$

Due to fixed-point arithmetic limits, overflow may occur while accumulating the mean-squared value when <m> is large. In these cases, a "very large number" is returned. Autoranging is supported by AVERAGE as follows: The first reading is taken with autoranging, and subsequent readings are taken on the same range as the first.

DSPEEK? <address> <data_page> 2040_{16}

Reads a 16-bit word from the control processor's Data Space RAM and returns it in the Query Response Register. Addressing is based on the Control Processor's memory map, and thus depends on how the data page bits (BD0-BD3) are set. Values of <data_page> between 0 and 15 inclusive, specify which page to read. If a value of $FFFF_{16}$ is specified for <data_page>, the current page is read. The value of <data_page> is irrelevant for addresses below 8000_{16}. Some low addresses may appear to have different contents when read via A24 access. This is because the control processor's internal RAM and registers are not available via A24.

DSPOKE <address> <data_page> <data> 2080_{16}

Writes the 16-bit word specified in <data> to the Control Processor's Data Space RAM. Many Control Processor registers are mapped into low RAM. Use this command carefully! The <address>, and <data_page> parameters have the same meanings as for the DSPEEK? command.

PSPEEK? <address> <code_page> <table_page> 2100_{16}

Reads a 16-bit word from the control processor's Program Space memory. Since the subsystem uses paging when accessing the Program Space, the <code_page> and <table_page> parameters are necessary. Valid values of <table_page> are 0, 1, or $FFFF_{16}$ with the latter specifying "use current page". The value of <table_page> is irrelevant for addresses below 8000_{16}. Valid values for <code_page> are 0, 1, 2, 3, or $FFFF_{16}$. The value of <code_page> matters only for the address range 4000_{16} through $7FFF_{16}$. In general, the PSPEEK? command reads flash memory, but for the address range 2000_{16} through $23FF_{16}$, it "usually" reads control processor RAM mapped to Program Space. A24 accesses can be used to view this region however.

Register Based Programming Fundamentals

Register based programming begins with a few fundamental register based operations. They are:

- Resetting the module
- General register access
- Executing register based commands
- Querying the module

These operations are explained here in detail. In the next section, "Programming Sequence" they will each be shown as a single operation.

Resetting the HP E1413

Resetting the module places it in the Ready for Program State. The module is reset according to the following figure:

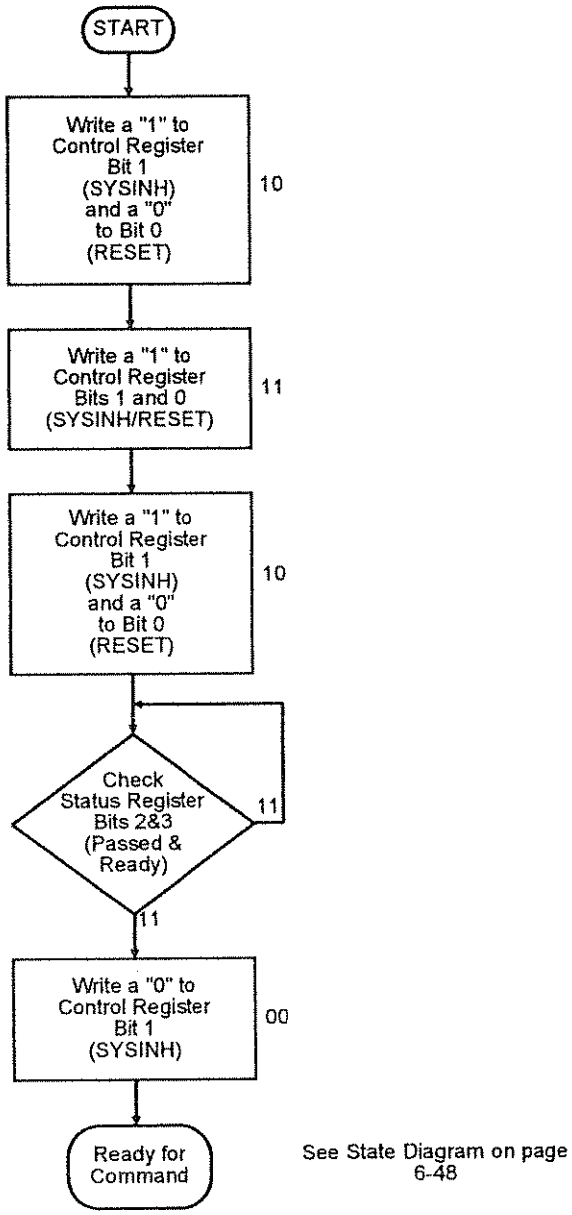


Figure D-3. Resetting the E1413A

Comments

1. The registers used are:
   - Control Register (base+04_{16})(write)
   - Status Register (base+04_{16})(read)
2. Writing a one (1) to bit 1 prevents the module from asserting the SYSFAIL line when the module is reset.

General Register Access

Any of the A16 registers can be read from while the module is in any state (see State Diagram on page D-48). You should avoid accessing the module's A24 areas other than the CVT while the module is scanning. In order to avoid unpredictable results, the Scan Control, Card Control, and Trigger Mode registers should not be written to unless the module is in the Waiting for Command State. To assure this, use the following procedure to write to these registers.

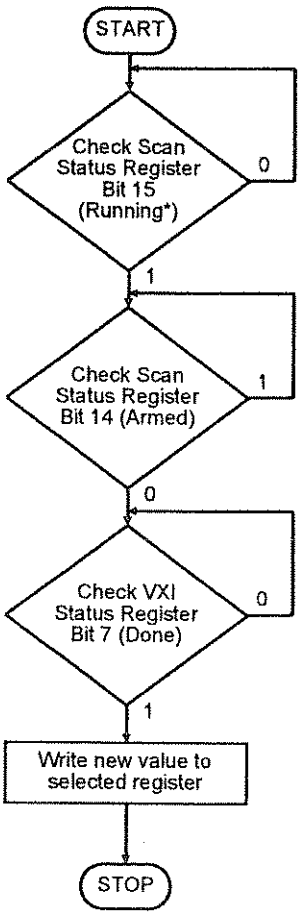


Figure D-4. General Register Access

Executing Register Based Commands

While some features of the HP E1413 are controlled by writing bit patterns to the various registers described in the previous section, other more complex operations are performed by the on-board microprocessor. Directing the microprocessor to perform these operations involves writing parameter values (if necessary) to one or more of the parameter registers and then writing the Register Based Command op code to the Command Register. Execute Register Based Commands according to the following figure:

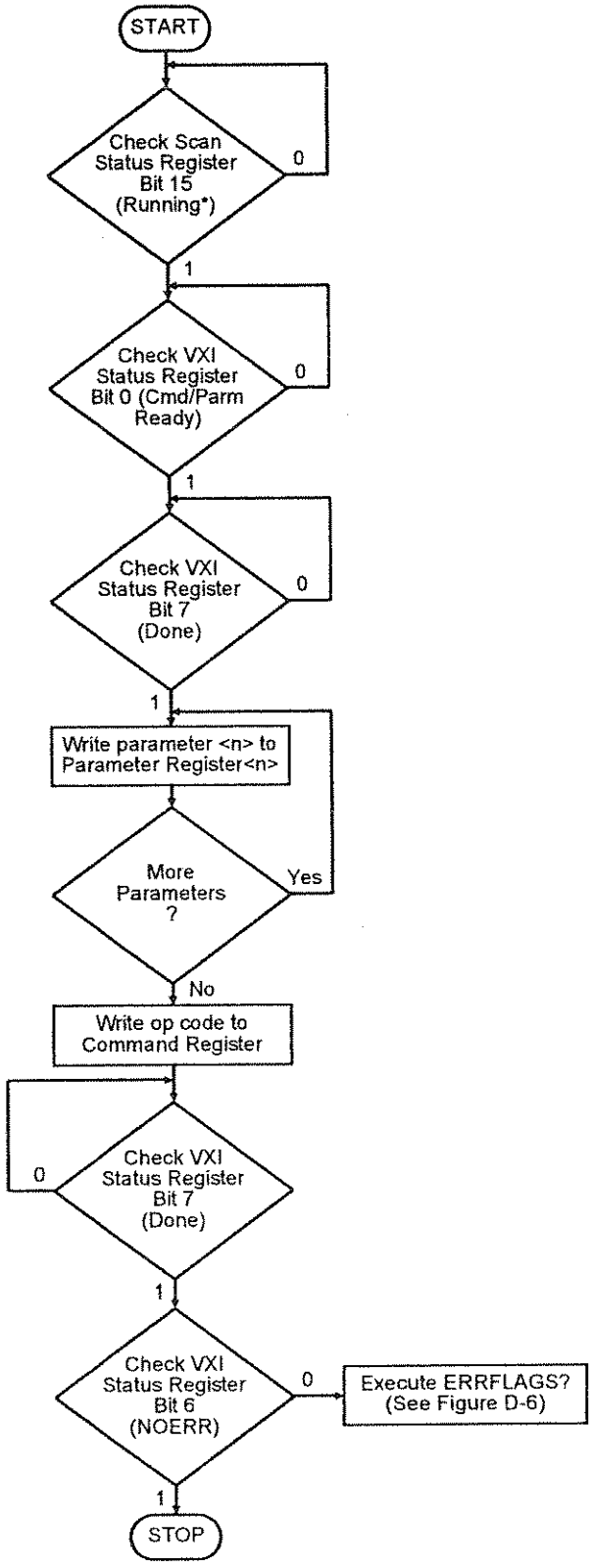


Figure D-5. Executing Commands

Comments

1. The registers used are:
   - Status Register (base+04_{16})
   - Parameter Register(s) (base+$0A_{16}$, $0C_{16}$, $0E_{16}$)
   - Command Register (base+08_{16})
   - Scan Status & Control Register (base+10_{16})
2. The parameters must be written to their registers before the command is written. Execution begins as soon as the command is written.
3. If command returns data, query for it before executing the ERRFLAGS? command.

Querying the Module

Some of the register based commands return information kept by the on-board microprocessor. When a command returns data (denoted in the Register Based Command Reference by the command name ending with a "?" character), that data will be accessed from the Query Response Register (base+08_{16}) as follows:

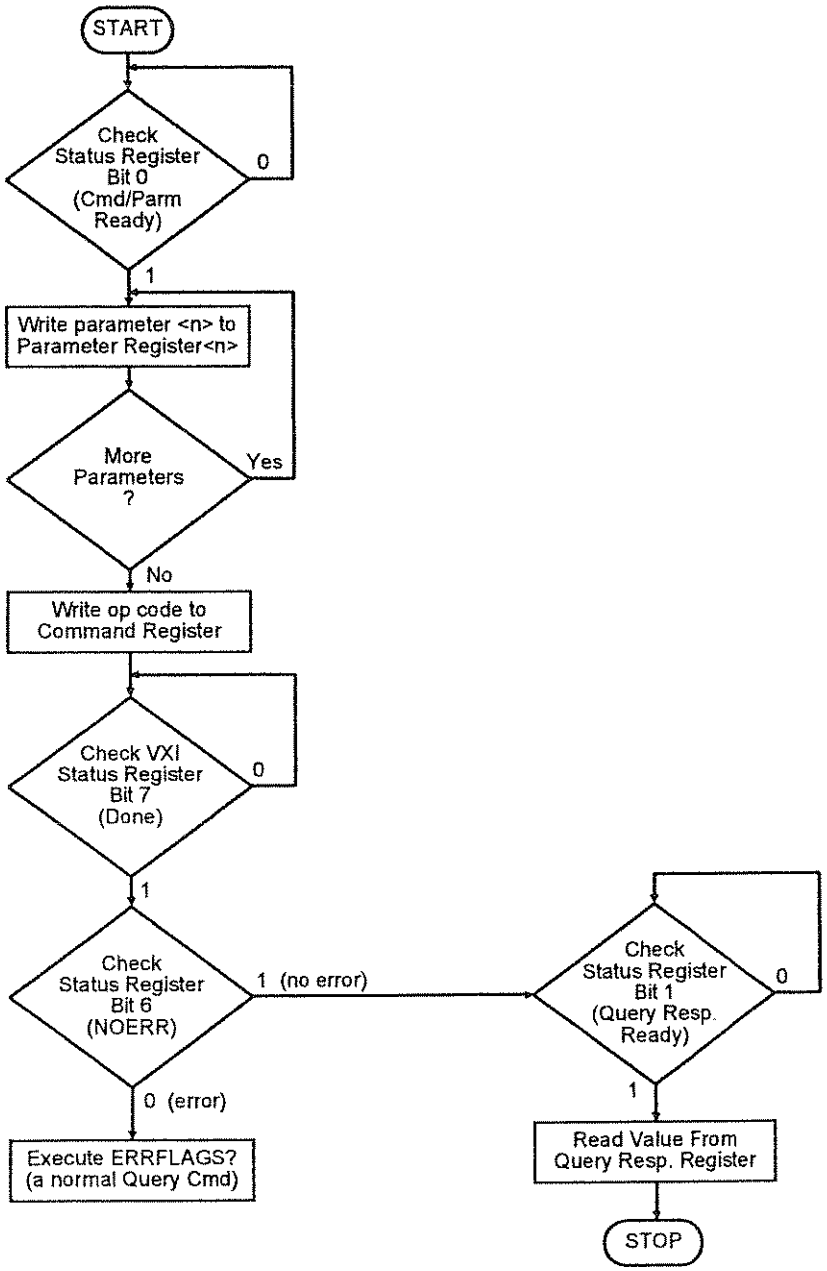


Figure D-6. Executing Queries

Comments

1. The registers used are:
   - Status Register (base+04_{16})
   - Parameter Register(s) (base+$0A_{16}$, $0C_{16}$, $0E_{16}$)
   - Command Register (base+08_{16})
   - Query Response Register (base+08_{16})
2. This procedure is simply executing a register based command (a query command) with the addition of monitoring the Query Response Register to determine when the returned value is available.

Control Processor States

The following state diagram shows the relationships between the control processor's states.

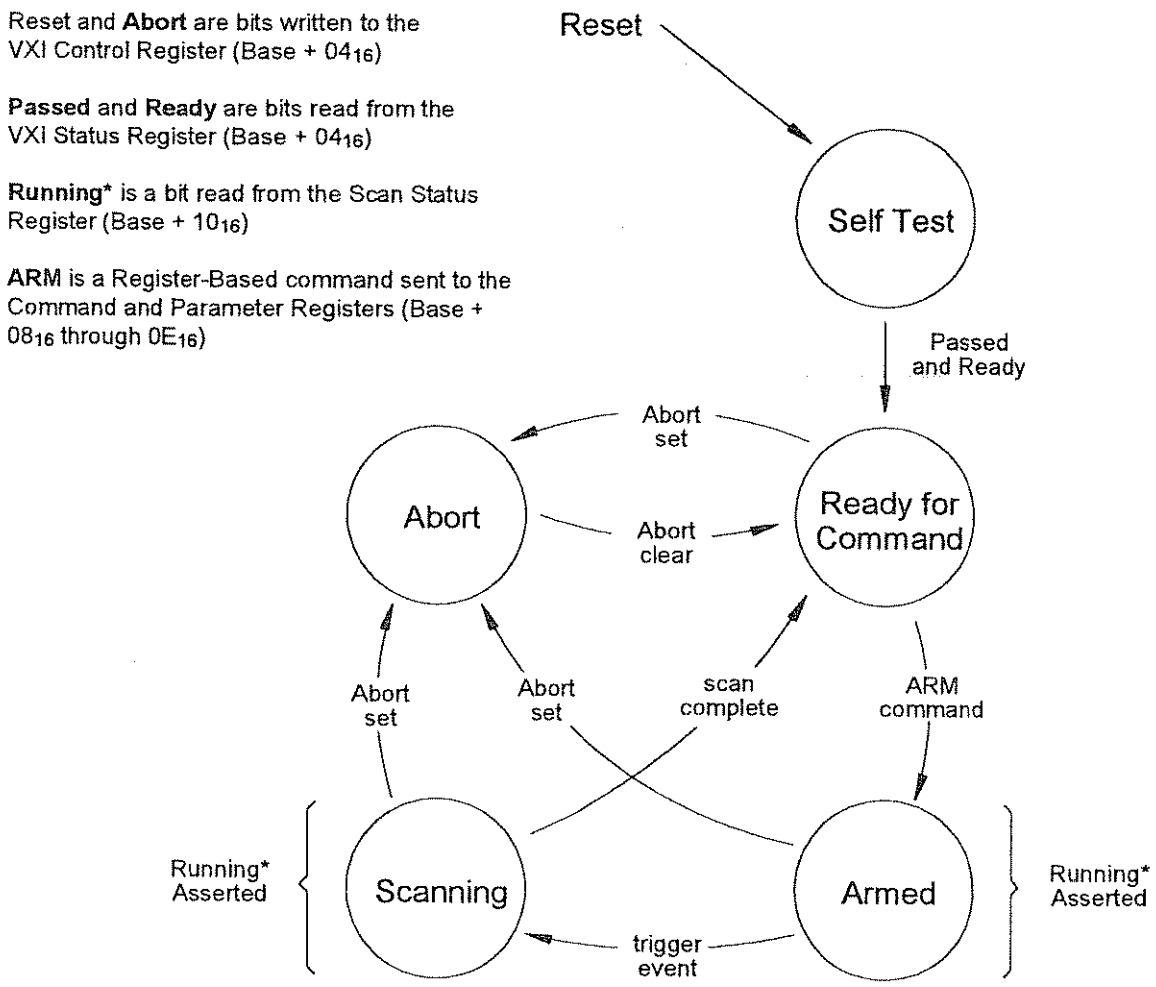


Figure D-7 Control Processor State Diagram

Programming Sequence

This guide is only a sequence of Register offsets and values to write to them. It is assumed that you have already studied the details of accessing registers and executing register-based commands in this chapter. You must follow register sequence exactly as shown.

Notes

1. See VXIbus Specification section C.2.1.1.2 and C.4.1.3 for address map configuration. Example shown programs HP E1413's A24 map to begin at address 200000_{16}
2. After every Command written to Command Register 08, the controller must wait for DONE (bit 7) and READY (bit 3) to be asserted in VXI Status register 04 before writing the next command.
3. After every Query written to Register 08, the controller must wait for DONE (bit 7) and Query Response Ready (bit 1) to be asserted in VXI Status Register 04 before reading Query Response from Query Response Register 08.
4. When sending a command which includes parameters, the Parameter Registers ($0A_{16}$, $0C_{16}$, $0E_{16}$) must be written before the Command Register 08.
5. Examples here may change to match measurement requirements.
6. ARM command does not set DONE (bit 7) in VXI Status Register 04 until measurement is complete. Wait for INITIATED (bit 13) in Scan Status Register 10_{16} before issuing any triggers.

Reset Module to Default State

Set up A24 Memory

Reg Offset (hex)	Value (hex)	
06	2000	note 1
04	FFFC	

Reset DSP Chip

Reg Offset (hex)	Value (hex)
04	8002
04	8003
04	8002

Loop reading VXI Status Reg 04 until bits 3 & 2 set (Passed and Ready)

04	8000

Reset CVT

Reg Offset (hex)	Value (hex)
08	0420 (CVTINIT cmd) note 2

Set Trigger Mode

Reg Offset (hex)	Value (hex)
2E	0008 (software trigger)

Set Trigger Count

Reg Offset (hex)	Value (hex)
0A	0001 (trigger count param) note 5
08	2330 (TRIGCOUNT cmd) notes 2,4

Interrupt Config

Reg Offset (hex)	Value (hex)
14	0001 (Int on IRQ1)

Read all SCP IDs

For each SCP address (total of eight times):

Reg Offset (hex)	Value (hex)
0A	40 X SCP Address
08	0800 (SCBREAD cmd) note 4

Read SCP ID response from Query Response Register 08 note 3

Set SCP Characteristics

For each SCP address (total of eight times):

Reg Offset (hex)	Value (hex)
0A	SCP Address (0-7)
0C	SCP attributes (see D-39)
08	0830 (SCPCHAR cmd) note 2

Programming Module After Reset Sequence

Clear FIFO

Reg Offset (hex)	Value (hex)
12	1000
12	0000

Link Channels to EU Conversions

Reg Offset (hex)	Value (hex)
0A	0000 (channel number)
0C	0000 (EU code for volts) note 5
08	0010 (ASSIGN cmd) notes 2,4
0A	0001 (channel number)
0C	0004 (EU code for ref thermistor) note 5
08	0010 (ASSIGN cmd) notes 2,4
0A	0002 (channel number)
0C	0008 (EU code for J thermocouple) note 5
08	0010 (ASSIGN cmd) notes 2,4
0A	0003 (channel number)
0C	0008 (EU code for J thermocouple) note 5
08	0010 (ASSIGN cmd) notes 2,4

	0A	0004 (channel number)
	0C	0008 (EU code for J thermocouple) note 5
	08	0010 (ASSIGN cmd) notes 2,4

Continue for all 64 channels

DSP checks SCP Gains

Reg Offset (hex)	Value (hex)
08	0820 (SCPGAINS cmd) note 2

Clear Scan List 1

Reg Offset (hex)	Value (hex)
08	0100 (NEWn cmd for Scan List 1) note 2

Define Scan List 1

Perform for each channel you want in scan list 1

Reg Offset (hex)	Value (hex)
0A	0000 (channel number)
0C	FFFF (range code , FFFF=autorange)
0E	7 (data_flags, 7=EU converted to FIFO & CVT)
08	0110 (APPEND cmd) notes 2,4

Select Scan List 1

Reg Offset (hex)	Value (hex)
10	0000 (See Scan Control Bits page D-11)

Arm the Module

Reg Offset (hex)	Value (hex)
0A	0000
08	0400 (ARM cmd) notes 2,4,6

Send Software Trigger

Reg Offset (hex)	Value (hex)
26	0001

Index

A

A16 Address space, D-4
Abbreviated Commands, 5-6
ABORt subsystem, 5-13
Accessing the CVT, 3-23
Accessing the FIFO, 3-24
Accuracy
    10K Ohm Thermistor, A-30 - A-31
    2250 Ohm Thermistor, A-26 - A-27
    5K Ohm Thermistor, A-28 - A-29
    DC Volts, A-2
    E Type Thermocouple, A-4 - A-7
    E Type Thermocouple (extended), A-8 - A-9
    J Type Thermocouple, A-10 - A-11
    K Type Thermocouple, A-12 - A-13
    R Type Thermocouple, A-14 - A-15
    Reference RTD, A-22 - A-23
    Reference Thermistor, A-20 - A-21
    RTD, A-24 - A-25
    S Type Thermocouple, A-16 - A-17
    T Type Thermocouple, A-18 - A-19
    Temperature, A-3
Additional topics, 3-25
ADDRess
    MEM:VME:ADDR, 5-56
ADDRess?
    MEM:VME:ADDR?, 5-56
Advanced FIFO data retrieval, 4-1
ALL?
    DATA:FIFO:ALL?, 5-75
AMPLitude
    OUTP:CURRent:AMPLitude, 5-59
    OUTPut:CURRent:AMPLitude?, 5-60
ARM subsystem, 5-14 - 5-16
ARM:SOURce, 5-15
ARM:SOURce?, 5-16
AVERage
    CALCulate:AVERage:COUNt?, 5-19
    CALCulate:AVERage[:STATe], 5-18
    CALCulate:AVERage[:STATe]?, 5-18

B

Base Address, D-4
    computing, D-4
    HP Command Module, D-5
Bulletin Board, HP T&M Help Line, 1-9
Byte, enabling events to the status, 4-22
Byte, reading the status, 4-22

C

CAL:CONF:RES, 5-30
CAL:CONF:VOLT, 5-30
CAL:SETup, 5-31
CAL:SETup?, 5-31
CAL:STORe, 5-32
CAL:TARE, 5-33
CAL:TARE, maximum capability, 4-28
CAL:TARE, Special Considerations, 4-28
CAL:TARE:RESet, 4-28, 5-35
CAL:TARE?, 5-35
CAL:VAL:RESistance, 5-35
CAL:VAL:VOLTage, 5-36
CAL:ZERO?, 5-37
CALCulate subsystem, 5-17 - 5-27
CALCulate:AVERage:COUNt, 5-18
CALCulate:AVERage:COUNt?, 5-19
CALCulate:AVERage[:STATe], 5-18
CALCulate:AVERage[:STATe]?, 5-18
CALCulate:CLIMits:FAIL?, 5-19 - 5-20
CALCulate:CLIMits:FLIMits?, 5-20
CALCulate:CLIMits:FLIMits:CURRent?, 5-21
CALCulate:CLIMits:FLIMits:POINts?, 5-21
CALCulate:LIMit:FAIL?, 5-23
CALCulate:LIMit:LOWer:DATA, 5-24
CALCulate:LIMit:LOWer:DATA?, 5-25
CALCulate:LIMit:LOWer[:STATe], 5-24
CALCulate:LIMit:LOWer[:STATe]?, 5-24
CALCulate:LIMit:UPPer:DATA, 5-26
CALCulate:LIMit:UPPer:DATA?, 5-27
CALCulate:LIMit:UPPer[:STATe], 5-25
CALCulate:LIMit:UPPer[:STATe]?, 5-26
CALCulate:LIMit[:STATe], 5-22
CALCulate:LIMit[:STATe]?, 5-22
CALibration subsystem, 5-28 - 5-37

Calibration, channel
*CAL?, 5-111
Calibration, control of, 1-7
Calibration, performing channel, 3-16
Channel calibration
*CAL?, 5-111
CHECksum?
DIAG:CHECK?, 5-38
Choosing the data retrieval method, 4-4
Clearing event registers, 4-24
Clearing the enable registers, 4-23
CLIMits:FAIL
CALCulate:CLIMits:FAIL?, 5-19 - 5-20
Command
Abbreviated, 5-6
Implied, 5-7
Linking, 5-10
Separator, 5-6
Command Quick Reference, 5-117 - 5-121
Command Reference, Common
*CAL?, 5-111
*CLS, 5-111
*ESE, 5-112
*ESE?, 5-112
*ESR?, 5-112
*IDN?, 5-112
*OPC, 5-113
*OPC?, 5-113
*RST, 5-113
*SRE, 5-114
*SRE?, 5-114
*STB?, 5-114
*TRG, 5-115
*TST?, 5-115
*WAI, 5-116
Command reference, register-based, D-25
Command Reference, SCPI, 5-12
CALCulate:LIMit, 5-22
ABORt subsystem, 5-13
ARM subsystem, 5-14 - 5-16
ARM:IMMediate, 5-15
ARM:SOURce, 5-15
ARM:SOURce?, 5-16
CALCulate subsystem, 5-17 - 5-27
CALCulate:AVERage:COUNt, 5-18
CALCulate:AVERage:COUNt?, 5-19
CALCulate:AVERage[:STATe], 5-18
CALCulate:AVERage[:STATe]?, 5-18
CALCulate:CLIMits:FAIL?, 5-19 - 5-20
CALCulate:CLIMits:FLIMits?, 5-20
CALCulate:CLIMits:FLIMits:CURRent?, 5-21
CALCulate:CLIMits:FLIMits:POINts?, 5-21
CALCulate:LIMit:FAIL?, 5-23
CALCulate:LIMit:LOWer:DATA, 5-24
CALCulate:LIMit:LOWer:DATA?, 5-25
CALCulate:LIMit:LOWer[:STATe], 5-24
CALCulate:LIMit:LOWer[:STATe]?, 5-24
CALCulate:LIMit:UPPer:DATA, 5-26
CALCulate:LIMit:UPPer:DATA?, 5-27
CALCulate:LIMit:UPPer[:STATe], 5-25
CALCulate:LIMit:UPPer[:STATe]?, 5-26
CALCulate:LIMit[:STATe], 5-22
CALibration subsystem, 5-28 - 5-37
CALibration:CONFigure:RESistance, 5-30
CALibration:CONFigure:VOLTage, 5-30
CALibration:SETup, 5-31
CALibration:SETup?, 5-31
CALibration:STORe, 5-32
CALibration:TARE, 5-33
CALibration:TARE:RESet, 5-35
CALibration:TARE?, 5-35
CALibration:VALue:RESistance, 5-35
CALibration:VALue:VOLTage, 5-36
CALibration:ZERO?, 5-37
DIAGnostic subsystem, 5-38 - 5-44
DIAGnostic:CHECksum?, 5-38
DIAGnostic:COMMand:SCPWRITE, 5-38
DIAGnostic:CUSTom:LINear, 5-39
DIAGnostic:CUSTom:PIECewise, 5-40
DIAGnostic:CUSTom:REFerence:TEMPerature, 5-41
DIAGnostic:INTerrupt:LINe, 5-41
DIAGnostic:INTerrupt:LINe?, 5-41
DIAGnostic:OTDetect:STATe, 5-42
DIAGnostic:OTDetect:STATe?, 5-43
DIAGnostic:QUERy:SCPREAD, 5-43
DIAGnostic:VERSion?, 5-43
FETCh?, 5-45
FETCh? subsystem, 5-45 - 5-46
FORMat subsystem, 5-47 - 5-48
FORMat:DATA, 5-47
FORMat:DATA?, 5-48
INITiate subsystem, 5-49 - 5-50
INITiate:CONTinuous, 5-49
INITiate:IMMediate, 5-50
INPut subsystem, 5-51 - 5-54
INPut:FILTer:LPASs:FREQuency?, 5-52
INPut:FILTer:LPASs:STATe, 5-52
INPut:FILTer:LPASs:STATe?, 5-53
INPut:GAIN, 5-53
INPut:GAIN?, 5-54
INPut:LPASs:FILTer:FREQuency, 5-51
MEMory subsystem, 5-55 - 5-58
MEMory:VME:ADDRess, 5-56
MEMory:VME:ADDRess?, 5-56
MEMory:VME:SIZE, 5-56
MEMory:VME:SIZE?, 5-57

MEMory:VME:STATe, 5-57
MEMory:VME:STATe?, 5-58
OUTPut subsystem, 5-59 - 5-64
OUTPut:CURRent:AMPLitude, 5-59
OUTPut:CURRent:AMPLitude?, 5-60
OUTPut:CURRent:STATe, 5-60
OUTPut:CURRent:STATe?, 5-61
OUTPut:SHUNt, 5-61
OUTPut:SHUNt?, 5-62
OUTPut:TTLTrg:SOURce, 5-62
OUTPut:TTLTrg:SOURce?, 5-63
OUTPut:TTLTrg :STATe, 5-64
OUTPut:TTLTrg :STATe?, 5-64
ROUTe subsystem, 5-65 - 5-70
ROUTe:SCAN, 5-65
ROUTe:SEQuence:DEFine, 5-66
ROUTe:SEQuence:DEFine?, 5-68
ROUTe:SEQuence:POINts?, 5-69
SAMPle subsystem, 5-71 - 5-72
SAMPle:TIMer, 5-71
SAMPle:TIMer?, 5-72
SENSe subsystem, 5-73 - 5-95
SENSe:DATA:COUN:HALF?, 5-76
SENSe:DATA:CVTable:RESet, 5-75
SENSe:DATA:CVTable?, 5-74
SENSe:DATA:FIFO:ALL?, 5-75
SENSe:DATA:FIFO:COUNt?, 5-76
SENSe:DATA:FIFO:HALF?, 5-77
SENSe:DATA:FIFO:MODE, 5-78
SENSe:DATA:FIFO:MODE?, 5-78
SENSe:DATA:FIFO:PART?, 5-79
SENSe:DATA:FIFO:RESet, 5-80
SENSe:FILTer:LPASs:STATe, 5-80
SENSe:FILTer:LPASs:STATe?, 5-80
SENSe:FUNCtion:CUSTom, 5-81
SENSe:FUNCtion:CUSTom:REFerence, 5-81
SENSe:FUNCtion:CUSTom:TCouple, 5-83
SENSe:FUNCtion:RESistance, 5-84
SENSe:FUNCtion:STRain:FBEN, 5-85
SENSe:FUNCtion:STRain:FBP, 5-85
SENSe:FUNCtion:STRain:FPO, 5-85
SENSe:FUNCtion:STRain:HBEN, 5-85
SENSe:FUNCtion:STRain:QUAR, 5-85
SENSe:FUNCtion:STRainHPO:, 5-85
SENSe:FUNCtion:TEMPerature, 5-87
SENSe:FUNCtion:VOLTage, 5-88
SENSe:REFerence, 5-89
SENSe:REFerence:TEMPerature, 5-91
SENSe:STRain:EXCitation, 5-91
SENSe:STRain:EXCitation?, 5-92
SENSe:STRain:GFACtor, 5-93
SENSe:STRain:GFACtor?, 5-93
SENSe:STRain:POISson, 5-94
SENSe:STRain:POISson?, 5-94
SENSe:STRain:UNSTrained, 5-94
SENSe:STRain:UNSTrained?, 5-95
STATus subsystem, 5-96 - 5-102
STATus:OPERation:CONDition?, 5-98
STATus:OPERation:ENABle, 5-98
STATus:OPERation:ENABle?, 5-99
STATus:OPERation:EVENt?, 5-99
STATus:PRESet, 5-100
STATus:QUEStionable:CONDition?, 5-101
STATus:QUEStionable:ENABle, 5-101
STATus:QUEStionable:ENABle?, 5-102
STATus:QUEStionable:EVENt?, 5-102
SYSTem subsystem, 5-103 - 5-104
SYSTem:CTYPe?, 5-103
SYSTem:ERRor?, 5-103
SYSTem:VERSion?, 5-104
TRIGger subsystem, 5-105 - 5-110
TRIGger:COUNt, 5-106
TRIGger:COUNt?, 5-106
TRIGger:IMMediate, 5-107
TRIGger:SOURce, 5-107
TRIGger:SOURce?, 5-108
TRIGger:TIMer, 5-109
TRIGger:TIMer:MODE, 5-108
TRIGger:TIMer:MODE?, 5-109
TRIGger:TIMer?, 5-110
Command sequences, defined, 1-9
Commands, FIFO reading transfer, 4-2
Commands, FIFO status, 4-2
Common Command Format, 5-6
Common mode rejection, A-1
Common mode voltage
  Maximum, A-1
Compensating for system wiring offsets, 4-27
Compensation, thermocouple reference, 4-29
CONDition
  STAT:OPER:CONDition?, 5-98
CONDition?
  STAT:QUES:CONDition?, 5-101
Configuring the HP E1413, 1-1
Connection
  recommended, 2-6
  signals to channels, 2-6
CONTinuous
  INIT:CONT, 5-49
Controlled reading count, 4-4
Controlling data conversion and destination, 4-7
Conversion, EU, C-1
Conversion, linking channels to EU, 3-7
COUNt
  CALCulate:AVERage:COUNt, 5-18
Count, controlled reading, 4-4

COUNt?
SENS:DATA:FIFO:COUNt?, 5-76
CTYPe?
SYST:CTYPe?, 5-103
CURRent
CALCulate:CLIMits:FLIMits:CURRent?, 5-21
Current Value Table
SENSe:DATA:CVTable?, 5-74
CUSTom
SENS:FUNC:CUSTom, 5-81
Custom, Linking Custom EU Tables, 3-15
Cutoff, setting filter, 3-6
CVT
Accessing the CVT, 3-23
Resetting the CVT, 3-24
SENSe:DATA:CVTable?, 5-74

D

DATA
CALCulate:LIMit:LOWer:DATA, 5-24
CALCulate:LIMit:UPPer:DATA, 5-26
FORMat:DATA, 5-47
FORMat:DATA?, 5-48
Data conversion/destination, controlling, 4-7
Data, retrieving, 3-23
DATA:FIFO:ALL?, 5-75
DEFine
ROUT:SEQ:DEF, 5-66
ROUT:SEQ:DEF?, 5-68
Description, module, 3-3
Detecting open transducers, 4-29
DIAG:CHECK?, 5-38
DIAG:CUST:REF:TEMP, 5-41
DIAG:INT:LINe, 5-41
DIAG:INT:LINe?, 5-41
DIAG:OTD:STATe, 5-42
DIAG:OTD:STATe?, 5-43
DIAG:VERSion?, 5-43
DIAGnostic
DIAGnostic:COMMand:SCPWRITE, 5-38
DIAGnostic:CUSTom:LINear, 5-39
DIAGnostic:CUSTom:PIECewise, 5-40
DIAGnostic:QUERy:SCPREAD, 5-43
DIAGnostic:COMMand:SCPWRITE, 5-38
DIAGnostic:CUSTom:LINear, 5-39
DIAGnostic:CUSTom:PIECewise, 5-40
DIAGnostic:QUERy:SCPREAD, 5-43
Directly, reading status groups, 4-24
Disabling flash memory access (optional), 1-7
Disabling the input protect feature (optional), 1-7
DSP, C-1
Dummy channel
Channel Data Modifier 7, 4-8
Data Conversion and Destination, 4-7

E

ENABle
STAT:OPER:ENABle, 5-98
STAT:QUES:ENABle, 5-101
ENABle?
STAT:OPER:ENABle?, 5-99
STAT:QUES:ENABle?, 5-102
Enabling events to the status byte , 4-22
Error Messages, B-1
Self Test, B-4
ERRor?
SYST:ERRor?, 5-103
EU, C-1
EU Conversion, C-1
EVENt?
STAT:OPER:EVENt?, 5-99
STAT:QUES:EVENt?, 5-102
Example program, 3-25
Example programs from the HP BBS, 1-9
Example programs, about, 1-9
Examples, operation status group, 4-22
Examples, questionable data group, 4-22
Examples, standard event group, 4-22
EXCitation
SENSe:STRain:EXCitation, 5-91
SENSe:STRain:EXCitation?, 5-92

F

FAIL
CALCulate:LIMit:FAIL?, 5-23
Fastest reading transfer, 4-6
FIFO reading transfer commands, 4-2
FIFO status commands, 4-2
FIFO, accessing the, 3-24
Flash Memory, C-1
Flash memory access, disabling, 1-7
FLIMits
CALCulate:CLIMits:FLIMits?, 5-20
Format
Common Command, 5-6
SCPI Command, 5-6
Format, specifying the data, 3-21
FORMat:DATA, 5-47
FORMat:DATA?, 5-48
FREQuency
INPut:FILT:FREQ, 5-51
FREQuency?
INP:FILT:FREQ?, 5-52

G

GAIN
channel, 5-111
INPut:GAIN, 5-53
GAIN?
INP:GAIN?, 5-54
Gains, setting SCP, 3-5
GFACtor
SENSe:STRain:GFACtor, 5-93
SENSe:STRain:GFACtor?, 5-93
Glossary, C-1

H

HALF?
SENS:DATA:FIFO:COUNt:HALF?, 5-76
SENS:DATA:FIFO:HALF?, 5-77
Help Line, HP T&M Bulletin Board, 1-9
HP E1413, configuring the, 1-1

I

IEEE +/- INF, 5-48
IMMediate
ARM:IMMediate, 5-15
INIT:IMM, 5-50
TRIG:IMMediate, 5-107
Impedance, input, A-1
Implied Commands, 5-7
INF, IEEE, 5-48
INIT:CONT, 5-49
INIT:IMM, 5-50
INITiate subsystem, 5-49 - 5-50
INP:FILT:FREQ?, 5-52
INP:FILT:LPAS:STAT, 5-52
INP:FILT:LPAS:STAT?, 5-53
INP:GAIN?, 5-54
Input impedance, A-1
Input protect feature, disabling, 1-7
INPut subsystem, 5-51 - 5-54
INPut:FILT:FREQ, 5-51
INPut:GAIN, 5-53
Installing signal conditioning plug-ons, 1-3
Inut voltage, maximum, A-1

L

LIMit
CALCulate:LIMit:LOWer[:STATe], 5-24
CALCulate:LIMit:LOWer[:STATe]?, 5-24
CALCulate:LIMit:UPPer[:STATe], 5-25
CALCulate:LIMit:UPPer[:STATe]?, 5-26
CALCulate:LIMit[:STATe], 5-22
LIMit:LOWer:DATA
CALCulate:LIMit:LOWer:DATA?, 5-25
LIMit:UPPer:DATA
CALCulate:LIMit:UPPer:DATA?, 5-27
LINe
DIAG:INT:LINe, 5-41
LINe?
DIAG:INT:LINe?, 5-41
Linking Commands, 5-10
Linking Custom EU Tables, 3-15
Linking resistance measurements, 3-8
Linking strain measurements, 3-13
Linking temperature measurements, 3-10
Linking voltage measurements, 3-8
Lists, defining the scan, 3-17

M

Maximum
Common mode voltage, A-1
Input voltage, A-1
Reading rate, A-1
Tare cal offset, A-2
Maximum CAL:TARE Offsets, 4-28
Maximum tare capability, 4-28
Measurement
Ranges, A-1
Resolution, A-1
Measurements, linking resistance, 3-8
Measurements, linking strain, 3-13
Measurements, linking temperature, 3-10
Measurements, linking voltage, 3-8
Measurements, thermocouple, 3-11
Measuring the reference temperature, 3-12
Meaurement
Accuracy DC Volts, A-2
MEM:VME:ADDR, 5-56
MEM:VME:ADDR?, 5-56
MEM:VME:SIZE, 5-56
MEM:VME:SIZE?, 5-57
MEM:VME:STATe, 5-57
MEM:VME:STATe?, 5-58
Messages, error, B-1
Method, choosing the data retrieval, 4-4
MODE
SENS:DATA:FIFO:MODE, 5-78
TRIG:TIM:MODE, 5-108
TRIG:TIM:MODE?, 5-109
Mode, selecting the FIFO, 3-22
MODE?
SENS:DATA:FIFO:MODE?, 5-78

Modes, Trigger, 4-11
Module description, 3-3

N

NaN, 5-48
Not-a-Number, 5-48

O

Offset
    A/D, 5-31, 5-111
    channel, 5-31, 5-111
Offsets, compensating for system wiring, 4-27
Offsets, unexpected, 4-28
Opening and wiring the terminal module, 2-8
Operation status group examples, 4-22
OUTP:CURRent:AMPLitude, 5-59
OUTP:CURRent:AMPLitude?, 5-60
OUTP:SHUNt, 5-61
OUTP:SHUNt?, 5-62
OUTP:TTLT :STATe, 5-64
OUTP:TTLT :STATe?, 5-64
OUTPut subsystem, 5-59 - 5-64
OUTPut:CURRent:STATe, 5-60
OUTPut:CURRent:STATe?, 5-61
OUTPut:TTLTrg:SOURce, 5-62
OUTPut:TTLTrg:SOURce?, 5-63
Overload readings, unexpected, 4-28

P

Paced scans, timer, 4-12
Parameter data and returned value types, 5-10
PART?
    SENS:DATA:FIFO:PART?, 5-79
Planning
    grouping channels to signal conditioning, 2-2
    planning wiring layout, 2-1
    sense vs. output SCPs, 2-3
    thermocouple wiring, 2-4
Plug-ons, installing signal conditioning, 1-3
Plug-ons, setting up signal conditioning, 3-5
POINts
    CALCulate:CLIMits:FLIMits:POINts?, 5-21
    ROUT:SEQ:POINts?, 5-69
POISson
    SENSe:STRain:POISson, 5-94
    SENSe:STRain:POISson?, 5-94
PRESet
    STAT:PRESet, 5-100
Program, example, 3-25
Programming sequence, 3-4

Q

Questionable data group examples, 4-22
Quick Reference, Command, 5-117 - 5-121

R

Ranges, measurement, A-1
Reading condition registers, 4-24
Reading event registers, 4-24
Reading rate, maximum, A-1
Reading status groups directly, 4-24
Reading the status byte, 4-22
Recommended measurement connections, 2-6
Reducing settling waits, 4-31
REFerence
    SENS:FUNC:CUST:REF, 5-81
    SENS:REFerence, 5-89
Reference junction, 2-7
Reference temperature sensor connections, 2-5
Register address space, D-4
Register addressing, D-4
Register addressing, base address, D-4
Register map, D-4
Register, the status byte group's enable, 4-23
Register-based command reference, D-25
Registers, clearing event, 4-24
Registers, clearing the enable, 4-23
Registers, reading condition, 4-24
Registers, reading event, 4-24
Rejection, common mode, A-1
Reset
    *RST, 5-113
    Resetting the CVT, 3-24
    SENS:DATA:CVT:RESet, 5-75
    SENS:DATA:FIFO:RESet, 5-80
RESistance
    CAL:CONF:RES, 5-30
    CAL:VAL:RESistance, 5-35
    SENS:FUNC:RESistance, 5-84
Resolution, measurement, A-1
Retrieval, advanced FIFO data, 4-1
ROUT:SCAN, 5-65
ROUT:SEQ:DEF, 5-66
ROUT:SEQ:DEF?, 5-68
ROUT:SEQ:POINts?, 5-69
ROUTe subsystem, 5-65 - 5-70
RTD and thermistor measurements, 3-10

S

SAMP:TIMer, 5-71
SAMP:TIMer?, 5-72
SAMPle subsystem, 5-71 - 5-72
SCAN
  ROUT:SCAN, 5-65
Scan Mode
  Timer Paced, 4-12
Scanning
  Continuous mode, 4-8
  Counted mode, 4-8
  Default mode, 4-8
  reducing settling waits, 4-31
SCP, C-1
  grouping channels to signal conditioning, 2-2
  sense vs. output SCPs, 2-3
SCPI Commands, 5-1
  Format, 5-6
Selecting the trigger source, 3-19
Selecting timer arm source, 3-20
Self Test, error messages, B-4
SENS:DATA:CVT:RESet, 5-75
SENS:DATA:FIFO:COUNt:HALF?, 5-76
SENS:DATA:FIFO:COUNt?, 5-76
SENS:DATA:FIFO:HALF?, 5-77
SENS:DATA:FIFO:MODE, 5-78
SENS:DATA:FIFO:MODE?, 5-78
SENS:DATA:FIFO:PART?, 5-79
SENS:DATA:FIFO:RESet, 5-80
SENS:FILT:LPAS:STATe, 5-80
SENS:FILT:LPAS:STATe?, 5-80
SENS:FUNC:CUST:REF, 5-81
SENS:FUNC:CUST:TC, 5-83
SENS:FUNC:RESistance, 5-84
SENS:FUNC:STRain, 5-85
SENS:FUNC:TEMPerature, 5-87
SENS:FUNC:VOLTage, 5-88
SENS:REF:TEMPerature, 5-91
SENS:REFerence, 5-89
SENSe subsystem, 5-73 - 5-95
SENSe:DATA:CVTable?, 5-74
SENSe:FUNC:CUSTom, 5-81
SENSe:STRain:EXCitation, 5-91
SENSe:STRain:EXCitation?, 5-92
SENSe:STRain:GFACtor, 5-93
SENSe:STRain:GFACtor?, 5-93
SENSe:STRain:POISson, 5-94
SENSe:STRain:POISson?, 5-94
SENSe:STRain:UNSTrained, 5-94
SENSe:STRain:UNSTrained?, 5-95
Separator, command, 5-6
Sequence, programming, 3-4
Setting current sources, 3-6
Setting filter cutoff, 3-6
Setting SCP gains, 3-5
Setting the logical address switch, 1-2
Settling
  reducing settling waits, 4-31
SETup
  CAL:SETup, 5-31
  CAL:SETup?, 5-31
SHUNt
  OUTP:SHUNt, 5-61
  OUTPut:SHUNt?, 5-62
Signal, connection to channels, 2-6
SIZE
  MEM:VME:SIZE, 5-56
SIZE?
  MEM:VME:SIZE?, 5-57
SOURce
  ARM:SOURce, 5-15
  ARM:SOURce?, 5-16
  OUTPut:TTLTrg:SOURce, 5-62
  TRIG:SOURce, 5-107
Source, selecting the trigger, 3-19
Source, selecting timer arm, 3-20
SOURce?
  TRIG:SOURce?, 5-108
Sources
  arm, 3-19
  trigger, 3-19
Sources, setting current, 3-6
Special Considerations, CAL:TARE, 4-28
Specifications, A-1
Standard Commands for Programmable Instruments, SCPI, 5-12
Standard event group examples, 4-22
STAT:OPER:CONDition?, 5-98
STAT:OPER:ENABle, 5-98
STAT:OPER:ENABle?, 5-99
STAT:OPER:EVENt?, 5-99
STAT:PRESet, 5-100
STAT:QUES:CONDition?, 5-101
STAT:QUES:ENABle, 5-101
STAT:QUES:ENABle?, 5-102
STAT:QUES:EVENt?, 5-102
STATe
  DIAG:OTD:STATe, 5-42
  DIAG:OTD:STATe?, 5-43
  INP:FILT:LPAS:STATe, 5-52
  INP:FILT:LPAS:STATe?, 5-53
  MEM:VME:STATe, 5-57
  MEM:VME:STATe?, 5-58
  OUTPut:CURRent:STATe, 5-60

OUTPut:CURRent:STATe?, 5-61
SENS:FILT:LPAS:STATe, 5-80
SENS:FILT:LPAS:STATe?, 5-80
Status bit precedence, D-8
STATus subsystem, 5-96 - 5-102
Step 1 setting up signal conditioning plug-ons, 3-5
Step 10 retrieving data, 3-23
Step 2 linking channels to EU conversion, 3-7
Step 3 performing channel calibration, 3-16
Step 4 defining the scan lists, 3-17
Step 5 setting the sample timer, 3-18
Step 6 setting up the trigger system, 3-19
Step 7 specifying the data format, 3-21
Step 8 selecting the FIFO mode, 3-22
Step 9 initiating the trigger system, 3-22
STORe
CAL:STORe, 5-32
STRain
SENS:FUNC:STRain, 5-85
Sub subsystem, 5-38 - 5-48, 5-55 - 5-58
Subsystem
ABORT, 5-13
ARM, 5-14 - 5-16
CALCulate, 5-17 - 5-27
CALibration, 5-28 - 5-37
DIAGnostic, 5-38 - 5-44
FETCh?, 5-45 - 5-46
FORMat, 5-47 - 5-48
INITiate, 5-49 - 5-50
INPut, 5-51 - 5-54
MEMory, 5-55 - 5-58
OUTPut, 5-59 - 5-64
ROUTe, 5-65 - 5-70
SAMPle, 5-71 - 5-72
SENSe, 5-73 - 5-95
STATus, 5-96 - 5-102
SYSTem, 5-103 - 5-104
TRIGger, 5-105 - 5-110
Supplying the reference temperature, 3-13
Switch, setting the logical address, 1-2
Syntax, Variable Command, 5-7
SYST:CTYPe?, 5-103
SYST:ERRor?, 5-103
SYST:VERSion?, 5-104
SYSTem subsystem, 5-103 - 5-104
System, initiating the trigger, 3-22
System, using the status, 4-21

T

TARE
CAL:TARE:RESet, 5-35
CAL:TARE?, 5-35
Tare cal offset, maximum, A-2
TARE?
CAL:TARE, 5-33
TCouple
SENS:FUNC:CUST:TC, 5-83
TEMPerature
DIAG:CUST:REF:TEMP, 5-41
SENS:FUNC:TEMPerature, 5-87
SENS:REF:TEMPerature, 5-91
Temperature accuracy, A-3
Temperature sensor connections, 2-5
Temperature, measuring the reference, 3-12
Temperature, supplying the reference, 3-13
Temperature, thermocouple reference, 3-12
Terminal Blocks, C-1
Terminal Module, C-1
description, 2-7
Terminal module, opening and wiring, 2-8
The status byte group's enable register, 4-23
Thermistor and RTD measurements, 3-10
Thermocouple measurements, 3-11
Thermocouple reference compensation, 4-29
Thermocouple reference temperature, 3-12
Timer
SAMP:TIMer, 5-71
SAMP:TIMer?, 5-72
TRIG:COUNt, 5-106
TRIG:TIMer, 5-109
Timer Paced Scans, 4-12
Timer, setting the sample, 3-18
TIMer?
TRIG:TIMer?, 5-110
Topics, additional, 3-25
Transducers, detecting open, 4-29
Transfer, fastest reading, 4-6
TRIG:COUNt, 5-106
TRIG:COUNt?, 5-106
TRIG:IMMediate, 5-107
TRIG:SOURce, 5-107
TRIG:SOURce?, 5-108
TRIG:TIM:MODE, 5-108
TRIG:TIM:MODE?, 5-109
TRIG:TIMer, 5-109
TRIG:TIMer?, 5-110
Trigger Modes, 4-11
TRIGger subsystem, 5-105 - 5-110
Trigger system
ABORt subsystem, 5-13
ARM subsystem, 5-14 - 5-16
INITiate subsystem, 5-49 - 5-50
Step 6 setting up, 3-19
TRIGger subsystem, 5-105 - 5-110
TTLTrg:SOURce

OUTPut:TTLTrg:SOURce?, 5-63
TTLTrg
OUTP:TTLT :STATe?, 5-64
OUTP:TTLTrg :STATe, 5-64

U

UNSTrained
SENSe:STRain:UNSTrained, 5-94
SENSe:STRain:UNSTrained?, 5-95
Using the status system, 4-21

V

Value types
parameter data, 5-10
returned, 5-10
Variable Command Syntax, 5-7
Verifying a successful configuration, 1-10
VERsion
DIAG:VERSion?, 5-43
VERsion?
SYST:VERSion?, 5-104
VOLTage
CAL:CONF:VOLT, 5-30
CAL:VALue:VOLTage, 5-36
SENS:FUNC:VOLTage, 5-88

W

Waits, reducing settling, 4-31
Wiring
planning for thermocouple, 2-4
planning layout, 2-1
signal connection, 2-6

Z

ZERO?
CAL:ZERO?, 5-37

SALES & SUPPORT OFFICES
Arranged alphabetically by country

HEADQUARTERS OFFICES
If there is no sales office listed for your area, contact one of these headquarters offices.

NORTH/CENTRAL AFRICA
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
CH-1217 **MEYRIN** 1, Switzerland
Tel: (022) 83 12 12
Telex: 27835 hmea
Cable: HEWPACKSA Geneve

ASIA
Hewlett-Packard Asia Ltd.
47/F, 26 Harbour Rd.,
Wanchai, **HONG KONG**
P.O. Box 863, Hong Kong
Tel: 5-8330833
Telex: 76793 HPA HX
Cable: HPASIAL TD

EASTERN EUROPE
Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O.Box 72
A-1222 **VIENNA**, Austria
Tel: (222) 2500-0
Telex: 1 3 4425 HEPA A

NORTHERN EUROPE
Hewlett-Packard S.A.
V. D. Hooplaan 241
P.O Box 999
NL-1183 AG **AMSTELVEEN**
The Netherlands
Tel: 20 547999
Telex: 18 919 hpner

SOUTH EAST EUROPE
Hewlett-Packard S.A.
World Trade Center
110 Avenue Louis Casai
1215 Cointrin, **GENEVA**, Switzerland
Tel: (022) 98 96 51
Telex: 27225 hpser

EASTERN USA
Hewlett-Packard Co.
4 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370

MIDWESTERN USA
Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800

SOUTHERN USA
Hewlett-Packard Co.
2000 South Park Place
ATLANTA, GA 30339
Tel: (404) 955-1500

WESTERN USA
Hewlett-Packard Co.
5161 Lankershim Blvd.
NORTH HOLLYWOOD, CA 91601
Tel: (818) 505-5600

MEDITERRANEAN AND MIDDLE EAST
Hewlett-Packard S.A.
Mediterranean and Middle East Operations
Atrina Centre
32 Kifissias Ave
Paradissos-Amarousion, **ATHENS**
Greece
Tel: 682 88 11
Telex: 21-6588 HPAT GR
Cable: HEWPACKSA Athens

OTHER INTERNATIONAL AREAS
Hewlett-Packard Co
Intercontinental Headquarters
3495 Deer Creek Road
PALO ALTO, CA 94304
Tel: (415) 857-1501
Telex: 034-8300
Cable: HEWPACK

ARGENTINA
Hewlett-Packard Argentina S.A.
Montaneses 2140/50
1428 **BUENOS AIRES**
Tel: 781-4059/69
Cable: HEWPACKARG

AUSTRALIA
Hewlett-Packard Australia Ltd
31-41 Joseph Street
P.O. Box 221
BLACKBURN, Victoria 3130
Tel: 895-2895
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia Ltd.
17-23 Talavera Road
P.O. Box 308
NORTH RYDE, N.S.W. 2113
Tel: 888-4444
Telex: 21561
Cable: HEWPARD Sydney

AUSTRIA
Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O. Box 72
A-1222 **VIENNA**
Tel: (0222) 2500-0
Telex: 134425 HEPA A

BELGIUM
Hewlett-Packard Belgium S.A./N.V
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru

BRAZIL
Hewlett-Packard do Brasil
I.e.C. Ltda.
Alameda Rio Negro, 750
ALPHAVILLE
06400 Barueri SP
Tel: (011) 421.1311
Telex: (011) 33872 HPBR-BR
Cable: HEWPACK Sao Paulo

Hewlett-Packard do Brasil
I.e.C. Ltda.
Praia de Botafago 228
6° Andar-conj 614
Edificio Argentina - Ala A
22250 **RIO DE JANEIRO**, RJ
Tel: (021) 552-6422
Telex: 21905 HPBR-BR
Cable: HEWPACK Rio de Janeiro

CANADA
Hewlett-Packard (Canada) Ltd
11120-178th Street
EDMONTON, Alberta T5S 1P2
Tel: (403) 486-6666

Hewlett-Packard (Canada) Ltd.
17500 Trans Canada Highway
South Service Road
KIRKLAND, Quebec H9J 2X8
Tel: (514) 697-4232
Telex: 058-21521

Hewlett-Packard (Canada) Ltd
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
Telex: 069-8644

Hewlett-Packard (Canada) Ltd.
2670 Queensview Dr.
OTTAWA, Ontario K2B 8K1
Tel: (613) 820-6483

CHINA, People's Republic of
China Hewlett-Packard Co., Ltd.
P.O. Box 9610, Beijing
4th Floor, 2nd Watch Factory Main Bldg.
Shuang Yu Shou, Bei San Huan Road
Hai Dian District
BEIJING
Tel: 28-0567
Telex: 22601 CTSHP CN
Cable: 1920 Beijing

DENMARK
Hewlett-Packard A/S
Kongevejen 25
DK-3460 **BIRKEROED**
Tel: (02) 81-66-40
Telex: 37409 hpas dk

FINLAND
Hewlett-Packard Oy
Pilspankalliontie 17
02200 **ESPOO**
Tel: 00358-0-88721
Telex: 121563 HEWPA SF

FRANCE
Hewlett-Packard France
Chemin des Mouilles
Boite Postale 162
69131 **ECULLY** Cedex (Lyon)
Tel: (78) 133-81-25
Telex: 310617F

Hewlett-Packard France
Parc d'activités du Bois Briard
Avenue du Lac
91040 **EVRY** Cedex
Tel: (60) 77-83-83
Telex: 692315F

Hewlett-Packard France
Zone Industrielle de Courtaboeuf
Avenue des Tropiques
91947 **LES ULIS** Cedex (Orsay)
Tel: (69) 07-78-25
Telex: 600048F

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrum Mitte
Hewlett-Packard-Strasse
D-6380 **BAD HOMBURG**
Tel: (06172) 16-0
Fax: (06172) 16-1309

Hewlett-Packard GmbH
Vertriebszentrun Südwest
Schickardstrasse 2
D-7030 **BÖBLINGEN**
Tel: (07031) 14-0
Fax: (07031) 14-6429

Hewlett-Packard GmbH
Vertriebszentrum Süd
Eschenstrasse 5
D-8028 **TAUFKIRCHEN**
Tel: (089) 61207-0
Fax: (089) 61207-300

GREECE
Hewlett-Packard A.E.
178, Kifissias Avenue
6th Floor
Halandri-**ATHENS**
Greece
Tel: 6471543, 6471673, 6472971
Telex: 221 286 HPHLGR

HONG KONG
Hewlett-Packard Hong Kong, Ltd.
G.P.O. Box 795
5th Floor, Sun Hung Kai Centre
30 Harbour Road
HONG KONG
Tel: 5-8323211
Telex: 66678 HEWPA HX
Cable: HEWPACK HONG KONG

ICELAND
Hewlett-Packard Iceland
Hoefdabakka 9
110 **REYKJAVIK**
Tel: (1) 67 1000

INDIA
Blue Star Ltd.
13 Community Center
New Friends Colony
NEW DELHI 110 065
Tel: 633182, 636674
Telex: 031-61120
Cable: BLUEFROST

INDONESIA
BERCA Indonesia P.T.
P.O.Box 2497/Jkt
Antara Bldg., 11th Floor
Jl. Medan Merdeka Selatan 17
JAKARTA-PUSAT
Tel: 343989
Telex: 46748 BERSAL IA

IRELAND
Hewlett-Packard Ireland Ltd.
82/83 Lower Leeson Street
DUBLIN 2
Tel: 0001 608800
Telex: 30439

ISRAEL
Computation and Measurement Systems (CMS) Ltd.
11 Masad Street
67060
TEL-AVIV
Tel: 388 388
Telex: 33569 Motil IL

ITALY
Hewlett-Packard Italiana S.p.A.
Via G. di Vittorio 9
I-20063 **CERNUSCO SUL NAVIGLIO**
(Milano)
Tel: (02) 923691
Telex: 334632

Hewlett-Packard Italiana S.p.A.
Viale C. Pavese 340
I-00144 **ROMA EUR**
Tel: (06) 54831
Telex: 610514

JAPAN
Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg.,
4-20 Nishinakajima, 5 Chome
Yodogawa-ku
OSAKA, 532
Tel: (06) 304-6021
Telex: YHPOSA 523-3624

Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido-Higashi, 3 Chome
Suginami-ku **TOKYO** 168
Tel: (03) 331-6111
Telex: 232-2024 YHPTOK

Yokogawa-Hewlett-Packard Ltd.
Yasuda Seimei Nishiguchi Bldg.
30-4 Tsuruya-cho, 3 Chome
Kanagawa-ku, **YOKOHAMA** 221
Tel: (045) 312-1252

SALES & SUPPORT OFFICES

Arranged alphabetically by country

KOREA
Samsung Hewlett-Packard Co. Ltd
Dongbang Yeoeuido Building
12-16th Floors
36-1 Yeoeuido-Dong
Youngdeungpo-Ku
SEOUL
Tel: 784-4666, 784-2666
Telex: 25166 SAMSAN K

MALAYSIA
Hewlett-Packard Sales (Malaysia)
Sdn. Bhd.
9th Floor
Chung Khiaw Bank Building
46, Jalan Raja Laut
50350 **KUALA LUMPUR**
Tel: 2986555
Telex: 31011 HPSM MA

MEXICO
Hewlett-Packard de Mexico,
S.A. de C.V.
Monte Pelvoux No. 111
Lomas de Chapultepec
11000 **MEXICO, D.F.**
Tel: 5-40-62-28, 72-66, 50-25
Telex: 17-74-507 HEWPACK MEX

NETHERLANDS
Hewlett-Packard Nederland B.V.
Startbaan 16
NL-1187 XR **AMSTELVEEN**
P.O. Box 667
NL-1180 AR **AMSTELVEEN**
Tel: (020) 547-6911
Telex: 13 216 HEPA NL

NORWAY
Hewlett-Packard Norge A/S
Osterndalen 16-18
P.O. Box 34
N-1345 **OESTERAAS**
Tel: 0047/2/24 60 90
Telex: 76621 hpnas n

PUERTO RICO
Hewlett-Packard Puerto Rico
101 Muñoz Rivera Av
Esu. Calle Ochoa
HATO REY, Puerto Rico 00918
Tel: (809) 754-7800

SAUDI ARABIA
Modern Electronics Establishment
Hewlett-Packard Division
P.O. Box 1228
Redec Plaza, 6th Floor
JEDDAH
Tel: 644 96 28
Telex: 4027 12 FARNAS SJ
Cable: ELECTA JEDDAH

SINGAPORE
Hewlett-Packard Singapore (Sales) Pte. Ltd
#08-00 Inchcape House
450-2 Alexandra Road
Alexandra P.O. Box 58
SINGAPORE, 9115
Tel: 4731788
Telex: 34209 HPSGSO RS
Cable: HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard So Africa (Pty.) Ltd
9 Eastern Service Road
Eastgate Ext. 3
SANDTON 2144
Tel: 802-5111, 802-5125
Telex: 4-20877 SA
Cable: HEWPACK Johannesburg

SPAIN
Hewlett-Packard Española, S.A.
Crta. de la Coruña, Km. 16, 400
Las Rozas
E-MADRID
Tel: (1) 637.00.11
Telex: 23515 HPE

SWEDEN
Hewlett-Packard Sverige AB
Skalholtsgatan 9, Kista
Box 19
S-16393 **SPÅNGA**
Tel: (08) 750-2000
Telex: (854) 17886
Telefax: (08) 7527781

SWITZERLAND
Hewlett-Packard (Schweiz) AG
7, rue du Bois-du-Lan
Case postale 365
CH-1217 **MEYRIN 1**
Tel: (0041) 22-83-11-11
Telex:27333 HPAG CH

TAIWAN
Hewlett-Packard Taiwan Ltd.
8th Floor, Hewlett-Packard Building
337 Fu Hsing North Road
TAIPEI
Tel: (02) 712-0404
Telex: 24439 HEWPACK
Cable:HEWPACK Taipei

TURKEY
Teknim Company Ltd.
Iran Caddesi No. 7
Karaklidere
ANKARA
Tel: 275800
Telex: 42155 TKNM TR

UNITED KINGDOM

ENGLAND
Hewlett-Packard Ltd.
Heathside Park Road
Cheadle Heath
STOCKPORT
Cheshire
SK3 ORB
Tel: 061-428-0828
Telex: 668068

Hewlett-Packard Ltd.
King Street Lane
Winnersh, **WOKINGHAM**
Berkshire RG11 5AR
Tel: 0734 784774
Telex: 847178

SCOTLAND
Hewlett-Packard Ltd.
SOUTH QUEENSFERRY
West Lothian, EH30 9TG
Tel: 031 331 1188
Telex: 72682

UNITED STATES

Alabama
Hewlett-Packard Co.
420 Wynn Drive
HUNTSVILLE, AL 35805
Tel: (205) 830-2000

Arizona
Hewlett-Packard Co.
8080 Pointe Parkway West
PHOENIX, AZ 85044
Tel: (602) 273-8000

California
Hewlett-Packard Co.
1421 S. Manhattan Av.
FULLERTON, CA 92631
Tel: (714) 999-6700

Hewlett-Packard Co.
5651 West Manchester Ave.
LOS ANGELES, CA 90045
Tel: (213) 337-8000
Telex: 910-325-6608

Hewlett-Packard Co.
9606 Aero Drive
SAN DIEGO, CA 92123
Tel: (619) 279-3200

Hewlett-Packard Co.
3003 Scott Boulevard
SANTA CLARA, CA 95054
Tel: (408) 988-7000
Telex: 910-338-0586

Colorado
Hewlett-Packard Co.
24 Inverness Place, East
ENGLEWOOD, CO 80112
Tel: (303) 649-5000

Connecticut
Hewlett-Packard Co.
47 Barnes Industrial Road South
WALLINGFORD, CT 06492
Tel: (203) 265-7801

Florida
Hewlett-Packard Co.
2901 N.W. 62nd Street
FORT LAUDERDALE, FL 33309
Tel: (305) 973-2600

Hewlett-Packard Co.
6177 Lake Ellenor Drive
ORLANDO, FL 32809
Tel: (305) 859-2900

Georgia
Hewlett-Packard Co.
2000 South Park Place
ATLANTA, GA 30339
Tel: (404) 955-1500
Telex: 810-766-4890

Illinois
Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800
Telex: 910-687-1066

Indiana
Hewlett-Packard Co.
11911 N. Meridian St.
CARMEL, IN 46032
Tel: (317) 844-4100

Louisiana
Hewlett-Packard Co.
160 James Drive East
ST. ROSE, LA 70087
P.O. Box 1449
KENNER, LA 70063
Tel: (504) 467-4100

Maryland
Hewlett-Packard Co.
3701 Koppers Street
BALTIMORE, MD 21227
Tel: (301) 644-5800
Telex: 710-862-1943

Hewlett-Packard Co.
2 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370

Massachusetts
Hewlett-Packard Co.
1775 Minuteman Road
ANDOVER, MA 01810
Tel: (617) 682-1500

Michigan
Hewlett-Packard Co.
39550 Orchard Hill Place Drive
NOVI, MI 48050
Tel: (313) 349-9200

Minnesota
Hewlett-Packard Co.
2025 W. Larpenteur Ave.
ST. PAUL, MN 55113
Tel: (612) 644-1100

Missouri
Hewlett-Packard Co.
1001 E. 101st Terrace Suite 120
KANSAS CITY, MO 64131-3368
Tel: (816) 941-0411

Hewlett-Packard Co.
13001 Hollenberg Drive
BRIDGETON, MO 63044
Tel: (314) 344-5100

New Jersey
Hewlett-Packard Co.
120 W. Century Road
PARAMUS, NJ 07653
Tel: (201) 265-5000

New Mexico
Hewlett-Packard Co.
7801 Jefferson N.E.
ALBUQUERQUE, NM 87109
Tel: (505) 823-6100

New York
Hewlett-Packard Co.
9600 Main Street
CLARENCE, NY 14031
Tel: (716) 759-8621

Hewlett-Packard Co.
7641 Henry Clay Blvd.
LIVERPOOL, NY 13088
Tel: (315) 451-1820

Hewlett-Packard Co.
3 Crossways Park West
WOODBURY, NY 11797
Tel: (516) 682-7800

North Carolina
Hewlett-Packard Co.
5605 Roanne Way
GREENSBORO, NC 27420
Tel: (919) 852-1800

Ohio
Hewlett-Packard Co.
15885 Sprague Road
CLEVELAND, OH 44136
Tel: (216) 243-7300

Hewlett-Packard Co.
9080 Springboro Pike
MIAMISBURG, OH 45342
Tel: (513) 433-2223

Hewlett-Packard Co.
675 Brooksedge Blvd.
WESTERVILLE, OH 43081
Tel: (614) 891-3344

Oklahoma
Hewlett-Packard Co.
3525 N.W. 56th St.
Suite C-100
OKLAHOMA CITY, OK 73112
Tel: (405) 946-9499

Oregon
Hewlett-Packard Co.
9255 S. W. Pioneer Court
WILSONVILLE, OR 97070
Tel: (503) 682-8000

Pennsylvania
Hewlett-Packard Co.
111 Zeta Drive
PITTSBURGH, PA 15238
Tel: (412) 782-0400

Hewlett-Packard Co.
2750 Monroe Boulevard
VALLEY FORGE, PA 19482
Tel: (215) 666-9000

Texas
Hewlett-Packard Co.
1826-P Kramer Lane
AUSTIN, TX 78758
Tel: (512) 835-6771

Hewlett-Packard Co.
10535 Harwin Drive
HOUSTON, TX 77036
Tel: (713) 776-6400

Hewlett-Packard Co.
930 E. Campbell Rd.
RICHARDSON, TX 75081
Tel: (214) 231-6101

Hewlett-Packard Co.
1020 Central Parkway South
SAN ANTONIO, TX 78232
Tel: (512) 494-9336

Utah
Hewlett-Packard Co.
3530 W. 2100 South St.
SALT LAKE CITY, UT 84119
Tel: (801) 974-1700

Virginia
Hewlett-Packard Co.
4305 Cox Road
GLEN ALLEN, VA 23060
Tel: (804) 747-7750

Washington
Hewlett-Packard Co.
15815 S.E. 37th Street
BELLEVUE, WA 98006
Tel: (206) 643-4000

Wisconsin
Hewlett-Packard Co.
275 N. Corporate Dr.
BROOKFIELD, WI 53005
Tel: (414) 794-8800

VENEZUELA
Hewlett-Packard de Venezuela C.A.
3A Transversal Los Ruices Norte
Edificio Segre 2 & 3
Apartado 50933
CARACAS 1050
Tel: (582) 239-4133
Telex: 251046 HEWPACK

YUGOSLAVIA
Do Hermes
General Zdanova 4
YU-11000 **BEOGRAD**
Tel: (011) 342 641
Telex: 11433